HP49G Entry Reference

Complete listing sorted by functionality
Edition 2.11, 30 May 2005

Carsten Dominik, Thomas Rast & Eduardo M. Kalinowski

Table of Contents

1 Introduction

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

1.1 Disclaimer and Acknowledgments

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

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

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

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

Here is a list of sources which have been used.

Programming in System RPL by Eduardo Kalinowski
: This book has been a major source for the database. The entire book has been reverse-engineered using pdftotext and then a variety of Emacs and Perl programs to extract and format the reference part of the book.

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

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

ML entry descriptions by Peter Geelhoed
: Peter Geelhoed created the initial version of the ML section for this document.

HP48/49 entry cross-reference by Joe Horn
: This document has been used to make a list of entries for the HP49 in the first place, and to add and double-check addresses for both calculators.

Various posts on `comp.sys.hp48`

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

Supported entry lists from HP

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

Further contributions

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

1.2 Terminology

1.2.1 Abbreviations used in Stack Diagrams

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

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

UserRPL stack diagrams use some additional abbreviations

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

1.2.2 Unsupported Entry Points

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

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

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

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

1.2.3 More Information

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

2 HP Objects

2.1 Binary Integers

2.1.1 Built-in BINTS 0-127

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
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
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 aka: FORTYSEVEN
332E7	BINT48	48d 30h aka: FORTYEIGHT
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
3339B	BINT66	66d 42h aka: FOURTWO
333A5	BINT67	67d 43h aka: FOURTHREE
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
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
335E9	BINT125	125d 7Dh
335F3	BINT126	126d 7Eh
335FD	BINT127	127d 7Fh

2.1.2 Built-in BINTS 127-255

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	(SYMBCMP)	146d 92h
3364D	(SYMBSYM)	154d 9Ah

33657	SYMBUNIT	158d 9Eh
3EAFB	(#9F)	159d 9Fh
33661	(backup)	159d 9Fh
3366B	SYMOB	160d A0h
33675	SYMREAL	161d A1h
3367F	(SYMCMP)	162d A2h
39E6B	(SYMARRY)	164d A4h
33689	(SYMLIST)	165d A5h
33693	SYMID	166d A6h
3369D	SYMLAM	167d A7h
336A7	(SYMSYMB)	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
3371F	BINT253	253d FDh
33729	BINT255d	255d FFh

2.1.3 Built-in BINTS 256-

33733	REALOBOB	256d 100h
3373D	#_102	258d 102h
33747	#SyntaxErr	262d 106h
33751	(BINT_263d)	263d 107h
3375B	(REALREALOB)	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	(Err#Cont)	318d 13Eh
33805	INTEGER337	337d 151h
3380F	(CMPOBOB)	512d 200h
33819	(Err#NoLstArg)	517d 205h
3A1C2	(#304)	772d 304h
33823	STRREALREAL	785d 311h
3B9FA	(#313)	787d 313h
3C11E	(ARRYREALOB)	1040d 410h
3B928	(#411)	1041d 411h
3382D	(ARRYREALREAL)	1041d 411h
33837	(ARRYREALCMP)	1042d 412h
3BA2D	(#414)	1044d 414h
3B93D	(#415)	1045d 415h
33841	(3ARRY)	1092d 444h
3C10F	(ARRYLISTOB)	1104d 450h
3B952	(#451)	1105d 451h
3384B	(ARRYLISTREAL)	1105d 451h
33855	(ARRYLISTCMP)	1106d 452h
3BA18	(#454)	1108d 454h
3B913	(#455)	1109d 455h
3A12D	(#4FF)	1279d 4FFh
3385F	(LISTREALOB)	1296d 510h
33869	(LISTREALREAL)	1297d 511h

3BA09	(#515)	1301d 515h
33873	(LISTLISTOB)	1360d 550h
277F6	(LN_0)	1541d 605h
27800	(LN_Neg)	1542d 606h
2780A	(InvalidEQ)	1543d 607h
27814	(Cureq#)	1544d 608h
2781E	(NoCureq#)	1545d 609h
27828	(EnterEq#)	1546d 60Ah
27832	(EnterName#)	1547d 60Bh
2783C	(SelPtype#)	1548d 60Ch
27846	(EmptyCat#)	1549d 60Dh
2768E	(#60E)	1550d 60Eh
27698	(NoStatPlot#)	1551d 60Fh
3387D	(IDREALOB)	1552d 610h
276AC	(SolvingFor#)	1553d 611h
276B6	(NoCurrent#)	1554d 612h
276C0	(PressSig+#)	1555d 613h
276CA	(SelectModl#)	1556d 614h
276D4	(NoAlarms#)	1557d 615h
276DE	(PressALRM#)	1558d 616h
276E8	(NextALRM#)	1559d 617h
27792	(PastDue#)	1560d 618h
2779C	(Acknowledge#)	1561d 619h
277A6	(KeyInAlrm#)	1562d 61Ah
277B0	(SelectRpt#)	1563d 61Bh
277BA	(IOSetupMenu#)	1564d 61Ch
277C4	(PlotType#)	1565d 61Dh
277CE	(NoExecAct#)	1566d 61Eh
277D8	(OffScreen#)	1567d 61Fh
277E2	(OnlyPtypes#)	1568d 620h
277EC	(StatName#)	1569d 621h
276F2	(ZoomPrompt#)	1570d 622h
276FC	(CatToStack#)	1571d 623h
27706	(XAutoZoom#)	1572d 624h
27710	(IR/wire#)	1576d 628h
2771A	(ASCII/bin#)	1577d 629h
27724	(#62A)	1578d 62Ah
2772E	(#62B)	1579d 62Bh

27738	(#62C)	1580d 62Ch
27742	(#62D)	1581d 62Dh
27788	(EnterMatrix#)	1582d 62Eh
33887	(IDLISTOB)	1616d 650h
33891	(FSTMACROROM#)	1792d 700h
3C17A	(#710)	1808d 710h
3C16B	(#750)	1872d 750h
08DF7	(#7FF)	2047d 7FFh
27878	(BINT800h)	2048d 800h
3B976	(#822)	2082d 822h
3C83C	(#82C)	2092d 82Ch
3B967	(#855)	2133d 855h
3C81E	(#85C)	2140d 85Ch
3389B	(PROGIDREAL)	2145d 861h
338A5	(PROGIDCMP)	2146d 862h
338AF	(PROGIDLIST)	2149d 865h
338B9	(PROGIDEXT)	2158d 86Eh
3E7FF	(#8F1)	2289d 8F1h
3E759	(#8FD)	2301d 8FDh
3E7E9	(#9F1)	2545d 9F1h
3E743	(#9FD)	2557d 9FDh
2774C	(Lackint#)	2561d A01h
27756	(Constant#)	2562d A02h
27882	Attn#	2563d A03h
338C3	ATTNERR	2563d A03h
27760	(Zero#)	2564d A04h
2776A	(RevSgn#)	2565d A05h
27774	(Extremum#)	2566d A06h
338CD	(SYMREALREAL)	2577d A11h
338D7	(SYMREALCMP)	2578d A12h
338E1	(SYMREALSYM)	2586d A1Ah
338EB	(SYMCMPREAL)	2593d A21h
338F5	(SYMCMPCMP)	2594d A22h
338FF	(SYMCMPSYM)	2602d A2Ah
33909	(SYMIDREAL)	2657d A61h
33913	(SYMIDCMP)	2658d A62h
3391D	(SYMIDLIST)	2661d A65h
33927	(SYMIDEXT)	2670d A6Eh

33931	(SYMSYMREAL)	2721d AA1h
3393B	(SYMSYMCMP)	2722d AA2h
33945	(3SYM)	2730d AAAh
3394F	(XFERFAIL)	3078d C06h
33959	(PROTERR)	3079d C07h
33963	(InvalServCmd)	3080d C08h
3396D	Connecting	3082d C0Ah
33977	(Retry)	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	(TYPEINT)	9748d 2614h
03FF9	(TYPEMATRIX)	9862d 2686h
03F8B	TYPEREAL	10547d 2933h
03FDB	(TYPEEREL)	10581d 2955h
03FA9	TYPEIDNT	10568d 2948h
03F95	(TYPECMP)	10615d 2977h
03F9F	(TYPELIST)	10868d 2A74h
03FC7	(TYPERRP)	10902d 2A96h
03FBD	(TYPESYMB)	10936d 2AB8h
03FE5	(TYPEEXT)	10970d 2ADAh
03FB3	(TYPECOL)	11677d 2D9Dh
03FA9	TYPEIDNT	10568d 2948h
03FD1	(TYPELAM)	11885d 2E6Dh
3C8DF	(#5B11)	23313d 5B11h
3D50D	(SYMRRANY)	41232d A110h
3D52B	(SYMRSYMANY)	41376d A1A0h
3D51C	(SYMSYMRANY)	43536d AA10h
2C4D2	(SYMSYMSYMANY)	43680d AAA0h
3B7AD	(#BBBB)	48059d BBBBh
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

2.1.4 Pushing Several BINTs

37287	ZEROZERO	(→ #0 #0)
37294	#ZERO#ONE	(→ #0 #1)
37305	#ZERO#SEVEN	(→ #0 #7)
36B12	ONEONE	(→ #1 #1) aka: ONEDUP
37315	#ONE#27	(→ #1 #27d)
37328	#TWO#ONE	(→ #2 #1)
3733A	#TWO#TWO	(→ #2 #2)
3734A	#TWO#FOUR	(→ #2 #4)
3735C	#THREE#FOUR	(→ #3 #4)
3736E	#FIVE#FOUR	(→ #5 #4)
37380	ZEROZEROZERO	(→ #0 #0 #0)
37394	ZEROZEROONE	(→ #0 #0 #1)
373A8	ZEROZEROTWO	(→ #0 #0 #2)
3558C	DROPZERO	(ob → #0)
37711	(3DROPZERO)	(ob ob ob → #0)
355A5	2DROP00	(ob ob → #0 #0)
3596D	DROPONE	(ob → #1)
36AD6	DUPZERO	(ob → ob ob #0)
36AEA	DUPONE	(ob → ob ob #1)
36B26	DUPTWO	(ob → ob ob #2)
36AFE	SWAPONE	(ob ob' → ob' ob #1)
35E75	ZEROSWAP	(ob → #0 ob)
360BB	ZEROOVER	(ob → ob #0 ob)
36568	ZEROFALSE	(→ #0 F)
35EA2	ONESWAP	(ob → #1 ob)

3657C	ONEFALSE	(→ #1 F)

2.1.5 Conversion

262F1	COERCE	(% → #)
35D08	COERCEDUP	(% → # #)
35EB6	COERCESWAP	(ob % → # ob)
3F481	COERCE2	(% %' → # #')
262EC	%ABSCOERCE	(% → #)
2F244	(Flag%isUser?)	(% → # flag) TRUE if real is greater 0, else FALSE.
2F31F	C%>#	(C% → # #')
05A03	HXS>#	(hxs → #)
2F17E	2HXSLIST?	({ hxs hxs' } → # #') Converts list of two hxs to two bints. Generates "Bad Argument Value" for invalid input.
05A51	CHR>#	(chr → #)
0EF006	^Z2BIN	(Z → #) Convert Z to bint. Returns FFFFF for overflows. Returns 0 for negative numbers.
19D006	^Z>#	(z → #) 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 → #2 #1) Converts 2 zints to bints.

2.1.6 Arithmetic Functions

03DBC	#+	(# #' → #+#')
03DEF	#1+	(# → #+1)
03E2D	#2+	(# → #+2)
355FD	#3+	(# → #+3)
35602	#4+	(# → #+4)
35607	#5+	(# → #+5)
3560C	#6+	(# → #+6)
35611	#7+	(# → #+7)
35616	#8+	(# → #+8)
3561B	#9+	(# → #+9)
35620	#10+	(# → #+10)

35625	(#11+)	(# → #+11)
3562A	#12+	(# → #+12)
03DE0	#-	(# #' → #-#')
2F13D	(DIFF_OR_ZERO)	(# #' → #'') If #' is greater than #, returns #0, otherwise returns #-#'.
03E0E	#1-	(# → #-1)
03E4E	#2-	(# → #-2)
355DF	#3-	(# → #-3)
355DA	#4-	(# → #-4)
355D5	#5-	(# → #-5)
355D0	#6-	(# → #-6)
355CB	(#7-)	(# → #-7)
355C6	(#8-)	(# → #-8)
355C1	(#9-)	(# → #-9)
03EC2	#*	(# #' → #*#')
2632D	#*OVF	(# #' → #*#') $0 \leq$ result $\leq$ FFFFF
03E6F	#2*	(# → #*2)
270DA	#3*	(# → #*2)
270BF	#5*	(# → #*2)
356B8	#6*	(# → #*6)
3569B	#8*	(# → #*8)
35675	#10*	(# → #*10)
03EF7	#/	(# #' → #r #q)
03E8E	#2/	(# → #/2) Rounded down.
36815	#1--	(# #' → #-#'+1) aka: #-+1
36851	#1-+	(# #' → #+#'-1) $1-+ is a typo in EXTABLE. aka: #+-1, $1-+
35552	#-#2/	(# #' → (#-#')/2)
357FC	#+DUP	(# #' → #+#' #+#')
35E39	#+SWAP	(ob # #' → #+#' ob)
36093	#+OVER	(ob # #' → ob #+#' ob)
3581F	#-DUP	(# #' → #-#' #-#')
35E4D	#-SWAP	(ob # #' → #-#' ob)
360A7	#-OVER	(ob # #' → ob #-#' ob)
35830	#1+DUP	(# → #+1 #+1)
35E61	#1+SWAP	(ob # → #+1 ob)

2F222	#1+ROT	(ob ob' # → ob' #+1 ob)
35841	#1-DUP	(# → #-1 #-1)
28071	#1-SWAP	(ob # → #-1 ob) aka: pull
3601B	#1-ROT	(ob ob' # → ob' #-1 ob)
281D5	#1-UNROT	(ob ob' # → #-1 ob ob')
35E89	#1-1SWAP	(# → 1 #-1) Returns the bint ONE and the result.
35912	DUP#1+	(# → # #+1)
3571E	DUP#2+	(# → # #+2)
35956	DUP#1-	(# → # #-1)
3674D	2DUP#+	(# #' → # #' #+#') aka: DUP3PICK#+
3683D	DROP#1-	(# ob → #-1)
357BB	SWAP#-	(# #' → #'-#)
3592B	SWAP#1+	(meta ob → meta&ob) aka: SWP1+
29786	('RSWP1+)	(# → nob #+1) nob is the next object in the runstream.
28099	SWAP#1+SWAP	(# ob → #+1 ob)
36829	SWAP#1-	(# ob → ob #-1)
280AD	SWAP#1-SWAP	(# ob → #-1 ob)
28989	(SWAPDROP#1-)	(ob # → #-1)
367ED	SWAPOVER#-	(# #' → #' #-#')
36775	OVER#+	(# #' → # #'+#)
367C5	OVER#-	(# #' → # #'-#)
28286	(OVER#1-)	(# #' → # #' #'')
36761	ROT#+	(# ob #' → ob #'+#)
367B1	ROT#-	(# ob #' → ob #'-#)
36801	ROT#1+	(# ob ob' → ob ob' #+1)
28001	ROT#1+UNROT	(# ob ob' → #+1 ob ob')
35E07	ROT#+SWAP	(# ob #' → #'+# ob) aka: ROT+SWAP
36789	3PICK#+	(# ob #' → # ob #'+#)
28804	(3PICK#1+)	(# ob ob' → # ob ob' #')
287E6	(3PICK#2+)	(# ob ob' → # ob ob' #')
3679D	4PICK#+	(# ob1 ob2 #' → # ob1 ob2 #'+#)
35E20	4PICK#+SWAP	(# ob1 ob2 #' → # ob1 #'+# ob2) aka: 4PICK+SWAP
35511	#MIN	(# #' → #'')
3551D	#MAX	(# #' → #'')

03EB1	#AND	(# #’ → #’’) Bitwise AND.

2.1.7 Tests

03D19	#=	(# #’ → flag)
03D4E	#<>	(# #’ → flag)
03CE4	#<	(# #’ → flag)
37466	(#<=)	(# #’ → flag)
03D83	#>	(# #’ → flag)
3747D	(#>=)	(# #’ → flag)
03CC7	#0<>	(# → flag)
03CA6	#0=	(# → flag)
3530D	#1<>	(# → flag)
352FE	#1=	(# → flag)
36711	#2<>	(# → flag)
352F1	#2=	(# → flag)
352E0	#3=	(# → flag)
366FD	#5=	(# → flag)
366BC	#<3	(# → flag)
36739	#>1	(# → flag) aka: ONE#>
358C2	2DUP#<	(# #’ → # #’ flag)
358F8	2DUP#>	(# #’ → # #’ flag)
363CE	ONE_EQ	(# → flag) Uses EQ test.
35268	OVER#=	(# #’ → # flag)
358DC	2DUP#=	(# #’ → # #’ flag)
36694	OVER#0=	(# #’ → # #’ flag)
352BD	DUP#0=	(# → # flag)
366A8	OVER#<	(# #’ → # flag)
3531C	DUP#1=	(# → # flag)
36725	OVER#>	(# #’ → # flag)
3532B	DUP#0<>	(# → # flag)
366D0	DUP#<7	(# → # flag) Returns TRUE if the argument is smaller than #7.
36676	2#0=OR	(# # → flag) Returns TRUE if either argument is zero.

2.2 Real Numbers

2.2.1 Built-in Real Numbers

2FB0A	%-MAXREAL	-9.99E499
30B24	(%-260)	-260
2FAB1	%-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	%0	0
2FB1F	%MINREAL	1E-499
2FF71	(%.05)	.05
27118	%.1	.1
2712D	(%.15)	.15
2FF47	(%.2776)	.2776
2FF1D	(%.2887)	.2887
2FF5C	(%.2943)	.2943
2FEF3	(%.461368)	.461368
2FF32	(%.522851)	.522851
339BE	%.5	.5
339D3	(%-.5)	-.5
2FF86	(%.99)	.99
2F94C	%1	1
270EE	(%1.8)	1.8
2F961	%2	2
339A9	%e	e
2F976	%3	3
2FAC6	%PI	π
2F98B	%4	4
2F9A0	%5	5
2F9B5	%6	6
2F9CA	%7	7

2F9DF	%8	8
2F9F4	%9	9
339E8	%10	10
2FCE6	%11	11
2FCFB	%12	12
2FD10	%13	13
2FD25	%14	14
2FD3A	%15	15
2FD4F	%16	16
2FD64	%17	17
2FD79	%18	18
2FD8E	%19	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
2FF08	(%50)	50
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

0CF0B5	(~%TICKSsec)	8192
2FCBC	(%9600)	9600
26DF7	(%14400)	14400 First available in ROM 1.22.
2FCD1	(%15360)	15360
2FCD1	(%15396)	15396
26E21	(%38400)	38400 First available in ROM 1.22.
26E36	(%57600)	57600 First available in ROM 1.22.
26E4B	(%115200)	115200 First available in ROM 1.22.
0CD0B5	(~%TICKSmin)	491520
0CB0B5	(~%HrTicks)	29491200
0C70B5	(~%TICKSweek)	4954521600
2FAF5	%MAXREAL	9.99E499
2F180	1REV	(→ 6.28318530718) (→ 360.) (→ 400.) Returns the angle of a full circle, corresponding to the current angular mode.

2.2.2 Built-in Extended Real Numbers

2FB49	%%0	0
2FBE5	%%.1	0.1
30DC8	%%.4	0.4
2FBFF	%%.5	0.5
2DA11	cfF	0.555... %%5/9 for C↔F conversion.
2FB63	%%1	1
2DA2B	cfC	1 For C↔K conversion.
2FB7D	%%2	2
2FB97	%%3	3
2FADB	%%PI	π
30017	PI/180	π/180
2FBB1	%%4	4
2FBCB	%%5	5
27A89	%%2PI	2π
30BEA	%%7	7

2FC19	%%10	10
30CC7	%%12	12
30CEB	%%60	60

2.2.3 Stack Manipulation Combined with Reals

282CC	(DROP%0)	(ob → %0)
2C4AA	(2DROP%0)	(ob ob' → %0)
2C4AA	(2DROP%0)	(ob ob' → %0)

2.2.4 Conversion

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

2.2.5 Real Functions

3035F	%+	(% %' → %+%')
25E69	%+SWAP	(ob % %' → %+%' ob)
26F36	%1+	(% → %+1)
3036C	%-	(% %' → %-%')
26F4A	%1-	(% → %-1)
30346	%>%%-	(% %' → %%-%%')

303A7	%*	(% %' → %*%')
35C18	%10*	(% → %*10)
303E9	%/	(% %' → %/%')
3045B	%^	(% %' → %^%')
302EB	%ABS	(% → %')
2C53B	(DUP%ABS)	(% → % %')
3030B	%CHS	(% → -%)
302C2	%SGN	(% → -1/0/1)
3049A	%1/	(% → 1/%)
30489	%>%%1/	(% → 1/%%)
304F4	%SQRT	(% → $\sqrt{a}$%)
3A4BE	(%2root)	(% → $\sqrt{a}$%) (% → C%) Computes square root of real, returns a complex number for negative arguments.
304E1	%>%%SQRT	(% → $\sqrt{a}$%%)
3A54B	(%SQ)	(% → %')
3051A	%EXP	(% → e^%)
3052D	%EXPM1	(% → e^%-1)
30559	%LN	(% → LN%)
30592	%LNP1	(% → LN(%+1))
3056C	%LOG	(% → LOG%)
305A5	%ALOG	(% → 10^%)
305DA	%SIN	(% → SIN%)
3062B	%COS	(% → COS%)
3067C	%TAN	(% → TAN%)
306AC	%ASIN	(% → ASIN%)
306DC	%ACOS	(% → ACOS%)
3070C	%ATAN	(% → ATAN%)
30799	%SINH	(% → SINH%)
307C5	%COSH	(% → COSH%)
307D8	%TANH	(% → TANH%)
307EB	%ASINH	(% → ASINH%)
307FE	%ACOSH	(% → ACOSH%)
30811	%ATANH	(% → ATANH%)
3031B	%MANTISSA	(% → %mant)
30824	%EXPONENT	(% → %expn)
30938	%FP	(% → %frac)
3094B	%IP	(% → %int)

30971	%FLOOR	(% → %maxint <=%)
3095E	%CEIL	(% → %minint >=%)
305C7	%MOD	(% %' → %rem)
30723	%ANGLE	(%x %y → %ang)
3A3D1	(%0%ANGLE)	(%x → %ang) %ANGLE with y=0;
30746	%>%%ANGLE	(%x %y → %%ang)
30F14	RNDXY	(% %places → %')
30F28	TRCXY	(% %places → %')
3084D	%COMB	(% %' → COMB(%,%'))
30860	%PERM	(% %' → PERM(%,%'))
30837	%NFACT	(% → %!) Calculates factorial of number.
30AAF	%FACT	(% → gamma(%+1)) Calculates gamma(x+1).
3046C	%NROOT	(% %n → %') Calculates the %nth root of the real number. Equivalent to user function XROOT.
3A30E	SWAP%NROOT	(%n % → %') Calculates the %nth root of the real number. Equivalent to user function XROOT.
300F9	%MIN	(% %' → %lesser)
300E0	%MAX	(% %' → %greater)
35DBC	%MAXorder	(% %' → %max %min)
309AD	%RAN	(→ %random) Returns next random number.
30A2F	%RANDOMIZE	(%seed →) System level RDZ: seeds the random number generator.
30A66	DORANDOMIZE	(% →) Stores given number as random number seed.
303B4	%OF	(% %' → %'/% * 100)
303F6	%T	(% %' → %pctotal)
3041B	%CH	(% %' → %pcchange)
3000D	%D>R	(%deg → %rad)
30040	%R>D	(%rad → %deg)
30E79	%REC>%POL	(%r %ang → %x %y)
30EA6	%POL>%REC	(%x %y → %r %ang)
30EDD	%SPH>%REC	(%r %ang %ph → %x %y %z)

2.2.6 Extended Real Functions

3032E	%%+	(%% %%' → %%+%%')
27012	(%%1+)	(%% → %%')
3033A	%%-	(%% %%' → %%-%%')
30385	%%*	(%% %%' → %%*%%')
3602F	%%*ROT	(ob ob' %% %%' → ob' %%+%%' ob)
35EDE	%%*SWAP	(ob %% %%' → %%+%%' ob)
36C7C	%%*UNROT	(ob ob' %% %%' → %%+%%' ob ob')
303D3	%%/	(%% %%' → %%/%%')
36C22	SWAP%%/	(%% %%' → %%'')
36BE6	%%/>%	(%% %%' → %)
3044A	%%^	(%% %%' → %%^%%')
51D006	^CK%%SQRT	(%% → %%/C%%)
30612	%%SINRAD	(%% → %%')
30767	%%ANGLERAD	(%% → %%')
302DB	%%ABS	(%% → %%abs)
306F3	%%ACOSRAD	(%% → %%rad)
3073A	%%ANGLE	(%%x %%y → %%ang)
30757	%%ANGLEDEG	(%%x %%y → %%deg)
306C3	%%ASINRAD	(%% → %%rad)
302FB	%%CHS	(%% → -%%)
3047D	%%1/	(%% → 1/%%)
30642	%%COS	(%% → %%cos)
30653	%%COSDEG	(%%deg → %%cos)
307B2	%%COSH	(%% → %%cosh)
30663	%%COSRAD	(%%rad → %%cos)
30507	%%EXP	(%% → e^%%)
30546	%%LN	(%% → ln %%)
30984	%%FLOOR	(%% → %%maxint)
		aka: %%INT
3057F	%%LNP1	(%% → %%ln(%%+1))
300C7	%%MAX	(%% %%' → %%max)
30E83	%%R>P	(%%x %%y → %%radius %%angle)
30EB0	%%P>R	(%%r %%ang → %%x %%y)
305F1	%%SIN	(%% → %%sin)
30602	%%SINDEG	(%%deg → %%sin)
30780	%%SINH	(%% → %%sinh)
304D5	%%SQRT	(%% → $\sqrt{a}$%%)
30693	%%TANRAD	(%%rad → %%tan)

2D817	(%%TANDEG)	(%%deg → %%tan)

2.2.7 Tests

302AC	%=	(% %' → flag)
302B7	%<>	(% %' → flag)
3025C	%<	(% %' → flag)
302A1	%<=	(% %' → flag)
30275	%>	(% %' → flag)
3028B	%>=	(% %' → flag)
3CA61	(XEQAND)	(% %' → flag) Logical AND for real numbers.
3CAE7	(XEQOR)	(% %' → flag) Logical OR for real numbers.
3CB5D	(XEQNOT)	(% → flag) Logical NOT for real numbers.
3CBCA	(XEQXOR)	(% %' → flag) Logical XOR for real numbers.
30156	%0=	(% → flag)
36C0E	DUP%0=	(% → flag)
301BA	%0<>	(% → flag) Can be used to change a user flag into a system flag.
30123	%0<	(% → flag)
30184	%0>	(% → flag)
301E2	%0>=	(% → flag)
3020A	%%<	(%% %%' → flag)
30296	%%<=	(%% %%' → falg)
3026A	%%>	(%% %%' → flag)
30280	%%>=	(%% %%' → flag)
30145	%%0=	(%% → flag)
2708A	(DUP%%0=)	(%% → %% flag)
301A6	%%0<>	(%% → flag)
30112	%%0<	(%% → flag)
301F6	%%0<=	(%% → flag)
30173	%%0>	(%% → flag)
301CE	%%0>=	(%% → flag)

2.3 Complex Numbers

2.3.1 Built-in Complex Numbers

27DE4	C%0	(0,0)
27E09	C%1	(1,0)
27DBF	C%-1	(-1,0)
27E2E	C%%1	(%%1,%%0)

2.3.2 Conversion

261D9	C%%>C%	(C%% → C%)
05C27	%>C%	(%re %im → C%)
362F2	SWAP%>C%	(%im %re → C%)
261FC	Re>C%	(%re → C%)
25E9C	C>Re%	(C% → %re)
25E9B	C>Im%	(C% → %im)
18C006	^E%%>C%%	(%%re %%im → C%%) Converts long reals to long complex.
261CF	%%>C%	(%%re %%im → C%)
25E82	C%>%%	(C% → %%re %%im)
25E83	C%>%%SWAP	(C% → %%im %%re)
05DBC	C%%>%%	(C%% → %%re %%im)
188006	^C2C%%	(C → C%%) Converts Gaussian integer to long complex.
189006	^ZZ2C%%ext	(Zre Zim → C%%) Converts Gaussian integer to long complex.
18B006	^C%>C%%	(C% → C%%) Converts complex to long complex.
15E006	^RIXCext	(Zre Zim → C) Convert integers to complex.
15F006	^IRXCext	(Zim Zre → C) Convert integers to complex.
160006	^IRXC2	

2.3.3 Functions

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	(C% → %) Absolute value.
261ED	C%CHS	(C% → -C%)
25E81	C%1/	(C% → 1/C%)
25E98	C%SQRT	(C% → $\sqrt{a}$C%)
25E95	C%SGN	(C% → C%/C%ABS)
261F2	C%CONJ	(C% → C%')
25E88	C%ARG	(C% → %)
25E91	C%EXP	(C% → e^C%)
25E92	C%LN	(C% → ln C%)
25E93	C%LOG	(C% → log C%)
25E87	C%ALOG	(C% → 10^C%)
25E96	C%SIN	(C% → sin C%)
25E8D	C%COS	(C% → cos C%)
25E99	C%TAN	(C% → tan C%)
25E89	C%ASIN	(C% → asin C%)
25E85	C%ACOS	(C% → acos C%)
25E8B	C%ATAN	(C% → atan C%)
25E97	C%SINH	(C% → sinh C%)
25E8E	C%COSH	(C% → cosh C%)
25E9A	C%TANH	(C% → tanh C%)
25E8A	C%ASINH	(C% → asinh C%)
25E86	C%ACOSH	(C% → acosh C%)
25E8C	C%ATANH	(C% → atanh C%)
05C72	(%%>C%%)	(%%re %%im → C%%)
261DE	C%%CHS	(C%% → -C%%)
261E3	C%%CONJ	(C%% → C%%')
515006	^ARG2	(im re → arg(ob)) ARG.
516006	^INTERNALARG2	
517006	^QUADRANT	(re im ?re>0 ?im>0 → newre newim Z) Returns Z0 Z1 Z-2 or Z-1 so that arg of corresponding complex number is Z * $\pi/2$ + theta where θ is in the interval [0,$\pi/2$]. The arguments on level 1 and 2 are flags.
51E006	^C%%SQRT	(C%% → C%%')

2.3.4 Tests

261E8	C%0=	(C% → flag)

261D4	C%%0=	(C%% → flag)

2.4 Character Strings

2.4.1 Built-in Characters

33D2B	CHR_00	'\00', CHR 0d 00h The NULL character.
33F77	CHR_Newline	'\0a', CHR 10d 0Ah
33D32	CHR_...	'...', CHR 31d 1Fh
33F93	CHR_Space	' ', CHR 32d 20h The space character.
33D39	CHR_DblQuote	'"', CHR 34d 22h
33D40	CHR_#	'#', CHR 35d 23h
33F70	CHR_LeftPar	'(', CHR 40d 28h
33F85	CHR_RightPar	')', CHR 41d 29h
33D47	CHR_*	'*', CHR 42d 2Ah
33D4E	CHR_+	'+', CHR 43d 2Bh
33D55	CHR_,	',', CHR 44d 2Ch
33D5C	CHR_-	'-', CHR 45d 2Dh
33D63	CHR_.	'.', CHR 46d 2Eh
33D6A	CHR_/	'/', CHR 47d 2Fh
33D71	CHR_0	'0', CHR 48d 30h
33D78	CHR_1	'1', CHR 49d 31h
33D7F	CHR_2	'2', CHR 50d 32h
33D86	CHR_3	'3', CHR 51d 33h
33D8D	CHR_4	'4', CHR 52d 34h
33D94	CHR_5	'5', CHR 53d 35h
33D9B	CHR_6	'6', CHR 54d 36h
33DA2	CHR_7	'7', CHR 55d 37h
33DA9	CHR_8	'8', CHR 56d 38h
33DB0	CHR_9	'9', CHR 57d 39h
33DB7	CHR_:	':', CHR 58d 3Ah
33DBE	CHR_;	';', CHR 59d 3Bh
33DC5	CHR_<	'<', CHR 60d 3Ch
33DCC	CHR_=	'=', CHR 61d 3Dh
33DD3	CHR_>	'>', CHR 62d 3Eh
33DDA	CHR_A	'A', CHR 65d 41h
33DE1	CHR_B	'B', CHR 66d 42h

33DE8	CHR_C	'C', CHR 67d 43h
33DEF	CHR_D	'D', CHR 68d 44h
33DF6	CHR_E	'E', CHR 69d 45h
33DFD	CHR_F	'F', CHR 70d 46h
33E04	CHR_G	'G', CHR 71d 47h
33E0B	CHR_H	'H', CHR 72d 48h
33E12	CHR_I	'I', CHR 73d 49h
33E19	CHR_J	'J', CHR 74d 4Ah
33E20	CHR_K	'K', CHR 75d 4Bh
33E27	CHR_L	'L', CHR 76d 4Ch
33E2E	CHR_M	'M', CHR 77d 4Dh
33E35	CHR_N	'N', CHR 78d 4Eh
33E3C	CHR_O	'O', CHR 79d 4Fh
33E43	CHR_P	'P', CHR 80d 50h
33E4A	CHR_Q	'Q', CHR 81d 51h
33E51	CHR_R	'R', CHR 82d 52h
33E58	CHR_S	'S', CHR 83d 53h
33E5F	CHR_T	'T', CHR 84d 54h
33E66	CHR_U	'U', CHR 85d 55h
33E6D	CHR_V	'V', CHR 86d 56h
33E74	CHR_W	'W', CHR 87d 57h
33E7B	CHR_X	'X', CHR 88d 58h
33E82	CHR_Y	'Y', CHR 89d 59h
33E89	CHR_Z	'Z', CHR 90d 5Ah
33FA1	CHR_[	'[', CHR 91d 5Bh
33FA8	CHR_]	']', CHR 93d 5Dh
33F9A	CHR_UndScore	'_', CHR 95d 5Fh
33E90	CHR_a	'a', CHR 97d 61h
33E97	CHR_b	'b', CHR 98d 62h
33E9E	CHR_c	'c', CHR 99d 63h
33EA5	CHR_d	'd', CHR 100d 64h
33EAC	CHR_e	'e', CHR 101d 65h
33EB3	CHR_f	'f', CHR 102d 66h
33EBA	CHR_g	'g', CHR 103d 67h
33EC1	CHR_h	'h', CHR 104d 68h
33EC8	CHR_i	'i', CHR 105d 69h
33ECF	CHR_j	'j', CHR 106d 6Ah
33ED6	CHR_k	'k', CHR 107d 6Bh

33EDD	CHR_l	'l', CHR 108d 6Ch
33EE4	CHR_m	'm', CHR 109d 5Dh
33EEB	CHR_n	'n', CHR 110d 6Eh
33EF2	CHR_o	'o', CHR 111d 6Fh
33EF9	CHR_p	'p', CHR 112d 70h
33F00	CHR_q	'q', CHR 113d 71h
33F07	CHR_r	'r', CHR 114d 72h
33F0E	CHR_s	's', CHR 115d 73h
33F15	CHR_t	't', CHR 116d 74h
33F1C	CHR_u	'u', CHR 117d 75h
33F23	CHR_v	'v', CHR 118d 76h
33F2A	CHR_w	'w', CHR 119d 77h
33F31	CHR_x	'x', CHR 120d 78h
33F38	CHR_y	'y', CHR 121d 79h
33F3F	CHR_z	'z', CHR 122d 7Ah
33FAF	CHR_{	'{', CHR 123d 7Bh
33FB6	CHR_}	'{', CHR 125d 7Dh
33F5B	CHR_Angle	'∠', CHR 128d 80h
33F69	CHR_Integral	'∫', CHR 132d 84h
33F62	CHR_Deriv	'∂', CHR 136d 88h
33F46	CHR_->	'→', CHR 141d 8Dh
33F4D	CHR_<<	'≪', CHR 171d ABh
33F54	CHR_>>	'≫', CHR 187d BBh
33F7E	CHR_Pi	'π', CHR 135d 87h
33F8C	CHR_Sigma	'Σ', CHR 133d 85h
33FBD	CHR_<=	'≤', CHR 137d 89h
33FC4	CHR_>=	'≥', CHR 138d 8Ah
33FCB	CHR_<>	'≠', CHR 139d 8Bh
37A78	(CHR_A8)	'\A8', CHR 168d A8h

2.4.2 Built-in Strings

055DF	NULL$	"" Empty string.
33B55	SPACE$	" " aka: `tok_`
272E5	(MARKED)	" " String of 2 spaces.

33B13	(14SPACES$)	" " String of 14 spaces.
33B39	NEWLINE$	"\0a" Newline.
27195	CRLF$	"\0d\0a" Carriage return and line feed.
33BB5	(toklparen)	"("
33BC1	(tokrparen)	")"
33A6B	(tok[)	"["
33A51	(tok])	"]"
33A77	tok{	"{"
33A83	(tok})	"}"
33AD7	tok<<	"≪"
33ACB	(tok>>)	"≫"
34048	$_LRParens	"()"
3401E	$_[]	"[]"
34010	$_{}	"{}"
34002	$_<<>>	"≪≫"
3402C	$_’’	"’’" Two single quotes.
3403A	$_::	"::"
34056	$_2DQ	"""" Two double quotes.
33B91	tok,	","
33B85	tok’	"’" One single quote.
33BFD	tok-	"-"
33B9D	tok.	"."
33C09	tok=	"="
272D9	tok->	"→"
2D848	tok_g	"g"
2D86D	tok_m	"m"
2D8AD	tok_s	"s"
33C4D	tok0	"0"
33C59	tok1	"1"
33C65	(tok2)	"2"
33C71	(tok3)	"3"
33C7D	(tok4)	"4"
33C89	(tok5)	"5"
33C95	(tok6)	"6"

33CA1	(tok7)	"7"
33BA9	(tok;)	";"
33CAD	tok8	"8"
33CB9	tok9	"9"
33ABF	tokESC	"\1B" Escape character.
33AE3	tokexponent	"E"
33B79	tokquote	""" One double quote.
33A8F	toksharp	"#"
33AA7	(tok$)	"$"
33AB3	(tok&)	"&"
33BD9	(tok*)	"*"
33BF1	(tok+)	"+"
33BE5	(tok/)	"/"
33AEF	(tokanglesign)	"∠"
33C21	(tokDER)	"∂"
33B45	($DER)	"der"
33AFB	(tokSIGMA)	"Σ"
33C15	(tokSQRT)	"$\sqrt{a}$"
33A9B	(tokuscore)	"_"
33B07	(tokWHERE)	"|"
33BCD	(tok^)	"^"
33D1F	($_...)	"\1F" Character 31, the forward arrow (system font) or dots (minifont).
2723F	(tok:)	":"
2724B	(tok‘)	"‘" One backquote.
2D933	(tok?)	"?"
340A4	$_RAD	"RAD"
340B4	$_GRAD	"GRAD"
33FF2	$_XYZ	"XYZ"
33FE2	$_R<Z	"R∠Z" "R<angle>Z"
33FD2	$_R<<	"R∠∠" "R<angle><angle>"
2D90F	(tokmol)	"mol"
2D8ED	(tokcd)	"cd"
2D8CD	(tokK)	"K"

2D88D	(tokA)	"A"
2D7FF	(tokdegR)	"\^oR" Degrees R.
2D7B3	(tokr)	"r"
2D7D3	(toksr)	"sr"
34076	$_EXIT	"EXIT"
34064	$_ECHO	"ECHO"
34088	$_Undefined	"Undefined"
33C2D	(tokCTGROB)	"GROB"
33C3F	(tokCTSTR)	"C$"
33B61	(tokUNKNOWN)	"UNKNOWN"
27221	(tokTO)	"TO"
2722F	(tokDIR)	"DIR"
27257	(tokELSE)	"ELSE"
27269	(tokEND)	"END"
27279	(tokUNTIL)	"UNTIL"
2728D	(tokREPEAT)	"REPEAT"
272A3	(tokNEXT)	"NEXT"
272B5	(tokSTEP)	"STEP"
272C7	(tokTHEN)	"THEN"
27C0B	($1:_)	"1: "
27EB4	(<Skip$)	"→SKIP"
27F00	(>Skip$)	"SKIP→"
27F4C	(<Del$)	"→DEL"
27F9F	(>Del$)	"DEL→"
3DF97	(tokIntercept)	"Intercept"
3DFB3	(tokSlope)	"Slope"
37F5C	(tokIF-prompt)	"IF-prompt"
34133	(tokCopyright)	"Copyright HP xxxx"
340CB	(tokVersion)	"Version HP49-B..."

2.4.3 Built-in Strings with Stack Manipulation

35D94	NULL$SWAP	(ob → $ ob) NULL$, then SWAP.
04D3E	DROPNULL$	(ob → NULL$) DROP then NULL$.
04D57	(TWODROPNULL$)	(ob ob' → NULL$) 2DROP then NULL$.

25EEC	NULL$TEMP	(→ $) Creates null string in temporary memory (NULL$, then <REF>TOTEMPOB).

2.4.4 Conversion

25F77	#>$	(# → $) Creates string from the bint (decimal).
25F72	#:>$	(# → "#: ") 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 → $) Converts identifier into string.
25EB3	DOCHR	(% → $) Creates string of the character with the number specified.
0F1006	^Z>S	(Z → $) Converts Z into a string (decimal).
2EFC1	hxs>$	(hxs → $) Uses current display mode and wordsize.
2EFC0	HXS>$	(hxs → $) Does <REF>hxs>$ and then appends base character.

2.4.5 Management

05A75	#>CHR	(# → chr) Returns character with the specified ASCII code.
37AA5	CHR>$	(chr → $* Strings) Converts a character into a string.
05636	LEN$	($ → #length) Returns length in bytes.
357E2	DUPLEN$	($ → $ #) DUP then LEN$.
05622	OVERLEN$	($ ob → $ ob #len) OVER then LEN$.
361DA	NEWLINE$&$	($ → "$\0a") Appends newline character to string. aka: NEWLINE&$
2F31A	APNDCRLF	($ → $') Appends carriage return and line feed to string.

050ED	CAR$	($ → chr) ($ → "") Returns first character of string as a string, or NULL$ for null string.
0516C	CDR$	($ → $') Returns string without first character, or NULL$ for null string.
378FA	POS$	($ $find start# → #pos) ($ $find start# → #0) Search for $find in $search, starting at position #start. Returns position of $find or 0 if not found. Same entry as POSCHR.
378FA	POSCHR	($search chr #start → #pos) ($search chr #start → #0) Same entry as <REF>POS$.
37906	POS$REV	($ $find #limit → #pos) ($ $find #limit → #0) Searches backwards from #limit to #1. Same entry as <REF>POSCHRREV.
37906	POSCHRREV	($seach chr #start → #pos) ($seach chr #start → #0) Same entry as <REF>POS$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; $' the substring after the first newline.
09A003	(^StrCutNchr)	($ #width → $') Replace SPACE chars with NEWLINE in order to fit the text in the given #width. This entry will produce lines longer than#width characters if a single word is longer than that. Used by ViewStrObject. Very fast (bang type).
09B003	(^StrCutNchr2)	($ #width #lines → $' #lines') Replace SPACE chars with NEWLINE in order to fit the text in the given #width. If a single word is longer than #width, the word is cut into pieces. The output will not be longer than #lines lines. #lines' gives the number of lines in $'.

05733	SUB$	($ #start #end → $') Returns substring between specified positions.
2F2C0	(XEQSUB$)	($ % %' → $') Same as <REF>SUB$ but uses real numbers as arguments.
3628E	#1-SUB$	($ #start #end+#1 → $') Does #1- and then SUB$.
362A2	1_#1-SUB$	($ #end → $') Returns substring with the first #end characters. aka: 1_#1-SUB
362B6	LAST$	($ #start → $') Returns substring from the specified start position to the end (inclusive).
362CA	#1+LAST$	($ #start-#1 → $') Returns substring from the specified start position to the end (exclusive).
29F0C	(DEL_END$)	($ → $') Removes the last character from a string.
35DA8	SUB$SWAP	(ob $ # #' → $' ob) SUB$ then SWAP.
2A5CA	SUB$1#	($ #pos → #') Returns bint with ASCII code of character at the specified position.
34C82	EXPAND	(hxs #nibs → hxs') Appends #nibs zero nibbles to the hxs.
05193	&$	($ $' → $+$') Concatenates two strings.
36FF6	&$SWAP	(ob $ $' → $+$' ob) &$ then SWAP.
353CD	!append$	($ $' → $+$') Tries &$, if not enough memory does !!append$?.
3533C	!insert$	($ $' → $'+$) Does SWAP then <REF>!append$.
35F6A	!append$SWAP	(ob $ $' → $+$' ob) !append$ then SWAP.
35369	!!append$?	($ $' → $+$') Attempts append "in place" if target is in tempob.
353F7	!!append$	($ $' → $+$') Tries appending "in place".
353EB	!!insert$	($ $' → $'+$) Tries inserting "in place".
0525B	>H$	($ chr → $') Prepends character to string
052EE	>T$	($ chr → $') Appends character to string.

35BD7	APPEND_SPACE	($ → $') Appends space to string.
35346	SWAP&$	($ $' → $'+$) Concatenates two strings.
2EED3	TIMESTR	(%dt %tm → "dy dt tm") Returns string representation of time, using current format. Example: "WED 06/24/98 10:00:45A"
25E7C	AND$	($1 $2 → $') Logical AND. Errors if strings are not the same length.
25EF0	OR$	($ $' → $'') Logical OR. Errors if strings are not the same length.
25F0D	XOR$	($ $' → $'') Logical XOR. Errors if strings are not the same length.
2647C	(!NOT$)	($ $' → $'' ???) Logical NOT "in place".
2646D	(!AND$)	($ $' → $'' ???) Logical AND. Does not check if strings are the same length.
26472	(!OR$)	($ $' → $'' ???) Logical OR, does not check if strings are the same length.
26477	(!XOR$)	($ $' → $'' ???) Logical XOR. Does not check if strings are the same length.
2F1A7	CHARSEDIT	(→) HP49 character browser. This is an interactive application from which characters can be echoed into the command line.

2.4.6 Parsing Strings

25EB7	DOSTR>	($ → ?) Internal version of <REF>STR→.
2EF62	palparse	($ → ob T) ($ → $ #pos $' F) Tries parsing a string into an object. If successful, returns object and TRUE, otherwise returns position of error, the offending part of the string $', and FALSE. If the string contains several arguments, the resulting object is a secondary containing these objects.

00E004	^algparse	($ → ob T) ($ → $ # #' F) Tries parsing a string into an object using algebraic mode. If successful, returns object and TRUE, otherwise returns the original string with information about the position of the error, and FALSE.
25E68	!*trior	(F → <SKIP>) (T T → <COLA>)
25E67	!*triand	(T T →) (F T → F T <SEMI>)
26206	tok8cktrior	($1 $1 → :: $1 <Ob1> ;) ($1 $2 → :: $1 <Ob2> <Rest> ;)
261BB	tok8trior	(GNT data $1 $1 → :: GNT data GetNextToken ;) (GNT data $1 $2 → :: $1 <Ob1> <Rest> ;)
29E67	nultrior	(NULL$ → :: ;) ($ → :: $ <Ob1> <Rest> ;)
25EDB	GetNextToken	(hxs-mask $ #start → hxs-mask $ #next $token)
2F33C	getmatchtok	(hxs-mask $ #loc $_tok → hxs-mask $ #next $match)
2EF6A	Parse.1	
2EF6B	Parse.2	
2EF6E	ParseFail	(ob $parsed #pos $' →) Uses DispBadToken to re-edit the parsed string and displays "Syntax Error".
2EF70	ParseFail2	
2EF6F	DispBadToken	(ob $parsed #pos $' →) Re-edits the parsed string, positions the cursor to the location of the error. Used by ParseFail.
2EF71	DispBadToken2	

2.4.7 Decompilation

2F191	!DcompWidth	(# →) Sets the width (in characters) of decompiled strings. This width is used to cut the resulting string (for stack display) or to break it into lines (mostly for editing). Note that most decompilation entries reset this value to the stack or editor width. Use stkdecomp$w and editdecomp$w to make sure the current width is used and not changed.
2F190	DcompWidth@	(→ #) Recalls the width of decompiled strings (in characters).

26459	`setStdWid`	(→) Sets `DcompWidth` to the standard value for stack display, either 19 or 30 characters, depending on system flag 72 (stack minifont). -- Flags: -72
2645E	`setStdEditWid`	(→) Sets `DcompWidth` to the width for editing, either 21 or 32 characters, depending on system flag 73 (edit minifont). -- Flags: -73
25F13	`stkdecomp$w`	(ob → $) Decompiles for stack display using the current `DcompWidth` to cut the string if it is too long.
25E6D	`1stkdecomp$w`	(ob → $) Calls `setStdWid` and decompiles for stack display (cutting the string if necessary).
2A842	`Decomp1Line`	(ob → $) Same as <REF>1stkdecomp$w.
2A904	`RPNDecomp1Line`	(ob → $) Same as <REF>Decomp1Line but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95
25E6F	`>Review$`	(id → $) Makes a string from the variable name and its contents (decompiled with <REF>Decomp1Line), for display with the review key. If the argument is a command, returns its name.
2A8E4	`DecompStd1Line32`	(ob → $) Sets 32 as `DcompWidth` and decompiles using `stkdecomp$w`.
2A9C4	`RPNDecompStd1Line32`	(ob → $) Same as <REF>DecompStd1Line32 but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95
2A8C9	`DecompStd1Line`	(ob → $) Calls `setStdWid` and decompiles, cutting if the string becomes too long.
2A9A4	`RPNDecompStd1Line`	(ob → $) Same as <REF>DecompStd1Line but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95

2A893	Decomp#Disp	(ob # → $) Calls setStdWid and decompiles ob (UserRPL components only), breaks the string into lines and returns the first #+1 lines. Used for multiline display in stack level 1.
2A964	RPNDecomp#Disp	(ob # → $) Same as Decomp#Disp but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95
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 multiline display in stack level 1.
2A944	RPNDecomp#Line	(ob # → $) Same as Decomp#Line but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95
25F11	editdecomp$w	(ob → $) Decompiles entire object for editing. It only decompiles the UserRPL components. Some System RPL entries like <REF>TakeOver are simply skipped, others are written as "External". Breaks the resulting strings into lines using the current DcompWidth.
25ECE	EDITDECOMP$	(ob → $) Calls setStdEditWid and the decompiles for editing like <REF>editdecomp$w.
2A85D	DecompEdit	(ob → $) Same as EDITDECOMP$.
2A924	RPNDecompEdit	(ob → $) Same as DecompEdit but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95
2AA43	AlgDecomp	(ob → $) Calls <REF>DecompEdit with a few checks around it.
25EAA	DECOMP$	(ob → $) Calls <REF>setStdWid and decompiles entire object (UserRPL components only). Breaks the string into lines using DcompWidth as width.
39CB3	(Ob,$>$')	(ob $ → "ob$") Applies <REF>DECOMP$ to ob and concatenates with the string.

39C9F	($,Ob>$')	($ ob → "$ob") Applies <REF>DECOMP$ to ob and concatenates with the string.
25EB1	DO>STR	($ → $) (ob → $) Internal version of →STR.
1A7006	^DO>STRID	(id/ob → $) Like <REF>DO>STR but without quotes for id.
2A8AE	DecompEcho	(ob → $) Calls `setStdEditWid` and decompiles the entire object (UserRPL only) into a single line.
2A984	RPNDecompEcho	(ob → $) Same as <REF>DecompEcho but enforce RPN mode (system flag 95 clear) during execution. -- Flags: -95
2F1BF	Decomp%Short	(% #width → $) Decompiles a real number into a string of the given #width. It will drop less significant 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 → $) The decompiler used by `stkdecomp$w`, `1stkdecomp$w`, `Decomp1Line`, `DecompStd1Line32`. `DcompWidth` must be set before this is called.
002004	^FSTR2	
003004	^FSTR3	(ob # → $) The decompiler used by `Decomp#Line`. `DcompWidth` must be set before this is called.
004004	^FSTR4	(ob → $) The decompiler used by `editdecomp$w`, `DecompEdit`, `EDITDECOMP$`. `DcompWidth` must be set before this is called.
005004	^FSTR5	(ob → $) The decompiler used by `DecompEcho`. `DcompWidth` must be set before this is called.
006004	^FSTR6	(ob # → $) The decompiler used by `Decomp#Line`. `DcompWidth` must be set before this is called.
007004	^FSTR7	(ob → $) The decompiler used by `DO>STR`. `DcompWidth` must be set before this is called.
008004	^FSTR8	

009004	^FSTR9	(ob → $) The decompiler used by DecompStd1Line. DcompWidth must be set before this is called.
00A004	^FSTR10	
00B004	^FSTR11	
00C004	^FSTR12	
00D004	^FSTR13	(ob → $) The decompiler used by DECOMP$. DcompWidth must be set before this is called.
35B82	palrompdcmp	(romptr → $ T) Decompiles a rompointer for the UserRPL stack. If it is a named rompointer, returns the name. Otherwise returns "XLIB n m".

2.4.8 String Tests

0556F	NULL$?	(ob → flag)
36252	DUPNULL$?	(ob → ob flag)
26436	($>$?)	($ $' → flag) String comparizon, alphabetically by character numbers.
2F321	CkChr00	($ → $ flag) Returns FALSE if string contains any null characters.

2.5 HEX Strings

2.5.1 Built-in HEX Strings

3ABD2	(hxsB010)	HXS 4 B010
399ED	(CHSpdata)	HXS 4 0108
3CB4A	(hxs0105)	HXS 4 0105
3A4B0	(PDataNSQRT)	HXS 4 010C
39C79	(hxs70107)	HXS 5 70107
39F2E	(hxs80108)	HXS 5 80108
39F2E	(hxs80108)	HXS 5 80108
3CA52	(hxs50105)	HXS 5 50105
3CAD8	(hxs40104)	HXS 5 40104
3CCA5	(hxs60106)	HXS 5 60106
3A17F	(ParseDataN^)	HXS 5 A0109
3DB8F	(hxsA0127)	HXS 5 A0127

3D719	(hxs014250)	HXS 6 014250
3A07D	(ParseDataPdiv)	HXS 7 8014050
3A18E	(ParseDataP^)	HXS 7 0405109
3D28F	(hxs0134250)	HXS 7 0134250
3D7C0	(hxs014360950)	HXS 9 014360950
39666	(hxs0140626250)	HXS A 0140626250
3D619	(hxs2214370B50)	HXS A 2214370B50
3D497	(INTGPDATA)	HXS C 014060626350
3D549	(SUMETCPDATA)	HXS C 014370606250

2.5.2 Conversion

059CC	#>HXS	(# → hxs) Length will be five.
2EFCB	%>#	(% → hxs) Converts real number into hxs. Should be called %>HXS.

2.5.3 General Functions

2EFBE	WORDSIZE	(→ #) Returns the current wordsize as a bint.
2EFAA	dostws	(# →) Sets the current wordsize.
055D5	NULLHXS	HXS 0 Puts a null hxs in the stack.
05566	(NULLHXS?)	(hxs → flag) Returns TRUE if the input is a null hxs.
0518A	&HXS	(hxs hxs' → hxs'') Appends hxs” to hxs’.
34C82	EXPAND	(hxs #nibs → hxs') Appends #nibs zero nibbles to the hxs.
05616	LENHXS	(hxs → #nibs) Returns length in nibbles.
05815	SUBHXS	(hxs #m #n → hxs') Returns sub hxs string.
2EFB9	bit+	(hxs hxs' → hxs'') Adds two hxs.
2EFC8	bit%#+	(% hxs → hxs') Adds real to hxs, returns hxs.
2EFC9	bit#%+	(hxs % → hxs') Adds real to hxs, returns hxs.

2EFBA	bit-	(hxs hxs' → hxs'') Subtracts hxs2 from hxs1.
2EFC6	bit%#-	(% hxs → hxs') Subtracts hxs from real, returns hxs.
2EFC7	bit#%-	(hxs % → hxs') Subtracts real from hxs, returns hxs.
2EFBC	bit*	(hxs hxs' → hxs'') Multiplies two hxs.
2EFC4	bit%#*	(% hxs → hxs') Multiplies real by hxs, returns hxs.
2EFC5	bit#%*	(hxs % → hxs') Multiplies hxs by real, returns hxs.
2EFBD	bit/	(hxs hxs' → hxs'') Divides hxs1 by hxs2.
2EFC2	bit%#/	(% hxs → hxs') Divides real by hxs, returns hxs.
2EFC3	bit#%/	(hxs % → hxs') Divides hxs by real, returns hxs.
2EFAC	bitAND	(hxs hxs' → hxs'') Bitwise AND.
2EFAD	bitOR	(hxs hxs' → hxs'') Bitwise OR.
2EFAE	bitXOR	(hxs hxs' → hxs'') Bitwise XOR.
2EFAF	bitNOT	(hxs → hxs') Bitwise NOT.
2EFB8	bitASR	(hxs → hxs') Arithmetic shift one bit to the right. The most significant bit (the sign) does not change.
2EFB6	bitRL	(hxs → hxs') Shifts circularly one bit to the left.
2EFB7	bitRLB	(hxs → hxs') Shifts circularly one byte to the left
2EFB4	bitRR	(hxs → hxs') Shifts circularly one bit to the right.
2EFB5	bitRRB	(hxs → hxs') Shifts circularly one byte to the right.
2EFB0	bitSL	(hxs → hxs') Shifts one bit to the left.
2EFB1	bitSLB	(hxs → hxs') Shifts one byte to the left.
2EFB2	bitSR	(hxs → hxs') Shifts one bit to the right.
2EFB3	bitSRB	(hxs → hxs') Shifts one byte to the right.

2.5.4 Tests

2EFCC	HXS==HXS	(hxs hxs' → %flag) == test
2F0EE	HXS#HXS	(hxs hxs' → %flag) ≠ test
2EFCF	HXS<HXS	(hxs hxs' → %flag) < test
2EFCD	HXS>HXS	(hxs hxs' → %flag) > test
2EFCE	HXS>=HXS	(hxs hxs' → %flag) ≥ test
2F0EF	HXS<=HXS	(hxs hxs' → %flag) ≤ test

2.6 Tagged Objects

05E81	>TAG	(ob $tag → tagged) Tags an object.
2F266	USER$>TAG	(ob $tag → tagged) Maximum of 255 characters in string.
2F223	%>TAG	(ob % → tagged) Converts real to string using current display mode and tags object.
05F2E	ID>TAG	(ob id/lam → tagged) Tags object with identifier or lam.
05E9F	({}>TAG)	({ id ob } → tagged)
37B04	TAGOBS	(ob $tag → tagged) (ob.. { $.. } → tagged...) Tags one or more objects.
05EC9	(TAG>)	(tagged → ob $tag)
37ABE	STRIPTAGS	(tagged → ob) Strips all tags from the object.
37AEB	STRIPTAGS12	(tagged ob' → ob ob') Strips all tags from the object in level two.

2.7 Arrays

2.7.1 General Functions

03562	(ARSIZE)	([] → #) Returns number of elements as a bint.

035A9	(DIMLIMITS)	`( [] → {#n #m} )`
		Returns list of array dimensions.
0371D	GETATELN	`( # [] → ob T )`
		`( # [] → F )`
		Gets one element from array.
03685	(ARRYEL?)	`( {#n #m} [] → # T )`
		`( {#n #m} [] → F )`
		Returns `TRUE` if array element exists.
03685	(FINDELN)	`( {} A → # flag )`
		Return index # of element `{}` in array.
16D006	^MDIMS	`( [[]] → #rows #cols T )`
		`( [] → #elem F )`
		Returns the size of an array. Equivalent to the HP48 command `MDIMS`.
35FD8	MDIMSDROP	`( [2D] → #m #n )`
		`MDIMS` followed by `DROP`.
16E006	^DIMLIMITS	`( [] → { # } )`
		`( [[]] → {# #} )`
		Returns the size of an array, like the User command `SIZE`, but the lengths are bints and not reals. Equivalent to the HP48 command `DIMLIMITS`.
35E006	^ARSIZE	`( [] → # )`
		Returns max # in an array.
36183	OVERARSIZE	`( [] ob → [] ob #elts )`
		Does `OVER` then <REF>ARSIZE.
260F8	PULLREALEL	`( [%] # → [%] % )`
		Gets real element.
260F3	PULLCMPEL	`( [C%] # → [C%] C% )`
		Gets complex element.
26102	PUTEL	`( [%] % # → [%]' )`
		`( [C%] C% # → [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% # → [C%]' )`
		Puts complex element at specified position. Warning: no copy to tempob first.
33B006	^MATTRAN	`( M → M' )`
		Matrix transposition.
331006	^Yext	`( V2 V1 → ob )`
		Scalar product of symbolic vectors, no check.
2F1D5	(MATR>C)	`( [%re] [%im] → [C%] )`
		Creates complex matrix from real and imaginary parts.

2F1D6	(MATC>R)	`( [C%] → [%re] [%im] )` Explodes complex matrix into real and imaginary parts.

2.7.2 Conversion

169006	^BESTMATRIXTYPE	`( ob → ob )` Converts symbolic matrix with real/cmplex entries to a numeric array.
172006	^CKNUMARRY	`( ob → ob )` Tests if ob is a numeric array. Tries to convert symbolic array to numeric array.

178006	^MATRIX2ARRAY	`( [] → [] )`
		`( [[]] → [[]] )`
		Tries to convert a symbolic matrix to a numeric one.

001007	^ListToArry	`( {}/{{}} → []/[[]] TRUE )`
		`( {}/{{}} → FALSE )`
		If possible, converts list of lists to normal array (containing only real or complex numbers) and returns `TRUE`. Otherwise, returns `FALSE`.

03442	(MAKEARRY)	`( {#n #m} ob → [] )` Makes array with all elements initialized to ob.

17F006	^XEQ>ARRY	`( ob1...obn {%n} → [] )`
		`( ob11...obmn {%m %n} → [[mxn]] )`
		Builds a matrix a la →`ARRY`.

180006	^XEQ>ARRAY1	
17C006	^XEQARRY>	`( [] → ob1...obn meta-arry )` Explodes a matrix a la →`ARRY`.
002007	^ArryToMatrix	`( [] → M )` Converts array to symbolic array.

2.7.3 Statistics

2EEDA	STATCLST	`( → )` Clears ΣDAT.
2EEDC	STATN	`( → N )` Internal `NΣ`.
2EEDF	STATSMIN	`( → % )` Internal `MINΣ`.
2EEDD	STATSMAX	`( → % )` Internal `MAXΣ`.

2EEDE	STATMEAN	`( → % )`
		`( → [] )`
		Internal `MEAN`.

2EEE0	STATSTDEV	(→ %) (→ []) Internal SDEV.
2EEE1	STATTOT	(→ %) (→ []) Internal TOT.
2EEE2	STATVAR	(→ %) (→ []) Internal VAR.
3DF92	(ListIntSlp)	(→ {}) List with the two strings "Intercept" and "Slope".

2.8 Unit Objects

2.8.1 Built-in Units

2D781	(SIbasis)	{ 1_kg 1_m... } Returns a list of the 10 base units of the HP49G.
2D837	(unit_kg)	1_kg
2D863	(unit_m)	1_m
2D883	(unit_A)	1_A
2D8A3	(unit_s)	1_s
2D8C3	(unit_K)	1_K
2D8E3	(unit_cd)	1_cd
2D905	(unit_mol)	1_mol
2D7A9	(unit_r)	1_r
2D7C9	(unit_sr)	1_sr
2D929	(unit_?)	1_?
2D7F5	(unit_R)	1_\^oR

2.8.2 Creating Units

2D74F	um*	* marker
2D759	um/	/ marker
2D763	um^	^ marker
2D76D	umP	Char prefix operator
2D777	umEND	Unit end operator
05481	EXTN	(ob1..obn #n → u) Builds a unit object.

2.8.3 General Functions

2F099	U>NCQ	(u → n%% cf%% qhxs)
		Returns the number, conversion factor to base units and a vector in the form:
		`[ kg m A s K cd mol r sr ? ]`
		where each element represents the exponent of that unit. For example, 1_N `U>NCQ` would return:
		`%%1 %%1 [ 1 1 0 -2 0 0 0 0 0 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 → u1') Change units of unit1 to units of unit2.
2F090	UMSI	(u → u') Equivalent to user word `UBASE`.
2F095	UMU>	(u → % u') Returns number and normalized part of unit.
2F019	UNIT>$	(u → $) Converts unit to string.
3900B	(UMFACT)	(u1 u2 → u) Equivalent to user word UFACT.
2F07B	U>nbr	(u → %) Returns number part of unit.
2F098	Unbr>U	(u % → u') Replaces number part of unit.
2F09A	TempConv	??? Used by `UMCONV` for the conversion of temperature units.
25EE4	KeepUnit	(% ob ob' → % ob) (% ob u → u' ob) If the level one object is a unit object, replaces the numeric part of it with the number on level 3. If not, just `DROP`.

2.8.4 Arithmetic Functions

2F081	UM+	(u u' → u'')
2F082	UM-	(u u' → u'')
2F080	UM*	(u u' → u'')
2F083	UM/	(u u' → u'')
2F097	UM^	(u % → u')
2F07D	UM%	(u %percent → u')

3B2A6	(SWAPUM%)	(%percent u → u')
2F07E	UM%CH	(u u' → %)
2F07F	UM%T	(u u' → %)
2F08F	UMMIN	(u u' → u?)
2F08E	UMMAX	(u u' → u?)
2F096	UMXROOT	(u u' → u'')
3A2FA	(SWAPUMXROOT)	(u u' → u'') DOes `SWAP` then <REF>UMXROOT.
2F08A	UMABS	(u → u')
2F08B	UMCHS	(u → u')
2F092	UMSQ	(u → u')
2F093	UMSQRT	(u → u')
2D949	UMSIGN	(u → %)
2D95D	UMIP	(u → u')
2D971	UMFP	(u → u')
2D985	UMFLOOR	(u → u')
2D999	UMCEIL	(u → u')
2D9CB	UMRND	(u → u')
2D9EE	UMTRC	(u → u')
2F08D	UMCOS	(u → u')
2F091	UMSIN	(u → u')
2F094	UMTAN	(u → u')

2.8.5 Tests

2F087	UM=?	(u u' → %flag)
2F07C	UM#?	(u u' → %flag)
2F086	UM<?	(u u' → %flag)
2F089	UM>?	(u u' → %flag)
2F085	UM<=?	(u u' → %flag)
2F088	UM>=?	(u u' → %flag)
2F076	puretemp?	([] []' → [] []' flag) Checks of the two arrays both denote pure temperature units, i.e. if both arrays are equal to `[0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]`

2.9 Composites

2.9.1 General Operations

0521F	&COMP	`( comp comp' → comp'' )` Concatenates two composites.
052FA	>TCOMP	`( comp ob → comp+ob )` Adds ob to tail (end) of composite.
08E33	(#>TCOMP+1)	`( comp # → comp&# #+1 )` Adds bint to tail of composite and increases bint by one
052C6	>HCOMP	`( comp ob → ob+comp )` Adds ob to head (beginning) of composite.
2949D	(!>HCOMP)	`( comp ob → ob+comp )` Tries do to `>HCOMP` in place???
294CF	(!>HCOMPcopy)	`( comp ob → ob+comp )` Calls <REF>!PTR>HCOMP if <REF>INHARDROM?, otherwise does `>TOPTEMP` on both args and then <REF>!?HCOMP.
29501	(!&HCOMP)	`( comp ob → ob+comp )` `>HCOMP` in place?
295BA	(!PTR>HCOMP)	`( comp PTR → PTR+comp )` Can be used if PTR is in HARDROM.
39C8B	(SWAP>HCOMP)	`( ob comp → ob+comp )` Does `SWAP` then `>HCOMP`.

05089	CARCOMP	`( comp → ob_head )`
		`( comp_null → comp_null )`
		Returns first object of the composite, or a null composite if the argument is a null composite.

361C6	?CARCOMP	`( comp T → ob )`
		`( comp F → comp )`
		If the flag is `TRUE`, does `CARCOMP`.

05153	CDRCOMP	`( comp → comp-ob_head )`
		`( comp_null → comp_null )`
		Returns the composite minus its first object, or a null composite if the argument is a null composite.

2825E	(TWONTHCOMPDROP)	`( comp → ob2 )` Gets the second element of composite.
2BC006	^LASTCOMP	`( comp → ob )` Gets the last element of composite. Does <REF>DUPLENCOMP then <REF>NTHCOMPDROP.
0567B	LENCOMP	`( comp → #n )` Returns length of composite (number of objects).
3627A	DUPLENCOMP	`( comp → comp #n )` Does `DUP` then <REF>LENCOMP.

055B7	NULLCOMP?	(comp → flag) If the composite is empty, returns TRUE.
36266	DUPNULLCOMP?	(comp → comp flag) Does DUP then <REF>NULLCOMP?.
056B6	NTHELCOMP	(comp #i → ob T) (comp #i → F) Returns specified element of composite and TRUE, or just FALSE if it could not be found.
35BC3	NTHCOMPDROP	(comp #i → ob) Does <REF>NTHELCOMP then DROP.
35D58	NTHCOMDDUP	(comp #i → ob ob) Does <REF>NTHCOMPDROP then DUP.
376EE	POSCOMP	(comp ob pred → #i) (comp ob pred → #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 → #pos) (comp ob → #0) POSCOMP with EQUAL as test.
37784	NTHOF	(ob comp → #i) (ob comp → #0) Does SWAP then <REF>EQUALPOSCOMP.
0FD006	^ListPos	(ob {} → #i / #0) Equivalent to NTHOF, but faster. However, it only works for lists.
37752	#=POSCOMP	(comp # → #i) (comp # → #0) POSCOMP with #= as test.
05821	SUBCOMP	(comp #m #n → comp') Returns a sub-composite. Makes all index checks first.
376B7	matchob?	(ob comp → T) (ob comp → ob F) Returns TRUE if ob is EQUAL to any element of the composite.
371B3	Embedded?	(ob1 ob2 → flag) Returns TRUE if ob2 is embedded in, or is the same as, ob1. Otherwise returns FALSE.

37798	`Find1stTrue`	(`comp test` → `ob T`) (`comp test` → `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 program below returns `%-4` and `TRUE`. `:: { %1 %2 %2 %-4 %-5 %6 } ' %0< Find1stTrue ;`
25F2C	`Find1stT.1`	Recursive internal function for `Find1stTrue`.
377C5	`Lookup`	(`ob test comp` → `nextob T`) (`ob test comp` → `ob F`) Tests every odd element (1,3,...) in the composite. If a test returns `TRUE`, the object after the tested one is returned, along with `TRUE`. If no object tests `TRUE`, `FALSE` is returned. For example, the program below returns `%6` and `TRUE`. `:: %0 ' %< { %1 %2 %3 %-4 %-5 %6 } Lookup ;`
377DE	`Lookup.1`	(`ob test` → `nextob T`) (`ob test` → `ob F`) `Return Stack:` (`comp` →) `Lookup` with the composite already pushed (with `>R`) onto the runstream. Called by `Lookup`.
37829	`EQLookup`	(`ob comp` → `nextob T`) (`ob comp` → `ob F`) `Lookup` with `EQ` as test.
37B54	`NEXTCOMPOB`	(`comp #ofs` → `comp #ofs' ob T`) (`comp #ofs` → `comp F`) Returns object at specified nibble offset from start. If the object is `SEMI` (i.e., the end of the composite has been reached) returns `FALSE`. To get the first element, use `FIVE` as offset value (to skip the prolog). `ZERO` works as well.

2.9.2 Building

05331	`(COMPN)`	(`obn..ob1 #n #prolog` → `comp`)
05459	`{}N`	(`obn..ob1 #n` → `{ obn..ob1 }`)
05445	`::N`	(`ob1..obn #n` → `:: ob1..obn ;`)
0546D	`SYMBN`	(`ob1..obn #n` → `sym`)

36F8D	top&Cr	(meta1 meta2 → symb) Does top& then <REF>SYMBN: .
286F6	(ONESYMBN)	(ob1..obn #n → symb)
05481	EXTN	(ob1..obn #n → u) Builds a unit object.
293F8	P{}N	(ob1..obn #n → {}) Build list with possible garbage collection.
2942F	(P::N)	(ob1..obn #n → seco) Build seco with possible garbage collection.
293C1	(PSYMBN)	(ob1..obn #n → sym) Build symb with possible garbage collection.

2.9.3 Exploding

054AF	INNERCOMP	(comp → obn..ob1 #n)
3622A	DUPINCOMP	(comp → comp obn..ob1 #n)
3623E	SWAPINCOMP	(comp obj → obj obn..ob1 #n)
35BAF	INCOMPDROP	(comp → obn..ob1)
35C68	INNERDUP	(comp → obn..ob1 #n #n)
2F0EC	ICMPDRPRTDRP	(comp → obn...ob4 ob2 ob1) Does <REF>INCOMPDROP then ROTDROP.
3BADA	(XEQLIST>)	(comp → obn..ob1 %n)
366E9	INNER#1=	(comp → obn..ob1 flag)
157006	^SYMBINCOMP	(symb → ob1 .. obN #n) (ob → ob #1) ({} → {} #1) Explodes symbolic object into meta. Other objects are converted into one-object metas by pushing #1 into the stack.
12A006	^2SYMBINCOMP	(ob1 ob2 → meta1 meta2) Does ^SYMBINCOMP for 2 objects.
158006	^CKINNERCOMP	({} → ob1 .. obN #n) (ob → ob #1) Explodes a list into a meta object. Other objects are converted into one-object metas by pushing #1 into the stack.
297EF	(INNERtop&)	(obn..ob1 #n comp → obm..ob1 #m) Explodes composite and adds to meta: INNERCOMP top& Adds composite objects to meta object.

2.9.4 Lists

055E9	NULL{}	(→ {}) Pushes a null list to the stack.

36ABD	DUPNULL{}?	({} → {} flag)
159006	^DUPCKLEN{}	({} → {} #n) (ob → ob #1) Return length of list, or 1 for non-lists.
29D18	ONE{}N	(ob → { ob })
36202	TWO{}N	(ob1 ob2 → { ob1 ob2 })
36216	THREE{}N	(ob1 ob2 ob3 → { ob1 ob2 ob3 })
361EE	#1-{}N	(ob1..obn #n+1 → {})
2B42A	PUTLIST	(ob #i {} → {}’) Replaces object at specified position. Assumes valid #i.
2FC006	^INSERT{}N	({} ob # → {}’) Insert object into list at given position. The position must be < than length of the list. If the position is zero, >TCOMP is used.
2FB006	^NEXTPext	(list → list1 list2) Extract in list2 all occurrances of the 1st object of list, the remaining objects are stored in list1. list1 = list-list2.
2FD006	^COMPRIMext	({} → {}’) Suppress multiple occurrances in the list.
15A006	^CKCARCOMP	({} → ob1) (ob → ob) Returns first element for lists, or object itself if it is not a list.
2EF5A	apndvarlst	({} ob → {}’) Appends ob to list if not already there.
0FE006	^AppendList	({} ob → {}’) Equivalent to apndvarlst, but faster.
4EB006	^prepvarlist	({} ob → {}’) Adds ob at the beginning of the list if not present. If ob is in list, move ob to the beginning of list. Unfortunately moving an entry does influence the sequence of the rest of the list unchanged.
100006	^SortList	(L pred → 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 example, the predicate would be a > test.
28A006	^PIext	({} → ob) Returns the product of all elements of the list.
25ED3	EqList?	(ob →) Is ob a list of equations? Returns T if ob is a list of at least two elements, and the second element is not a list itself.

2.9.5 Secondaries

055FD	`NULL::`	(→ `:: ;`) Returns null secondary.
37073	`Ob>Seco`	(ob → `::` ob `;`) Does `ONE` then <REF>::N.
3705A	`?Ob>Seco`	(ob → `::` ob `;`) If the object is not a secondary, does `Ob>Seco`.
37087	`2Ob>Seco`	(ob1 ob2 → `::` ob1 ob2 `;`) Does `TWO` then <REF>::N.
3631A	`::NEVAL`	(ob1..obn #n → ?) Does <REF>::N then <REF>EVAL.

2.10 Meta Objects

2.10.1 Stack Functions

29A35	`(dup)`	(meta → meta meta)
0326E	`NDROP`	(1..n #n →)
37032	`DROPNDROP`	(1..n #n ob →)
35FB0	`#1+NDROP`	(ob 1..n #n →) aka: `N+1DROP`
28211	`NDROPFALSE`	(ob1..obn #n → F)
391006	`^NDROPZERO`	(obn..ob1 #n → #0) Replace Meta object with empty Meta object. Should be called dropZERO.
29A5D	`psh`	(meta1 meta2 → meta2 meta1) Should be called `swap`.
29A8F	`roll2ND`	(meta1 meta2 meta3 → meta2 meta3 meta1) Should be called `rot`.
29B12	`unroll2ND`	(meta1 meta2 meta3 → meta3 meta1 meta2) Should be called `unrot`.
3695A	`SWAPUnNDROP`	(meta1 meta2 → meta2) Should be called `swapdrop`.
36946	`SWAPUnDROP`	(meta1 meta2 → meta2 ob1..obn) Swaps two metas and drops the count. Should be called `swapDROP`.
36FA6	`metaROTDUP`	(meta1 meta2 meta3 → meta2 meta3 meta1 meta1) Should be called `rotdup`.

2.10.2 Combining Functions

296A7	top&	(meta1 meta2 → meta1&meta2)
2973B	pshtop&	(meta1 meta2 → meta2&meta1)
29722	(top&top&)	(meta1 meta2 meta3 → meta1&meta2&meta3)
36FBA	ROTUntop&	(meta1 meta2 meta3 → meta2 meta3&meta1)
36FCE	roll2top&	(meta1 meta2 meta3 → meta3 meta1&meta2) aka: rolltwotop&
2963E	psh&	(meta1 meta2 meta3 → meta1&meta3 meta2)

2.10.3 Meta and Object Operations

3592B	SWAP#1+	(meta ob → meta&ob) aka: SWP1+
34431	DUP#1+PICK	(n..1 #n → n..1 #n n)
2979A	('R'RROT2+)	(meta → meta&nob&nob1) Takes nob and nob1 from run stream and adds them to the meta.
34504	get1	(ob meta → meta ob)
36147	OVER#2+UNROL	(meta ob → ob meta)
29693	psh1top&	(meta ob → ob&meta)
28071	pull	(meta&ob → meta ob) aka: #1-SWAP
28085	pullrev	(ob&meta → meta ob)
29137	(pulldroppull)	(meta&ob1&ob2 → meta ob1)
2899D	(2pull2DROP)	(meta&ob1&ob2 → meta)
29821	psh1&	(meta1 meta2 ob → ob&meta1 meta2)
298C0	psh1&rev	(meta1 meta2 ob → ob&meta1 meta2)
2F193	UobROT	(ob meta1 meta2 → meta1 meta2 ob)
29754	pullpsh1&	(meta1 meta2&ob → ob&meta1 meta2)
406006	^addt0meta	(meta1&ob meta2 → meta1 meta2) Removes the last object of meta1.
29972	pshzer	(meta → #0 meta)
2F38E	xnsgeneral	(meta → LAM3&meta&LAM1) Uses contents of LAM1 and LAM3.
2F38F	xsngeneral	(meta → meta&LAM3&LAM1) Uses contents of LAM1 and LAM3.

2.10.4 Other Operations

3760D	SubMetaOb	(meta #start #end → meta') Gets a sub-meta. Does range checks.

37685	SubMetaOb1	`( ob1..obi..obn #n #i #n #i → ob1..obi #n #i )`
		This function can be used to take the first i objects of a meta, if you follow it with `SWAPDROP`. Example:
		`:: %1 %2 %3 %4 %5 BINT5`
		`   BINT3 BINT5 BINT3`
		`   SubMetaOb1 ;`
		results in:
		`%1 %2 %3 #5 #3`

33F006	^submeta	`( meta #begin #end → meta' )` Extracts submeta from a meta.

2F356	metatail	`( ob1..obn-i..obn #i #n+1 → ob1..ob..obn-i #n-i obn-i+1..obn #i )`
		#n is the count of the objects in meta. Takes the last #i elements of meta and creates a new one. Example:
		`:: %1 %2 %3 %4 %5`
		`   BINT2 BINT6 metatail ;`
		Results:
		`%1 %2 %3 #3 %4 %5 #2`

385006	^metasplit	`( meta #i → 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 → meta2 meta1 flag )` Test equality of 2 metas.
3BF006	^EQUALPOSMETA	`( Meta ob → 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 → Meta2 Meta1 ob #pos )` Returns last occurrence of ob in Meta1 or in Meta2. #pos is >0 if in meta2, is <0 if in meta1 (#pos=MINUSONE-#).
198006	^METAINT?	`( Meta → Meta flag )` Tests if Meta is an integer.
199006	^METAPOSINT?	`( Meta → Meta flag )` Tests if Meta is a positive integer smaller than Zsmall.

2.11 Symbolics

2.11.1 General Operations

0546D	SYMBN	`( ob1..obn #n → sym )`

2BD8C	(Cr)	`ob1..obn #n -> symb` Does `'R`, `SWAP#1+` then <REF>SYMBN . Creates a symbolic from the meta in the stack and the next object in the runstream. This object is added to the end of the symbolic.
055F3	(NULLSYMB)	(→ `sym`) Puts a null algebraic in the stack.
286E7	symcomp	(`ob` → `ob'`) If ob is symbolic, does nothing, otherwise `ONE SYMBN`.
2F073	SWAPcompSWAP	(`ob ob'` → `ob'' ob'`) Does `SWAP symcomp SWAP`.
28ACE	(DROP?symcomp)	(`%/C%/Z/id/lam ob'` → `%/C%/Z/id/lam`) (`ob ob'` → `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` → `%/C%/Z/id/lam`) (`ob #1` → `symb`) (`ob #` → `symb`) If # is `BINT1`, calls `DROP?symcomp`. If it is any other number, calls `SYMBN`.
2F25E	(SPLITEQ)	(`sym` → `arg1 arg2`) Internal version of EQ→.
2F242	(EXPR>)	(`sym` → `arg1..argn %n ob`) Internal version of OBJ→.
25EA2	CRUNCH	(`ob` → `%`) Internal version of →`NUM`.
2F110	(FINDVAR)	(`sym` → `{}`) Returns a list of the variables of the equation, recursing into programs and functions in the equation.
462006	^EQUATION?	(`ob` → `ob flag`) Returns `TRUE` if ob is a symbolic finishing by `x=`.
463006	^USERFCN?	(`ob` → `ob flag`) Returns `TRUE` if ob is a symbolic finishing by `xFCNAPPLY`.

29CB9	uncrunch	(→)
		Clears numeric results flag (system flag 3) for the next command only. Example:
		`SYMCOLCT = :: uncrunch colct ;`
		`--`
		Flags: -3

`2BCA2`	`cknumdsptch1`	(`sym` → `symf`) Used by one argument functions to evaluate a symbolic or numeric routine according to numeric results flag. Usage: `:: cknumdsptch1 <sym> <num> ;` If numeric mode, `CRUNCH` is applied to the level one object and `COLA` is applied to <num>. If symbolic mode, `ckseval1:` is called. Example: `:: cknumdsptch1 MetaRE xRE ;` `--` Flags: -3
`2BB21`	`sscknum2`	(`sym sym` → `symf`) Used by two argument functions to evaluate function according to current numeric mode. `Usage: :: sscknum2 <sym> <num> ;` In numeric mode both arguments are CRUNCHed and <num> is COLAd. Else, `cksseval2:` is called. Example: `SYM+ = :: sncknum2 Meta+ x+ ;`
`2BB3A`	`sncknum2`	(`sym % → symf`) `Usage: :: sncknum2 <sym> <num> ;` In symbolic mode uses `cksneval2:`. Example: `SYM+O = :: sncknum2 Meta+Con x+ ;`
`2BB53`	`nscknum2`	(`% sym → symf`) `Usage: :: nscknum2 <sym> <num> ;` In symbolic mode uses cknseval2:. Example: `O+SYM = :: nscknum2 Con+Meta x+ ;`

2.11.2 Derivatives

`2C07B`	`D/D*`	Derivative of multiplication.
`2C086`	`D/D+`	Derivative of addition.
`2C091`	`D/D-`	Derivative of subtraction.
`2C09C`	`D/D/`	Derivative of division.
`2C10B`	`D/D=`	Derivative of equality.
`2C116`	`D/DABS`	Derivative of ABS.
`2C13A`	`D/DACOS`	Derivative of ACOS.
`2C145`	`D/DACOSH`	Derivative of ACOSH.

2C150	D/DALOG	Derivative of ALOG.
2C2B5	D/DAPPLY	
2C15B	D/DARG	Derivative of ARG.
2C166	D/DASIN	Derivative of ASIN.
2C171	D/DASINH	Derivative of ASINH.
2C17C	D/DATAN	Derivative of ATAN.
2C187	D/DATANH	Derivative of ATANH.
2C192	D/DCHS	Derivative of CHS.
2C1B0	D/DCONJ	Derivative of CONJ.
2C1CE	D/DCOS	Derivative of COS.
2C1D9	D/DCOSH	Derivative of COSH.
2C289	D/DDER	Derivative of derivative.
2C1E4	D/DEXP	Derivative of EXP.
2C21B	D/DIFTE	Derivative of IFTE.
2C29F	D/DINTEGRAL	Derivative of integral.
2C1EF	D/DINV	Derivative of INV.
2C1FA	D/DLN	Derivative of LN.
2C205	D/DLNP1	Derivative of LNP1.
2C210	D/DLOG	Derivative of LOG.
2C226	D/DSIN	Derivative of SIN.
2C231	D/DSINH	Derivative of SINH.
2C23C	D/DSQ	Derivative of SQ.
2C247	D/DSQRT	Derivative of SQRT.
2C2AA	D/DSUM	Derivative of `SUM`.
2C252	D/DTAN	Derivative of TAN.

2C25D	D/DTANH	Derivative of TANH.
2C294	D/DWHERE	
2C268	D/D^	Derivative of power.
2C273	D/D^X	
2C27E	D/D^Y	

2.11.3 Other Functions

2EF26	SYMSHOW	(sym id/lam → symf)
2F2A9	XEQSHOWLS	(sym {} → symf)

2.11.4 Meta Symbolics Functions

29986	pshzerpsharg	(meta → M_last M_rest) Pushes last sub-expression in meta. If meta is a valid expression M_rest will be empty.
3701E	pZpargSWAPUn	(meta → M_rest M_last) <REF>pshzerpsharg then <REF>psh .
36FE2	plDRPpZparg	(meta&ob → M_last M_rest) Drops ob then calls <REF>pshzerpsharg .
3F1006	^DIVMETAOBJ	(o1...on #n ob → {o1/ob...on/ob}) Division of all elements of a meta by ob. Tests if o=1.

2.12 Library and Backup Objects

2.12.1 Port Operations

25EEB	NEXTLIBBAK	(#addr → backup/library #nextaddr) Gets next `library` or backup.

2.12.2 Rompointers

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

35C40	DUPROMPTR@	(ROMPTR → ROMPTR ob T) (ROMPTR → ROMPTR F) Does DUP then ROMPTR@.
02FEF	(ROMSEC)	(ROMPTR → ?) Recalls contents of rompointer and EVAL. Generates "Undefined XLIB Error" if not found.
35A88	?>ROMPTR	(ob → ob') If ROM-WORD? and TYPECOL? then RPL@.
35AAB	?ROMPTR>	(ob → ob') If <REF>TYPEROMP? and content exists <REF>INHARDROM? then return contents.
35BFF	RESOROMP	(→ ob) Recalls contents of next object in the runstream (which must be a rompointer).
07E76	(PTR>ROMPTR)	(ob → ROMPTR T) (ob → F) If the object is a library command, returns its rompointer and TRUE, if not just FALSE.
081FB	(ROMPTRDECOMP)	(ROMPTR → id T) (ROMPTR → F) If the library command exists and has a name, returns that name and TRUE, otherwise FALSE.
07C18	(COMPILEID)	(id → id T) (id → ROMPTR T) (id → F) Searches id in current path, if found returns TRUE. Else searches attached libraries. If nothing was found, return FALSE.
34FCD	ROM-WORD?	(ob → flag)
34FC0	DUPROM-WORD?	(ob → ob flag)

2.12.3 Libraries

07709	TOSRRP	(# →) Attaches library to HOME directory. -- <REF>TEXT:Libraries
076AE	OFFSRRP	(# →) Detaches library from HOME directory. -- <REF>TEXT:Libraries
0778D	(ONSRRP?)	(# → flag) Returns TRUE if library is attached to HOME directory.
2F2A7	XEQSETLIB	(% →) Internal ATTACH.

015002	`(^GETLIBS)`	`( → {} )` Returns a list of all attached libraries in the format `{ { "Title1" #id1 } { "Title2" #id2 } ...}` This is used for the `library` menu, so libraries without titles are skipped.
014002	`(^LIBS)`	`( → {} )` Resturns a list of all attached libraries in the format `{ "Title1" #id1 #port1 "Title2" ... }` This is the internal version of the User word `LIBS`, and it also lists libraries without title. -- <REF>TEXT:Libraries
07638	`SETHASH`	`( #libnum hxs → )`
265DA	`(GetLibExt)`	`( ob1..obn #msg #lib → ob1'..obm' flag )` Call the message handler of `library` #lib. The flag is `TRUE` if the `library` is attached and has a message handler, `FALSE` otherwise. Note that `library` message handlers usually require extra arguments on the stack which may also be modified during the call. The handling of most but not all messages leaves the #msg unchanged on the stack, so most of the time, obm' = #msg. -- <REF>TEXT:Libraries
25F2E	`(ExecGetLibsExtentions_sup)`	`( ob1..obn #msg → ob1'..obm' )` Calls the message handlers of all attached libraries with the specified #msg. Note that `library` message handlers usually require extra arguments on the stack which may also be modified during the call. -- <REF>TEXT:Libraries
08199	`(ROMPARTNAME)`	`( #libnum → id T )` `( #libnum → F )` Returns title of `library` as an ID, and `TRUE`. If `library` is not found, returns just `FALSE`.
081DE	`(LIB>#)`	`( lib → #libnum T )` Returns number of `library`.
08081	`(ROMPART>ADDR)`	`( #libnum → #addr T )` `( #libnum → F )` Recalls `library` addres `+` 10 (prolog and length skipped).
080BF	`(ROMPARTSIZE)`	`( #libnum → #nibbles-10 T )` `( #libnum → F )` Returns size of `library`.

080DA	(NEXTROMPID)	(#libnum → #nextlibnum T) (#libnum → F) If specified library exists, #libnum is returned with TRUE.
08112	(GETHASH)	(#libnum → hxs_table T) (#libnum → F) Gets specified library's hash table.
08130	(GETMSG)	(#libnum → [] T) (#libnum → F) Gets specified library's message table. -- <REF>TEXT:Libraries
0764E	SETMESG	([$] #libnum →) Sets message table of specified library. -- <REF>TEXT:Libraries
0813C	(GETLINK)	(#libnum → hxs_table T) (#libnum → F) Gets specified library's link table.
08157	(GETCONFIG)	(#libnum → ob T) (#libnum → F)
07F86	(ROMPART)	(rrp → {#lib1..#libn} T) (ROMPTR → #libnum) Gets the list of libraries attached to the directory, along with TRUE. If the argument is a rom pointer, returns the library number of this pointer.
2F2C6	(XEQXRCL)	(:%port:%libnum → lib) Puts a pointer to the library with romidid %libnum in port %port onto the stack. The argument is a tagged real. The tag can also be '&' in order to search all ports. The library is not yet in TEMPOB, you need to execute TOTEMP in order the achieve this.

2.12.4 Backup Objects

081D9	BAKNAME	(bak → id T) Returns backup's name
0905F	BAK>OB	(bak → ob) Gets backup object.

3 General SysRPL Entries

3.1 Stack Operations

03188	DUP	(ob → ob ob)
35CE0	DUPDUP	(ob → ob ob ob)
2D5006	^3DUP	(3 2 1 → 3 2 1 3 2 1)
28143	NDUPN	(ob #n → ob..ob #n) (ob #0 → #0)
35FF3	DUPROT	(1 2 → 2 2 1)
3457F	DUPUNROT	(1 2 → 2 1 2) aka: SWAPOVER
36133	DUPROLL	(1..n #n → 1 3..n #n 2)
281FD	(DUPROLLSWAP)	(1..n #n → 1 3..n 2 #n)
3432C	DUP4UNROLL	(1 2 3 → 3 1 2 3)
3611F	DUPPICK	(n..1 #n → n..1 #n n-1)
35D30	DUP3PICK	(1 2 → 1 2 2 1) aka: 2DUPSWAP
34431	DUP#1+PICK	(n..1 #n → n..1 #n n)
29362	(DUP#2+PICK)	(n..1 #n → n..1 #n n+1)
031AC	2DUP	(1 2 → 1 2 1 2)
35D30	2DUPSWAP	(1 2 → 1 2 2 1) aka: DUP3PICK
36CA4	2DUP5ROLL	(1 2 3 → 2 3 2 3 1)
031D9	NDUP	(1..n #n → 1..n 1..n)
03244	DROP	(1 →)
357CE	DROPDUP	(1 2 → 1 1)
37032	DROPNDROP	(1..n #n ob →)
35733	DROPSWAP	(1 2 3 → 2 1)
3574D	DROPSWAPDROP	(1 2 3 → 2) aka: ROT2DROP, XYZ>Y
36007	DROPROT	(1 2 3 4 → 2 3 1)
3606B	DROPOVER	(1 2 3 → 1 2 1)
03258	2DROP	(1 2 →)
341D2	3DROP	(1 2 3 →) aka: XYZ>
341D7	4DROP	(1..4 →) aka: XYZW>
341DC	5DROP	(1..5 →)
341E8	6DROP	(1..6 →)

341F4	7DROP	(1..7 →)
0326E	NDROP	(1..n #n →)
35FB0	#1+NDROP	(ob 1..n #n →) aka: N+1DROP
2F0A1	RESETDEPTH	(ob1..obn obn+1..obx #n → ob1..obn) Drops all but #n levels of the stack.
28335	(KEEP)	(ob1..obn ob1'..obm' #m → ob1'..obm') Drops all stack levels above #m.
0314C	DEPTH	(1..n → 1..n #n)
371F9	UStackDepth	(→ #) The depth of the stack, similar to DEPTH.
28187	reversym	(1..n #n → n..1 #n)
03223	SWAP	(1 2 → 2 1)
3576E	SWAPDUP	(1 2 → 2 1 1)
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
29808	('Rswapop)	(1 2 → nop 2) Replaces level two with the next object in the run-stream.
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 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>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)
35F06	4ROLLSWAP	(1 2 3 4 → 2 3 1 4)
36043	4ROLLROT	(1 2 3 4 → 2 4 1 3) aka: FOURROLLROT
360E3	4ROLLOVER	(1 2 3 4 → 2 3 4 1 4)
34257	5ROLL	(1 2 3 4 5 → 2 3 4 5 1) aka: FIVEROLL
358A7	5ROLLDROP	(1 2 3 4 5 → 2 3 4 5)
34281	6ROLL	(1..6 → 2..6 1) aka: SIXROLL
342EA	7ROLL	(1..7 → 2..7 1) aka: SEVENROLL
342BB	8ROLL	(1..8 → 2..8 1) aka: EIGHTROLL
34318	(9ROLL)	(1..9 → 2..9 1)
03325	ROLL	(1..n #n → 2..n 1)
35FC4	ROLLDROP	(1..n #n → 2..n)
35D80	ROLLSWAP	(1..n #n → 2..n-1 1 n)
344F2	#1+ROLL	(ob 1..n #n → 1..n ob)
34517	#2+ROLL	(a b 1..n #n → b 1..n a)
2D6006	^#3+ROLL	(obn+3...obn...ob1 #n → obn+2...ob1 obn+3)
344DD	#+ROLL	(1..n+m #n #m → 2..n+m 1)
344CB	#-ROLL	(1..n-m #n #m → 2..n-m 1)
3422B	UNROT	(1 2 3 → 3 1 2) aka: 3UNROLL, XYZ>ZXY
35D1C	UNROTDUP	(1 2 3 → 3 1 2 1)
35872	UNROTDROP	(1 2 3 → 3 1) aka: SWAPDROPSWAP, XYZ>ZX
343BD	UNROT2DROP	(1 2 3 → 3) aka: ROTROT2DROP, XYZ>Z
341BA	UNROTSWAP	(1 2 3 → 3 2 1) aka: SWAPROT, XYZ>ZYX
360CF	UNROTOVER	(1 2 3 → 3 1 2 1)
3422B	3UNROLL	(1 2 3 → 3 1 2) aka: UNROT, XYZ>ZXY
34331	4UNROLL	(1 2 3 4 → 4 1 2 3) aka: FOURUNROLL, XYZW>WXYZ
35D44	4UNROLLDUP	(1 2 3 4 → 4 1 2 3 3)
343CF	4UNROLL3DROP	(1 2 3 4 → 4) aka: XYZW>W

36057	4UNROLLROT	(1 2 3 4 → 4 3 2 1)
34357	5UNROLL	(1 2 3 4 5 → 5 1 2 3 4) aka: FIVEUNROLL
3438D	6UNROLL	(1..6 → 6 1..5) aka: SIXUNROLL
35BEB	7UNROLL	(1..7 → 7 1..6)
3615B	8UNROLL	(1..8 → 8 1..7)
28225	(9UNROLL)	(1..9 → 9 1..8)
3616F	10UNROLL	(1..10 → 10 1..9)
0339E	UNROLL	(1..n #n → n 1..n-1)
34552	#1+UNROLL	(ob 1..n #n → n ob 1..n-1)
34564	#2+UNROLL	(a b 1..n #n → n a b 1..n-1)
3453D	#+UNROLL	(1..n+m #n #m → n+m 1..n+m-1)
3452B	#-UNROLL	(1..n-m #n #m → n-m 1..n+m-1)
032C2	OVER	(1 2 → 1 2 1)
35CF4	OVERDUP	(1 2 → 1 2 1 1)
35D6C	OVERSWAP	(1 2 → 1 1 2) aka: OVERUNROT
35D6C	OVERUNROT	(1 2 → 1 1 2) aka: OVERSWAP
36CF4	OVER5PICK	(1 2 3 4 → 1 2 3 4 3 1)
37046	2OVER	(1 2 3 4 → 1 2 3 4 1 2)
34485	3PICK	(1 2 3 → 1 2 3 1)
35F1A	3PICKSWAP	(1 2 3 → 1 2 1 3)
360F7	3PICKOVER	(1 2 3 → 1 2 3 1 3)
36CCC	3PICK3PICK	(1 2 3 → 1 2 3 1 2)
2F1C6	DROP3PICK	(1 2 3 4 → 1 2 3 1)
3448A	4PICK	(1 2 3 4 → 1 2 3 4 1)
35F2E	4PICKSWAP	(1 2 3 4 → 1 2 3 1 4)
36CE0	SWAP4PICK	(1 2 3 4 → 1 2 4 3 1)
3610B	4PICKOVER	(1 2 3 4 → 1 2 3 4 1 4)
3448F	5PICK	(1 2 3 4 5 → 1 2 3 4 5 1)
34494	6PICK	(1..6 → 1..6 1)
34499	7PICK	(1..7 → 1..7 1)
3449E	8PICK	(1..8 → 1..8 1)
344A3	(9PICK)	(1..9 → 1..9 1)
344A8	(10PICK)	(1..10 → 1..10 1)
032E2	PICK	(1..n #n → 1..n 1)
373D0	(UNPICK)	(1..n ob #n → ob 2..n)
37408	(#1+UNPICK)	(1..n ob #n-1 → ob 2..n)

3741A	(#+UNPICK)	(1..n ob #n-#m #m → ob 2..n)
3742B	(#1-UNPICK)	(1..n ob #n+1 → ob 2..n)
34436	#1+PICK	(1..n #n-1 → 1..n 1)
34451	#2+PICK	(1..n #n-2 → 1..n 1)
34465	#3+PICK	(1..n #n-3 → 1..n 1)
34474	#4+PICK	(1..n #n-4 → 1..n 1)
34417	#+PICK	(1..n+m #n #m → 1..n+m 1)
34405	#-PICK	(1..n-m #n #m → 1..n-m 1)

3.2 Temporary Environments

3.2.1 Built-in IDs and LAMs

272FE	NULLID	(→ id) Null (empty) identifier.
27308	(EvalNULLID)	(→) Evaluates the empty identifyer, therefore enters the hidden directory.
27308	NULLID1	(→ id) Null (empty) identifier.
27308	NULLID!	(→) Evaluate empty identifier.
2B3AB	NULLLAM	(→ lam) Puts NULLLAM in the stack.
3EA01	(ID_CST)	ID CST
3EF97	(ID_S)	ID S
2715F	(ID_X)	ID X
27155	'IDX	(→ id) Puts ID X unevaluated on the stack.
272F3	(CUREQ)	ID EQ
27937	(ID_SIGMADAT)	ID ΣDAT
27AE9	('IDPAR)	(→ id) Puts ID PPAR unevaluated on the stack. -- <REF>TEXT:Reserved|PPAR
2799A	(ID_PPAR)	ID PPAR
27B2F	(ID_TPAR)	ID TPAR
27B25	('IDTPAR)	(→ id)
27B11	(ID_VPAR)	ID VPAR
27B07	('IDVPAR)	(→ id)
2799A	(ID_PYR)	ID PYR

2798A	(ID_FV)	ID FV
2797D	(ID_PMT)	ID PMT
27972	(ID_PV)	ID PV
27963	(ID_I%YR)	IT I%TR
2795A	(ID_N)	ID N
27946	(ID_SIGMAPAR)	ID ΣPAR
271D8	(ID_STARTERR)	ID STARTERR
271D3	(IDSTARTERR)	{ ID STARTERR }
271B9	(ID_STARTUP)	ID STARTUP
271B1	(ListSTARTUP)	{ ID STARTUP }
271A3	(IDIOPAR)	ID IOPAR

3.2.2 Conversion

05B15	$>ID	($ → ID)
362DE	DUP$>ID	($ → $ ID)
05AED	(ID>LAM)	(id → lam)
05B01	(LAM>ID)	(lam → id)

3.2.3 Temporary Environments Words

074D0	BIND	(obn..ob1 {lamn..lam1} →) Binds n objects to n differently named lams.
074E4	DOBIND	(obn..ob1 lamn..lam1 #n →) Binds n objects to n differently named lams.
36518	1LAMBIND	(ob →) Binds one object to a null named lam.
36513	DUP1LAMBIND	(ob → ob) Does DUP then <REF>1LAMBIND.
155006	^2LAMBIND	(ob1 ob2 →) Binds two objects to null named lams.
156006	^3LAMBIND	(ob1 ob2 ob3 →) Binds three objects to null named lams.
0DE0B0	~nNullBind	(obn..ob1 #n →) Binds #n objects to null named lams. 1LAM has the count, 2LAM the first object. Decompiles to :: ' NULLLAM CACHE ;
36A77	dvarlsBIND	(ob →) Binds ob to LAM 'dvar.
07497	ABND	(→) Abandons topmost temporary environment.

2A7CF	(ABNDTrue)	(→ T) Does <REF>ABND then TRUE.
2A7E3	(ABNDFalse)	(→ F) Does FALSE then <REF>ABND .
34D00	CACHE	(obn..ob1 #n lam →) Binds all objects under the same name. 1LAM has the count.
34EBE	DUMP	(NULLLAM → ob1..obn #n) Inverse of CACHE. Always does garbage collection.
34D58	SAVESTACK	(→) Caches stack to SAVELAM.
2EF72	CacheStack	(→) Caches the stack using SAVESTACK if UNDO is on and Suspend is OK. If there was a previous environment caching the stack, it is abandoned first.
34FA6	undo	(→) Dumps SAVELAM.
07943	@LAM	(lam → ob T) (lam → F) Tries recalling object from lam. If successful, returns object and TRUE, otherwise returns just FALSE.
07D1B	STOLAM	(ob lam →) Tries storing object in lam. Generates "Undefined Local Name" error if lam is not found.
02FD6	(DoLam)	(lam → ob) (lam → !error!) Tries recalling object from lam, generates "Undefined Local Name" error if not found.
078E9	(FIRST@LAM)	(lam → ob T) (lam → F) @LAM for first environment only.
078F5	(NTH@LAM)	(lam #n → ob T) (lam #n → F) @LAM for nth environment only.
075A5	GETLAM	(#n → ob) Gets contents of nth topmost lam.
34616	1GETLAM	(→ ob)
34620	2GETLAM	(→ ob)
3462A	3GETLAM	(→ ob)
34634	4GETLAM	(→ ob)
3463E	5GETLAM	(→ ob)
34648	6GETLAM	(→ ob)
34652	7GETLAM	(→ ob)
3465C	8GETLAM	(→ ob)
34666	9GETLAM	(→ ob)

34670	10GETLAM	(→ ob)
3467A	11GETLAM	(→ ob)
34684	12GETLAM	(→ ob)
3468E	13GETLAM	(→ ob)
34698	14GETLAM	(→ ob)
346A2	15GETLAM	(→ ob)
346AC	16GETLAM	(→ ob)
346B6	17GETLAM	(→ ob)
346C0	18GETLAM	(→ ob)
346CA	19GETLAM	(→ ob)
346D4	20GETLAM	(→ ob)
346DE	21GETLAM	(→ ob)
346E8	22GETLAM	(→ ob)
346F2	(23GETLAM)	(→ ob)
346FC	(24GETLAM)	(→ ob)
34706	(25GETLAM)	(→ ob)
34710	(26GETLAM)	(→ ob)
3471A	(27GETLAM)	(→ ob)
075E9	PUTLAM	(ob #n →) Stores new contents to nth topmost lam.
34611	1PUTLAM	(ob →)
3461B	2PUTLAM	(ob →)
34625	3PUTLAM	(ob →)
3462F	4PUTLAM	(ob →)
34639	5PUTLAM	(ob →)
34643	6PUTLAM	(ob →)
3464D	7PUTLAM	(ob →)
34657	8PUTLAM	(ob →)
34661	9PUTLAM	(ob →)
3466B	10PUTLAM	(ob →)
34675	11PUTLAM	(ob →)
3467F	12PUTLAM	(ob →)
34689	13PUTLAM	(ob →)
34693	14PUTLAM	(ob →)
3469D	15PUTLAM	(ob →)
346A7	16PUTLAM	(ob →)
346B1	17PUTLAM	(ob →)
346BB	18PUTLAM	(ob →)

346C5	19PUTLAM	(ob →)
346CF	20PUTLAM	(ob →)
346D9	21PUTLAM	(ob →)
346E3	22PUTLAM	(ob →)
346ED	(23PUTLAM)	(ob →)
346F7	(24PUTLAM)	(ob →)
34701	(25PUTLAM)	(ob →)
3470B	(26PUTLAM)	(ob →)
34715	(27PUTLAM)	(ob →)
3471F	(DUP1PUTLAM)	(ob → ob)
34729	(DUP2PUTLAM)	(ob → ob)
34797	DUP4PUTLAM	(ob → ob) Does DUP then <REF>4PUTLAM .
34724	(1GETLAMSWAP)	(ob → ob' ob) Does <REF>1GETLAM then SWAP.
3472E	(2GETLAMSWAP)	(ob → ob' ob) Does <REF>2GETLAM then SWAP.
364FF	1GETABND	(→ 1lamob) Does <REF>1GETLAM then <REF>ABND .
35DEE	1ABNDSWAP	(ob → 1lamob ob) Does <REF>1GETABND then SWAP.
35F42	1GETSWAP	(ob → 1lamob ob) Does <REF>1GETLAM then SWAP.
2F318	1GETLAMSWP1+	(# → 1lamob #+1) Does <REF>1GETLAM then SWAP#1+.
3632E	2GETEVAL	(→ ?) Does <REF>2GETLAM then <REF>EVAL .
3483E	GETLAMPAIR	(#n → #n ob lam F) (#n → #n T) Gets lam contents and name (10 = 1lam, 20 = 2lam, etc.)
347AB	DUPTEMPENV	(→) Duplicates topmost tempenv (clears protection word).
2B3A6	1NULLLAM{}	(→ {}) Puts a list with one NULLLAM in the stack.
271F4	(2NULLLAM{})	(→ {}) Puts a list with two times NULLLAM in the stack.
27208	(3NULLLAM{})	(→ {}) Puts a list with three times NULLLAM in the stack.
2B3B7	4NULLLAM{}	(→ {}) Puts a list with four times NULLLAM in the stack.
27AB7	(8NULLLAM{})	(→ {}) Puts a list with eight times NULLLAM in the stack.

3.3 Error Handling

3.3.1 General Words

26067	ERRBEEP	(→) Beeps.
04CE6	ERROR@	(→ #) Returns current error number.
04D0E	ERRORSTO	(# →) Stores new error number.
36883	ERROROUT	(# →) Stores new error number and calls ERRJMP.
04D33	ERRORCLR	(→) Stores zero as new error number.
04ED1	ERRJMP	(→) Invokes error handling sub-system.
04E07	GETEXITMSG	(→ $) Gets EXITMSG (user defined error message).
04E37	EXITMSGSTO	($ →) Stores $ as EXITMSG.
25EAE	DO#EXIT	(# →) Stores new error number, does <REF>AtUserStack and then <REF>ERRJMP.
25EB0	DO%EXIT	(% →) Same as above, but takes real number as argument.
25EAF	DO$EXIT	($ →) Stores string as EXITMSG, #70000 as error number, does <REF>AtUserStack and then <REF>ERRJMP .
04EA4	ABORT	(→) Does <REF>ERRORCLR and <REF>ERRJMP .
04E5E	ERRSET	(→) Sets new error trap.
04EB8	ERRTRAP	(→) Error trap marker. If no error happens, still removes all temporary environments created since ERRSET.
04D87	JstGetTHEMESG	(# → $) Fetches message from message table. To get a message from a library, use the formula: libnum*#100+msgnum. -- <REF>TEXT:Libraries aka: JstGETTHEMSG

04D64	GETTHEMESG	(# → $) If #70000 then does <REF>GETEXITMSG, else does <REF>JstGetTHEMESG . -- <REF>TEXT:Libraries
39332	(?GetMsg)	(# → $msg) (ob → ob) If the argument is a bint, does JstGETTHEMSG to fetch a message. Other arguments are returned unchanged. -- <REF>TEXT:Libraries
04DD7	(SPLITmsg)	(#msg → #error #libnum) Splits message number into error and library numbers. -- <REF>TEXT:Libraries

3.3.2 Error Generating Words

04FB6	SETMEMERR	Error 001h Generates "Insufficient Memory" error.
04FC2	(SETDIRRECUR)	Error 002h Generates "Directory Recursion" error.
04FCE	(SETLAMERR)	Error 003h Generates "Undefined Local Name" error.
05016	SETROMPERR	Error 004h Generates "Undefined XLIB Name" error.
04FAA	(SETLBERR)	Error 006h Generates "Power Lost" error.
04FDA	(SETCORPORT)	Error 008h Generates "Invalid Card Data" error.
04FE6	(SETOBINUSE)	Error 009h Generates "Object In Use" error.
04FF2	SETPORTNOTAV	Error 00Ah Generates "Port Not Available" error.
04FFE	(SETNOROOM)	Error 00Bh Generates "No Room In Port" error.
0500A	(SETXNONEXT)	Error 00Ch Generates "Object Not In Port" error.
26508	(NOEQERR)	Error 104h Generates "No Current Equation" 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.
26521	(SETUNDOERR)	Error 124h Generates "LAST STACK Disabled" error.
262E2	SETSTACKERR	Error 201h Generates "Too Few Arguments" error.
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	(# →) Calls ERRJMP if the error number is any of {13E 123 DFF}.
37226	(ListErrspecial)	(→ {}) List of error numbers handled specially by Sig?ErrJmp. This is simply { #13E #123 #DFF }
25F10	ederr	(→) Error handler for applications which use savefmt1 to save the current display format. Calls <REF>rstfmt1 and then errors out.

3.4 Conditionals

3.4.1 Boolean Flags

2602B	COERCEFLAG	(T → %1) (F → %0) Converts system flag to user flag, drops current stream.
301BA	%0<>	(% → flag) Can be used to change a user flag into a system flag.
03A81	TRUE	(→ T)
27E87	TrueTrue	(→ T T)
36540	TrueFalse	(→ T F) aka: TRUEFALSE
09378	(TRUESWAP)	(ob → T ob)
03AC0	FALSE	(→ F)

36554	`FalseTrue`	(→ F T) aka: `FALSETRUE`
283E8	`FalseFalse`	(→ F F)
27E9B	`failed`	(→ F T)
35280	`DROPTRUE`	(ob → T)
2D7006	`^2DROPTRUE`	(ob ob' → T)
28DAB	`(3DROPTRUE)`	(ob1 ob2 ob3 → T)
35289	`DROPFALSE`	(ob → F)
35B32	`2DROPFALSE`	(ob1 ob2 → F)
28D38	`(4DROPFALSE)`	(ob1..ob4 → F)
28E05	`(5DROPFALSE)`	(ob1..ob5 → F)
28211	`NDROPFALSE`	(ob1..obn #n → F)
2812F	`SWAPTRUE`	(ob1 ob2 → ob2 ob1 T)
374AA	`(SWAPFALSE)`	(ob1 ob2 → ob2 ob1 F)
374BE	`SWAPDROPTRUE`	(ob1 ob2 → ob2 T)
28239	`(SWAPDROPFALSE)`	(ob1 ob2 → ob2 F)
35EF2	`XYZ>ZTRUE`	(ob1 ob2 ob3 → ob3 T)
2962A	`RDROPFALSE`	(→ F) Puts `FALSE` in the stack and drops rest of current stream.
29616	`(RDROPTRUE)`	(→ T) Puts `TRUE` in the stack and drops rest of current stream.
03AF2	`NOT`	(flag → flag') Returns `FALSE` if the input is `TRUE`, and vice-versa.
03B46	`AND`	(flag1 flag2 → flag) Returns `TRUE` if both flags are `TRUE`.
03B75	`OR`	(flag1 flag2 → flag) Returns `TRUE` if either flag is `TRUE`.
03ADA	`XOR`	(flag1 flag2 → flag) Returns `TRUE` if flags are different.
365F9	`ORNOT`	(flag1 flag2 → 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`.

3.4.2 General Tests

03B2E	EQ	(ob1 ob2 → flag) Returns TRUE if both objects are the same, i.e., they occupy the same physical space in memory. Only the addresses of the objects are tested.
36621	2DUPEQ	(ob1 ob2 → ob1 ob2 flag) Does 2DUP then EQ.
3664E	EQOR	(flag ob1 ob2 → flag') Does EQ then OR.
3607F	EQOVER	(ob3 ob1 ob2 → ob3 flag ob3) Does EQ then OVER.
3663A	EQ:	(ob → flag) EQ with the next object in the current stream.
36635	DUPEQ:	(ob → ob flag) Does DUP then EQ:.
03B97	EQUAL	(ob1 ob2 → flag) Returns TRUE if the objects are equal (but not necessarily the same), i.e., their prologs and contents are the same.
3CCB4	(SAME)	(ob1 ob2 → %1/%0) Does EQUAL, then COERCEFLAG. Identical to what <REF>xSAME does.
3660D	EQUALNOT	(ob1 ob2 → flag) Returns TRUE if the objects are different.
36662	EQUALOR	(flag ob1 ob2 → flag') Does EQUAL then OR.
0FF006	^Contains?	(ob1 ob2 → 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. Otherwise, tests if ob1 and ob2 are equal.

3.4.3 True/False Tests

34AA1	?SEMI	(T → :: ;) (F → :: <ob1> <rest> ;)
34A92	NOT?SEMI	(T → :: <ob1> <rest> ;) (F → :: ;)
3692D	?SEMIDROP	(ob T → :: ob ;) (ob F → :: <ob1> <rest> ;)
34BD8	NOT?DROP	(ob T → :: ob <ob1> <rest> ;) (ob F → :: <ob1> <rest> ;)
35F56	?SWAP	(ob1 ob2 T → :: ob2 ob1 <ob1> <rest> ;) (ob1 ob2 F → :: ob1 ob2 <ob1> <rest> ;)
35DDA	?SKIPSWAP	(ob1 ob2 T → :: ob1 ob2 <ob1> <rest> ;) (ob1 ob2 F → :: ob2 ob1 <ob1> <rest> ;)

35F97	?SWAPDROP	(ob1 ob2 T → :: ob1 <ob1> <rest> ;) (ob1 ob2 F → :: ob2 <ob1> <rest> ;)
35F7E	NOT?SWAPDROP	(ob1 ob2 T → :: ob2 <ob1> <rest> ;) (ob1 ob2 F → :: ob1 <ob1> <rest> ;)
070FD	RPIT	(T ob → :: ob <ob1> <rest> ;) (F ob → :: <ob1> <rest> ;) ob is actually executed, and not pushed in the stack.
070C3	RPITE	(T ob1 ob2 → :: ob1 <ob1> <rest> ;) (F ob1 ob2 → ob2 <ob1> <rest> ;) ob1 or ob2 is actually executed, and not pushed in the stack.
34AF4	COLARPITE	(T ob1 ob2 → :: ob1 ;) (F ob1 ob2 → :: ob2 ;) ob1 or ob2 is actually executed, and not pushed in the stack.
34B4F	2'RCOLARPITE	Return to composite and ITE there.
34A22	IT	(T → :: <ob1> <rest> ;) (F → :: <ob2> <rest> ;)
0712A	?SKIP	(T → :: <ob2> <rest> ;) (F → :: <ob1> <rest> ;) aka: NOT_IT
34B3E	ITE	(T → :: <ob1> <ob3> <rest> ;) (F → :: <ob2> <rest> ;)
36865	COLAITE	(T → :: <ob1> ;) (F → :: <ob2> ;)
34ABE	ITE_DROP	(ob T → :: <ob2> <rest> ;) (ob F → :: ob <ob1> <rest> ;)
36EED	ANDITE	(f1 f2 → :: <ob1> <ob3> <rest> ;) (f1 f2 → :: <ob2> <rest> ;)
349F9	case	(T → :: <ob1> ;) (F → :: <ob2> <rest> ;)
34A13	NOTcase	(T → :: <ob2> <rest> ;) (F → :: <ob1> ;)
36D4E	ANDcase	(f1 f2 → :: <ob1> ;) (f1 f2 → :: <ob2> <rest> ;)
36E6B	ANDNOTcase	(f1 f2 → :: <ob1> ;) (f1 f2 → :: <ob2> <rest> ;)
359E3	ORcase	(f1 f2 → :: <ob1> ;) (f1 f2 → :: <ob2> <rest> ;)
3495D	casedrop	(ob T → :: <ob1> ;) (ob F → :: ob <ob2> <rest> ;)
3494E	NOTcasedrop	(ob T → :: ob <ob2> <rest> ;) (ob F → :: <ob1> ;)
34985	case2drop	(ob1 ob2 T → :: <ob1> ;) (ob1 ob2 F → :: ob1 ob2 <ob2> <rest> ;)

34976	NOTcase2drop	(ob1 ob2 T → :: ob1 ob2 <ob2> <rest> ;)
		(ob1 ob2 F → :: <ob1> ;)
349B1	caseDROP	(ob T → :: ;)
		(ob F → :: ob <ob1> <rest> ;)
349C6	NOTcaseDROP	(ob T → :: ob <ob1> <rest> ;)
		(ob F → :: ;)
368FB	casedrptru	(ob T → T)
		(ob F → :: ob <ob1> <rest> ;)
		Note: should be called caseDRPTRU.
365B3	casedrpfls	(ob T → F)
		(ob F → :: ob <ob1> <rest> ;)
		Note: should be called caseDRPFLS.
36B3A	NOTcsdrpfls	(ob T → :: ob <ob1> <rest> ;)
		(ob F → F)
		Note: should be called NOTcaseDRPFLS.
349D6	case2DROP	(ob1 ob2 T → :: ;)
		(ob1 ob2 F → :: ob1 ob2 <ob1> <rest> ;)
349EA	NOTcase2DROP	(ob1 ob2 T → :: ob1 ob2 <ob1> <rest> ;)
		(ob1 ob2 F → :: ;)
365CC	case2drpfls	(ob1 ob2 T → F)
		(ob1 ob2 F → :: ob1 ob2 <ob1> <rest> ;)
		Note: should be called case2DRPFLS.
3652C	caseTRUE	(T → T)
		(F → :: <ob1> <rest> ;)
36914	NOTcaseTRUE	(T → :: <ob1> <rest> ;)
		(F → T)
365E5	caseFALSE	(T → F)
		(F → :: <ob1> <rest> ;)
2B2C5	NOTcaseFALSE	(T → :: <ob1> <rest> ;)
		(F → F)
359AD	COLAcase	(T → :: <ob1> ;)
		(F → :: <ob2> <rest> ;)
		Drops the rest of current stream and executes case in the stream above.
359C8	COLANOTcase	(T → :: <ob2> <rest> ;)
		(F → :: <ob1> ;)
		Drops the rest of current stream and executes NOTcase in the stream above.

3.4.4 Binary Integer Tests

363B5	#=?SKIP	(#m #n → :: <ob2> <rest> ;)
		(#m #n → :: <ob1> <rest> ;)
363E2	#>?SKIP	(#m #n → :: <ob1> <rest> ;)
		(#m #n → :: <ob2> <rest> ;)
35C54	#=ITE	(#m #n → :: <ob1> <ob3> <rest> ;)
		(#m #n → :: <ob2> <rest> ;)

36F29	#<ITE	(#m #n → :: <ob1> <ob3> <rest> ;) (#m #n → :: <ob2> <rest> ;)
36F3D	#>ITE	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> <ob3> <rest> ;)
348D2	#=case	(#m #n → :: <ob1> ;) (#m #n → :: <ob2> <rest> ;)
348E2	OVER#=case	(#m #n → :: #m <ob1> ;) (#m #n → :: #m <ob2> <rest> ;)
34939	#=casedrop	(#m #n → :: <ob1> ;) (#m #n → :: #m <ob2> <rest> ;) Note: should be called OVER#=casedrop.
36590	#=casedrpfls	(#m #n → F) (#m #n → :: #m <ob1> <rest> ;) Note: should be called OVER#=caseDRPFLS.
36D9E	#<>case	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> ;)
36D76	#<case	(#m #n → :: <ob1> ;) (#m #n → :: <ob2> <rest> ;)
36DCB	#>case	(#m #n → :: <ob2> <rest> ;) (#m #n → :: <ob1> ;)
34A7E	#0=?SEMI	(#0 → :: ;) (# → :: <ob1> <rest> ;)
36383	#0=?SKIP	(#0 → :: <ob2> <rest> ;) (# → :: <ob1> <rest> ;)
36F15	#0=ITE	(#0 → :: <ob1> <ob3> <rest> ;) (# → :: <ob2> <rest>)
36ED4	DUP#0=IT	(#0 → :: #0 <ob1> <rest> ;) (# → :: # <ob2> <rest> ;)
36F51	DUP#0=ITE	(#0 → :: #0 <ob1> <ob3> <rest> ;) (# → :: # <ob2> <rest> ;)
348FC	#0=case	(#0 → :: <ob1> ;) (# → :: <ob2> <rest> ;)
348F7	DUP#0=case	(#0 → :: #0 <ob1> ;) (# → :: # <ob2> <rest> ;)
3490E	DUP#0=csedrp	(#0 → :: <ob1> ;) (# → :: # <ob2> <rest> ;)
36D21	DUP#0=csDROP	(#0 → :: ;) (# → :: # <ob1> <rest> ;)
36D8A	#1=case	(#1 → :: <ob1> ;) (# → :: <ob2> <rest> ;)
3639C	#1=?SKIP	(#1 → :: <ob2> <rest> ;) (# → :: <ob1> <rest> ;)
36DB2	#>2case	(#0/#1/#2 → :: <ob2> <rest> ;) (# → :: <ob1> ;)

25E72	?CaseKeyDef	(# #’ → :: ’ ob1 T ;) (# #’ → :: <ob2> <rest> ;) Compares two bints. If equal, quotes the next object from the runsream and returns it along with TRUE.
25E73	?CaseRomptr@	(# #’ → ob T) (# #’ → F) (# #’ → :: <ob2> <rest> ;) Compares two bints. If equal, tries to resolve the rompointer which must be the next object in the runstream. The ROMPTR@ pushes TRUE when successful, so this entry can be used directly for key handlers.

3.4.5 Real and Complex Number Tests

2B149	%0=case	(%0 → :: %0 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
36DDF	j%0=case	(%0 → :: <ob1> ;) (ob → :: <ob2> <rest> ;)
2B15D	C%0=case	(C%0 → :: C%0 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
2B11C	num0=case	(0 → :: 0 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex zero are TRUE conditions for this test.
2B1A3	%1=case	(%1 → :: %1 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
2B1C1	C%1=case	(C%1 → :: C%1 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
2B176	num1=case	(1 → :: 1 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex one are TRUE conditions for this test.
2B20C	%2=case	(%2 → :: %2 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
2B22A	C%2=case	(C%2 → :: C%2 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
2B1DF	num2=case	(2 → :: 2 <ob1> ;) (ob → :: ob <ob2> <rest> ;) Both a real and a complex two are TRUE conditions for this test.
2B289	%-1=case	(%-1 → :: %-1 <ob1> ;) (ob → :: ob <ob2> <rest> ;)
2B2A7	C%-1=case	(C%-1 → :: C%-1 <ob1> ;) (ob → ob <ob2> <rest> ;)

2B25C	`num-1=case`	`( -1 → :: -1 <ob1> ; )`
		`( ob → :: ob <ob2> <rest> ; )`
		Both a real and a complex -1 are `TRUE` conditions for this test.

3.4.6 Meta Object Tests

2AFFB	`MEQ1stcase`	`( meta&ob1 ob2 → ob1=ob2 ? case )` Meta&ob1 ob2 ob1=ob2 ? `case`
2AF37	`AEQ1stcase`	`( meta&ob → ob=nob ? case )` Meta&ob ob=nob ? `case`
2B01B	`MEQopscase`	`( meta1&ob1 meta2&ob2 ob3 → )` Meta1&ob1 Meta2&ob2 ob3
2B06A	`AEQopscase`	`meta1&ob1 meta2&ob2` Meta1&ob1 Meta2&ob2

2B083	`Mid1stcase`	`( meta&ob → ob is id )`
		`lam ? case`
		Meta&ob ob is id or lam ? `case`

2AE32	`M-1stcasechs`	`( Meta&NEG → Meta COLA )`
		`( Meta → Meta SKIP )`
		`( Meta&(%<0) → Meta&ABS(%) COLA )`
		Meta&NEG Meta `COLA` ; Meta Meta `SKIP` Meta&(%<0) Meta&ABS(%) `COLA`

3.4.7 General Object Tests

36EBB	`EQIT`	`( ob1 ob1 → :: <ob1> <rest> ; )`
		`( ob1 ob2 → :: <ob2> <rest> ; )`

36F01	`EQITE`	`( ob1 ob1 → :: <ob1> <ob3> <rest> ; )`
		`( ob1 ob2 → :: <ob2> <rest> ; )`

36D3A	`jEQcase`	`( ob1 ob1 → :: <ob1> ; )`
		`( ob1 ob2 → :: <ob2> <rest> ; )`

34999	`EQcase`	`( ob1 ob1 → :: ob1 <ob1> ; )`
		`( ob1 ob2 → :: ob1 <ob2> <rest> ; )`
		Note: Should be called `OVEREQcase`.

359F7	`REQcase`	`( ob → :: ob <ob2> ; )`
		`( ob → :: ob <ob3> <rest> ; )`
		`EQcase` with the next object in the runstream.

34920	`EQcasedrop`	`( ob1 ob1 → :: <ob1> ; )`
		`( ob1 ob2 → :: ob1 <ob2> <rest> ; )`
		Note: should be called `OVEREQcasedrop`.

35A10	`REQcasedrop`	`( ob → <ob2> ; )`
		`( ob → <ob3> <rest> ; )`
		`EQcasedrop` with the next object in the runstream.

36D62	`EQUALcase`	`( ob1 ob1 → :: <ob1> ; )`
		`( ob1 ob2 → :: <ob2> <rest> ; )`

36E7F	EQUALNOTcase	(ob1 ob1 → :: <ob2> <rest> ;)
		(ob1 ob2 → :: <ob1> ;)
36D08	EQUALcasedrp	(ob ob1 ob2 → :: <ob1> ;)
		(ob ob1 ob2 → :: ob <ob2> <rest> ;)
2AD81	EQUALcasedrop	(ob1 ob2 → :: <ob1> ;)
		(ob1 ob2 → :: ob1 <ob2> <rest> ;)
29E99	tok=casedrop	($ $' → :: <ob1> ;)
		($ $' → :: $ <ob2> <rest> ;)
		Note: should be called OVERtok=casedrop.
2ADBD	nonopcase	(seco → :: seco <ob2> <rest> ;)
		(ob → :: ob <ob1> ;)
2B0CC	idntcase	(id → :: id <ob1> ;)
		(ob → :: ob <ob2> <rest> ;)
36E93	dIDNTNcase	(id → :: id <ob2> <rest> ;)
		(ob → :: ob <ob1> ;)
2B0EF	idntlamcase	(id/lam → :: id <ob1> ;)
		(ob → :: ob <ob2> <rest> ;)
36DF3	REALcase	(% → :: <ob1> ;)
		(ob → :: <ob2> <rest> ;)
3EB9D	(dREALcase)	(% → :: % ob1 ;)
		(ob → :: ob <ob2> <rest> ;)
36EA7	dREALNcase	(% → :: % <ob2> <rest> ;)
		(ob → :: ob <ob1> ;)
36E07	dARRYcase	([] → :: [] <ob1> ;)
		(ob → :: ob <ob2> <rest> ;)
36E43	dLISTcase	({} → :: {} ob1 ;)
		(ob → :: ob <ob2> <rest> ;)
260C6	NOTLISTcase	({} → :: {} <ob2> <rest> ;)
		(ob → :: ob <ob1> ;)
260D0	NOTSECOcase	(seco → :: seco <ob2> <rest> ;)
		(ob → :: ob <ob1> ;)
260CB	NOTROMPcase	(romp → :: romp <ob2> <rest> ;)
		(ob → :: ob <ob1> ;)
2ADE0	numb1stcase	(%/C%/[]/[L] → :: <ob1> ;)
		(ob → :: ob2 <rest> ;)
		If %, C%, [ ] or [L] then <REF>COLA, else <REF>SKIP .
36E2F	(dZINTcase)	(zint → :: zint ob1 ;)
		(ob → :: ob <ob2> <rest> ;)

3.4.8 Miscellaneous

36F65	UserITE	(#set → :: <ob1> <ob3> <rest> ;)
		(#clr → :: <ob2> <rest> ;)
36F79	SysITE	(#set → :: <ob1> <ob3> <rest> ;)
		(#clr → :: <ob2> <rest> ;)

36C4F	caseDoBadKey	(T → :: DoBadKey ;) (F → :: <ob1> <rest> ;) aka: caseDEADKEY
36C36	caseDrpBadKy	(ob T → :: DoBadKey ;) (ob F → :: ob <ob1> <rest> ;)
361B2	caseERRJMP	(T → :: ERRJMP ;) (F → :: <ob> <rest> ;)
36B53	caseSIZEERR	(T → :: SIZEERR ;) (F → :: <ob> <rest> ;)
36B67	NcaseSIZEERR	(T → :: <ob> <rest> ;) (F → :: SIZEERR ;)
36BAA	NcaseTYPEERR	(T → :: <ob1> <rest> ;) (F → :: TYPEERR ;)
25EEE	NoEdit?case	(→ :: <ob1> <rest> ;) (→ :: <rest> ;) Tests if there is no edit line active.
36E57	EditExstCase	(→ :: <ob1> <rest> ;) (→ :: <rest> ;) Tests if there is an edit line active.
2BE36	(AlgebraicModecase)	(→ :: <ob1> ;) (→ :: <ob2> <rest>) Tests for algebraic mode and does case.

3.5 Runstream Control

06E8E	NOP	(→) Does nothing.
39CD5	xNEGNEG	(→) Does nothing, decompiles to :: CK1&Dispatch BINT0 NOP ; There like NOP, but requires an argument.
06EEB	'R	(→ ob) Pushes next object in return stack (i.e., the first object in the composite above this one) to the stack (skipping it). If top return stack is empty (contains SEMI), a null secondary is pushed and the pointer is not advanced.
06F66	'REVAL	(→ ?) Does <REF>'R then <REF>EVAL.
36A27	'R'R	(→ ob1 ob2) Does <REF>'R twice.
34BEF	ticR	(→ ob T) (→ F) Pushes next object in return stack to stack and TRUE, of just FALSE if the top return stack body is empty. In this case, it is dropped.

36A4A	'RRDROP	(→ ob) Does <REF>'R , then <REF>RDROP.
06F9F	>R	(:: →) Pushes :: to top of return stack (skips prolog, i.e., the composite will be executed automatically).
0701F	R>	(→ ::) Creates and pops a secondary from top return stack body to stack.
07012	R@	(→ ::) Like <REF>R>, but the return stack is not popped.
0716B	IDUP	(→) Pushes interpreter pointer into the return stack.
06F8E	EVAL	(ob →) Evaluates object.
262FB	COMPEVAL	(comp →) EVAL just pushes a list back, this one executes it.
34BAB	2@REVAL	(→) EVAL first object in the stream above the previous one.
34BBB	3@REVAL	(→) EVAL first object in the stream above the stream above the previous one.
34A31	GOTO	(→) Jumps to next address in stream. Address is a five-nibble address, not a system binary. Can only be used to jump to the middle of programs, cannot jump to a program prolog.
34A46	?GOTO	(flag →) If TRUE, jumps, else skips five nibbles.
34A59	NOT?GOTO	(flag →) If FALSE jumps, else skips five nibbles.
26111	RDUP	(→) Duplicates top return stack level.
06FB7	RDROP	(→) Pops the return stack.
343E1	2RDROP	(→) Pops two return stack levels.
343F3	3RDROP	(→) Pops three return stack levels.
36342	DROPRDROP	(ob →) Does DROP then <REF>RDROP .
3597F	RDROPCOLA	(→) Does <REF>RDROP then <REF>COLA .
34144	RSWAP	(→) Swap in the return stack.

2644A	(RROLL)	(#n →) Rolls nth return stack level to top of return stack.
368C9	RSKIP	(→) Skips first object in the return stack (i.e., the first object in the composite above this one).
2B8BE	(OBJ>R)	(ob →) 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 elements.
2B8E6	(R>OBJ)	(→ ob) Gets an object from the return stack.
0312B	SEMI	(→) DROP the rest of the current stream.

3.5.1 Quoting Objects

06E97	'	(→ nob (nextob)) Pushes next object in the stream to the stack (skipping it).
38837	xSILENT'	(→ nextob) Put the next ob in the runstream on the stack. Quoter used in UserRPL.
3696E	DUP'	(ob → ob nob) Does DUP then '.
36996	DROP'	(ob → nob) Does DROP then '.
36982	SWAP'	(ob1 ob2 → ob2 ob1 nob) Does SWAP then '.
369AA	OVER'	(ob1 ob2 → ob1 ob2 ob1 nob) Does OVER then '.
369BE	STO'	(ob id/lam → nob) Does STO then '.
369D2	TRUE'	(→ T nob) Pushes TRUE and the next object to the stack.
369FF	FALSE'	(→ F nob) Pushes FALSE and the next object to the stack.
369E6	ONEFALSE'	(→ #1 F nob) Pushes ONE, FALSE and the next object to the stack.
36A13	#1+'	(# → #+1 nob) Does #1+ then '.
36306	'NOP	(→ NOP) Pushes NOP to the stack.
3619E	'ERRJMP	(→ ERRJMP) Pushes ERRJMP to the stack.

2B90B	'DROPFALSE	(→ DROPFALSE) Pushes DROPFALSE to the stack.
25E6A	'DoBadKey	(→ DoBadKey) Pushes DoBadKey to the stack.
25E6B	'DoBadKeyT	(→ DoBadKey T) Pushes <REF>DoBadKey and TRUE to the stack.
2F32E	DROPDEADTRUE	(ob → DoBadKey T) Makes the user drop dead, then pushes TRUE.
36BBE	('x*)	(→ x*) Pushes <REF>x* (User word *) to the stack.
36BD2	'xDER	(→ xDER) Pushes xDER (User word ∂) to the stack.
27B43	'IDFUNCTION	(→ xFUNCTION) Pushes xFUNCTION (User word FUNCTION) to the stack.
27B6B	'IDPOLAR	(→ xPOLAR) Pushes xPOLAR (User word POLAR) to the stack.
27B57	('IDCONIC)	(→ xCONIC) Pushes xCONIC (User word CONIC) to the stack.
27B7F	'IDPARAMETER	(→ xPARAMETRIC) Pushes xPARAMETRIC (user word PARAMETRIC) to the stack.
27B93	('IDTRUTH)	(→ xTRUTH) Pushes xTRUTH (user word TRUTH) to the stack.
27BA7	('IDSCATTER)	(→ xSCATTER) Pushes xSCATTER (user word SCATTER) to the stack.
27BBB	('IDHISTOGRAM)	(→ xHISTOGRAM) Pushes xHISTOGRAM (user word HISTROGRAM) to the stack.
27BCF	('IDBAR)	(→ xBAR) Pushes xBAR (user word BAR) to the stack.
27BE3	('IDFAST3D)	(→ xFAST3D) Pushes xFAST3D (user word FAST3D) to the stack.
29ED0	'Rapndit	(meta ob1...ob4 → meta&ob ob1...ob4) Takes ob from runstream and appends it to the meta starting in level 5.
36AA4	'xDEREQ	(ob → flag) Is ob eq to user command xDER?

3.5.2 Skipping Objects

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.
281E9	(DROPCOLA)	Does DROP then COLA
0714D	SKIP	Skips 1 obj in the runstream.
0715C	(2SKIP)	Skips 2 objs 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.

3.6 Loops

3.6.1 Indefinite Loops

0716B	IDUP	($\rightarrow$) Pushes interpreter pointer into the return stack.
071A2	BEGIN	($\rightarrow$) Pushes interpreter pointer into the return stack.
071AB	AGAIN	($\rightarrow$) 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 $\rightarrow$) If FALSE then <REF>AGAIN, otherwise <REF>RDROP .
3640F	NOT_UNTIL	(flag $\rightarrow$) NOT then <REF>UNTIL .
35B96	#0=UNTIL	(# $\rightarrow$ #) Actually, should be called DUP#0=UNTIL.
071EE	WHILE	(flag $\rightarrow$) If TRUE does nothing, otherwise <REF>RDROP then <REF>2SKIP .
36428	NOT_WHILE	(flag $\rightarrow$) NOT then <REF>WHILE .

36441	DUP#0<>WHILE	(# →) Try to guess what it does.

3.6.2 Definite Loops

073F7	DO	(#stop #start →)
073C3	ZERO_DO	(#stop →)
364C8	DUP#0_DO	(#stop → #stop)
073CE	ONE_DO	(#stop →)
073DB	#1+_ONE_DO	(#stop →)
364E1	toLEN_DO	({} → {}) From ONE to #elements.
07334	LOOP	(→)
073A5	+LOOP	(# →) Increments index by specified number.
364AF	DROPLOOP	(ob →)
36496	SWAPLOOP	(ob1 ob2 → ob2 ob1)
07321	(STOPLOOP)	(→) Destroys topmost loop environment.
34AAD	SEMILOOP	(→)
07221	INDEX@	(→ #) Recalls topmost loop counter value.
3645A	DUPINDEX@	(ob → ob #)
3646E	SWAPINDEX@	(ob1 ob2 → ob2 ob1 #)
36482	OVERINDEX@	(ob1 ob2 → ob1 ob2 ob1 #)
367D9	INDEX@#-	(# → #')
07270	INDEXSTO	(# →) Stores new topmost loop counter value.
07249	ISTOP@	(→ #) Recalls topmost loop stop value.
07295	ISTOPSTO	(# →) Stores new topmost loop stop value.
283FC	ISTOP-INDEX	(→ #)
07258	JINDEX@	(→ #) Recalls second topmost loop counter value.
072AD	JINDEXSTO	(# →) Stores new second topmost loop counter value.
07264	JSTOP@	(→ #) Recalls second topmost loop stop value.
072C2	JSTOPSTO	(# →) Stores new second topmost loop stop value.

3709B	ExitAtLOOP	(→)
		Does not exit loop immediately. Just stores zero as the stop value, so all objects until the next LOOP will be evaluated. aka: ZEROISTOPSTO

3.7 Memory Operations

3.7.1 Recalling, Storing and Purging

0797B	@	(id/lam → ob T)
		(id/lam → F)
		Basic recalling function.
2B3D5	(@DROP)	(id/lam → ob)
		(id/lam →)
		DOES <REF>@ then DROP.
35C2C	DUP@	(id/lam → id/lam ob T)
		(id/lam → id/lam F)
		Does DUP then <REF>@.
35A5B	SAFE@	(id/lam → ob T)
		(id/lam → F)
		For lams does <REF>@. For ids does <REF>?ROMPTR> to the ob found.
35A56	DUPSAFE@	(id/lam → id/lam ob T)
		(id/lam → id/lam F)
		Does DUP then <REF>SAFE@.
25EF7	SAFE@_HERE	(id → ob F)
		(id → T)
		Same as <REF>SAFE@, but works only in the current directory.
2F064	Sys@	(ID → ob T)
		(ID → F)
		Switches temporarily to the HOME directory and executes @ there.
2F2A3	XEQRCL	(id → ob)
		Same as <REF>SAFE@, but errors if variable is not found. Also works for lams, but you get the wrong error.
3F2EA	(DUPXEQRCL)	(id → id ob)
		Tries to recall, errors if not existent.
2F24E	LISTRCL	({path id} → ob)
		Recalls from specified path.
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.

2F2D5	EVALNOCKSTO	(ob id/lam →) Same as <REF>EvalNoCK:_ <REF>STO.
2F2D5	(EVLNCKSTO)	(ob id →) Does EvalNoCk: xSTO
35A29	SAFESTO	(ob id/lam →) For ids, does <REF>?>ROMPTR to the object before storing.
2F380	SysSTO	(ob ID →) Switches temporarily to the HOME directory and executes <REF>STO there.
25E79	XEQSTOID	(ob id/lam →) Same as <REF>SAFESTO, but will only store in the current directory and will not overwrite a directory. aka: ?STO_HERE
25F0C	XEQStoKey	(ob ID →)
3E823	xSTO>	(ob id →) (ob symb →) Like <REF>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 →) Inputs value for a variable and stores it.
085D3	REPLACE	(newob oldob → newob) Replaces oldob (in memory) with newob.
08C27	PURGE	(id →) Purges variable. Does no type check first.
25E78	?PURGE_HERE	(id →) Like <REF>PURGE, but only works in current directory.
1D3006	^SAFEPURGE	(idnt/lam →) Purge idnt/lam if it exist.
2C388	MOVEVAR	Move the variable to a different directory. Stack diagram unknown - level 1 must be rrp, but level two??
08696	CREATE	(ob id →) Creates a variable in the current directory. Errors if id is or contains current directory. Assumes id is not a pco.
25EC4	DoHere:	(→) Next object in the runstream is evaluated for the current directory only.
36A8B	'LAMLNAMESTO	(ob →) STO to LAM LAMLNAME.

3.7.2 Directories

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

`08DF2`	`(CREATERRP)`	`( id → )`
		Creates an empty directory. Does not check if the name is already used.
		`:: # 7FF CRDIR# SWAP CREATE ;`

`25EA1`	`CREATEDIR`	`( id → )` Creates an empty directory. Calls <REF>?PURGE_HERE first to delete the original.

`08326`	`LASTRAM-WORD`	`( rrp → ob T )`
		`( rrp → F )`
		Recalls first object in directory.

`25EE7`	`LastNonNull`	`( rrp → ob T )`
		`( rrp → F )`
		Recalls first object in directory (not null named).

`08376`	`PREVRAM-WORD`	`( ob → ob' T )`
		`( ob → F )`
		Recalls next object in directory.

`25EF2`	`PrevNonNull`	`( ob → ob' T )`
		`( ob → F )`
		Recalls next object in directory (not null named).

`082E3`	`RAM-WORDNAME`	`( ob → id )` Recalls name of object in current directory.
`25F14`	`XEQPGDIR`	`( id → )` Purges a directory. Checks references, etc. first.
`2F296`	`XEQORDER`	`( {id1 id2..} → )` Orders the variables in the directory by moving the given variables to the beginning of the directory.
`25EB9`	`DOVARS`	`( → {id1 id2..} )` Returns list of variables from current directory.
`25EB8`	`DOTVARS%`	`( % → {} )` Returns a list of variables in the current directory with user type given by the number. Internal TVARS if a single number was given.
`0BD002`	`^DOTVARS{}`	`( {# #' ...} → {} )` Returns a list of variables in the current directory with user type given by any of the numbers in the list. This is the core of the TVARS program.
`2C3FA`	`(DOTVARS)`	`( {# #' ...} → {} )` Pointer to `^DOTVARS{}`.
`25EF1`	`PATHDIR`	`( → {HOME dir1 dir2..} )` Returns current path.

2F265	`UPDIR`	(→) Goes to parent directory.
08309	`(MYRAMROMPAIR)`	(`rrp` → `rrp'` T) (`rrp` → F) Gets parent directory. Returns `FALSE` if parent directory is HOME.
08DD4	`(SYSRRP?)`	(`rrp` → `flag`) Is `rrp` HOME?
08D5A	`CONTEXT@`	(→ `rrp`) Recalls current directory.
08D08	`CONTEXT!`	(`rrp` →) Sets new current directory.
25917	`(LastContext!)`	(`rrp` →)
2591C	`(LastContext@)`	(→ `rrp`)
08DD4	`(SYSRRP?)`	(`rrp` → `flag`) Is `rrp` HOME?
08D82	`(STOPSIGN@)`	(→ `rrp`) Recalls last directory.
08D4A	`(STOPSIGN!)`	(`rrp` →) Stores new last directory.
08D92	`HOMEDIR`	(→) Sets HOME as current directory. aka: `SYSCONTEXT`
08DC4	`(SYSSTOPSIGN)`	(→) Sets HOME as last directory.
3712C	`SaveVarRes`	(→) Binds current and last directories to two nullnamed lams.
37186	`RestVarRes`	(→) First sets HOME as both the current and last directories (in case an error happens). Then, restores the current and last directories from 1LAM and 2LAM.

3.7.3 The Hidden Directory

3714A	`SetHiddenRes`	(→) Sets the hidden directory as the current and last directories.
370C3	`WithHidden`	(→ ?) Executes next command in hidden directory.
370AF	`RclHiddenVar`	(`id` → `ob` T) (`id` → F) Recalls variable in hidden directory. Same as `:: WithHidden @ ;`

37104	StoHiddenVar	(ob id →) Stores variable in hidden directory. Same as `:: WithHidden STO ;`
37118	PuHiddenVar	(id →) Purges variable in hidden directory. Same as `:: WithHidden PURGE ;`

3.7.4 Temporary Memory

06657	TOTEMPOB	(ob → ob') Copies object to TEMPOB and returns pointer to the new copy.
35C90	TOTEMPSWAP	(ob1 ob2 → ob2' ob1) Does TOTEMPOB then SWAP.
25E9F	CKREF	(ob → ob') If object is in TEMPOB, is not embedded in a composite and not referenced, does nothing. Else copies it to TEMPOB and returns the copy.
3700A	SWAPCKREF	(ob1 ob2 → ob2 ob1') Does SWAP then <REF>CKREF.
06B4E	INTEMNOTREF?	(ob → ob flag) If the object is in TEMPOB area, is not embedded in a composite and is not referenced, returns the object and TRUE, otherwise returns the object and FALSE.
06B3E	(FREEINTEMP?)	(ob → ob flag) Tests if object is in TEMPOB area and not in a composite.
01E0E8	˜INTEMPOB?	(ob → ob flag)
065D9	(PTRREFD?)	(ob → ob flag) Tests if object is referenced.
065E5	(REFERENCED?)	(ob → ob flag) Tests if object is referenced or in composite.
06BC2	(NOTREF?)	(ob → ob flag) Tests if object is not referenced or in composite. (:: REFERENCED? NOT ;)
06DDE	(>TOPTEMP)	(ob → ob') Moves object to top ob TEMPOB area. Does not garbage collection.
064BD	(TOTEMPOBADJ)	(ob → ob ob') Makes a standalone copy by moving references to a new copy.
064D6	(DOADJ1)	(ob1 ob2 → ob1 ob') Moves references from ob2 to ob1 (ob1 in TEMPOB area).

064E2	(DOADJ)	(ob1 ob2 → ob1 ob') Moves references from ob2 to ob1 (ob1 in TEMPOB area). References to body of ob2 are moved too.

3.8 Time and Alarms

26120	SLOW	(→) 15 millisecond delay.
26125	VERYSLOW	(→) 300 millisecond delay.
2F37E	SORTASLOW	(→) 1.2 second delay (4 x VERYSLOW).
2612A	VERYVERYSLOW	(→) 3 second delay.
2F2D4	dowait	(%secs →) Waits specified number of seconds.
3005E	%>HMS	(% → %hms) Converts from decimal to H.MMSS format.
30912	%%H>HMS	(%% → %%hms) Same as %>HMS, but for long reals.
30077	%HMS>	(%hms → %) Converts from H.MMSS format to decimal.
3008B	%HMS+	(%hms1 %hms2 → %hms) Adds time in hms format.
300B3	%HMS-	(%hms1 %hms2 → %hms) Subtracts time in hms format.
2EECF	TOD	(→ %time) Returns current time.
2F388	VerifyTOD	(%time → %time) Checks for validaty of time. Errors if not valid.
2EED0	DATE	(→ %date) Returns current date.
2F03B	(>DATE)	(%date →) Sets date, errors if % is not a valid 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	(→ hxs) Returns tick count. aka: SysTime
2EED3	TIMESTR	(%dt %tm → "dy dt tm") Returns string representation of time, using current format. Example: "WED 06/24/98 10:00:45A"

2F329	Date>d$	(%date → $) Returns string representation of date, using current format.
2F381	TOD>t$	(%time → $) Returns string represent the time, using current format.
2F1AB	Date>hxs13	(%date → hxs) Converts date to ticks.
2F003	(Ticks>Date)	(hxs → %date) Returns date from hxs of internal alarm list format.
2F002	(Ticks>TOD)	(hxs → %time) Returns time from hxs of internal alarm list format.
2F004	(Ticks>Rpt)	(hxs → %rpt) Converts hxs in internal alarm list format to repetition interval.

3.8.1 Alarms

2F178	ALARMS@	(→ {}) Returns internal alarms list.
2F37F	STOALM	(%date %time acti %rep → %) Stores an alarm. %repeat is the number of ticks between every repetition. Since there are 8192 ticks in a second, 60 seconds in a minute, and 60 minutes in an hour, to make an alarm that repeats every hour, %repetition would be 8192*60*60 = 29491200. Returns real number representing the position of the alarm in the list.
2F0AC	PURGALARM%	(% →) Internal <REF>xDELALARM.
2F314	RCLALARM%	(%n → {}) Recalls nth alarm. List is in the format of STOALARMLS.
25FA9	ALARM?	(→ flag) Returns TRUE if an alarm is due.
2F113	FNDALARM{}	
2F336	FindNext	

3.9 System Functions

3.9.1 User and System Flags

2614D	`SetSysFlag`	(# →) Sets the system flag with number #. <REF>TEXT:Flags
26044	`ClrSysFlag`	(# →) Clears the system flag with number #. <REF>TEXT:Flags
26170	`TestSysFlag`	(# → flag) Returns `TRUE` if system flag is set. <REF>TEXT:Flags
26152	`SetUserFlag`	(# →) Set the user flag with number #. <REF>TEXT:Flags
26049	`ClrUserFlag`	(# →) Clear the user flag with number #. <REF>TEXT:Flags
26175	`TestUserFlag`	(# → flag) Returns `TRUE` if user flag is set. <REF>TEXT:Flags
2F259	`RCLSYSF`	(→ hxs) Recalls system flags from 1 to 64. <REF>TEXT:Flags
2F25F	`(STOSYSF)`	(hxs →) Stores system flags from 1 to 64. <REF>TEXT:Flags
2F23E	`DOSTOSYSF`	(hxs →) Stores system flags from 1 to 64, checking for changes in `LASTARG` flag.
2F25A	`(RCLSYSF2)`	(→ hxs) Recalls system flags from 65 to 128.
2F260	`(STOSYSF2)`	(hxs →) Stores system flags from 65 to 128.
2F25B	`RCLUSERF`	(→ hxs) Recalls user flags from 1 to 64.
2F261	`(STOUSERF)`	(hxs →) Stores user flags from 1 to 64.
2F25C	`(RCLUSERF2)`	(→ hxs) Recalls user flags from 65 to 128.
2F262	`(STOUSERF2)`	(hxs →) Stores user flags from 65 to 128.
2F3A9	`(STOALLFcont)`	(hxs_usr hxs_sys →) Stores user and system flags from 1 to 64. First is user flags, second is system flags.
2F3AA	`(STOALLFcont2)`	(hxs_sys1 hxs_usr1 hxs_sys2 hxs_usr2 →) Expects 4 hxs and stores them as user and system flags.

3B76C	(DOSTOALLF)	({} →) 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	(→) Save system flags in a virtual stack. <REF>TEXT:Flags
25F22	RestoreSysFlags	(→) Restore system flags from virtual stack, popping that level. <REF>TEXT:Flags
2ABF0	RunSafeFlags	`Run Stream:` (ob →) Evaluates the next object in the runstream, but saves and restores the system flags around it. Uses `DoRunSafe`. This is very useful. <REF>TEXT:Flags
2AB69	RunInApprox	`Run Stream:` (ob →) Eval next object in runstream with system flags 20, 21 clear and 22, 105, 102, 120 set. -- Flags: -20 -21 -22 -105 -102 -120
2AC0E	DoRunSafe	(ob → hxs1 hxs2) Evaluate ob and put the system flags as they were before the evaluation on the stack. Used by `RunSafeFlags` and `RunSafeFlagsNoError`.

2ABD7	RunSafeFlagsNoError	`Run Stream:`
		(ob →)
		`:: 'R DoRunSafe 2DROP ;`

2EFA5	DOHEX	(→) Switch stack display format of HEX strings to hexadecimal. <REF>TEXT:Flags
2EFA8	DODEC	(→) Switch stack display format of HEX strings to decimal. <REF>TEXT:Flags
2EFA6	DOBIN	(→) Switch stack display format of HEX strings to binary.
2EFA7	DOOCT	(→) Switch stack display of HEX strings to octal.
2EFBF	BASE	(→ #) Returns #10h, #10d, #10b or #10o. In decimal terms, 16 for hexadecimal base, 10 for decimal base, 8 for octal base or 2 for binary base.

`2605D`	`DOSTD`	(→) Internal version of user word `STD`.
`26053`	`DOFIX`	(`#` →) Internal version of user word `FIX`.
`26058`	`DOSCI`	(`#` →) Internal version of user word `SCI`.
`2604E`	`DOENG`	(`#` →) Internal version of user word `ENG`.
`261A7`	`savefmt1`	(→) Saves the current number format, and changes to `STD` mode.
`261A2`	`rstfmt1`	(→) Restores the number format saved by `savefmt1`. Only one set of flags can be saved, there is no nesting of these entries.
`2FFDB`	`SETRAD`	(→) Set angular mode to RAD.
`25EF3`	`RAD?`	(→ `flag`) Is angular mode RAD?
`2FFBD`	`SETDEG`	(→) Set angular mode DEG.
`2FFEF`	`SETGRAD`	(→) Set angular mode GRAD.
`25EBA`	`DPRADIX?`	(→ `flag`) Returns `TRUE` if current radix is `"."`.
`256AC`	`UNDO_OFF`	(→) Turns saving of the last stack for UNDO off.
`256A7`	`UNDO_ON`	(→) Turns saving of the last stack for UNDO on.
`256A2`	`UNDO_ON?`	(→ `flag`) Tests if last stack saving for UNDO is on.
`25E6C`	`1A/LockA`	(→) Equivalent to pressing the ALPHA key, turns on ALPHA mode for either 1 keypress or until the next ALPHA keypress, depending on system flag 60. `--` Flags: -60

3.9.2 Hardware Tests

2F3BF	(IsApple)	(→ flag) Can be used to distinguish the old Saturn HP49G from the new ARM-based hp48gII and hp49g+. The entry returns TRUE on the new machines. On an HP49G, this entry is not present. But you can test on both machines with the following ML program:
		`CODE`
		`  $80B`
		`  XM=0`
		`  ?XM=0`
		`  SKIPYES { }`
		`  GOVLNG ="PushF/TLoop"`
		`ENDCODE`
		First available in ROM 1.22.

2F3C0	(IsMidApple)	(→ flag) Tests for the hp48gII. Returns TRUE on hp48gII, FALSE on hp49g+ and HP49G+. First available in ROM 1.22.
2F3C1	(IsBigApple)	(→ flag) Check for the hp49g+. Returns TRUE on hp49g+, FALSE on hp48gII and HP49G+. Use this entry to test for the large screen. First available in ROM 1.22.

3.9.3 General Functions

25EB2	DOBEEP	(%freq %dur →) Beeps. Analog to user function BEEP.
261AC	setbeep	(#ms #Hz →) Also beeps.
0C4002	^SERIAL	(→ $) Return a string with the Serial number of the unit.
041A7	TurnOff	(→) Internal OFF.
041ED	DEEPSLEEP	(→ flag) Puts HP into deepsleep mode. Returns TRUE if "Invalid Card Data" message.
01118	LowBat?	(→ flag) Returns TRUE if low battery.
0426A	ShowInvRomp	(→) Flashes "Invalid Card Data" message.
2EE5D	?FlashAlert	(→) Displays system warnings.

04544	(AlertStatus)	(→ #) Gets last system warning: #0h = OK #1h = Alarm #2h = LowBat (S) #4h = LowBat (P1) #8h = LowBat (P2)
04575	(Alert$)	(# → $) Recalls system warning message.
2F237	(DOAPWL)	(→) Forces a warm start but does not log a warmstart event.
04912	(LiteSlp)	(→) Enters light sleep mode.
05F42	GARBAGE	(→) Forces garbage collection.
05F61	MEM	(→ #) Returns amount of free memory in nibbles. Does not do garbage collection. (The user word does.)
05902	OSIZE	(ob → #) Returns object size in nibbles. Forces garbage collection.
05944	OCRC	(ob → #nib hxs) Returns size in nibbles and checksum as hxs.
2F257	OCRC%	(ob → hxs %bytes) Returns checksum and size in bytes.
2F267	VARSIZE	(id → hxs %bytes) Returns checksum and size in bytes of specified variable.
394C8	INHARDROM?	(ob → ob flag) Is object address < #80000h?
05AB3	CHANGETYPE	(ob #prolog → ob') Changes prolog of object, does TOTEMPOB.
25F90	>LANGUAGE	(# →) Sets the current language for messages. Internal version of x→LANGUAGE.
25F95	LANGUAGE>	(→ #) Returns the current language for messages. Internal version of the xLANGUAGE→ command.
256BE	NOBLINK	(→) Clears the BLINKFLAG, SysNib5.
25E71	?BlinkCursor	(→) Makes the cursor Blink if in App-mode or Editline.

3.10 The Virtual Stack

25F1E	`PushVStack`	`( obn..ob1 → obn..ob1 )`
		`Virtual Stack:`
		`( → [obn..ob1] )`
		Pushes the RPN stack onto the Virtual Stack. The RPN stack is unchanged.

25F1F	`PushVStack&Clear`	`( obn..ob1 → )`
		`Virtual Stack:`
		`( → [obn..ob1] )`
		Does `PushVStack` and then clears the RPN stack.

25F1A	`PopMetaVStackDROP`	`( → obn..ob1 )`
		`Virtual Stack:`
		`( [obn..ob1] → )`
		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.)

25F1B	`PopVStack`	`( obm..ob1 → 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`	`( → obn..ob1 )`
		`Virtual Stack:`
		`( [obn..ob1] → [obn..ob1] )`
		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`	`( obm..ob1 → obn'..ob1' )`
		`Virtual Stack:`
		`( [obn'..ob1'] → [obn'..ob1'] )`
		Copies the topmost virtual stack into the RPN stack. The Virtual Stack is not changed, but the current RPN stack is lost.

26265	`PushMetaVStack`	`( obn..ob1 #n → obn..ob1 #n )`
		`Virtual Stack:`
		`( → [obn..ob1] )`
		Pushes #n objects as a new virtual stack. Any other objects in the RPN stack are not pushed. The RPN stack is unchanged.

25F1D	`PushMetaVStack&Drop`	`( obn..ob1 #n → )`
		`Virtual Stack:`
		`( → [obn..ob1] )`
		Does `PushMetaVStack` then drops the pushed objects. Any other objects present in the RPN stack are neither pushed nor dropped.

25F19	`PopMetaVStack`	`( → obn..ob1 #n )`
		`Virtual Stack:`
		`( [obn..ob1] → )`
		Insers the contents of the most recent virtual stack into the RPN stack, followed by the count. The previous contents of the RPN stack are not lost.

2624C	`GetMetaVStack`	`( → obn..ob1 #n )`
		`Virtual Stack:`
		`( [obn..ob1] → [obn..ob1] )`
		Inserts the objects from the topmost virtual stack into the RPN stack, along with the count. The Virtual Stack is unchanged.

265D5	`(SetMetaVStack)`	`( obn'..ob1' #n → )`
		`Virtual Stack:`
		`( [obn..ob1] → [obn'..ob1'] )`
		Modify the elements of the Virtual Stack according to a meta on the stack. The meta on the RPN stack and the fist level of the Virtual Stack must have the same number of elements!

25F20	`PushVStack&Keep`	`( obn..ob1 obm'..ob1' #m → obm'..ob1' #m )`
		`Virtual Stack:`
		`( → [obn..ob1] )`
		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`	`( obn..ob1 obm'..ob1' #m → obm'..ob1' )`
		`Virtual Stack:`
		`( → [obn..ob1] )`
		Does `PushVStack&Keep` and then `DROP`.

25F1C	`PopVStackAbove`	`( obm'..ob1' → obn..ob1 obm'..ob1' )`
		`Virtual Stack:`
		`( [obn..ob1] → )`
		Pops the contents of the topmost virtual stack (like <REF>PopMetaVStackDROP would have done) into the RPN stack, but *above* the current contents of the RPN stack. This undoes `PushVStack&Keep` (or `PushVStack&KeepDROP`).

26215	`DropVStack`	`( → )`
		`Virtual Stack:`
		`( [obn..ob1] → )`
		Drops the topmost virtual stack from the Virtual Stack.

26229	GetElemTopVStack	(#i → obi) Virtual Stack: ([obn..ob1] → [obn..ob1]) Returns the ith object from the topmost virtual 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 →) Virtual Stack: ([obn..ob(n-i)..ob1] → [obn..new_ob..ob1]) 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: ([obn..ob1] → [obn..ob1]) 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 (generally thought of as 1), # 1 is at level 2, etc. Note: no checking wheter #i is valid.
2626A	PutElemBotVStack	(new_ob #i →) Virtual Stack: ([obn..obi..ob1] → [obn..new_ob..ob1]) Replaces the ith object from the topmost virtual stack with new_ob, counting from the bottom. Note: no checking wheter #i is valid.
26233	GetVStackProtectWord	(→ #) Hacking stuff: Gets the protection word of the last VStack level.
2622E	SetVStackProtectWord	(# →) Hacking stuff: Sets the protection word of the last VStack level.
26251	InitVirtualStack	

3.11 Kermit

27142	LAMLNAME	
2F350	'LamKPSto	
2EEBB	SENDLIST	({} →) Internal SEND.

2EEBC	GETNAME	($/id/lam →) Internal KGET.
2EEBD	DOFINISH	(→) Internal FINISH.
2EEBE	DOPKT	($ $' →) Internal PKT.
2EEC1	DOBAUD	(% →) Internal BAUD.
2EEC2	DOPARITY	(% →) Internal PARITY.
2EEC3	DOTRANSIO	(% →) Internal TRANSIO.
2EEC4	DOKERRM	(→ $) Internal KERRM.
2EEC5	DOBUFLEN	(→ % 0/1) Internal BUFLEN.
2F12E	(DOSTIME)	Internal STIME.
2EEC6	DOSBRK	(→) Internal SBRK.
2F130	(DOXMIT)	($ →) Internal XMIT.
2EEC7	DOSRECV	(% →) Internal SRECV.
2EEC9	CLOSEUART	(→) Internal CLOSEIO.
2EECB	DOCR	(→) Internal CR.
2EECD	DODELAY	(% →) Internal DELAY.
2F34B	KDispRow2	
2F34C	KDispStatus2	
2F333	EXCHINITPK	
2F372	SENDEOT	
2F374	SENDNAK	
2F373	SENDERROR	
2F376	SENDPKT	
2F0E7	InitIOEnv	
2F0E6	KERMOPEN	
2EEC0	DOOPENIO	
2F2FF	OpenIO	
2F35D	OpenIOPrt	
2F31A	APNDCRLF	($ → $') Appends carriage return and line feed to string.
2EECA	docr	

2F346	IOCheckReal	
271A3	(IDIOPAR)	ID IOPAR
2716D	StdIOPAR	(→ {}) Default IOPAR: { 9600 0 0 0 3 1 }.
2EEBF	GetIOPAR	(→ %baud % % % % %) Recalls IOPAR and explodes it into the stack.
2F062	StoIOPAR	({} →) STO the list of IO parameters in the HOME directory in the variable IOPAR.
2F37B	SetIOPARErr	Error C12h Generates "Invalid IOPAR" error.
27A3A	StdPRTPAR	
2F063	StoPRTPAR	
2F338	GetChkPRTPAR	
2F312	OpenUartClr	
2F313	OpenUart?Clr	
2F0BC	PRINT	
2F362	PRINTxNLF	
2F36A	REMAP	
2EECE	SetEcma94	
2F177	AllowPrlcdCl	
2F361	PrintGrob	
2F37D	SetServMode	
2F325	ClrServMode	
2F377	SendSetup	
2F386	TRPACKETFAIL	
2F343	IncrLAMPKNO	Increases packet number.
2F33A	GetKermPkt#	
2F3A8	(RecvNextPkt)	
2F34F	KVISLF	($ → $') String translation for transfer from HP to PC. Inserts <cr> (character 12) in front of every newline (character 10), and translates characters >127 to the corresponding backslash escape. Which translations are being made depends upon the current translation mode (the last number in the IOPAR variable, can be set with DOTRANSIO). 0: No translation 1: CRLF translation 2: CRLF and characters 128-159 (80h-9Fh) 3: CRLF and characters 128-255 (80h-FFh)

2F34E	KVIS	($ → $') Like <REF>KVISLF, but never translates newlines.
2F34D	KINVISLF	($ → $' $'') String translation for transfer from PC to HP. Translates digraphs in the string to characters and removes <cr> (character 12) in front of newline characters. Which translations are actually made depends upon the current translation mode, see KVISLF. $" contains any incomplete trailing backslash sequence in the original string.
2F33B	GETKP	
2F371	SENDACK	
2F375	SENDNULLACK	
2F319	ACK_INIT	
2F15A	CHOOSE_INIT	
2F331	ENCODE1PKT	
2F330	ENCODE	
2F32A	DECODE	
2F387	UARTBUFLEN	
2EEC8	FLUSHRSBUF	
2F364	PUTSERIAL	
2F33F	GETSERIAL	
2F389	VERSTRING	(→ $) Returns version string.
25F06	UART?	
25F07	UARTxcp	
2F3A7	(SEND_PACKET)	
2F292	XEQIOBACKUP	
00C10	kermpktmsg	
00C0E	kermrecvmsg	
00C0D	kermsendmsg	

4 Input and Output

4.1 Checking for Arguments

4.1.1 Number and Type of Arguments

262B0	CK0	(→) Saves current command to LASTCKCMD. Marks stack below level 1 to STACKMARK.
262B5	CK1	(ob → ob) Saves current command to LASTCKCMD. Verifies that there is at least one object in the stack, if not generates a "Too Few Arguments" error. Saves stack mark to STACKMARK. If Last Arg is enabled then saves the argument.
262BA	CK2	(ob1 ob2 → ob1 ob2) Like <REF>CK1, but checks for at least two arguments.
262BF	CK3	(ob1...ob3 → ob1...ob3) Like <REF>CK1, but checks for at least three arguments.
262C4	CK4	(ob1...ob5 → ob1...ob5) Like <REF>CK1, but checks for at least four arguments.
262C9	CK5	(ob1...ob5 → ob1...ob5) Like <REF>CK1, but checks for at least five arguments.
262CE	CKN	(ob1...obn %n → ob1..obn #n) Checks for a real in level one. Then checks for that number of arguments. Finally, converts the real to a bint.
262D3	(CKN+1)	(ob1...obn+1 %n → ob1..obn #n) Checks for a real in level one. Then checks for n+1 of arguments. Finally, converts the real to a bint.
26292	CK0NOLASTWD	(→) Like <REF>CK0, but does not save current command.
26297	CK1NOLASTWD	(ob → ob) Like <REF>CK1, but does not save current command.
2629C	CK2NOLASTWD	(ob1 ob2 → ob1 ob2) Like <REF>CK2, but does not save current command.
262A1	CK3NOLASTWD	(ob1...ob3 → ob1...ob3) Like <REF>CK3, but does not save current command.

262A6	CK4NOLASTWD	(ob1...ob4 → ob1...ob4) Like <REF>CK4, but does not save current command.
262AB	CK5NOLASTWD	(ob1...ob5 → ob1...ob5) Like <REF>CK5, but does not save current command.
25F25	CKNNOLASTWD	(ob1...obn %n → ob1..obn #n) Like <REF>CKN, but does not save current command.
2631E	CK&DISPATCH0	(→) Dispatches on stack argument. Does not convert ZINTs to REAL. -- <REF>CK&DISPATCH1 <REF>CK&DISPATCH2 <REF>TEXT:Dispatch_Types
26328	CK&DISPATCH1	(→) Dispatches on stack arguments, stripping tags and converting ZINTS to REALS (HP49 only) if necessary. -- <REF>CK&DISPATCH0 <REF>CK&DISPATCH2 <REF>TEXT:Dispatch_Types
26323	CK&DISPATCH2	(→) Equivalent to <REF>CK&DISPATCH1. -- <REF>CK&DISPATCH0 <REF>TEXT:Dispatch_Types
26300	CK1&Dispatch	(→) Combines <REF>CK1 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch_Types
26305	CK2&Dispatch	(→) Combines <REF>CK2 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch_Types
2630A	CK3&Dispatch	(→) Combines <REF>CK3 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch_Types
2630F	CK4&Dispatch	(→) Combines <REF>CK4 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch_Types

26314	`CK5&Dispatch`	(→) Combines <REF>CK5 with <REF>CK&DISPATCH1. -- <REF>TEXT:Dispatch_Types
25F9A	`0LASTOWDOB!`	(→) Clears command save by last CK<n> command. <REF>CK0 aka: `0LASTOWDOB!`, `0LastRomWrd!`
2EF6C	`AtUserStack`	(→) `:: CK0NOLASTWD 0LASTOWDOB! ;`
25E9E	`CK1NoBlame`	(→) `:: 0LASTOWDOB! CK1NOLASTWD ;`
354CB	`'RSAVEWORD`	(→) Stores first object in the composite above the actual to LASTCKCMD. aka: `'RSaveRomWrd`
26319	`EvalNoCK`	(comp → ?) Evaluates composite without saving as current command. If first command is CK<n>&Dispatch it is replaced by `CK&DISPATCH1`. If first command is CK<n> it is skipped. Any other first command is also skipped!
25F29	`(EvalNoCK:)`	`Run Stream:` (ob →) <REF>EvalNoCK with the next object in the runstream as argument.
25F29	`('EvalNoCK:_sup)`	`Run Stream:` (ob →) <REF>EvalNoCK with the next object in the runstream as argument. aka: `EvalNoCK:`
2A9E9	`RunRPN:`	`Run Stream:` (ob →) 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. -- Flags: -95

4.1.2 Type Checking

36B7B	`CKREAL`	(% → %) (Z → %) Checks for real. If a ZINT, convert to real. Else `SETTYPEERR`.
184006	`^CK1Z`	($/#/hxs → Z) CHecks for an integer. Converts strings, bints or hxs's to zints. Errors for other object types.

185006	^CK2Z	(ob ob' → Z Z') Like <REF>^CK1Z, but for two objects.
186006	^CK3Z	(ob ob' ob'' → Z Z' Z'') Like <REF>^CK1Z, but for three objects.
3F33F	(CKARRY)	(→) Checks for array.
3F3C1	(CKLIST)	(→) Checks for list.
3D2B4	CKSYMBTYPE	(→) Checks for quoted name (name as symbolic).
2EF07	nmetasyms	(meta → meta) Checks for meta containing %, C%, unit, id, lam or symb.
03C64	TYPE	(ob → #prolog) Returns address of prolog of object.
3BC43	XEQTYPE	(ob → ob %type) System version of user word TYPE, but this keeps the object.
3511D	TYPEREAL?	(ob → flag)
35118	DUPTYPEREAL?	(ob → ob flag) aka: DTYPEREAL?
3512C	TYPECMP?	(ob → flag)
35127	DUPTYPECMP?	(ob → ob flag)
3510E	TYPECSTR?	(ob → flag)
35109	DUPTYPECSTR?	(ob → ob flag) aka: DTYPECSTR?
35136	DUPTYPEARRY?	(ob → ob flag) aka: DTYPEARRY?
3513B	TYPEARRY?	(ob → flag ???)
35292	TYPERARRY?	(ob → flag)
352AD	TYPECARRY?	(ob → flag)
35195	TYPELIST?	(ob → flag)
35190	DUPTYPELIST?	(ob → ob flag) aka: DTYPELIST?
3504B	TYPEIDNT?	(ob → flag)
35046	DUPTYPEIDNT?	(ob → ob flag)
350E1	TYPELAM?	(ob → flag)
350DC	DUPTYPELAM?	(ob → ob flag)
194006	^TYPEIDNTLAM?	(ob → flag) Tests if ob is ID or lam.
2F0D4	(NotIDorLAM?)	(ob → ob flag) Tests if ob is neither an ID nor a LAM.
35168	TYPESYMB?	(ob → flag)
35163	DUPTYPESYMB?	(ob → ob flag)

350FF	TYPEHSTR?	(ob → flag)
350FA	DUPTYPEHSTR?	(ob → ob flag)
35186	TYPEGROB?	(ob → flag)
35181	DUPTYPEGROB?	(ob → ob flag)
351A4	TYPETAGGED?	(ob → flag)
3519F	DUPTYPETAG?	(ob → ob flag)
351B3	TYPEEXT?	(ob → flag) Is ob a unit object?
351AE	DUPTYPEEXT?	(ob → ob flag) Is ob a unit object?
3514A	TYPEROMP?	(ob → flag)
35145	DUPTYPEROMP?	(ob → ob flag)
350F0	TYPEBINT?	(ob → flag)
350EB	DUPTYPEBINT?	(ob → ob flag)
35159	TYPERRP?	(ob → flag)
35154	DUPTYPERRP?	(ob → ob flag)
3503C	TYPECHAR?	(ob → flag)
35037	DUPTYPECHAR?	(ob → ob flag)
35177	TYPECOL?	(ob → flag) Is on a secondary?
35172	DUPTYPECOL?	(ob → ob flag) Is ob a secondary? aka: DTYPECOL?
350D2	TYPEAPLET?	(ob → flag)
350CD	DUPTYPEAPLET?	(ob → ob flag)
35087	TYPEFLASHPTR?	(ob → flag)
35082	DUPTYPEFLASHPTR?	(ob → ob flag)
350C3	TYPEFONT?	(ob → flag)
350BE	DUPTYPEFONT?	(ob → ob flag)
350B4	TYPELNGCMP?	(ob → flag)
350AF	DUPTYPELNGCMP?	(ob → ob flag)
350A5	TYPELNGREAL?	(ob → flag)
350A0	DUPTYPELNGREAL?	(ob → ob flag)
35096	TYPEZINT?	(ob → flag)
35091	DUPTYPEZINT?	(ob → ob flag)
182006	^TYPEZ?	(ob → flag)
183006	^DUPTYPEZ?	(ob → ob flag)
114007	^TYPEGAUSSINT?	(ob → flag) Checks if ob is Gaussian integer. First available in ROM 1.11.

115007	^DTYPEGAUSSINT?	(ob → ob flag) Checks if ob is Gaussian integer. First available in ROM 1.11.
116007	^DUPTYPEGAUSSINT?	(ob → ob flag) Checks if ob is Gaussian integer. First available in ROM 1.11.
3505A	(TYPEBAK?)	(ob → flag)
35055	(DUPTYPEBAK?)	(ob → ob flag)
35069	(TYPELIB?)	(ob → flag)
35064	(DUPTYPELIB?)	(ob → ob flag)
35078	(TYPEMATRIX?)	(ob → flag)
35073	(DUPTYPEMATRIX?)	(ob → ob flag)
35073	(DTYPEMATRIX?)	(ob → ob flag)
351C2	(TYPEEXT0?)	(ob → flag)
351BD	(DUPTYPEEXT0?)	(ob → ob flag)
187006	^CK1Cext	(ob → flag) Checks if object is integer or Gaussian integer.
181006	^CKALG	(ob → ob) Checks that an object is real/cmplx/unit or idnt/lam/symbolic.
25E77	?OKINALG	(ob → ob flag) Is object allowed in algebraics?
171006	^DTYPFMAT?	(ob → ob flag) Tests if object is a symbolic matrix.
191006	^IDNTLAM?	(ob → ob flag) Tests if ob is `idnt` or lam.
192006	^FLOAT?	(ob → ob flag) Tests if ob is real or complex.
195006	^REAL?	(ob → ob flag) Tests if ob is real, zint or hxs.
196006	^TYPEREALZINT?	(ob → flag) Tests if ob is real, zint or hxs.
193006	^CKSYMREALCMP	(ob → ob) Does "Bad Argument Type" error if ob is not a real, complex or symbolics.

4.2 Keyboard Control

4.2.1 Converting Keycodes

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` → `%rc.p`) Converts from system key representation format 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
25EDD	`H/WKey>KeyOb`	
25EEA	`ModifierKey?`	(`#kc #pl` → `flag`) Is the key any of the three modifiers right-shift, left-shift, or alpha?
2594E	`KeyOb@`	(→ `id/romptr`) Returns the object assigned the the key which caused the current program to be executed, or whatever has been stored with `KeyOb!`
25949	`KeyOb!`	(`ob` →) Store ob as the `KeyOb`.
2593F	`KeyOb0`	(→) Clear the `KeyOb`.
25944	`(KeyOb0?)`	(→ `flag`) Is the `KeyOb` clear?

4.2.2 Waiting for Keys

261CA	`FLUSHKEYS`	(→) Flushes the key buffer. aka: `FLUSH`
04708	`CHECKKEY`	(→ `#kc T`) (→ `F`) Returns next key in the key buffer (if there is one), but does not pop it. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
04714	`GETTOUCH`	(→ `#kc T`) (→ `F`) Pops next key from key buffer (if there is one). Does handle shift-hold keys. -- <REF>TEXT:Keycodes
25ED6	`GETKEY`	(→ `#kc flag`) Get a single keypress from the keybuffer, waits if necessary. The key is returned along with `TRUE`. If an exception happens, returns `FALSE`. The exception is not handled. Does handle shift-hold keys. -- <REF>TEXT:Keycodes

25ED7	GETKEY*	(→ #kc T) (→ F F) (→ {Alrmlist} T F) Get a single keypress from the keybuffer, waits if necessary. The key is returned along with TRUE. If an exception happens (error or alarm), the exceptions is handled and the entry returns FALSE. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
25ED9	GetKeyOb	(→ ob) Wait for a single key and return the object associated with this key. Does handle shift-hold keys. -- <REF>TEXT:Keycodes
25EC5	DoKeyOb	(ob →) Execute ob as if it had been assigned to a key and the key had been pressed.
047C7	REPKEY?	(#kc → flag) Returns TRUE if the key is being pressed. -- <REF>TEXT:Keycodes
25EF5	REPEATER	(→) Takes two objects from the runstream, a BINT and a program. The BINT must represent a keycode. The program is evaluated at least once, and then again and again as long as the specified key is being pressed. -- <REF>TEXT:Keycodes
25EF6	REPEATERCH	(→) Same as REPEATER, but slower, so more appropriate for scrolling and cursor motions. -- <REF>TEXT:Keycodes
25EE3	KEYINBUFFER?	(→ flag) Returns TRUE if there is at least a key in the key buffer.
25F0B	WaitForKey	(→ #kc #flag) Returns next full key press. Does *not* handle shift-hold keys. -- <REF>TEXT:Keycodes

2F268	`Wait/GetKey`	(% → ?) Internal `WAIT` command. Does *not* handle shift-hold keys. -- <REF>TEXT:Keycodes

4.2.3 The ATTN Flag

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

4.2.4 Bad Keys

25EBF	`DoBadKey`	(→) Beeps.
25ECD	`DropBadKey`	(ob →) Beeps.
25E6E	`2DropBadKey`	(ob ob' →) Beeps.

4.2.5 User Keys

25F09	`UserKeys?`	(→ flag) Does `BINT62 TestSysFlag`.
25967	`GetUserKeys`	(→ {}) Returns user keys list (internal format). -- <REF>TEXT:Reserved|UserKeys

2F3B3	`(StoUserKeypatch)`	`( ob #kc #p →  )` Assigns an object to a key, specified in system format. If ob is `NULL{}`, then this actually deletes a key assignment. -- <REF>TEXT:Reserved|UserKeys
25962	`(UserKeys!)`	`( {} →  )` Stores user keys (list is in internal format). -- <REF>TEXT:Reserved|UserKeys
25958	`(UserKeys0)`	`( →  )`
2595D	`(UserKeys0?)`	`( → flag )`
25621	`(NonUsrKeyOK?)`	`( → flag )` Returns `TRUE` if the keys not defined do their normal actions.
25617	`(SetNUsrKeyOK)`	`( →  )` Keys not defined do their normal actions.
2561C	`(ClrNUsrKeyOK)`	`( →  )` Keys not defined just beep when pressed.
25EE5	`Key>StdKeyOb`	`( #kc #pl → 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 → ob )` If user mode is on, recalls the user object assigned to a key. If user mode is off, recalls the standard assignment instead.
25E76	`?Key>UKeyOb`	
255006	`^KEYEVAL`	`( % → ? )` Keystroke evaluation. If % is negative, the standard key is always evaluated.
25600	`(Do1User?)`	`( → flag )` Checks if the 1USR flag is set. -- Flags: -61
25605	`(SetDo1User)`	`( →  )` Sets the 1USR flag. -- Flags: -61
2560A	`(ClrDo1User)`	`( →  )` Clears the 1USR flag. -- Flags: -61
25621	`(NonUsrKeyOK?)`	`( → flag )` Returns `TRUE` if the keys not defined do their normal actions.

4.3 The Menu

4.3.1 Menu Properties

04A41	`GETDF`	`( #menukey → ob )` Gets the definition of a menu key from THOUCHTAB. #menukey = #1..#6
04A0B	`GETPROC`	`( #menukey → ob )` Gets the definition of a menu key from THOUCHTAB. #menukey = #1..#6. With #7, get the executor.
04A4C	`(SETDF)`	
04A57	`(SETPROC)`	
2581B	`(BadMenu?)`	`( → flag )` Does the menu need an update?
25820	`(SetBadMenu)`	`( → )` Mark the mennu as bad.
25825	`(ClrBadMenu)`	`( → )` Mark the menu as OK.

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.

2587C	`(LabelDef@)`	`( → ob )` Recall the current definition of LebelDef.
25908	`LastMenuDef!`	`( menu → )` Sets the definition of the last menu. menu is a MenuList or a program, or a Rompointer.
2590D	`LastMenuDef@`	`( → menu )` Recalls the definition of the last menu. menu is a MenuList or a program, or a Rompointer.
25903	`(LastMenuDef?)`	`( ob → )` Is there a value for `LastMenuDef?`
25EFB	`SaveLastMenu`	`( → )` Stores row and definition of current menu as the last menu.

260A8	`LastMenuRow!`	(#n →) 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@`	(→ #n) 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.
2584F	`(MenuData!)`	(ob →) Store ob as the current `MenuData` definition.
25854	`(MenuData@)`	(→ ob) Recall the current `MenuData` definition.
2585E	`(GetMenuData)`	(→)
2582D	`(MenuDef?)`	(→) Is there a current menu definition?
25840	`(MenuDef!)`	(ob →) Store ob as the current menu definition.
25845	`MenuDef@`	(→ menu) Recalls the current menu definition. menu is a MenuList or a program, or a Rompointer.
258EF	`(MenuExitAct!)`	(ob →) Store ob as exit action.
25EEF	`NoExitAction`	(→) Sets `NOP` as ExitAction. Mostly used to avoid that the menu is saved as the previous menu when a new Menu gets installed.
258F4	`(MenuExitAct@)`	(→ ob) Recall the current definition of `MenuExitAct`.
258FE	`(DoMenuExit)`	(→) Execute the current definition of `MenuExitAct`.
260B7	`MenuRow!`	(#n →) 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,...
260BC	`MenuRow@`	(→ #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.
2589F	`MenuKeyLS!`	(ob → ob) Set the action for left-shifted menu keys. The program receives the action part of the menu item as an argument, i.e. `{ob-NS ob-LS ob-RS}`.
25F02	`StdMenuKeyLS`	({ob-NS ob-LS ob-RS} → ?) The content of `MenuKeyLS` for standard menus.
258A4	`(MenuKeyLS@)`	(→ ob) Recall the current definition of `MenuKeyLS`.

258AE	(DoMenuKeyLS)	??? Execute the current definition of `MenuKeyLS`.
2588B	MenuKeyNS!	(og → ob) Set the action for unshifted menu keys. The program receives the action part of the menu item as an argument, i.e. ob-NS or `{ob-NS ob-LS ob-RS}`.
25890	MenuKeyNS@	(→ ob) Recall the action for unshifted menu keys.
25EFC	SetKeysNS	(ob →) Sets ob as MenuKeysNS, `DoBadKey` to LS & RS.
25F03	StdMenuKeyNS	(ob-NS → ?) ({ob-NS ob-LS ob-RS} → ?) The content of `MenuKeyNS` for standard menus.
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}`.
258B8	(MenuKeyRS@)	(→ ob) Recall the current definition of `MenuKeyRS`.
258C2	(DoMenuKeyRS)	??? Execute the current definition of `MenuKeyRS`.
25809	(Rebuild?)	(→ flag) Does the menu need a rebuild?
2580E	SetRebuild	(→) Sets the flag that the menu needs to be rebuild.
25813	(ClrRebuild)	(→) Clear the menu Rebuild flag.
258C7	ReviewKey!	(ob →) Store a program which is called with the review key (RS DOWN). The program has the stack diagram (→)
258CC	(ReviewKey@)	(→ ob) Recall the current definition of the review program.
258D6	(DoReview)	(→) Execute the program stored with `ReviewKey!`. This program should show information about the commands in the current menu page. The default program just displays the full names of the menu entries (retrieved with `GETPROC >Review$`).
25863	MenuRowAct!	(ob →) Stores ob as the RowAct menu property.
25868	(MenuRowAct@)	(→ ob) Recall the current `MenuRowAct` property.

25872	(DoMenuRowAct)	`???` Execute the current `MenuRowAct` program.
257F7	(Track?)	`( → flag )` Is there a Trach action defined for the current menu?
257FC	(SetTrack)	`( ob → )` Set the program which should be executed when the current directory changes. For many menus, this is just a `NOP`, but for example the `VAR` menu needs it to display the correct variables.
25801	(ClrTrack)	`( → )` Clear the `TrackAct` program.
258EA	(DoTrack)	`( → )` Execute the current `TrackAct` program.
25EE2	InitTrack:	`( → )` Execute the program which is next in the runstream if the directory changes. Used by the `VAR` menu to set first menurow when diretory changes, or by the CST menu to rebuild it.

258DB	(TrackAct!)	`( ob → )`
		Store a program for the track action. This program should have a stack diagram
		`( → ).`

258E0	(TrackAct@)	`( → ob )` Recall the current `TrackAct` program.

4.3.2 Building Menus

275C6	TakeOver	`( → )` Override the default menu key executer. If this is the first entry in a program, the program can be used in edit mode. When the first in a program in the label slot of a menu key, the program is evaluated to get the label object (most likely a grob).
27FED	NullMenuKey	`( → )` A placeholder for an empty menu key when defining menu lists.

275EE	Modifier	`( → )`
		`:: TakeOver ;`

27620	MenuMaker	`( → ob )`
		Quotes next object, and also provides `TakeOver`. The disassembly is
		`:: TakeOver 'R ;`
		Normally this is used like this:
		`:: MenuMaker menu InitMenu ;`

25EE0	`InitMenu`	(`menu` →) menu is `{}` or :: settings `{}` ; Settings override the default settings installed by `InitMenu`.
25EC6	`DoMenuKey`	(`menu` →) `:: SetDA12NoCh InitMenu ;`
25EE1	`InitMenu%`	(`%mnu.pg` →) (`%0` →)
25EDA	`GetMenu%`	(→ `%`)
25F00	`StartMenu`	(`menu #n` →) #n is the index of the first menu key on the page, use 1 for the first page, 7 for the second etc. `StartMenu` does ExitAction (Previous menu!), sets the default menu properties and page. Then it evaluates menu, stores result to MenuKeys and executes `SetThisRow`.
25EFE	`SetThisRow`	(→) Builds a new `TOUCHTAB`, `SetBadMenu`.
25EE8	`LoadTouchTbl`	(`MenuKey1 .. MenuKeyN #n` →) Builds new `TOUCHTAB` from menukeys.

4.3.3 Menu Display

2EF66	`SysMenuCheck`	(→) Checks menu validity. If `DA3NoCh?` then nothing. If `Track?` then ?DoTrackAct@. If `Rebuild?` then `SetThisRow`.
2DFCC	`?DispMenu`	(→) Redisplays the menu now if no key is waiting in the buffer. Even better is this: `:: DA3OK?NOTIT ?DispMenu ;`
2DFF4	`DispMenu.1`	(→) Displays menu now.
2DFE0	`DispMenu`	(→) :: `DispMenu.1 SetDAsValid` ;

4.3.4 Displaying Menu Labels

2E0D5	`Grob>Menu`	(`#col grob` →) Displays grob as menu label.
2E0F3	`Str>Menu`	(`#col $` →) Displays string as menu label.
2E11B	`Id>Menu`	(`#col id` →) Displays id as menu label.
2E107	`Seco>Menu`	(`#col ::` →) Does `EVAL` then `DoLabel`.

25886	DoLabel	(#col ob →) 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 → grob') Inserts $ into grob using CENTER$3x5 with y=5.
08E007	^WRITEMENU	($6...$1 →) Displays the six strings as menu keys.

4.3.5 General Entries

25EA6	CheckMenuRow	(# → # #')
25EFD	SetSomeRow	(#n →) with Mod(n,FFFFFh)= 0.
2589A	DoMenuKeyNS	(#n →)
275FD	MenuKey	(→) Takes NOB from Runstream.
2F15B	CLEARMENU	(→)
25F2B	CHECKMENU	(→)
3EA01	(ID_CST)	ID CST
2C2C0	nCustomMenu	(→) Installs the CST menu.
25EFF	SolvMenuInit	(→) Sets MenuKeyNS/LS/RS, ReviewKey and LabelDef properties needed by the Solver menu.
25ECC	DoSolvrMenu	(→) Installs the solver menu which is also available via 75 MENU.
25EC7	DoNameKeyLRS	
25EC8	DoNameKeyRS	
25EC3	DoFirstRow	(→) Sets the first row of the current menu.
25EC9	DoNextRow	
25ECB	DoPrevRow	

4.4 InputLine and Inputforms

4.4.1 Inputline

2EF5F	`InputLine`	`( args → $ T )`
		`( args → $ ob1..obn T )`
		`( args → ob1..obn T )`
		`( args → F )`
		`args = $pr $line #pos`
		`#I/R #I/A #alph`
		`menu #row attn #parse`

2F154	`(Ck&Input1)`	`( $1 $2 → $3 )` This is what the User command INPUT does if level 1 is a string.
2F155	`(Ck&Input2)`	`( $1 {} → $3 )` This is what the User command INPUT does if level 1 is a list.
2F344	`InputLAttn`	
2F345	`InputLEnter`	

4.4.2 Inputform

020004	`^IfMain`	`( l1..ln f1..fm #n #m msg $ → ob1..obn T )`
		`( l1..ln f1..fm #n #m msg $ → F )`
		
		`l = $ #x #y`
		`f = msg #x #y #w #h #type legal`
		`dec $hlp ChDat ChDec res init`
		Starts an input form using the new engine.

2C371	`DoInputForm`	`( l1..ln f1..fm #n #m msg $ → ob1..obn T )`
		`( l1..ln f1..fm #n #m msg $ → F )`
		
		`l = $ #x #y`
		`f = msg #x #y #w #h #type legal`
		`dec $hlp ChDat ChDec res init`
		Starts an input form using the old engine.

0050B0	`~IFMenuRow1`	`( → {} )` Returns the menu for the first menu row of an InputForm.
0060B0	`~IFMenuRow2`	`( → {} )` Returns the menu for the second menu row of an InputForm.

4.4.3 The input form message handler commands

021004	`^IfSetFieldVisible`	`( # T/F(fld/lbl) T/F(val) → )`
		`( # T/F(fld/blb) #0 → 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 current setting.

022004	`^IfSetSelected`	`( # T/F(fld/lbl) T/F(val) → )`
		`( # T/F(fld/blb) #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 → )` Sets the grob of a field or a label (modifies the data saved in the data string).
024004	`^IfSetFieldValue`	`( val # → )` Sets the value of a field (full handling, including GROB setting).
026004	`^IfGetFieldValue`	`( # → val )` Gets the value of the Nth field.
027004	`^IfGetCurrentFieldValue`	`( → )` Gets the value of the current field.
025004	`^IfSetCurrentFieldValue`	`( val → )` Sets the value of the current field.
028004	`^IfGetFieldMessageHandler`	`( # → prg )` Retrieves a field message handler.
029004	`^IfGetFieldType`	`( # → #type )` Retrieves the field type.
02A004	`^IfGetFieldObjectsType`	`( # → {} )` Retrieves the field object type list.
02B004	`^IfGetFieldDecompObject`	`( # → val )` Retrieves the field decomp value.
02C004	`^IfGetFieldChooseData`	`( # → {} )` Retrieves the field data for choose.
02D004	`^IfGetFieldChooseDecomp`	`( # → val )` Retrieves the field decomp value in case of choose.
02E004	`^IfGetFieldResetValue`	`( # → val )` Retrieves the field reset value.
02F004	`^IfSetFieldResetValue`	`( val # → )` Changes the field reset value.
030004	`^IfGetFieldInternalValue`	`( # → val )` Retrieves the field internal value.
031004	`^IfDisplayFromData`	`( → )` Displays the datastring on the screen. Takes care of the command line size.
032004	`^IfGetNbFields`	`( → #n )` Recalls the number of fields from the data string.
033004	`^IfCheckSetValue`	`( # val → )` Checks or uncheck a check field.

034004	`^IfCheckFieldtype`	`( ob → ob flag )` Checks if an object meets the current field type requirements.

04C004	`^IfGetPrlgFromTypes`	`( {} → {}' )`
		`( #FFFFF → #0 )`
		Generates a list of the allowed prologs for a field.

035004	`^IfReset`	`( → )` Resets all fields, set as the current value their reset value. Used to explode the datalist on the stack to work on it.
036004	`^IfSetField`	`( # → )` Makes a different field "current".

037004	`^IfKeyChoose`	`( → val )`
		`( → )`
		If the current field is a choose field, displays the posibilities and let the user choose. A value is returned only if the user does not press ⟨CANCEL⟩.

038004	`^IfKeyEdit`	`( → (cmd line) )` Edits the current field value if possible. You cannot edit a choose and a label choose field.

039004	`^IfKeyTypes`	`( → (cmd line) )`
		`( → )`
		Displays a Choose box with all the possible types for this field. A command line is opened only if the user replies with OK.

03A004	`^IfKeyCalc`	`( → val )` Puts the value of the field on the stack and `HALT`. Allows to the user to compute a new value.
03B004	`^IfKeyInvertCheck`	`( → )` Inverts the current check field value.
03C004	`^IfONKeyPress`	`( → )` On Key handler. Gives the oportunity to the user to perform his own program. 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.

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`	`( → )` Initializes the internal depth counter. This has to be used when running a command modifying the stack
042004	`^IfMain2`	`( $dat handl {} → F )` `( $dat handl {} → ob1...obn 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`	`( → ob1...obn )` 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) → #x #y #w #h )` Gets the size and position of an object.
047004	`^IfSetAllLabelsMessages`	`( $dat bmsg #n → $dat )` Sets the text of a set of labels.
048004	`^IfSetAllHelpStrings`	`( $dat bmsg #n → $dat )` Sets the Help String of all fields.
04D004	`^IsUncompressDataString`	`( $dc → $dat )` Uncompresses a compressed data string.
049004	`^IfCreateTitleGrob`	
046004	`^IfDisplayFromData2`	
041004	`^IfSetTitle2`	

4.5 The Filer

067004	`^Filer`	`( → )` Calls the standard filer.
06D004	`^FILER_MANAGER`	`( {path} {args} → flag )` `{args} = { item1 item2 ... }` `item = {name loc action [prog] [key]} ... }` Customized Filer, browsing all object types. {path} is the starting path for the filer, it can be an empty list for HOME. Tagging the empty list with "0", "1" or "2" makes the filer start in the corresponding port. flag is `FALSE` when filer is exited with ON, otherwise `TRUE`. <REF>Filer_Action_Reference

06E004	^FILER_MANAGERTYPE	({types} {path} {args} →) {args} = { item1 item2 ... } item = {name loc action [prog] [key]} ... } Customized filer for selected types only. The types are prologue addresses like { DOFONT DORRP DOBAK } etc. <REF>FILER_MANAGER <REF>Filer_Action_Reference
06F004	^FontBrowser	(→ font T) Uses the File Manager to search for fonts.

4.6 The Browser Engines

4.6.1 The HP48 Browser Engine

0000B3	~Choose	(::Appl $Title ::Convert {} offset → {}’ T) (::Appl $Title ::Convert {} offset → ob T) (::Appl $Title ::Convert {} offset → F) The return value is a list if checkfields are enabled, otherwise it is just the selected object. Only FALSE is returned when the user presses ⟨CANCEL⟩. -- <REF>TEXT:Browser48
0050B3	~ChooseMenu0	(→ {}) Menus with "OK". -- <REF>TEXT:Browser48
0060B3	~ChooseMenu1	(→ {}) Menus with "CANCL", "OK". -- <REF>TEXT:Browser48
0070B3	~ChooseMenu2	(→ {}) Menus with "CHK", "CANCL", "OK". -- <REF>TEXT:Browser48
0630B3	~ChooseSimple	($title {items} → ob T) ($title {items} → F) Simple interface to the HP48 choose engine. On the HP49G, calls ^RunChooseSimple. -- <REF>TEXT:Browser48
004002	^RunChooseSimple	($title {items} → ob T) ($title {items} → F) Simple interface to the HP48 choose engine. -- <REF>TEXT:Browser48

`09F002`	`^DoCKeyCheck`	(→) Toggle check on current item. -- <REF>TEXT:Browser48
`0A0002`	`^DoCKeyChAll`	(→) Check all elements. -- <REF>TEXT:Browser48
`0B0002`	`^DoCKeyUnChAll`	(→) Uncheck all items. -- <REF>TEXT:Browser48
`09E002`	`^DoCKeyCancel`	(→) Simulate Cancel. -- <REF>TEXT:Browser48
`09D002`	`^DoCKeyOK`	(→) Simulate OK. -- <REF>TEXT:Browser48
`0B3002`	`^LEDispPrompt`	(→) Redraw title. -- <REF>TEXT:Browser48
`0B2002`	`^LEDispList`	(→) Redraw browser lines. -- <REF>TEXT:Browser48
`0B1002`	`^LEDispItem`	(`#` →) Redraw one line. -- <REF>TEXT:Browser48
`0150B3`	`(~BBMoveTo)`	(`#` →) Moves selection to line and updates display. -- <REF>TEXT:Browser48
`0190B3`	`(~BBRecalOff&Disp)`	(`flag` →) Recalculates offset of selected item in page, and redraws lines if the flag is `TRUE`. -- <REF>TEXT:Browser48
`0220B3`	`(~BBRunEntryProc)`	(→) Sends message 85 to ::Appl, thus running the user-defined start-up procedure. -- <REF>TEXT:Browser48

0230B3	(~BBReReadPageSize)	(→) Re-reads the size of the page (message 57). -- <REF>TEXT:Browser48
0240B3	(~BBReReadHeight)	(→) Re-reads the height of the browser line (message 58). -- <REF>TEXT:Browser48
0250B3	(~BBReReadCoords)	(→) Re-reads the coordinates of the browser box (message 63). -- <REF>TEXT:Browser48
0260B3	(~BBReReadWidth)	(→) Re-reads the width of the browser line (message 59). -- <REF>TEXT:Browser48
0280B3	(~BBRunENTERAction)	(→) Sends message 96 to ::Appl, thus running the OK action. It does not check the value returned and never exits. -- <REF>TEXT:Browser48
0290B3	(~BBRunCanclAction)	(→) Sends message 91 to ::Appl, thus running the ⟨CANCEL⟩ action. It does not check the value returned and never exits. -- <REF>TEXT:Browser48
02F0B3	(~BBReDrawBackgr)	(→) Redraws the background. -- <REF>TEXT:Browser48
0370B3	(~BBGetNGrob)	(#n → grob) Returns nth element as a grob. -- <REF>TEXT:Browser48
0380B3	(~BBGetNStr)	(#n → $) Returns nth element as a string. -- <REF>TEXT:Browser48
03B0B3	(~BBRereadChkEnbl)	(→) Re-reads whether checkmarks are enabled. (Message 61). -- <REF>TEXT:Browser48

03C0B3	(~BBRereadFullScr)	(→) Re-reads whether to use full-screen mode. (Message 60). -- <REF>TEXT:Browser48
03D0B3	(~BReReadMenus)	(→) Re-reads the menu. (Message 83). -- <REF>TEXT:Browser48
03E0B3	(~BBReReadNElems)	(→) Re-reads the number of elements. (Message 62). -- <REF>TEXT:Browser48
03F0B3	(~BBGetN)	(#n → ob) Returns nth element. -- <REF>TEXT:Browser48
04B0B3	(~BBIsChecked?)	(#n → flag) Returns whether the given element is checked. -- <REF>TEXT:Browser48
0520B3	(~BBUpArrow)	(→ grob) Returns up arrow as grob -- <REF>TEXT:Browser48
0530B3	(~BBDownArrow)	(→ grob) Returns down arrow as grob -- <REF>TEXT:Browser48
0540B3	(~BBSpace)	(→ grob) Returns a space as grob. -- <REF>TEXT:Browser48
0590B3	(~BBPgDown)	(→) Go down one page. -- <REF>TEXT:Browser48
05A0B3	(~BBPgUp)	(→) Go up one page. -- <REF>TEXT:Browser48
05B0B3	(~BBEmpty?)	(→ flag) Returns TRUE if the browser has no elements. -- <REF>TEXT:Browser48

05C0B3	(~BBGetDefltHeight)	(→ #) Returns height of lines based on the font that will be used. This value is the default height of the browser. Equivalent to FPTR 2 64. -- <REF>TEXT:Browser48
0100E0	~BRbrowse	
0A5003	^BRDispItems	
0A4003	^BRdone	
0AB003	^BRGetItem	
0A6003	^BRinverse	
0130E0	~BRoutput	
070004	^BrowseMem.1	
0190E0	~BRRclC1	(→) `:: LAM 'BR5 ;`
0180E0	~BRRclCurRow	`:: LAM 'BR3 ;`
0030E0	~BRStoC1	`:: ' LAM 'BR5 STO ;`
0A7003	^BRViewItem	

4.6.2 The HP49 Browser Engine

072002	(^Choose3)	(meta $title #pos ::handler → ob T) (meta $title #pos ::handler → F) The main choose engine. -- <REF>TEXT:Browser49
073002	(^Choose3Save)	(meta $title #pos ::handler → ob T) (meta $title #pos ::handler → F) Save and restore HARDBUFF/2 around a `^Choose3` call. -- <REF>TEXT:Browser49
074002	(^Choose3Index)	(meta $title #pos ::handler → #idx T) (meta $title #pos ::handler → F) Same as `^Choose3`, but returns the index of the selected item instead of the item itself. #idx starts at zero. -- <REF>TEXT:Browser49

06E002	(^Choose2)	(meta $title #pos → ob T) (meta $title #pos → F) Call ^Choose3 with empty message handler. -- <REF>TEXT:Browser49
06F002	(^Choose2Save)	(meta $title #pos → ob T) (meta $title #pos → F) Save and restore HARDBUFF/2 around a ^Choose2 call. -- <REF>TEXT:Browser49
070002	(^Choose2Index)	(meta $title #pos → #idx T) (meta $title #pos → F) Call Choose3Index with empty message handler. This is just `:: 'DROPFALSE FPTR2 ^Choose3Index ;` -- <REF>TEXT:Browser49
005002	(^sysCHOOSE)	($title {} %sel → ob %1) ($title {} %sel → %0) Equivalent to User RPL CHOOSE command. -- <REF>TEXT:Browser49
075002	(^ChooseDefHandler)	(→ ::handler) Pushed the default message handler (the one used by the ⟨CAT⟩ key) on the stack. -- <REF>TEXT:Browser49
088002	(^SaveHARDBUFF)	(→) Save HARDBUFF and HARDBUFF2 is a safe place. -- <REF>TEXT:Browser49
089002	(^RestoreHARDBUFF)	(→) Restore HARDBUFF and HARDBUFF2 saved with SaveHARDBUFF. -- <REF>TEXT:Browser49
077002	(^Choose3OK)	(→) The OK action executed by Choose3 if OK or ENTER is pressed. -- <REF>TEXT:Browser49
076002	(^Choose3CANCL)	(→) The CANCEL action executed by Choose3 if ⟨CANCL⟩ or ⟨ON⟩ is pressed. -- <REF>TEXT:Browser49

4.7 The Parametrized Outer Loop (POL)

2B475	`ParOuterLoop`	(`Disp Keys NonAppKeys? DoStdKeys? menu #row suspendOK? ExitCond AppErr` →)
2B4AC	`POLSaveUI`	(`Disp Keys NonAppKeys? DoStdKeys? menu #row suspendOK? ExitCond AppErr` →) Saves current UI to `LAMSavedUI`.
2B542	`POLSetUI`	`<see>ParOuterLoop` Sets new UI, same arguments as to `ParOuterLoop`.
2B628	`POLKeyUI`	(→) Displays, reads and evaluates keys according to set UI.
2B6CD	`POLRestoreUI`	(→) Restores saved UI from `LAMSavedUI`.
2B6B4	`POLResUI&Err`	(→) Restores saved UI and executes `ERRJMP`.
29F25	`AppDisplay!`	(`ob` →)
29F35	`AppDisplay@`	(→)
29F55	`AppKeys!`	(`ob` →)
29F75	`AppKeys0`	???
29F65	`(AppKeys@)`	
2A055	`AppExitCond!`	(`ob` →)
2A065	`AppExitCond@`	(→ `ob`)
2A145	`AppError!`	(`ob` →)
2A158	`AppError@`	(→ `ob`)
25690	`AppMode?`	(→ `flag`) Is currently a POL active?
25695	`SetAppMode`	(→)
2569A	`ClrAppMode`	(→)
2564D	`SetNAppKeyOK`	(→)
25652	`(ClrNAppKeyOK)`	(→)
2565A	`DoStdKeys?`	(→ `flag`)
2565F	`SetDoStdKeys`	(→)
25664	`(ClrDoStdKeys)`	(→)
25F04	`SuspendOK?`	(→ `flag`) Does the current user interface allow suspension?
27E72	`nohalt`	(→ `ob`) `:: LAM 'nohalt ;`
2566C	`(AppSuspOK?)`	(→)
25671	`SetAppSuspOK`	(→)
25676	`ClrAppSuspOK`	(→)

2B709	InitPOLVars	

4.8 Editor Commands

4.8.1 Status

257A2	EditLExists?	(→ flag) Does an EditLine exist?
2EEED	NoEditLine?	(→ flag) Does no EditLine exist?
2F196	RCL_CMD	(→ $) Returns a copy of the current command line to the stack. Same as EDITLINE$.
2EEEB	EDITLINE$	(→ $) Returns a copy of the current command line to the stack. Same as RCL_CMD.
2F197	RCL_CMD2	(→ $) 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 EditLine.
2EF87	RCL_CMD_POS	(→ #) Recalls the current cursor position.
26585	CURSOR@	(→ #) Recalls the current cursor position.
26594	(CURSOR_PART)	(→ #) Recalls the current cursor row (line). There is no such entry for the column, but CURSOR_OFF FIRSTC@ #+ can be used for this purpose.
2F158	(ChrAtCur)	(→ chr) Returns the character under the cursor. At the end of the file, returns CHR_00.
2EEEA	CURSOR_END?	(→ flag) Checks if the cursor is at the end of a line or at the end of the file. Works by checking the current character against newline and CHR_00.
2EF91	CAL_CURS_POS	(#l #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	(#l #c → #) 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	(→ $) Recalls a string with current editor settings. Can be used together with STO_CMD_MODE to save and restore the state of the EditLine, when temporarily leaving the editor with HALT or when calling a program which must temporarily change settings.
2F198	STO_CMD_MODE	($ →) Stores a mode string similar to the one obtained by RCL_CMD_MODE.
26599	(CURSOR_PART+)	
2659E	(CURSOR_PART-)	
265A3	(CURPART->1)	
265A8	(CURPART->CR+)	
26562	(CURSORPLUS)	
26567	(CURSORMINUS)	
26571	(?CURSOR+)	
2658F	(CURSOR-)	

4.8.2 Display Window

264B3	(TOPLINE!)	(# →) Sets the line of the current editor content which should be displayed at the top of the editor window.
264B8	(TOPLINE@)	(→ #) Recalls the line number of the first displayed line.
264BD	(TOPLINE+)	(→) Increases TOPLINE by one. If the cursor leaves the screen, cursor and display window are moved to the beginning of the file.
264C2	(TOPLINE-)	(→) Decreases TOPLINE by one. If the cursor leaves the screen, cursor and display window are moved to the beginning of the file.
264CC	FIRSTC@	(→ #) Column of the left display window edge.
264DB	FIRSTC+	(→) Increases the position of the left window ege by one.

264D6	(FIRSTC-)	(→) Decreases the position of the left window ege by one.
264D1	SETFIRSTC_0	(→) Sets the position of the left display window edge to zero.
26030	CURSOR_OFF	(→ #) Cursor column relative to left edge of display window.
26580	CURSOR_OFF+	(→) Increases the CURSOR offset by one.
2657B	CURSOR_OFF0	(→) Sets the cursor offset to zero.
26576	(CURSOR_OFF!)	(# →) Sets the cursor offset.

4.8.3 Inserting Text

2EF74	CMD_PLUS	($ →) Inserts string at current cursor position in EditLine.
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.
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 EditLine.
2EEE4	Echo$Key	($/chr →) Same as CMD_PLUS.
2EEE3	EchoChrKey	($/chr →) Same as CMD_PLUS, but first ?TogU/LCase.
2F11C	Echo$NoChr00	($ →) Inserts string at current cursor position in EditLine.
25EC1	DoDelim	(→) Takes a character or string from the runstream and inserts it.

25EC2	`DoDelims`	`( → )` Takes a character or a string from the runstream, inserts it and moves the cursor back by one character.
25795	`INSERT_MODE`	`( → )` Turns insert mode on. In insert mode, new characters do not overwrite old ones.
2577F	`(TOGGLE_I/R)`	`( → )` Toggles the insert/overwrite flag.
2ACB0	`?TogU/LCase`	`( chr → chr' )` Toggle upper/lowercase of character if some condition is fulfilled.
25790	`INSERT?`	`( → flag )` Returns `TRUE` if insert mode is active.

4.8.4 Deleting Text

2EF82	`CMD_DEL`	`( → )` 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.
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`	`( → )` Clears the entire EditLine.

2EEE7	`InitEdLine`	`( → )`
		`:: DEL_CMD ;`

2F2F0	`DO<Del`	`( → )` Deletes left to beginning of word. Same as the ⟨←DEL⟩ button in the editor TOOL menu.
2F2F1	`DO>Del`	`( → )` Deletes right to beginning of next word, Same as the ⟨DEL→⟩ button in the editor TOOL menu.
2F2F9	`DODEL.L`	`( → )` 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`	`( → )` Deletes to beginning of line. Same as the ⟨RS⟩+⟨←DEL⟩ in the editor TOOL menu.

2F2DE	DoFarDel	($\rightarrow$) Deletes to end of line. Same as ⟨RS⟩+⟨Del→⟩ in the editor TOOL menu.

4.8.5 Moving the Cursor

2EF8B	STO_CURS_POS	(# $\rightarrow$) 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	(# $\rightarrow$) Same as `STO_CURS_POS`, but moves the right edge of the editor window to the cursor column.
2EF8D	STO_CURS_POS3	(# $\rightarrow$) 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	(# $\rightarrow$) Behaves with respect to editor window positioning like <REF>STO_CURS_POS2, but with respect to invisible chars like <REF>STO_CURS_POS3.
2EF8F	STO_CURS_POS_VIS	(# $\rightarrow$) Like <REF>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	(# $\rightarrow$) ({# #'} $\rightarrow$) Sets the cursor to the given position. For the list argument, the numbers are row and column.
2611B	SETCURSOR	
2EF7C	CMD_NXT	($\rightarrow$) Moves cursor to next char, like Right Arrow.
2EF7B	CMD_BAK	($\rightarrow$) Moves cursor to the left. Same as as Left Arrow.
2EF80	CMD_DOWN	($\rightarrow$) Moves cursor to the next line. Same as Down Arrow.

2EF7F	`CMD_UP`	(→) Moves cursor to the previous line, like Up Arrow.
2EF7D	`CMD_DEB_LINE`	(→) Moves cursor to the beginning of line. Same as RS+LEFT.
2EF7E	`CMD_END_LINE`	(→) Moves cursor to the end of line. Same as RS+RIGHT.
2EF7A	`CMD_PAGED`	(→) Moves cursor one page down, like LS+DOWN.
2EF77	`CMD_PAGEL`	(→) Moves cursor one page left, like LS+LEFT.
2EF78	`CMD_PAGER`	(→) Moves cursor one page right, like LS+RIGHT.
2EF79	`CMD_PAGEU`	(→) Moves cursor one page up, like LS+UP.
2F2EE	`DO<Skip`	(→) Skips left to beginning of word. Same as the ←SKIP button in the editor TOOL menu.
2F2EF	`DO>Skip`	(→) Skips right to the beginning of the next word. Same as the SKIP→ button in the editor TOOL menu.
2F2E4	`DO>BEG`	(→) Goes to begin of selection (if active) or to beginning of EditLine. Same as ⟨→BEG⟩ button in the editor TOOL menu.
2F2E5	`DO>END`	(→) Goes to end of selection. Same as the ⟨→END⟩ button in the editor TOOL menu. When there is no selection, does not move.
2F2E6	`GOTOLABEL`	(→) Brings up the CHOOSE-box with labels in the EditLine. Same as the LABEL button in the editor TOOL/GOTO menu.

4.8.6 Selection, Cut and Paste, the Clipboard

2EF83	`CMD_STO_DEBUT`	(# →) Sets begin marker, like ⟨RS⟩+⟨BEGIN⟩, but takes position from stack.
2EF84	`CMD_STO_FIN`	(# →) Sets end marker, like ⟨RS⟩+⟨END⟩, but takes position from stack.

2EF85	`RCL_CMD_DEB`	(→ #) (→ #0) Recalls the position of the `BEGIN` marker. If the selection has been cleared, returns `ZERO`.
2EF86	`RCL_CMD_FIN`	(→ #) (→ #0) Recalls the position of the `END` marker. If the selection has been cleared, returns `ZERO`.
2F2DC	`ClearSelection`	(→) Unselects the selected text without changing the contents of the editor. Sets both begin and end marker to `ZERO`.
2EF93	`VERIF_SELECTION`	(→ flag) Returns `TRUE` when the `END` marker is not `ZERO`, indicating that the selection is active. Use this command as a check before doing anything with the selection.
2EF8A	`CMD_COPY`	(→) Copies selected string, like ⟨RS⟩+⟨COPY⟩.
2EF88	`CMD_CUT`	(→) Cuts string. Really is "delete", does not copy to kill buffer. So a "normal" CUT would be `:: CMD_COPY CMD_CUT ;`
2EF89	`CUT.EXT`	(→ $) ML routine used by `CMD_CUT`. Should not be used on its own since it does not move the cursor position.

2F2FA	`CMD_COPY.SBR`	(→ $)
		Puts the selection as a string on the stack. This command is font/style aware. It is recommended not to use it because it may get the wrong text style if the cursor is not re-positioned to the beginning of the selection first. If you don't use fonts,
		`:: RCL_CMD`
		`   RCL_CMD_DEB RCL_CMD_FIN`
		`   SUB$ ;`
		does something similar.

2EF94	`PASTE.EXT`	($ →) Pastes from stack with treatment of fonts and styles. Inserts the string on stack level 1 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`	(→) 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`	`( → )` 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@)`	`( → $ )` Recalls `Clipboard` contents to stack.
2A0A5	`(Clipboard0)`	`( → )` Clears the `Clipboard`.
2A0B5	`(Clipboard?)`	`( → flag )` Is there anything on the `Clipboard?`

4.8.7 Search and Replace

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 GET.W<-.
2F2F4	`GET.W<-`	`( # → #' )` 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->.
2576D	`(CaseSensitive?)`	`( → flag )` Is the flag for case-sensitive search currently set?
25772	`(SetCaseSensitive)`	`( → )` Set case-sensitive seatch.
25777	`(ClrCaseSensitive)`	`( → )` Set case-insensitive search.
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.
2A0C5	`(FindPattern!)`	`( $ → )` Sets the find pattern.
2A0D5	`(FindPattern@)`	`( → $ )` Recalls the current find pattern. If there is not current pattern, this returns PTR 0 - so always check first with `FindPattern?`.
2A0E5	`(FindPattern0)`	`( → )` Deletes the current find pattern.

2A0F5	`(FindPattern?)`	(→ `flag`) Checks if a find pattern has been defined.
2A105	`(ReplacePattern!)`	($ →) Sets the replace pattern.
2A115	`(ReplacePattern@)`	(→ $) Recalls the current replace pattern. If there is not current pattern, this returns PTR 0 - so always check first with `ReplacePattern?`.
2A125	`(ReplacePattern0)`	(→) Deletes the current replace pattern.
2A135	`(ReplacePattern?)`	(→ `flag`) Checks if a replace pattern has been defined.
2F2E8	`DOFIND`	(→) Same as the FIND menu button in the editor TOOL/SEARCH menu. Pops up the FIND input form.
2F2EA	`DONEXT`	(→) Finds next. Same as the NEXT button in the editor TOOL/SEARCH menu. Uses the pattern set with `FindPattern!`.
2F2E9	`DOREPL`	(→) Same as the REP button in the editor TOOL/SEARCH menu. Pops up the REPLACE input form.
2F2EB	`DOREPLACE`	(→) Replaces current match. Same as the R button in the editor TOOL/SEARCH menu. Uses the pattern set with `ReplacePattern!`.
2F2EC	`DOREPLACE/NEXT`	(→) Replaces current match and move to next match. Same as the R/N button in the editor TOOL/SEARCH menu.
2F2ED	`REPLACEALL`	(→) Replaces all matches in buffer. Same as the ALL button in the editor TOOL/SEARCH menu.
2F2FC	`REPLACEALLNOSCREEN`	(→) Like <REF>REPLACEALL, but does not update the screen. Much faster this way.

4.8.8 Evaluation

2F2DF	`EditSelect`	(→) 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`	`(` → `)` Evaluates the current line and replace it with the result of the evaluation. Similar to `EVAL.SELECTION`, but without the need to select the line first.
2F2FB	`EVAL.SELECTION`	`(` → `)` Evaluates the current selection and replace it with the result of the evaluation. Same as the EXEC button in the editor TOOL menu.
2F2F8	`EXEC_CMD`	`( cmd algflag` → `obsel )` Runs a command on the selection in the Editline. 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` → `)` Saves current user interface, evaluate ob and restore the user interface. Can be used to run applications from inside another application.

4.8.9 Starting the Editor

2F19A	`ViewLevel1`	`( ob` → `ob' )` Edits the object in level 1.
2F2DA	`AlgCharEdit`	
2F1AF	`AlgObEdit`	`( ob` → `ob' )` Used instead of `ViewLevel1` if in Algebraic mode. Does not execute STARTED and EXITED.
2F1AD	`CharEdit`	
2B2F2	`(DoLevel1:)`	`( ob` → `ob' )` Evaluates the next object in the runstream, which usually in an editing command like <REF>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.
257BE	`ClrNewEditL`	
2F1A8	`EditFont`	
2EEE5	`EditLevel1`	`( ob` → `ob' )`

`2F1AE`	`ObEdit`	`( ob → ob' T )`
		`( ob → F )`
		Edits object. When the user cancels, only `FALSE` is returned. Otherwise the changed object along with `TRUE` is returned.

`2F1AC`	`StrEdit`	

`011004`	`^EQW3Edit`	`( symb → symb' T )`
		`( symb → 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`.

`2EEE9`	`EditString`	`( $ → )` Starts editing the string in the command line when the current program exits. This is the entry to use if a program should exit with the command line. Use `InitEdLine` before this entry to clear the command line (if desired) - if not, the string is inserted into the existing command line. All code after this entry will be executed *before* control is handed to the editor application. For example: (see code below) Note that when you press ENTER after editing, the command line will be parsed normally.

`2B351`	`Rcl&Do:`	`( id → )`
		Executes the program which is next in the runstream on the contents of the variable. The program typically is an edit command, with the stack diagrams
		`( ob → ob' T )`
		`( ob → F )`
		If the flag is `TRUE`, ob' is stored back into the original variable.

```
::
  "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)
;
```

2B31A	`Roll&Do:`	`( # → )`
		Does `ROLL` and then executes the program which is next on the runsteam. So the program is applied to the object on level #. Typically, this is an edit command, with the stack diagram
		`( ob → ob )`
		After the program exits, `UNROLL` is used to put the object back to the right stack position. This entry is probably used in the interactive stack.

2F09B	`(Rcl&Edit)`	`( id → )` Uses `Rcl&Do:` to edit the contents of the variable.
2F09C	`(Rcl&View)`	`( id → )` Uses `Rcl&Do:` to view the contents of the variable.
2F09D	`(Roll&Edit)`	`( # → )` Uses `Roll&Do:` to edit the contents of specified stack level.
2F09E	`(Roll&View)`	`( # → )` Uses `Roll&Do:` to view the contents of specified stack level.

4.8.10 Miscellaneous

25ED2	`EditMenu`	`( → {} )` Returns the Editor menu.
2EF73	`?Space/Go>`	`( → )` 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.

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 surrounded by spaces.

2EF75	`AddTrailingSpace`	`( $ → $' )` Adds a trailing space to the string on level1 unless the string already ends with a space.

26855	`CMDSIZE`	`( → # )`
		ML entry point to get the size of the EditLine. As ML entries cannot be called directly from SysRPL, don't use it unless you know the necessary magic.
		`:: RCL_CMD LEN$ ;`
		works for us assembler dummies ;-)

2EF9A	CommandLineHeight	(→ #pix) Returns the number pixel rows occupied by visible part of the EditLine.
2F2DB	DOTEXTINFO	(→) Displays the info screen about the Editline. Same as the INFO button in the editor TOOL menu.
2F2F6	GET_CUR_FONT.EXT	(→ #) Returns the ID (as a system binary) of the font used for the character under the cursor.
2EF96	NO_AFFCMD	(→) 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	(→) Redisplays the command line.
2F19F	?DispCommandLine	(→) Redisplays the command line if necessary.

2F2F7	PUT_STYLE	(# →)
		Changes the style at point. If the selection is active, changes the style of the text in the selection. Otherwise changes the style of text typed subsequently. Takes a BINT from the stack which is the number of the style. In think the ITALI button in the editor TOOL/STYLE menu could be implemented with the following program:
		`:: ERRSET PUT_STYLE`
		`   ERRTRAP ERRJMP ;`
		PUT_STYLE does not ABND its temporary environment, so you need the ERRTRAP construction to work around this bug.

2F2F5	PUT_FONTE	(# →) Changes the font at point. Works similar to the PUT_STYLE command.
2F2E7	SELECT.FONT	(→) Pops up the CHOOSE box to select a font. Same as the FONT button in the editor TOOL/STYLE menu.
2F2E0	ViewEditGrob	(→) at cursor Views the grob currently edited in the Editline near the cursor. If the EditLine contains `GROB 10 10 FFFFFF...` move the cursor to the "1" of the first "10". Then this entry point will display the grob.

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`	`( → {} )` Returns the ⟨←DEL⟩ menu key.
27F9A	`>DelKey`	`( → {} )` Returns the ⟨DEL→⟩ menu key.
27EAF	`<SkipKey`	`( → {} )` Returns the ⟨←SKIP⟩ menu key.
27EFB	`>SkipKey`	`( → {} )` Returns the ⟨SKIP→⟩ menu key.
2EEE6	`InitEd&Modes`	`( → )` :: `InitEdLine InitEdModes` ;

2EEE7	`InitEdLine`	`( → )`
		`:: DEL_CMD ;`

2EEE8	`InitEdModes`	`( → )`
2F05E	`SaveLastEdit`	`( $ → )` Calls CMD_STO if history is on.
2F326	`CMDSTO`	`( $ → )` Adds string to the list of the last 4 commands, accessible with the ⟨CMD⟩ key.

4.9 Entries Related to the Equation Writer

010004	`^EQW3`	
01D004	`^EQW3Code`	
01C004	`^EQW3CursorOff`	
01B004	`^EQW3CursorOn`	

011004	`^EQW3Edit`	`( symb → symb' T )`
		`( symb → 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`.

012004	`^EQW3StartEdit`	
016004	`^EQW3ViewLeft`	
014004	`^EQW3ViewLeftX`	

013004	^EQW3ViewMargin
017004	^EQW3ViewRight
018004	^EQW3ViewRightRPL
015004	^EQW3ViewRightX
2F192	DoNewEqw

4.10 Entries Related to the Matrix Editor and Matrix Operations

2F142	DoNewMatrix	(→ []/[[]]) Start matrix editor to enter a new matrix.
007007	^DoNewMatrixReal	(→ []/[[]]) Start matrix editor to enter a real matrix. ZINTs are converted to reals.
008007	^DoNewMatrixCplx	(→ []/[[]]) Start matrix editor to enter a complex matrix. ZINTs and REALS are converted to complex.
00B007	^DoNewMatrixRealOrCplx	([] → [[]]) Will edit an array of either reals or complex numbers.
2F13C	DoOldMatrix	([] → []') Edit an existing matrix.
009007	^DoOldMatrixReal	([] → []') Edit an existing real matrix in the matrix editor.
00A007	^DoOldMatrixCplx	([] → []') Edit an existing complex matrix in the matrix editor.
006007	^RunDoNewMatrix	(→ []/[[]]) Start matrix editor for new matrix.
005007	^RunDoOldMatrix	([] → []') Edit any kind of Arry/matrix.

4.11 The Display

4.11.1 Display Organization

26166	TOADISP	(→) Sets the text display as the active.
2616B	TOGDISP	(→) Sets the graphic display as the active.
25FA4	ABUFF	(→ textgrob) Returns the text grob to the stack.

26076	GBUFF	(→ graphgrob) Returns the graphic grob to the stack. The HP49 extable address for ExitAction! is the same, but this must be a bug.
2608F	HARDBUFF	(→ dispgrob) Returns the current grob to the stack.
26094	HARDBUFF2	(→ menugrob) Returns the menu grob to the stack.
25EDE	HARDHEIGHT	(→ #height) Returns the height of HARDBUFF.
25ED5	GBUFFGROBDIM	(→ #height #width) Returns dimensions of graphic grob.

4.11.2 Preparing the Display

25EF4	RECLAIMDISP	(→) Activates the text grob, clears it and sets the default size.
2EE7D	ClrDA1IsStat	(→) Suspends clock display.
2EEFD	MENUOFF?	(→ flag) Returns TRUE if the menu grob is off.
2F034	TURNMENUOFF	(→) Turns off menu display, enlarges ABUFF to fill screen.
2F031	TURNMENUON	(→) Turns menu grob on.
2EEFC	MENUOFF	(→)
26247	GetHeader	(→ #) Gets header size in lines (0-2).
26283	SetHeader	(# →) Sets header size in lines (0-2).
26099	HEIGHTENGROB	(grob #rows →) Heightens graph or text grob.
260A3	KILLGDISP	(→) Clears graph display by setting it to NULLGROB. See DOERASE.
2EEF9	DOERASE	(→) Erases the graphics display grob without changing its size.

4.11.3 Immediate Refresh

2EF67	`SysDisplay`	(→) Redisplays all required areas. Does it immediately, without waiting for the current command to finish.
2F19F	`?DispCommandLine`	(→) Redisplays the command line if necessary.
2F19E	`DispCommandLine`	(→) Redisplays the command line.
2EE5A	`DispEditLine`	(→) Just calls `DispCommandLine`.
2DFCC	`?DispMenu`	(→) Redisplays the menu now if no key is waiting in the buffer. Even better is this: `:: DA3OK?NOTIT ?DispMenu ;`
2DFF4	`DispMenu.1`	(→) Displays menu now.
2DFE0	`DispMenu`	(→) :: `DispMenu.1 SetDAsValid` ;
2C341	`?DispStack`	(→) Redisplays the stack now if necessary.
2C311	`?DispStatus`	(→) Redisplays the status area now if necessary.
2C305	`DispStatus`	(→) Displays the status area now.
2C2F9	`DispStsBound`	(→) Displays a horizontal line at y=14, normally the separation between header and stack.
2EE5B	`DispTime?`	
2A7F7	`DispTimeReq?`	(→ `flag`) Is time display required? Checks system flag 40 and something else.
048F9	`(ShowClk?)`	(→ `flag`) Checks both `DispTime?` and `DispTimeReq?`.
2F300	`DispILPrompt`	(→) Redisplays the InputLine prompt, i.e. refreshes the region between the command line and the header during `InputLine`. Requires a string (the prompt) in 4LAM.
26260	`nDISPSTACK`	(`$prompt #height #header flag flag` →) Used by `DispILPrompt`.

4.11.4 Controlling Display Refresh

2EE8D	`ClrDA1OK`	(→)
2EE8E	`ClrDA2aOK`	(→)
2EE8F	`ClrDA2bOK`	(→)

2EE90	ClrDA2OK	(→)
2EE6E	ClrDA3OK	(→)
2EE6D	ClrDAsOK	(→)
2EE62	DA1OK?	(→ flag)
2EE82	(DA2aOK?)	(→ flag)
2EE84	(DA2bOK?)	(→ flag)
2EE86	(DA2OK?)	(→ flag)
2EE63	DA3OK?	(→ flag)
2EE88	(DAsOK?)	(→ flag)
2EE66	DA2aLess1OK?	(→ flag)
2BF3A	DA1OK?NOTIT	(→) Does DA1OK?, NOT then IT.
2BF53	DA2aOK?NOTIT	(→) DA2aOK?, NOT then IT.
2BF6C	DA2bOK?NOTIT	(→) DA2bOK?, NOT then IT.
2BF85	DA3OK?NOTIT	(→) Does DA3OK?, NOT then IT.
2EE69	SetDA1Temp	(→)
2EE8A	SetDA2aTemp	(→)
2EE6A	SetDA2bTemp	(→)
2EEA7	ClrDA2bTemp	(→)
2F37A	SetDA2OKTemp	(→)
2EE6B	SetDA3Temp	(→)
2EE71	SetDA12Temp	(→)
2EE64	SetDAsTemp	(→)
2EEA3	(SetDA2aTempF)	(→)
2EEA5	SetDA2bTempF	(→)
2EEA9	(SetDA3TempF)	(→)
2EE67	SetDA1Valid	(→)
2EF98	SetDA2aValid	(→)
2EE68	SetDA2bValid	(→)
2EE91	SetDA2Valid	(→)
2EF99	SetDA3Valid	(→)
2EE92	(SetDAsValid)	(→)
2EE97	(SetDA1ValidF)	(→)
2EEA0	SetDA3ValidF	(→)
2EE78	SetDA1Bad	(→)
2EE74	ClrDA1Bad	(→)

2EEB0	DA1Bad?	(→ flag)
2EE79	SetDA2aBad	(→)
2EE83	(SetDA2aBadT)	(→ T) (SetDA2aBad TRUE)
2EE75	ClrDA2aBad	(→)
2EEB1	DA2aBad?	(→ flag)
2EE7A	SetDA2bBad	(→)
2EE85	(SetDA2bBadT)	(→ T) (SetDA2bBad TRUE)
2EEB3	ClrDA2bBad	(→)
2EEB2	DA2bBad?	(→ flag)
2EE7B	SetDA3Bad	(→)
2EE87	(SetDA3BadT)	(→ T) (SetDA3Bad TRUE)
2EEB5	ClrDA3Bad	(→)
2EEB4	DA3Bad?	(→ flag)
2EE72	SetDA1NoCh	(→)
2EEBA	(DA1NoCh?)	(→ flag)
2EE73	SetDA2aNoCh	(→)
2EEB9	(DA2aNoCh?)	(→ flag)
2EE76	SetDA2bNoCh	(→)
2EE81	ClrDA2bNoCh	(→)
2EEB7	DA2bNoCh?	(→ flag)
2EE93	SetDA2NoCh	(→)
2EE6F	SetDA12NoCh	(→)
2EE77	SetDA3NoCh	(→)
2EEB6	(ClrDA3NoCh)	(→)
2EE70	SetDA13NoCh	(→)
2EE94	SetDA23NoCh	(→)
2EE65	SetDA12a3NCh	(→) aka: SetDA12a3NoCh
2F379	SetDA123NoCh	(→)
2EE7C	SetDAsNoCh	(→)
2EE6C	SetDA2aEcho	(→)
2EEAC	SetDA1IsStat	(→)
2EEAE	SetNoRollDA2	(→)
2EEAF	ClrNoRollDA2	(→)
2EEAD	(NoRollDA2?)	(→ flag)
2EEAB	DA1IsStatus?	(→ flag)

2EE7F	`SetDA2bIsEdL`	(→)
2EE7E	`DA2bIsEdL?`	(→ `flag`)
2EE80	`ClrDA2bIsEdL`	(→)
2EE8B	`MENoP&FixDA1`	
2EF59	`MENP&FixDA12`	
25EA8	`Ck&Freeze`	(`%` →) Internal FREEZE.

4.11.5 Clearing the Display

25E7E	`BLANKIT`	(`#startrow #rows` →) Clears #rows from `HARDBUFF`, starting at #startrow.
26021	`CLEARVDISP`	(→) Clears `HARDBUFF`.
2EED4	`Clr8`	(→) Clears top eight rows (first status line).
2EED5	`Clr8-15`	(→) Clears 2nd status line.
2F15E	`Clr16`	(→) Clears top 16 rows.
2EF5E	`BlankDA1`	(→) Clears status area from `HARDBUFF`.
2F31C	`BlankDA2a`	(→) Clears display area DA2a.
2F31B	`BlankDA2`	(→) Clears display areas DA2a and DA2b.
2EE5C	`BlankDA12`	(→) Clears display areas DA1 and DA2
261C0	`CLCD10`	(→) Clears status and stack areas.
261C5	`CLEARLCD`	(→) Clears whole display.
2EF05	`DOCLLCD`	(→) Like user word <REF>CLLCD.

4.11.6 Annunciator and Modes Control

2613E	`SetLeftAnn`	(→) Sets left-shift annunciator.
2603A	`ClrLeftAnn`	(→) Clears left-shift annunciator.
26148	`SetRightAnn`	(→) Sets right-shift annunciator.

2603F	`ClrRightAnn`	(→) Clears right-shift annunciator.
26139	`SetAlphaAnn`	(→) Sets alpha annunciator.
26035	`ClrAlphaAnn`	(→) Clears alpha annunciator.
25EE9	`LockAlpha`	(→) Sets alpha mode, annunciators, etc.
25F08	`UnLockAlpha`	(→) Clears alpha mode, annunciators, etc.
2649F	`(ClrBusyAnn)`	(→) Clears the busy annunciator.
264A4	`(ClrI/OAnn)`	(→)
26143	`SetPrgmEntry`	(→) Sets program-entry mode.
264F4	`(ClrPrgmEntry)`	(→) Clears program-entry mode.
2610C	`PrgmEntry?`	(→ `flag`) Is program-entry mode set?
25726	`(LOWERCASE?)`	(→ `flag`) Is the flag for lowercase letter entry set?
2572B	`(SETLOWERCASE)`	(→) Set the flag for lowercase letter entry.
25730	`(CLRLOWERCASE)`	(→) Clear the flag for lowercase letter entry.
25738	`(TOGLOWERCASE)`	(→) Toggle the flag for lowercase letter entry.
25EBE	`Do1st/2nd+:`	(→ `:: <ob1> ; (PRG mode)`) (→ `:: <ob2> <rest> ; (no PRG mode)`) If in program mode, executes the next object after it. If not in program mode, executes the rest of the stream starting at the second object after it.
25719	`SetAlgEntry`	(→) Sets algebraic-entry mode.
2571E	`ClrAlgEntry`	(→) Clears algebraic-entry mode.
256EA	`AlgEntry?`	(→ `flag`) Is algebraic-entry mode set?
25EDF	`ImmedEntry?`	(→ `flag`) Returns `TRUE` if immediate-entry mode (program and algebraic-entry modes cleared).
25E74	`?ClrAlg`	(→) Clears AlgEntry mode if set.
25E75	`?ClrAlgSetPr`	(→) Clears AlgEntry mode if set and sets ProgramEntry mode.

4.11.7 Window Coordinates

2F384	`TOP8`	`( → HBgrob #x1 #y #x1+131 #y1+8 )` Returns coordinates of first status line.
2F36C	`Rows8-15`	`( → HBgrob #x1 #y1+8 #x1+131 #y1+16 )` Returns coordinates of second status line.
2F383	`TOP16`	`( → HBgrob #x1 #y1 #x1+131 #y1+16 )` Returns coordinates of status area.
2617F	`WINDOWCORNER`	`( → #y #x )` Gets coordinates of corner of window. Note the order of #x and #y.
2EED6	`HBUFF_X_Y`	`( → HBgrob #x #y )` Returns current grob and window coordinates.
2F352	`LEFTCOL`	`( → #x )` Gets x-coordinate of left column.
2F36B	`RIGHTCOL`	`( → #x )` Gets x-coordinate of right column.
2F385	`TOPROW`	`( → #y )` Gets y-coordinate of top row.
2F31D	`BOTROW`	`( → #y )` Gets y-coordinate of bottom row.
26198	`WINDOWXY`	`( #y #x → )` Sets corner coordinates. The name really should be WINDOWYX

4.11.8 Scrolling the Display

26193	`WINDOWUP`	`( → )` Moves display one pixel up.
26184	`WINDOWDOWN`	`( → )` Moves display one pixel down.
26189	`WINDOWLEFT`	`( → )` Moves display one pixel left.
2618E	`WINDOWRIGHT`	`( → )` Moves display one pixel right.
2F370	`SCROLLUP`	`( → )` Moves display one pixel up, checks for corresponding key being pressed.
2F36D	`SCROLLDOWN`	`( → )` Moves display one pixel down, checks for corresponding key being pressed.
2F36E	`SCROLLLEFT`	`( → )` Moves display one pixel left, checks for corresponding key being pressed.

2F36F	SCROLLRIGHT	(→) Moves display one pixel right, checks for corresponding key being pressed.
2F34A	JUMPTOP	(→) Jumps to top of display.
2F347	JUMPBOT	(→) Jumps to bottom of display.
2F348	JUMPLEFT	(→) Jumps to left of display.
2F349	JUMPRIGHT	(→) Jumps to right of display.
2F38D	WINDOWTOP?	(→ flag) Is window at the top?
2F38A	WINDOWBOT?	(→ flag) Is window at the bottom?
2F38B	WINDOWLEFT?	(→ flag) Is window at the left?
2F38C	WINDOWRIGHT?	(→ flag) Is window at the right?

4.11.9 Displaying Objects

2F21D	ViewObject	(ob →)
2F21E	ViewStrObject	(flag $ → F) Flag decides if it should be possible to toggle TEXT/GRAPH.
2F21F	ViewGrobObject	(flag grob → F) Flag decides if it should be possible to toggle TEXT/GRAPH.
25F12	sstDISP	(ob →) Displays ob in status line. Used for single stepping during debugging.
0C1007	^SCROLLext	(grob →) Launches PICT environment.
2EF61	WINDOW#	(#x #y →) Internal `PVIEW`, displays PICT starting at the given coordinates.

4.11.10 Displaying Text

25EB4	DODISP	(ob %row →) Displays any object in specified row.
25FB8	DISPROW1	($ →) aka: `DISP@01`, `BIGDISPROW1`

25EAB	DISPROW1*	($ →) Displays relative to window corner.
0C8002	(^DISPROW1_plus)	($ →) Only useful on ROM 1.22-2.0! Deprecated since ROM 2.0! Write text to the first line of the extended header on the 49G+ (pixel rows 1-8). This messes up the second row, so this entry should only be used together with DISPROW2_plus. A good way to automatically do the right thing is DISPSTATUS2. First available in ROM 1.22.
25FBD	DISPROW2	($ →) aka: DISP@09, BIGDISPROW2
25EAC	DISPROW2*	($ →) Displays relative to window corner.
0C9002	(^DISPROW2_plus)	($ →) Only useful in ROM 1.22-2.0! Deprecated since ROM 2.0! Write text to the second line of the extended header on the 49G+ (pixel rows 9-16). Should be used together with DISPROW1_plus. First available in ROM 1.22.
25FC2	DISPROW3	($ →) aka: DISP@17, BIGDISPROW3
25FC7	DISPROW4	($ →) aka: DISP@25, BIGDISPROW4
25FCC	DISPROW5	($ →)
261F7	DISPROW6	($ →)
25FD1	DISPROW7	($ →)
25FD6	DISPROW8	($ →) May not be possible depending on the size of the font and whether the menu is on or off.
25FDB	DISPROW9	($ →) May not be possible depending on the size of the font and whether the menu is on or off.
25FE0	DISPROW10	($ →) May not be possible depending on the size of the font and whether the menu is on or off.
25FB3	DISPN	($ #row →) aka: BIGDISPN
25EBC	Disp5x7	($ #start #max →) Displays string on multiple lines, starting at #start and no using more than #max rows. New lines must be manually specified. Segments longer than 22 characters are truncated and appended with "...".
2F038	(Save16)	(→ grob) Returns top 16 rows.

2F3CF	(Save16Patch)	(→ grob) Get the Header area as a grob. On the 49G+, this gets the extra 16 lines of the screen. On a machine with small screen (48gII,49G), this is equivalent to `Save16`. Also, starting from ROM 2.0, this again just calls `Save16`. First available in ROM 1.22.
2F3B6	(Restore16)	(grob →) Restores top 16 rows.
2F3D0	(Rest16Patch)	(grob →) Display grob in the top 16 rows of the display. This works with the extended screen on the 49G+ - on a smaller screen, it is equivalent to `Restore16`. Also, starting from ROM 2.0, this again just calls `Restore16`. First available in ROM 1.22.
25EAD	DISPSTATUS2	($ →) Displays message in status area using two lines.
38C00	(DoPrompt)	($ →) `DISPSTATUS2` and freeze status area.
2EEFF	DispCoord1	($ →) 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 above the menu.
25FEA	DISPLASTROWBUT1	($ →) Displays $ in the last stack display row. If menu is turned on it can cover displayed text.

4.11.11 Messages and Boxes

25ED4	FlashMsg	($ →) Displays message in status area, then restores it to normal.
2EE61	FlashWarning	($ →) Displays message in a message box and beeps. Waits for OK to be pressed.

2F1A5	`AskQuestion`	`( $ → flag )` Use the string to ask the user a question with yes/no in a choose box. If you prefer a YES/NO menu, this can be implemented like this, using ~DoMsgBox:

```
::
  15 10     (BINTs, don't know what they
do)
  MINUSONE  (could also be a grob)
  '
  ::
    NoExitAction
    { NullMenuKey NullMenuKey
      NullMenuKey NullMenuKey
      { "NO"  :: TakeOver FALSETRUE 2PUTLAM
; }
      { "YES" :: TakeOver TrueTrue  2PUTLAM
; }
     }
  ;
  ROMPTR2 ~DoMsgBox
;
```

02E002	`^DoAlert`	`( $ → )` Displays alert messagebox, a message box with a little alert grob in the upper left corner.
2EE60	`DoWarning`	`( $ → )` Displays message, beeps and freezes status area.
007002	`^Ck&DoMsgBox`	`( $ → )` Displays a message box and waits for the user to press OK.
0000B1	`~DoMsgBox`	`( $ #x #y grob menu → T )` Displays a message box with a grob in the upper left corner and the specified menu. If no grob is desired, use `MINUSONE`. The meaning of #x and #y is unclear - it seems that any BINT will do.
0040B1	`~MsgBoxMenu`	`( → {} )` The messsage box menu, with just the OK key.

4.11.12 Fonts

2621A	`FONT>`	`( → font )` Recalls system font.
2625B	`MINIFONT>`	`( → minifont )` Recalls the current minifont.
25F15	`>FONT`	`( font → )` Sets system font.

2620B	>MINIFONT	(minifont →) Sets the current minifont.
26288	StackLineHeight	(→ #) Returns height of text grob minus size of header and menu.
26242	GetFontStkHeight	(→ #) Returns stack font height (used for display stack rows). aka: StackFontHeight
26238	GetFontCmdHeight	(→ #) Returns command line font height (used for editing objects).
2623D	GetFontHeight	(→ #) Returns system font height.
26210	CHECK_SCAN_FONT	
026FE	DOMINIFONT	
06F004	^FontBrowser	(→ font T) Uses the File Manager to search for fonts.
2621F	FSCANFONT	
26256	INITMKFONT	
26904	Init_MetaKernelFont	
2627E	SCANFONT	

4.12 Graphics

4.12.1 Built-in Grobs

27AA3	(NULLPAINT)	(→ grob) 0x0 Null grob
27D3F	CROSSGROB	(→ grob) 5x5 Cross cursor ("+")
27D5D	MARKGROB	(→ grob) 5x5 Mark symbol ("x")
27D7B	(NullMenuLbl)	21x8 normal menu key
2E25C	(InvLabelGrob)	21x8 inverse menu key
279F6	(StdBaseLabel)	21x8 inverted nomal menu key grob
2E198	(BoxLabelGrobInv)	21x8 inverted box label grob
2E1FA	(DirLabelGrobInv)	21x8 inverted DIR label grob
0860B0	~grobAlertIcon	9x9 Alert grob
0870B0	~grobCheckKey	21x8 Check Key menu grob A tickmark and "CHK" in a menu grob.

4.12.2 Dimensions

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

4.12.3 Grob Handling

2607B	`GROB!`	`( grob1 grob2 #x #y → )` Stores grob1 into grob2. Bang type.
2EFDB	`(GROB+)`	`( grob1 grob2 → grob )` Combines two grobs using bitwise OR. Errors when grobs have different sizes.
2F342	`GROB+#`	`( flag grob1 grob2 #x #y → grob' )` Inserts grob2 into the specified position of grob1, using OR (if flag is `TRUE`) or XOR (if flag is `FALSE`). Does all necessary checks first.
26080	`GROB!ZERO`	`( grob #x1 #y1 #x2 #y2 → 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 → grob' )` Returns specified portion of grob.
25F0E	`XYGROBDISP`	`( #x #y grob → )` Stores grob in HARDBUFF with upper left corner at (#x,#y). HARDBUFF is expanded if necessary.
25ED8	`GROB>GDISP`	`( grob → )` Stores new graph grob.
260B2	`MAKEGROB`	`( #height #width → grob )` Creates a blank grob.
2F0DB	`MAKEPICT#`	`( #w #h → )` Creates blank graph grob. Minimum size is 131x64. Smaller grobs will be automatically resized.
2609E	`INVGROB`	`( grob → grob' )` Inverts grob data bits. Bang type.

260E4	PIXON	(#x #y →) Sets pixel in text grob.
260DF	PIXOFF	(#x #y →) Clears pixel in text grob.
260EE	PIXON?	(#x #y → flag) Is pixel in text grob on?
260DA	PIXON3	(#x #y →) Sets pixel in graph grob.
260D5	PIXOFF3	(#x #y →) Clears pixel in graph grob.
260E9	PIXON?3	(#x #y → flag) Is pixel in graph grob on?
280C1	ORDERXY#	(#x1 #y1 #x2 #y2 → #x1' #y1' #x2' #y2') Orders the bints to be appropriate for defining a rectangle in a grob. Swaps #x1 and #x2 if #x2<#x1. Swaps #y1 and #y2 if #y2<#y1.
280F8	ORDERXY%	(%x1 %y1 %x2 %y2 → %x1' %y1' %x2' %y2') ORDERXY# with real numbers.
2EF9F	LINEON	(#x1 #y1 #x2 #y2 →) Draws a line in text grob.
2EFA0	LINEOFF	(#x1 #y1 #x2 #y2 →) Clears a line in text grob.
2EFA1	TOGLINE	(#x1 #y1 #x2 #y2 →) 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 →) Draws a line in graph grob. x1<x2 is not required.
2EFA3	LINEOFF3	(#x1 #y1 #x2 #y2 →) Clears a line in graph grob.
2EFA4	TOGLINE3	(#x1 #y1 #x2 #y2 →) Toggles a line in graph grob.
2F382	TOGGLELINE#3	(#x1 #y1 #x2 #y2 →) Toggles line in graph grob. x1<x2 is not required.
2F32C	DRAWBOX#	(#x1 #y1 #x2 #y2 →) Draws rectangle in graph grob.
2EF03	DOLCD>	(→ grob) Returns current display.
2EF04	DO>LCD	(grob →) Grob to display.
0BF007	^GROBADDext	(grob2 grob1 → grob) Vertical grob addition. grob2 will be above grob1.

4.12.4 Greyscale Graphics

25592	`SubRepl`	(`grb1 grb2 #x1 #y1 #x2 #y2 #W #H` → `grb1'`) Replace a part of grb1 with a part of grb2 in REPLACE mode.
25597	`SubGor`	(`grb1 grb2 #x1 #y1 #x2 #y2 #W #H` → `grb1'`) Replace a part of grb1 with a part of grb2 in OR mode.
2559C	`SubGxor`	(`grb1 grb2 #x1 #y1 #x2 #y2 #W #H` → `grb1'`) Replace a part of grb1 with a part of rgb2 in XOR mode.
25565	`LineW`	(`grb #x1 #y1 #x2 #y2` → `grb'`) Draw a white line.
2556F	`LineG1`	(`grb #x1 #y1 #x2 #y2` → `grb'`) Draw a light grey line.
25574	`LineG2`	(`grb #x1 #y1 #x2 #y2` → `grb'`) Draw a dark grey line.
2556A	`LineB`	(`grb #x1 #y1 #x2 #y2` → `grb'`) Draw a black line.
25579	`LineXor`	(`grb #x1 #y1 #x2 #y2` → `grb'`) XOR a line.
2F218	`CircleW`	(`grb #Cx #Cy #r` → `grb'`) Draw a white circle.
2F216	`CircleG1`	(`grb #Cx #Cy #r` → `grb'`) Draw a light grey circle.
2F217	`CircleG2`	(`grb #Cx #Cy #r` → `grb'`) Draw a dark grey circle.
2F215	`CircleB`	(`grb #Cx #Cy #r` → `grb'`) Draw a black circle
2F219	`CircleXor`	(`grb #Cx #Cy #r` → `grb'`) XOR a circle.
2557E	`Sub`	(`grb #x1 #y1 #x2 #y2` → `grb' flag`) Get a part of a grob.
25583	`Repl`	(`grb1 grb2 #x #y` → `grb1'`) Copy grb2 into grb1 in REPLACE mode.
25588	`Gor`	(`grb1 grb2 #x #y` → `grb1'`) Copy grb2 into grb1 in OR mode.
2558D	`Gxor`	(`grb1 grb2 #x #y` → `grb1'`) Copy grb2 into grb1 in XOR mode.
255A1	`Grey?`	(`grob` → `flag`) Is grob a Greyscale Grob?
255B0	`ScrollVGrob`	(`grb #W #X #Yd #Ys #h` → `grb'`) Scroll up and down a portion of a graphical object.
255BA	`PixonW`	(`grb #x #y` → `grb'`) Make a pixel white.
255C4	`PixonG1`	(`grb #x #y` → `grb'`) Make a pixel light grey.
255C9	`PixonG2`	(`grb #x #y` → `grb'`) Make a pixel dark grey.

255BF	PixonB	(grb #x #y → grb') Make a pixel black.
255CE	PixonXor	(grb #x #y → grb') Apply XOR to a pixel.
255D3	FBoxW	(grb #x1 #y1 #x2 #y2 → grb') Make a white filled rectangle.
255D3	FBoxG1	(grb #x1 #y1 #x2 #y2 → grb') Make a light grey filled rectangle.
255D8	FBoxG2	(grb #x1 #y1 #x2 #y2 → grb') Make a dark grey filled rectangle.
255DD	FBoxB	(grb #x1 #y1 #x2 #y2 → grb') Make a black filled rectangle.
255E2	FBoxXor	(grb #x1 #y1 #x2 #y2 → grb') Apply XOR to a filled rectangle.
255E7	LBoxW	(grb #x1 #y1 #x2 #y2 → grb') Draw a white rectangle.
255EC	LBoxG1	(grb #x1 #y1 #x2 #y2 → grb') Draw a light grey rectangle.
255F1	LBoxG2	(grb #x1 #y1 #x2 #y2 → grb') Draw a dark grey rectangle.
255F6	LBoxB	(grb #x1 #y1 #x2 #y2 → grb') Draw a black rectangle.
255FB	LBoxXor	(grb #x1 #y1 #x2 #y2 → grb') Apply XOR to a rectangle.
2F21B	ToGray	(grb → grb'/grb) Convert a B&W grob to Greyscale.
2F21A	Dither	(grb → grb'/grb) Convert a greyscale grob to B&W
255B5	Distance	(#Δx #Δy → #SQRT(Δx^2+Δy^2)) Compute the distance between two points.

4.12.5 Creating Menu Label Grobs

2E166	MakeStdLabel	($ → grob) Makes standard menu label.
2E189	MakeBoxLabel	($ → grob) Makes label with a box.
2E1EB	MakeDirLabel	($ → grob) Makes directory label.
2E139	(MakeDir/StdLabel)	(ob → grob) Makes directory label if ob is a directory (rrp), otherwise calls MakeStdLabel.
2E24D	MakeInvLabel	($ → grob) Makes inverse label.

25E7F	`Box/StdLabel`	`( $ flag` → `grob )` If `TRUE` makes box label, otherwise makes standard label.
25F01	`Std/BoxLabel`	`( $ flag` → `grob )` If `TRUE` makes standard label, otherwise makes box label.

25E80	`Box/StdLbl:`	`(` → `grob )`
		Does `Box/StdLabel` with the next two objects from the stream.
		`Usage: :: Box/StdLbl: $ <test> ;`

2E0D5	`Grob>Menu`	`( #col grob` → `)` Displays grob as menu label.
2E0F3	`Str>Menu`	`( #col $` → `)` Displays string as menu label.
2E11B	`Id>Menu`	`( #col id` → `)` Displays id as menu label.
2E107	`Seco>Menu`	`( #col ::` → `)` Does `EVAL` then `DoLabel`.
25886	`DoLabel`	`( #col ob` → `)` If ob is of one of the supported types, displays a menu label. If not, generates a "Bad Argument Type" error.

2E094	`(StdLabelDef)`	`( #col grob` → `)`
		`( #col $` → `)`
		`( #col id` → `)`
		`( #col ::` → `)`
		Works by dispatching the object type.

4.12.6 Converting Strings to Grobs

25F7C	`$>GROB`	`( $` → `grob )` Makes grob of the string using the system font. Linefeed does *not* make new line.
25F86	`$>GROBCR`	`( $` → `grob )` Makes grob of the string using the system font. Linefeed *does* make new line.
25F81	`$>grob`	`( $` → `grob )` Makes grob of the string using the minifont. Linefeed does *not* make new line.
25F8B	`$>grobCR`	`( $` → `grob )` Makes grob of the string using the minifont. Linefeed *does* make new line.
05F0B3	`(~$>grobOrGROB)`	`( $` → `grob )` Converts string to a grob using either the current font or the minifont, depending on system flag 90.

25F24	`RIGHT$3x6`	`( $ #n → flag grob )` Transforms string into grob (using the minifont), then takes all characters starting after column #n. flag is `FALSE` if #n is greater than the width of the grob. In this case, the whole grob is returned.
25FEF	`CENTER$3x5`	`( grob #x #y $ #w → grob' )` Creates grob from string (using the minifont) and embeds it at specified position (#x, #y). The grob is centered around #x and the to is put at #y. #w represents the maximum width of the grob created. If the text is wider, it is truncated. Bangtype.
2E2AA	`MakeLabel`	`( $ #w #x grob → grob' )` Inserts $ into grob using `CENTER$3x5` with y=5.
02F002	`(^MkTitle)`	`( $ → grob )` Create a title grob. This is the text embedded in a dot matrix pattern, as used for Choose boxes etc. The size of the grob is 131x7.
25FF9	`LEFT$3x5`	`( grob #x #y $ #w → grob' )` Like <REF>CENTER$3x5, but the left corner of the text is positioned at #x.
26071	`ERASE&LEFT$3x5`	`( grob #x #y $ #w → grob' )` Like <REF>LEFT$3x5, but erase background first.
26008	`LEFT$3x5Arrow`	`( grob #x #y $ #w → grob' )` Like <REF>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 <REF>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 <REF>LEFT$3x5CR, but show truncation with arrows.
25FF4	`CENTER$5x7`	`( grob #x #y $ #w → grob' )` Same as `CENTER$3x5`, but using system font.
25FFE	`LEFT$5x7`	`( grob #x #y $ #w → grob' )` Like <REF>CENTER$5x7, but the left corner of the text is positioned at #x.
2606C	`ERASE&LEFT$5x7`	`( grob #x #y $ #w → grob' )` Like <REF>LEFT$5x7, but erase background first.
26003	`LEFT$5x7Arrow`	`( grob #x #y $ #w → grob' )` Like <REF>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 → grob') Like <REF>LEFT$5x7, but interpret newlines.
2600D	LEFT$5x7CRArrow	(grob #x #y $ #w → grob') Like <REF>LEFT$5x7CR, but show truncation with arrows.

4.12.7 Creating Grobs from Other Objects

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

4.13 Plotting

27AE9	('IDPAR)	(→ id) Puts ID PPAR unevaluated on the stack. -- <REF>TEXT:Reserved|PPAR
2799A	(ID_PPAR)	ID PPAR
2F162	CHECKPICT	(→) Checks size of GBUFF. If it is smaller than 131x64 sets GBUFF back to its default size (131x64).
2EF06	CKPICT	(xPICT →) Checks for user word xPICT on level 1. Errors (SETTYPEERR) if there is another object.
2F258	PICTRCL	(xPICT → grob) Does CKPICT, then recalls GBUFF and does TOTEMPOB.
2F355	MAKEPVARS	(→ {}) Creates the default PPAR variable in the current directory and returns its value. -- <REF>TEXT:Reserved|PPAR

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. -- <REF>TEXT:Reserved|PPAR
2F33D	GETPARAM	`( # → ob )` Extracts the #th item from PPAR. No error checking! -- <REF>TEXT:Reserved|PPAR
2F0FF	GETXMIN	`( → % )` Recalls XMIN from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved|PPAR
2F366	PUTXMIN	`( % → )` Sets a new value for XMIN. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2F0FE	GETXMAX	`( → % )` Recalls XMAX from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved|PPAR
2F365	PUTXMAX	`( % → )` Sets a new value for XMAX. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2F100	GETYMIN	`( → % )` Recalls YMIN from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved|PPAR
2F368	PUTYMIN	`( % → )` Sets a new value for YMIN. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2F10E	GETYMAX	`( → % )` Recalls YMAX from the PPAR list if existent. If not, the default PPAR is created in the current directory. -- <REF>TEXT:Reserved|PPAR

2F367	`PUTYMAX`	(% →) Sets a new value for YMAX. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2F107	`GETPMIN&MAX`	(→ C% C%) -- Returns PMIN and PMAX. -- <REF>TEXT:Reserved|PPAR
2EEF2	`PUTINDEP`	(ID →) Internal `xINDEP` if the arg is an ID.
2EEF3	`PUTINDEPLIST`	({} →) Internal `xINDEP` if the arg is a list.
2F0E8	`INDEPVAR`	(→ id) Recalls the independent variable. If a list, extract first element. `:: GETINDEP DUPTYPELIST? ?CARCOMP ;`
2F106	`GETINDEP`	(→ id) (→ {}) Recalls the independent variable field in PPAR. -- <REF>TEXT:Reserved|PPAR
2EEF5	`GETPTYPE`	(→ name) Recalls the plot type using `GETPARAM`. -- <REF>TEXT:Reserved|PPAR
2EEF6	`PUTPTYPE`	(name →) Sets a new plot type. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2F10D	`GETRES`	(→ %) Recalls the plot resolution using `GETPARAM`. -- <REF>TEXT:Reserved|PPAR
2EEF4	`PUTRES`	(% →) Set new plot resolution. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2F33E	`GETSCALE`	(→ % %') Recalls the plot scale parameters. -- <REF>TEXT:Reserved|PPAR

2EEF1	PUTSCALE	(% %' →) Set new plot scale. PPAR is created if necessary. -- <REF>TEXT:Reserved|PPAR
2EEEF	AUTOSCALE	(→) Internal AUTO.
2EF60	DOGRAPHIC	(→) Sets the scroll mode of PICTURE and is essentially the same as { } PVIEW.
2F109	GETXPOS	
2F007	getxpos	
2F340	GETYPOS	
2F008	getypos	
25ECF	EQUATION	(→ ob T) (→ F) Recall the current equation, stored in the 'EQ' variable, and TRUE. If there is no 'EQ' variable on the path, just returns FALSE.
2F339	GetEqN	(#n → ob T) (#n → NULL$ F) Get the #nth equation, if EQ is a list of equations.
25EB5	DORCLE	(→ ob) Recalls the contents of the EQ variable, errors if it does not exist.
25EB6	DOSTOE	(ob →) Stores ob into the variable EQ.
2F297	XEQPURGEPICT	(xPICT →) If object in level one is xPICT, erases the graphic display. Otherwise, errors.
00113	CRER	
2F328	CROSSMARKON	
2EEFA	CROSS_HAIRS	
2EEFB	CROSS_OFF	
2F105	GDISPCENTER	(→) Moves to center of graphics display
2F10A	GetRes	
2EEF8	HSCALE	
2EEF7	VSCALE	
2F35E	PLOTERR	
2F35F	PlotOneMore?	
2F0C5	PLOTPREP	

2EF01	DOPX>C	({ hxs hxs' } $\rightarrow$ C%) Converts a list of two hex strings into a complex number. Used for plotting coordinates. Inverse operation is DOC>PX.
2EF02	DOC>PX	(C% $\rightarrow$ { hxs hxs' }) Converts a complex coordinate point into list of two HXS numbers. Inverse operation is DOPX>C.

5 The HP49G CAS

5.1 Type Checking and Conversion

157006	^SYMBINCOMP	(symb → ob1 .. obN #n) (ob → ob #1) ({} → {} #1) Explodes symbolic object into meta. Other objects are converted into one-object metas by pushing #1 into the stack.
12A006	^2SYMBINCOMP	(ob1 ob2 → meta1 meta2) Does ^SYMBINCOMP for 2 objects.
4D7006	^VXXLext	(ob Lvar → Q) Converts object to internal form. The object can be a symbolic, a symbolic vector or a symbolic matrix. If the conversion was not successfull, vxxxlflag is cleared.
400006	^R2SYM	(lvar ob → ob) Back conversion of a scalar object.
4D8006	^METALISTVXXL	(Meta → Meta) Conversion of all elements of a meta object with respect to the variables in LAM1.
4D9006	^VXXLFext	(n/d → Z1/Z2) Conversion of a fraction which does not depend on any variables.
4DA006	^VXXL1ext	(n → Z) Conversion of an object which does not depend on any variables.
4DB006	^VXXL0	(ob → Q) Conversion of object with respect to Lvar in LAM1.
4DC006	^VXXL2NR	(Meta → Q) Converts symbolic meta to internal form (LAM1=Lvar). Set nocareflag to avoid square root problems.
4DD006	^VXXL2	(Meta → Q) Converts symbolic meta to internal form (LAM1=Lvar).
167006	^TYPEIRRQ?	(ob → flag) Is ob an irrquad?
168006	^DTYPEIRRQ?	(ob → ob flag) DUP, then ^TYPEIRRQ?.
177006	^CKMATRIXELEM	(ob → ob) Checks that ob is a valid internal matrix element. Look for CK[]NCK for user matrix element.

18F006	^CKFPOLYext	(ob → ob) Errors if list contains secondaries or empty lists.
190006	^CK2FPOLY	(ob ob → ob ob) Does CKFPOLYext on two objects.
19E006	^CLEANIDLAM	(ob → ob) Suppresses SYMB if not needed.

5.2 Integers

5.2.1 Built-in Integers

2733F	(Z-9)	-9
2734B	(Z-8)	-8
27357	(Z-7)	-7
27363	(Z-6)	-6
2736F	(Z-5)	-5
2737B	(Z-4)	-4
27387	(Z-3)	-3
27393	(Z-2)	-2
2739F	(Z-1)	-1
273AB	(Z0)	0
273B6	(Z1)	1
273C2	(Z2)	2
273CE	(Z3)	3
273DA	(Z4)	4
273E6	(Z5)	5
273F2	(Z6)	6
273FE	(Z7)	7
2740A	(Z8)	8
27416	(Z9)	9
27422	(Z10)	10
2742F	(Z12)	12
2743C	(Z24)	24
27449	(Z100)	100
274A9	(ZINT1_0)	(→ 1 0) Pushes the ZINTS 1 and 0.
2E0006	^DROPZ0	(ob → z0)
2DF006	^DROPZ1	(ob → z1)
392006	^2DROPZ0	(2 1 → z0)

3B3006	^NDROPZ0	(obn...ob1 #n → z0) Replaces meta with Z0.
3B4006	^NDROPZ1	(obn...ob1 #n → z1) Replaces meta with Z1.
274A4	(INTERNALiX)	{ 1 0 0 } List with the three ZINTS 1, 0, and 0.
27C70	(Z0ONE)	(→ ZINT 0 #1)

5.2.2 Conversion Functions

0EE006	^#>Z	(# → Z) Converts bint to zint.
0F5006	^R>Z	(% → z) Converts real to zint. Do not call this entry if the number if not an integer.
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 → Z / Error) Checks if HXS is a proper zint number and trims it.
0F2006	^S>Z	($ → z) Converts decimal in a string into a zint.
0F3006	^S>Z?	($ → z T) ($ → $ F) If possible, converts string into a zint and returns TRUE. If not, keeps the original string and returns FALSE.
184006	^CK1Z	($/#/hxs → Z) CHecks for an integer. Converts strings, bints or hxs's to zints. Errors for other object types.
185006	^CK2Z	(ob ob' → Z Z') Like <REF>^CK1Z, but for two objects.
186006	^CK3Z	(ob ob' ob'' → Z Z' Z'') Like <REF>^CK1Z, but for three objects.
202006	^CK&CONVINT	(symb → zint) (symb → :: zint zint' ;) Check that a sym is a zint or Gauss integer, convert it.
203006	^CK&CONV2INT	(symb symb' → zint zint') (symb symb' → :: zint1 zint2 ; :: zint3 zint4 ;) Check that 2 sym are zint or Gauss integer, convert them.
205006	^CONVBACKINT	(zint|c → symb)

204006	^CONVBACK2INT	(zint|c zint|c → symb symb)
0F4006	^Z>ZH	(Z → Z') Converts decimal Z to hex Z.
18E006	^Z2Sext	(Z → '$Z') Converts Z to string number. The number is embedded in a symbolic to enable using it in algebraics.

5.2.3 General Integer Operations

101006	^ZTrim	(Z → 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 → |Z|) Takes the absolute value of Z. If Z is already positive then does nothing. Else duplicate object and change sign.
50B006	^ZABS	(Z → Z') Absolute value.
590006	(^ZSQ)	(Z → Z') Computes the square of a zint.
0E0006	^ZSQRT	(Z → Z' 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 → Z') Make Z modulo N.
0DD006	^ZMod	(Z1 Z2 → Z')
105006	^ZNMax	(Z1 Z2 → NormMax[Z1,Z2]) Returns the integer with the greatest absolute value. (Returns Z1 if |Z1|≥|Z2|; returns Z2 if |Z1|<|Z2|).
106006	^ZNMin	(Z1 Z2 → NormMin[Z1,Z2]) Returns the integer with the smallest absolute value. (Returns Z1 if |Z1|≤|Z2|; returns Z2 if |Z1|>|Z2|).
10D006	^ZBits	(Z → Z #bits) Calculates number of bits used in Z.
10E006	^ZBit?	(Z #bit → Z flag) Tests if a bit in Z is set. Count starts from zero, as opposed to ZBits.
2B7006	^ZGCDext	(Z2 Z1 → Z) Integer GCD.

2B8006	`^ZGcd`	(Z2 Z1 → Z) This is the same entry as `ZGCDext`.
20A006	`^IEGCD`	Internal EGCD for integers.
3D6006	`^IEGCDext`	(a b → d u v) Bezout for integers. d=au+bv=gcd(a,b).
3D9006	`^INEGCD`	(a b → d u v)
3DA006	`^EGCDSWAP`	
3DB006	`^EGCDNEWG`	
07C007	`^#FACT`	(# → 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 → z!) Factorial for long integers.
215006	`^PA2B2`	(z/% → a+bi) Internal `PA2B2`.

5.2.4 Integer Factorization and Prime Numbers

0C9006	`^ZFactor`	(Zs → Lf) Factors signed long integer.
0CA006	`^NFactor`	(z → {}) Factors positive long integer.
0CB006	`^NFactorSpc`	(z → {}) Semi-factors positive long integer. This is regular factorization with an extra 'hopeless?' test.
0CD006	`^SFactor`	(S → 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.
0CE006	`^SPollard`	(S → 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, although the starting value is different.

`0CF006`	`^BFactor`	`( N → 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.
`0D0006`	`^BrentPow`	`( Za Z1 Z2 Zn #k → 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≠Z2 makes no sense for k≠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?`	`( Z → flag )` 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.
`0D2006`	`^ZIsPrime?`	`( Z → flag )` Probabilistic primality test for a positive integer.
`0D3006`	`^SIsPrime?`	`( S → flag )` Tests if positive short Z is prime. M-R test fails for integers ≤ 3, so we just test them separately at the start. For convenience lets define 0 and 1 to be primes also.
`0D4006`	`^BIsPrime?`	`( S → flag )` Test if positive long Z is prime.
`0D5006`	`^BRabin`	`( Z #base → Z flag )` Performs Miller-Rabin test for long positive integer. Returns `TRUE` if base witnesses composite. Else returns `FALSE`.
`0D6006`	`^ZTrialDiv2`	`( Z → 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 → flag )` Trial division primality test for a positive integer. works for $Z \geq 3$ (return false for Z=2).

0D8006	^ZTrialDiv	(Z → 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 2000^2 will also be automatically detected.
0C7006	^Prime+	(Z → Z') Returns next prime (Z' > Z).
0C8006	^Prime-	(Z → Z') Returns previous prime (Z' < Z).

5.2.5 Gaussian Integers

274A9	(Z1Z0)	(1,0)
27516	(Z0Z1)	(0,1)
2754B	(Z-1Z0)	(-1,0)
2756C	(Z1Z1)	(1,1)
114007	^TYPEGAUSSINT?	(ob → flag) Checks if ob is Gaussian integer. First available in ROM 1.11.
115007	^DTYPEGAUSSINT?	(ob → ob flag) Checks if ob is Gaussian integer. First available in ROM 1.11.
116007	^DUPTYPEGAUSSINT?	(ob → ob flag) Checks if ob is Gaussian integer. First available in ROM 1.11.
187006	^CK1Cext	(ob → flag) Checks if object is integer or Gaussian integer.
15D006	^CXRIext	(C → Zre Zim) Returns real and imaginary part of Gaussian integer.
2B5006	^CGCDext	(C2 C1 → C) GCD for Gauss integers.
4D5006	^CSQFFext	(C → { factor1 mult1 ... 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>> ; → { fact1 mult1 ... factn multn }) Factorization of irrquads and Gauss integers.

4D6006	^SUMSQRext	(Z → 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 → |C|^2) Square modulus of a Gauss integer.

5.2.6 Integer Tests

265C1	Z=	(Z Z' → flag)
265C6	Z<>	(Z Z' → flag)
265BC	Z<	(Z Z' → flag)
265D0	Z<=	(Z Z' → flag)
265B7	Z>	(Z Z' → flag)
265CB	Z>=	(Z Z' → flag)
0F8006	^QIsZero?	(Q → flag) Tests if Q is zero. Assumes list contains only lists or hexes!.
0F7006	^DupQIsZero?	(Q → Q flag) Duplicates Q and tests if Q is zero. Assumes list contains only lists or hexes!.
0FA006	^ZIsOne?	(Z → flag) Tests if Z is Z1.
0F9006	^DupZIsOne?	(Z → Z flag) Duplicates Z, and returns TRUE if Z is 1.
109006	^DupZIsTwo?	(Z → Z flag) Returns TRUE if Z is 2.
0FC006	^ZIsNeg?	(Z → flag) Tests if Z is negative.
0FB006	^DupZIsNeg?	(Z → Z flag) Tests if Z is negative.
10A006	^DupZIsEven?	(Z → Z flag) Tests if Z is even.
107006	^ZNLT?	(Z1 Z2 → flag) TRUE if |Z1|<|Z2|.
19A006	^OBJINT?	(z/% → z flag) Tests if Obj is an integer.
19B006	^OBJPOSINT?	(z/% → 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.

199006	^METAPOSINT?	(Meta → Meta flag) Tests if Meta is a positive integer smaller than Zsmall.
0CC006	^DupTypeS?	(Z → Z flag) Tests if Z is short ($\leq$ 64 bits).

5.3 Matrix Operations

5.3.1 Creating and Redimensioning Matrices

371006	^MATIDN	(M/z/% → M') Creates identity matrix.
372006	^MATCON	(M ob → [ob]) Creates constant matrix from matrix.
373006	^MAKEARRY	({#el} ob → []) ({#rows #cols} ob → [[]]) Creates constant matrix/array, initializing all elements with ob. ob may be symbolic, real, complex or zint.
345006	^DIMRANM	({} → M') Creates symbolic random matrix from dimensions.
344006	^MATRANM	(M → M') Changes all elements of matrix to elements generated randomly.
374006	^OBJDIMS2MAT	(ob {} → M) Creates constant matrix from dimension and ob.
375006	^LCPROG2M	(#n #m prg → M) Fills a matrix of specified size using a program. prg must take two arguments and return one argument. On entry MAKE2DMATRIX provide the indexes as Z integers.
376006	^MAKE2DMATRIX	(#n #m prg → M) Creates 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.
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	(M {} → M') Changes size of a matrix, removing elements and/or adding zeros, as necessary.

342006	`^VRRDM`	`( []/[[]] {} → [] )` Vector Right ReDiMension: adds 0 to the right.
343006	`^VRRDMmeta`	`( meta #l → meta-#l )` Meta Right ReDiMension: adds 0 to the right.

5.3.2 Conversion

16A006	`^{}TO[]`	`( {} → [] )` Converts from list-of-lists representation to matrix. No checks on the element type.

17A006	`^LIST2MATRIX`	`( {} → [] )`
		`( {{}} → [[]] )`
		`( ob → ob )`
		Converts a symbolic list to a matrix. Does not check that matrix is a valid one. Use `DTYPFMAT?` to do that.

16B006	`^[]TO{}`	`( [] → {} )` Converts from matrix to list-of-lists.

179006	`^MATRIX2LIST`	`( [] → { } )`
		`( [[]] → {{}} )`
		`( ob → ob )`
		Converts a symbolic matrix to a list.

17E006	`^ARRAY2MATRIX`	`( [] → [] )`
		`( [[]] → [[]] )`
		Converts array to symbolic array if necessary.

175006	`^SAMEMATRIX`	`( M1 M2 → 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 → 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`	`( []/[[]] → {}/{{}} )` Converts normal array (containing only real or complex numbers) to list of lists; errors for symbolic arrays.
17D006	`^MATEXPLODE`	`( [[ob1..obn]] → ob1..obn [[ob1..obn]] )`

5.3.3 Tests

16C006	`^DUPNULL[]?`	`( ob → ob flag )` Tests for a null array.
359006	`^NULLVECTOR?`	`( V → flag )` Returns true if vector is null.

16F006	^CKSAMESIZE	(arry1 arry2 → arry1 arry2 flag) Tests if arry1 and 2 have the same size.
170006	^DTYPENDO?	(ob → ob flag) Tests if object is a square symbolic matrix. Convert numeric array to symbolic matrix.
173006	^2DMATRIX?	(ob → ob flag) Tests if object is a 2D matrix.

5.3.4 Calculations with Matrices

320006	^MAT+	(M2 M1 → M2+M1)
321006	^MADD	(M2 M1 → M2+M1)
322006	^MAT-	(M2 M1 → M2-M1)
323006	^MSUB	(M2 M1 → M2-M1)
324006	^VADD	(V2 V1 → V2+V1)
325006	^VSUB	(V2 V1 → V2-V1)
326006	^MAT*	(M2 M1 → M2*M1) Matrix product with size and type checking.
327006	^MMMULT	(M2 M1 → M2*M1)
328006	^MVMULT	(M V → V') Product of matrix by vector.
329006	^SCL*MAT	(ob M → M*ob) Scalar times matrix.
32A006	^MAT*SCL	(M ob → M*ob) Matrix times scalar.
32B006	^VPMULT	(V ob → V') Multiplies vector by a scalar.
335006	^MATSQUARE	(M → M*M)
32C006	^MAT^	(M z/% → M') Integral matrix power.
32D006	^MATCROSS	([] []' → []'') Vector product.
32E006	^MATDOT	(V2 V1 → ob) Scalar product with checking.
32F006	^RNDARRY	(M % → M) Rounds array.
330006	^TRCARRY	(M % → M) Truncates array.
332006	^MAT/SCL	(M ob → M/ob) Divides matrix by scalar.
333006	^MAT/	(V M → M^-1*V) "Divides" Vector by matrix.
334006	^MATCHS	(M → -M)

34E006	^MATINV	(M → M^-1)
336006	^MATCONJ	(M → M')
337006	^MATRE	(M → re[M])
338006	^MATIM	(M → im[M])
339006	^MATTRACE	(M → trace) Matrix trace.
33A006	^MATTRN	(M → M') Matrix transposition and conjugation.
33C006	^mattran	(M → Meta-M') Transposes matrix, returns meta-matrix.
33D006	^mattrn	(Meta-M → Meta-M') Transposes meta-matrix.
346006	^MATDET	(M → det) Determinant, expanding all (not row reduction).
347006	^MATRDET	(M → det) Determinant using row reduction.
348006	^MATFNORM	(M → ob) Frobenius norm.
349006	^MATRNORM	(M → ob) Row norm.
34A006	^MATCNORM	(M → ob) Column norm.
174006	^MATRIXDIM	(ob → #) Returns symbolic matrix dimensionality of an object.

5.3.5 Linear Algebra and Gaussian Reduction

34C006	^MATREF	(M → M') Returns matrix in Row-Echelon form.
34B006	^MATRREF	(M → M') Returns matrix in Reduced Row-Echelon form.
34F006	^MATREFRREF	(M #full_ref → 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 → M') Multiplies row #i of symbolic matrix M by constant. 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 → M') Does Lj <- c*Li+Lj. ncol is not used, it's here because of the stack state at call-time from inside laRCI.
350006	^INXREDext	(Lvar #full_ref M → Lvar pivot M)
351006	^METAMATRED	(Meta-M Lvar #full_red → meta-M Lvar pivot)

352006	`^METAPIVOT`	(`meta-M #l #c` → `meta-M #l #l' #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'.
353006	`^PIVOTNORM`	
354006	`^PIVOTFLOAT`	(`float` → `float_modulus`)
34D006	`^MATRANK`	(`M` → `Z/%`) Rank of a matrix.

5.3.6 Linear System Solver

080007	`^LINSOLV`	(`b a` → `y`) Solves y'=ay+b.
0F4007	`^SOLVEMETASYST`	(`meta-M` → `d meta-sol T`) (`meta-M` → `F`) Solves linear system in meta representation. Meta-sol has been reduced to the same denominator d.
0F5007	`^REDUCEMETASYST`	(`meta-M` → `meta->M'`) Reduces linear system in meta representation.
0F6007	`^REDUCEMETAPSYST`	(`meta-M` → `meta-M'`) Reduces linear system in meta representation. Does not reduce last column of meta-matr. This is useful to solve linear system with parameters in the last column.
0F7007	`^SOLVECRAMER`	(`meta-M` → `d meta-sol T`) (`meta-M` → `F`) Solves cramer system. Meta-matr must be fully reduced. Meta-sol is reduced to the same denominator. d flag is `FALSE` if dimension do not match.
355006	`^SYSText`	(`M linc` → `linc linc' res cas_p`)
356006	`^STOSYSText`	(`M2 M1` → `M2 list`)
357006	`^MAKESYSText`	(`M_eq M_inc` → `M_eq M lidnt flag`) Converts linear equations to a matrix and checks that equation are linear with respect to lidnt.
358006	`^VARGENext`	

5.3.7 Other Matrix Operations

35A006	`^FINDELN`	(`{} A` → `# flag`) Returns index # of element `{}` in array.
35B006	`^PULLEL[S]`	(`A #` → `A el`) Extracts element of index # from array. Array type test is made in assembly for array speed.

35C006	^BANGARRY	`( el # M → M' )` Puts el at index # of matrix M.
35D006	^PUT[]	`( el #i V → V )` Replaces #i-th vector component by element.

17B006	^LENMATRIX	`( [] → #el )`
		`( [[]] → #row )`

33E006	^MATSUB	`( M rmin nrows cmin ncols { #m #n } → M' )` Extracts submatrix from a matrix.
340006	^MATREPL	`( M1 M2 → 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 begins in matrix d at row r and column c.
35F006	^MATRIX>DIAG	`( A ncols+1 ndiags → V )` Extracts diagonal terms. ncols+1 is there because `MATRIX>DIAG` is called inside `la>DIAG`.
360006	^MATRIXDIAG>	`( ncol+1 diagV dlen dims{} → M )` Constructs a matrix from a vector of diagonal terms.
361006	^la+ELEMsym	`( V ob %i → V' )` Inserts element in symbolic vector at row %i.

362006	^INSERTROW[]	`( V ob #i → V )`
		`( M V #i → M' )`
		Inserts element/vector in symbolic vector/matrix at row #i. Checks for 0 < #i < #n + 1, but does not check for matrix/vector size.

363006	^insertrow[]	`( ob #i meta → meta )` Inserts element/vector in meta-object at position #i. Checks for 0 < #i < #n + 1, but does not check for vector size.
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 #i → M )` 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 → 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' → M )` Exchanges columns c and c' of a symbolic matrix.
36A006	^MATRIXRSWAP	`( M #r #r' → M )` Exchanges lines r and r' of a symbolic matrix.

0AC003	^SWAPROWS	(M % %' → M') SWAP two rows in matrix. Internal version of xRSWP. First available in ROM 1.11.
36B006	^MATRIX-ROW	(M #r → 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 → M cc) Extracts column #r from matrix. Checks boundaries.
36E006	^METAMATCSWAP	(meta-M #c #c' → meta-M) Exchanges columns c and c' of a meta-matrix.
36F006	^METAMATRSWAP	(meta-M #l #l' → meta-M) Exchanges lines l and l' of a meta-matrix (or vector).
370006	^STOMAText	(M →) Stores matrix in 'MATRIX' in current directory.
378006	^ADDMATOBJext	(arry ob → arry arry) (ob arry → arry arry) Used for addition of numeric matrix and symbolic object.
379006	^VUNARYOP	(v op → V) Applies unary op(v[i]) to get V[i].
37A006	^VBINARYOP	(V2 V1 binop → V) Works even if V2 and V1 do not have not the same dimension.
37B006	^PEVAL	(V r → P[r]) Horner evaluation, where elements of V represent coefficients of a polynomial.

5.3.8 Eigenvalues, Eigenfunctions, Reduction

37C006	^MATEGVL	(M → V) Computes eigenvalues of a matrix like <REF>xEGVL.
37F006	^MATEGV	(M → V) Computes eigenvalues/eigenvectors of a matrix like <REF>xEGV.
37E006	^MADJ	(M → M^-1 P[M] P[lambda]) Computes inverse, matrix polynomial and characteristic polynomial.
380006	^JORDAN	(M → pmin pcar {evect} {eval}) (pmadj pcar → pmin pcar {evect} {eval}) Eigenvalue/eigenfunctions computation.

22D006	`^FLAGJORDAN`	`( M →  )` Internal `JORDAN`.
381006	`^QXA`	`( symb lidnt → M lidnt )` Converts symbolic quad form to matrix quad form.
224006	`^FLAGQXA`	`( symb lidnt → M lidnt )` Internal `QXA`.
382006	`^AXQ`	`( M lidnt → symb lidnt )` Converts matrix quad form to qymbolic quad form.
225006	`^FLAGAXQ`	`( M lidnt → symb lidnt )` Internal `AXQ`.
383006	`^GAUSS`	`( symb → D P symb' )` Gauss reduction of quadratic form (symbolic).
226006	`^FLAGGAUSS`	`( symb lidnt → symb' )` Internal `GAUSS`.
384006	`^SYLVESTER`	`( M → D P )` Gauss reduction of a quadratic form (matrix).
227006	`^FLAGSYLVESTER`	`( M → P D )` Internal `SYLVESTER`.
228006	`^PCAR`	`( [[]] → symb )` Internal `PCAR`.

5.4 Symbolic Expression Handling

5.4.1 Basic Operations and Function Application

125006	`^x+ext`	`( ob2 ob1 → ob2+ob1 )` Symbolic addition, tests for infinities.
126006	`^x-ext`	`( ob2 ob1 → ob2-ob1 )` Symbolic subtraction, tests for infinities.
127006	`^x*ext`	`( ob2 ob1 → ob2*ob1 )` Symbolic multiplication, tests for infinities.
129006	`^x/ext`	`( ob2 ob1 → ob2/ob1 )` Symbolic division, tests for infinities.
12B006	`^x^ext`	`( ob power → ob^power )` Power.
12C006	`^EXPAND^`	`( x y → x^y=exp[y*ln[x]] )` Power with simplifications. If y is a fraction of integers, use XROOT^ instead.
4FB006	`^QNeg`	`( ob → -ob )` Symbolic negation.
4FC006	`^RNEGext`	`( ob → -ob )` Symbolic negation.
4FA006	`^SWAPRNEG`	`( ob2 ob1 → ob1 -ob2 )` Does `SWAP` then symbolic negation.

4FE006	`^RREext`	(`ob` → `Re(ob)`) Symboloc real part.
4FD006	`^SWAPRRE`	(`ob2 ob1` → `ob1 Re(ob2)`) `SWAP`, then `RREext`.
500006	`^RIMext`	(`ob` → `Im(ob)`) Symbolic imaginary part.
4FF006	`^SWAPRIM`	(`ob1 ob2` → `ob2 Im(ob1)`) `SWAP`, then `RIMext`.
501006	`^xREext`	(`symb` → `symb'`) Complex real part. Expands only `+ - * / ^`.
503006	`^xIMext`	(`symb` → `symb'`) Complex imaginary part. Expands only `+ - * / ^`.
505006	`^RCONJext`	(`ob` → `Conj(ob)`) Symbolic complex conjugate.
507006	`^xSYMCONJ`	
50D006	`^xABSext`	(`ob` → `abs(ob)`) Symbolic ABS function.
50A006	`^RABSext`	(`ob` → `abs(ob)`) Internal ABS. Internal representation.
50F006	`^xSYMABS`	
512006	`^xSYMSIGN`	
514006	`^xSYMARG`	
519006	`^CXIRext`	
52A006	`^xINVext`	(`ob` → `1/ob`) Symbolic inversion.
557006	`^xSYMINV`	(`symb` → `1/symb`) Symbolic inversion.
553006	`^xSQext`	(`symb` → `sq(symb)`) Symbolic square.
2EF53	`(SYMSQ)`	(`symb` → `symb^2`) Calls `^xSYMSQ` for symbolic objects and `xSQ` for other objects.
555006	`^xSYMSQ`	(`symb` → `symb^2`)
51B006	`^SXSQRext`	(`ob` → `sqrt(ob)`) Does not take care of the sign.
51C006	`^XSQRext`	(`ob` → `sqrt(ob)`) Tries to return a positive square root if nocareflag is cleared.
52B006	`^xvext`	(`ob` → `sqrt(ob)`) Symbolic square root, tests for 0 and 1.
552006	`^xSYMSQRT`	(`symb` → `sqrt(symb)`)
521006	`^CKLN`	(`ob` → `ln(ob)`) Symbolic LN with special handling for fractions. Does not use the internal representation.

522006	`^xLNext`	`( ob → ln(ob) )` Symbolic LN, without fraction handling.
524006	`^xSYMLN`	
525006	`^EXPANDLN`	`( ob → ln(ob) )` Symbolic LN using internal representation. Before switching to internal representation, test for ABS, 0 and 1 and, in real mode, test if ob=exp(x).
528006	`^REALLN`	`( ob → ln(ob) )` Internal natural logarithm for a real argument.
526006	`^CMPLXLN`	`( ob → ln(ob) )` Internal complex natural logarithm.
527006	`^LNATANext`	`( ob → ln(ob) )` Internal natural logarithm for complex.
529006	`^xEXPext`	`( y d n → exp(y*n/d*i*π) )` 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 → 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 → cos(ob) )`
533006	`^xACOSext`	`( ob → acos(ob) )` Symbolic ACOS. Tests for 0, infinity and tables.
53F006	`^xSYMACOS`	`( ob → acos(ob) )`
52D006	`^xSINext`	`( ob → 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 → sin(ob) )`
532006	`^xASINext`	`( ob → asin(ob) )` Symbolic ASIN. Tests for 0, infinity and tables.
53D006	`^xSYMASIN`	`( ob → asin(ob) )`
52E006	`^xTANext`	`( ob → tan(ob) )` Symbolic TAN. Tests for 0 and multiplies of $\pi/12$. Also tests if ob=atan(x).
53A006	`^xSYMTAN`	`( ob → tan(ob) )`
534006	`^xATANext`	`( ob → atan(ob) )` Symbolic ATAN. Tests for 0, infinity and tables.
541006	`^xSYMATAN`	`( ob → atan(ob) )`
52F006	`^xCOSHext`	`( ob → cosh(ob) )` Symbolic COSH. Tests for 0, infinity and acosh(x).
545006	`^xSYMCOSH`	`( ob → cosh(ob) )`
54E006	`^xACOSHext`	`( symb → acosh(symb) )` Symbolic ACOSH.
550006	`^xSYMACOSH`	`( symb → acosh(symb) )`

530006	^xSINHext	(ob → sinh(ob)) Symbolic SINH. Tests for 0, infinity and asinh(x).
543006	^xSYMSINH	(ob → sinh(ob))
54B006	^xASINHext	(symb → symb') Symbolic ASINH.
54D006	^xSYMASINH	(symb → asinh(symb))
531006	^xTANHext	(ob → tanh(ob)) Symbolic TANH. Tests for 0 and atanh(x).
547006	^xSYMTANH	(ob → tanh(ob)) Symbolic TANH.
548006	^xATANHext	(symb → symb') Symbolic ATANH.
54A006	^xSYMATANH	(ob → atanh(ob))
55B006	^xSYMD>R	
55D006	^xSYMR>D	
55F006	^xSYMFLOOR	(symb → symb')
561006	^xSYMCEIL	(symb → symb')
563006	^xSYMIP	(symb → symb')
565006	^xSYMFP	(symb → symb')
567006	^xSYMXPON	(symb → symb')
569006	^xSYMMANT	(symb → symb')
56B006	^xSYMLNP1	(symb → symb')
56D006	^xSYMLOG	(symb → symb')
56F006	^xSYMALOG	(symb → symb')
571006	^xSYMEXPM1	(symb → symb')
572006	^factorial	(symb → symb!) Symbolic factorial.
573006	^facts	(symb → symb!) Symbolic factorial.
575006	^xSYMFACT	(symb → symb!)
578006	^xSYMNOT	(symb → symb')
128006	^x=ext	(ob2 ob1 → ob2=ob1)
12E006	^xssSYMXROOT	
3AC006	^xssSYM+	
3AE006	^xssSYM-	
3B0006	^xssSYM*	
3B2006	^xssSYM/	
3B6006	^xssSYM^	
3B8006	^xSYMCHS	
130006	^xssSYMMIN	

132006	^xssSYMMAX
134006	^xssSYM<?
136006	^xssSYM<=?
138006	^xssSYM>?
13A006	^xssSYM>=?
13C006	^xssSYM=?
13E006	^xssSYM#?
140006	^xssSYM%
142006	^xssSYM%CH
144006	^xssSYM%T
146006	^xssSYMMOD
148006	^xssSYMTRCXY
14A006	^xssSYMRNDXY
14C006	^xssSYMCOMB
14E006	^xssSYMPERM
150006	^xssSYMOR
152006	^xssSYMAND
154006	^xssSYMXOR

5.4.2 Trigonometric and Exponential Operators

408006	^COS2TAN/2	(symb → symb') x → (1-(tan(x/2))^2)/(1+(tan(x/2))^2)
40B006	^SIN2TAN/2	(symb → symb') x → 2 tan(x/2)/(1+(tan(x/2))^2)
40E006	^TAN2TAN/2	(symb → symb') x → 2 tan(x/2)/(1-(tan(x/2))^2)
412006	^COS2TAN	(symb → symb2) x → 1/sqrt(1+(tan(x))^2)
414006	^SIN2TAN	(symb → symb') x → tan(x)/sqrt(1+(tan(x))^2)
41A006	^LNP12LN	(symb → symb') x → ln(x+1)
41B006	^LOG2LN	(symb → symb') x → log(x)
41C006	^ALOG2EXP	(symb → symb') x → alog(x)
41D006	^EXPM2EXP	(symb → symb') x → exp(x)-1
41E006	^SQRT2LNEXP	(symb → symb') x → exp(ln(x)/2)

41F006	^sqrt2lnexp	(meta → meta')
		x → exp(ln(x)/2)
420006	^TAN2EXP	(symb → symb')
		x → (exp(i2x)-1)/(i*(exp(i2x)+1))
422006	^ASIN2LN	(symb → symb')
		x → = i*ln(x+sqrt(x^2-1))+pi/2.
424006	^ACOS2LN	(symb → symb')
		x → ln(x+sqrt(x^2-1))/i
427006	^TAN2SC	(symb → symb')
		x → sin(x)/cos(x)
42A006	^SIN2TC	(symb → symb')
		x → cos(x)*tan(x)
42C006	^COS2ext	(symb → symb')
		x → sqrt(1-(sin(x))^2).
42E006	^SIN2ext	(symb → symb')
		x → sqrt(1-(cos(x))^2).
431006	^ATAN2ASIN	(symb → symb')
		x → asin(x/sqrt(x^2+1))
434006	^ASIN2ATAN	(symb → symb')
		x → atan(x/sqrt(1-x^2))
437006	^ASIN2ACOS	(symb → symb')
		x → π/2-acos(x)
43C006	^ACOS2ASIN	(symb → symb')
		x → π/2-asin(x)
43D006	^ATAN2LNext	(symb → symb')
		x → i/2*ln((i+x)/(i-x))
440006	^TAN2SC2	(symb → symb')
		x → (1-cos(2x))/sin(2x)
442006	^TAN2CS2	(symb → symb')
		x → sin(2x)/(1+cos(2x))
444006	^SIN2EXPext	(symb → symb')
		x → (e^(i*x)-1/e^(i*x))/(2i)
446006	^COS2EXPext	(symb → symb')
		x → (e^(i*x)+1/e^(i*x))/2
448006	^SINH2EXPext	(symb → symb')
		x → (e^x-1/e^x)/2
44A006	^COSH2EXPext	(symb → symb')
		x → (e^x+1/e^x)/2
44C006	^TANH2EXPext	(symb → symb')
		x → (e^2x-1)/(e^2x+1)
44E006	^ASINH2LNext	(symb → symb')
		x → ln(x+sqrt(x^2+1))
450006	^ACOSH2LNext	(symb → symb')
		x → ln(x+sqrt(x^2-1))

452006	`^ATANH2LNext`	`( symb → symb' )` x → ln((1+x)/(1-x))/2
454006	`^XROOT2ext`	`( symb1 symb2 → symb' )` x y → exp(ln(y)/x)
45A006	`^LN2ATAN`	`( symb → symb' )` x → ln(x)

5.4.3 Simplification, Evaluation and Substitution

45B006	`^VAR=LIST`	`( idnt {} → {}' )` Replaces all elements of the initial list by idnt=element.
464006	`^SYMBEXEC`	`( ob symb → ob' )` 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 change of variable works for change of user functions.
465006	`^MEVALext`	`( 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 vxxlflag is set SIGN var are leaved unchanged.
466006	`^CASNUMEVAL`	`( symb list1 list2 → symb' )` Evaluation of a symbolic. The lists' formats are list1={idnt/lam1... idnt_n/lam_n} list2={value1...value_n}. The idnt's/lam's in list1 are *not* evaluated before replacing value1...value_n.
467006	`^CASCOMPEVAL`	`( symb → symb' )` Evaluation of a symbolic.
468006	`^REPLACE2BY1`	`( symb idnt a → symb' )` Evaluation of 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 <REF>REPLACE2BY1 but prevents evaluation of INT.
46A006	`^SYMBWHERE`	
46B006	`^CASCRUNCH`	`( ob → % )` Like <REF>CRUNCH but in approximate mode.
46C006	`^APPROXCOMPEVAL`	`( symb → symb' )` Like <REF>CASCOMPEVAL but in approximate mode.
11A007	`^ALGCASCOMPEVAL`	`( expr → expr )` First available in ROM 1.11.

297006	^SLVARext	(Lvar → Lvar') Simplifies all elements of the list that are supposed to be variables.
298006	^SIMPLIFY	(symb → symb') Simplifies one object like <REF>xEVAL.
299006	^SIMP1ext	(symb → symb') Simplifies one object like <REF>xEXPAND. Object must be a symbolic, a real or a complex number.
29A006	^SYMEXPAN	(symb → symb') Simplifies one object like <REF>xEXPAN. Object must be symb/real/cmplx.
29B006	^SIMPVAR	(ob → ob') Simplifies variable.
2A0006	^SIMPSYMBS	(inf sup fcn var → int(inf,sup,fcn,var))
2A1006	^SYMINTEGRAL	
2A2006	^SIMPUSERFCN	(ob1..obn #n ob → 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	(V1..Vn #n fcn → f[]) Evaluates a user function with stack checking.
2A4006	^SIMP|	(ob list → ob') Executes the WHERE operator.
2A9006	^SIMPext	(ob1 ob2 → ob1' ob2') Simplifies two objects in internal representation. Checks that o2 is not a complex or an irrquad because decomposition of the corresponding fraction with larg would generate a "Try to recover Memory".
2AA006	^SIMPEXTOK	
2AC006	^SLOWSIMP2L	
2AD006	^SIMPGCDext	(o1 o2 gcd → o1/gcd o2/gcd) Divides o1 and o2 by gcd.
2AE006	^SIMP3ext	(a b → g a'' b'') Calculates g = gcd(a,b) and a"=a/g and b"=b/g.
2AF006	^SIMP3LISText	
2B0006	^SIMP3LSTSLOW	
2B9006	^TSIMP2ext	(symb → symb) Transcendental simplifications. Converts only sqrt ^ and XROOT to EXP/LN. LN are returned as -1/INV[-LN[]] for use by SERIES.
2BA006	^TSIMPext	(symb → symb) Transcendental simplifications. Convert transcendental functions to EXP and LN.
2BB006	^TSIMP3ext	(symb → symb)

5.4.4 Collection and Expansion

26E006	`^COLCext`	`( symb → symb' )` Factorization with respect to the current variable of symb and factorization of the integer content of symb.
2FE006	`^TCOLLECT`	`( symb → symb' )` Performs trigonometric linearization and then collects sines and cosines of the same angle.
2FF006	`^SIGMAEXPext`	`( symb → symb' )` Conversion to exp and ln with exponential linearization.
300006	`^LINEXPext`	`( symb → Meta )` Meta = arg_exp1 coef1 ... arg_expn coefn #2n.
301006	`^SIGMAEXP2ext`	`( Meta → symb )` Back conversion from arg_exp/coef_meta to symbolic.
303006	`^SINEXPA`	`( symb → symb' )` Expands SIN.
316006	`^LNEXPA`	`( symb → symb' )` Expands LN.
31C006	`^MTRIG2SYMB`	`( Meta → symb )` Back conversion of trig-meta to symbolic.
309006	`^COSEXPA`	`( symb → symb' )` Expands COS.
30F006	`^EXPEXPA`	`( symb → symb' )` Expands EXP.
31B006	`^LINEXPA`	`( symb → Meta )` Alternates trig operator and coefficient.
31D006	`^LNCOLCext`	`( symb → symb' )` Collects logarithms.
31F006	`^TEXPAext`	`( symb → symb )` Main transcendental expansion program.
26F006	`^SYMCOLCT`	
270006	`^COLC1`	
271006	`^COLC2`	
240006	`^EXLR`	`( 'a=b' → a b )` `( ob → X ob )` Internal equation splitter.

5.4.5 Trigonometric Transformations

407006	`^HALFTAN`	`( symb → symb' )` Converts trigonometric functions to TAN of the half angle.
411006	`^TRIGTAN`	`( symb → symb' )` Convert sin and cos to tan of the same angle.
416006	`^TRIGext`	`( symb → symb' )` Applies sin^2+cos^2=1 to simplify trigonometric expressions. If flag -116 is set, tries to keep only sin, else only cos.
417006	`^HYP2EXPext`	`( symb → symb' )` Converts hyperbolic functions to exp and ln. Converts XROOT and ^ to exp and ln.
418006	`^EXPLNext`	`( symb → symb' )` Converts all transcendental functions to exp and ln.
419006	`^SERIESEXPLN`	`( symb → symb' )` Converts sqrt, ^ and XROOT to EXP/LN.
426006	`^TAN2SCext`	`( symb → symb' )` Converts tan to sin/cos.
429006	`^SIN2TCext`	`( symb → symb' )` Converts sin to cos*tan.
430006	`^ATAN2Sext`	`( symb → symb' )` Converts ATAN to ASIN using asin(x)=atan(x/sqrt(1-x^2)).
433006	`^ASIN2Text`	`( symb → symb' )` Converts ASIN to ATAN using asin(x)=atan(x/sqrt(1-x^2)).
436006	`^ASIN2Cext`	`( symb → symb' )` Converts ASIN to ACOS using asin(x)=pi/2-acos(x).
43A006	`^ACOS2Sext`	`( symb → symb' )` Converts ACOS to ASIN using acos(x)=pi/2-asin(x).
43F006	`^TAN2SC2ext`	`( symb → symb' )` Converts TAN to SIN/COS of the double angle. If flag -116 is set calls `TAN2SC2`, else `TAN2CS2`.
456006	`^LN2ext`	`( symb → symb' )` If symb contains x, returns -1/inv(-ln(x)), else ln(x). Used by SERIES.
457006	`^SINCOSext`	`( symb → symb' )` Converts exp and ln to exp*sin+cos and ln+i*atan.

5.4.6 Division, GCD and LCM

3E8006	`^PSEUDODIV`	`( Q2 Q1 → a Q2*a/Q1 Q2*a/Q1 )`

3E9006	^IDIV2	
3EA006	^BESTDIV2	(o2 o1 → quo mod)
3EB006	^CDIV2ext	
3EC006	^QUOText	(o2 o1 → o2 div o1) Euclidean quotient of 2 objets (works even if o2 mod o1=0).
3ED006	^NEWDIVext	(ob2 ob1 → quo mod) Euclidean division, ob2 and ob1 may be fractions of returns a fraction of Q.
3F3006	^QUOTOBJext	(a_a-1...a0 bb_1...b0 #b #a flag → r q) SRPL Euclidean division: step 2 computes the remainder r only if flag is TRUE.
3F4006	^DIVISIBLE?	(a b → a/b T) (a b → ob F) Returns TRUE and quotient if b divides a, otherwise returns FALSE.
3F5006	^QDiv?	(a b → a/b T) (a b → F) Returns TRUE and quotient if b divides a, otherwise returns FALSE.
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 → 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 → ob) Calculates LCM of the denominators of the elements of Meta.
2B1006	^LPGCDext	({} → {} ob) Calculates the GCD of all the elements in the list. The algorithm is far from optimal.
2B2006	^SLOWGCDext	(c 1 A B → c* gcd(A,B)) Euclidean algorithm for polynomial GCD. Used if A or B contains irrquads. c is the GCD of the contents of the original polynomials returned after failure of GCDHEUext.
2B3006	^QGcd	(ob2 ob1 → gcd) Generic internal GCD. (LAM2: GCDext ob1, ob2 → pgcd).
2B4006	^GCDext	

5.5 Symbolic Meta Handling

5.5.1 Basic Expression Manipulation

157006	`^SYMBINCOMP`	`( symb → ob1 .. obN #n )`
		`( ob → ob #1 )`
		`( {} → {} #1 )`
		Explodes symbolic object into meta. Other objects are converted into one-object metas by pushing #1 into the stack.

386006	`^m-1&m+1`	`( meta → 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`	`( meta → meta 1&meta&/ )` Duplicates the meta, and inverts the expression represented by it.
388006	`^1&meta`	`( Meta → 1&Meta )` Prepends the number 1 to the meta.
389006	`^meta/2`	`( Meta → Meta&2&/ )` Divides the expression by two.
38A006	`^addt2`	`( Meta → Meta&2 )` Appends the number 2 to the meta.
38B006	`^addt/`	`( Meta → Meta&/ )` Appends division to meta.
38C006	`^meta2*`	`( Meta → 2&Meta&* )` Multiplies the expression by 2.
459006	`^metai*`	`( meta → meta*i )` Multiplies meta by i.
38D006	`^meta1-sq`	`( Meta → 1&Meta&SQ&- )` Changes x into 1-x^2, where x is the original expression.
38E006	`^metasq+1`	`( Meta → Meta&SQ&1&+ )` Changes x into x^2+1, where x is the original expression.
38F006	`^metasq-1`	`( Meta → Meta&SQ&1&- )` Changes x into x^2-1, where x is the original equation.
390006	`^meta-1`	`( Meta → Meta&1&- )` Subtracts one from the expression.
398006	`^addt^`	`( Meat → Meta&^ )` Append power operator to meta object.
39C006	`^top&addt*`	`( meta2 meta1 → meta2*meta1 )` `top& addt*`. No checks.
39D006	`^top&addt/`	`( meta2 meta1 → meta2/meta1 )` `top& addt/`. No checks.

39E006	`^addti`	`( meta → meta&i )` Appends i (the Imaginary unit) to expression.

5.5.2 Basic Operations and Function Application

393006	`^metaadd`	`( Meta1 Meta2 → Meta1+Meta2 )` Adds 2 meta objects with trivial simplifications. `metaadd` checks for Meta1/2=Z0 `ONE`.
3AB006	`^MetaAdd`	`( Meta2 Meta1 → Meta2+Meta1 )` Adds 2 meta objects with trivial simplifications. Checks for infinities then call `metaadd`.
1CE006	`^ckaddt+`	`( Meta1 Meta2 → Meta1+Meta2 )` Adds 2 meta objects with trivial simplifications.
394006	`^metasub`	`( Meta1 Meta2 → Meta1+Meta2 )` Subtracts 2 meta objects with trivial simplifications. `metasub` checks for Meta1/2=Z0 `ONE`.
3AD006	`^MetaSub`	`( Meta2 Meta1 → Meta2-Meta1 )` Subtracts 2 meta objects with trivial simplifications. Checks for infinities then call `metasub`.
1CF006	`^ckaddt-`	`( Meta1 Meta2 → Meta1+Meta2 )` Subtracts 2 meta objects with trivial simplifications.
395006	`^metamult`	`( Meta1 Meta2 → Meta1*Meta2 )` Multiplies 2 meta objects with trivial simplifications. Checks for meta1, meta2= `Z0` or `Z1`, checks for `xNEG`.
3AF006	`^MetaMul`	`( Meta2 Meta1 → Meta2*Meta1 )` Multiplies 2 meta objects with trivial simplifications. Checks for infinities/0 then call `metamult`.
1CD006	`^ckaddt*`	`( Meta1 Meta2 → Meta1*Meta2 )` Multiplies 2 meta objects with trivial simplifications.
396006	`^metadiv`	`( Meta2 Meta1 → 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 → Meta2/Meta1 )` Divide 2 meta objects with trivial simplifications. Checks for infinities and 0 then call `metadiv`.
3F1006	`^DIVMETAOBJ`	`( o1...on #n ob → {o1/ob...on/ob} )` Division of all elements of a meta by ob. Tests if o=1.
397006	`^meta^`	`( Meta ob → 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.

399006	`^metapow`	`( Meta2 Meta1 → Meta2^Meta1 )` Elevates expression to a power (any other expression). If length of Meta1 is `ONE`, calls `meta^`.
3B5006	`^MetaPow`	`( Meta2 Meta1 → Meta2^Meta1 )` Power. Checks for infinities then calls `metapow`.
39B006	`^metaxroot`	`( Meta2 Meta1 → Meta2&XROOT&Meta1 )` Root of expression.
3B9006	`^metaneg`	`( meta → meta )` Checks only for meta finishing by `xNEG`.
3BA006	`^metackneg`	`( meta → meta )` Like <REF>metaneg but checks for meta=ob `ONE`.
3B7006	`^MetaNeg`	`( Meta → Meta )` Negates meta. Only checks for final <REF>xNEG in meta.
502006	`^xSYMRE`	`( meta → meta' )` Meta complex real part. Expands only `+ - * / ^`.
504006	`^xSYMIM`	`( meta → meta' )` Meta complex imaginary part. Expands only `+ - * / ^`.
50E006	`^addtABS`	`( Meta → Meta' )` Meta ABS. Does a CRUNCH first to find sign.
510006	`^addtABSEXACT`	`( Meta → Meta' )` Meta ABS. No crunch, sign is only found using exact methods.
511006	`^addtSIGN`	`( Meta → Meta' )` Meta SIGN.
513006	`^addtARG`	`( Meta → Meta' )` Meta ARG.
12D006	`^addtXROOT`	`( Meta2 Meta1 → Meta' )` Meta XROOT. XROOT(o2,o1) is o1^[1/o2], compared to o2^o1.
12F006	`^addtMIN`	`( Meta2 Meta1 → Meta' )` Meta MIN.
131006	`^addtMAX`	`( Meta2 Meta1 → Meta' )` Meta MAX.
133006	`^addt<`	`( Meta2 Meta1 → Meta' )` Meta <.
135006	`^addt<=`	`( Meta2 Meta1 → Meta' )` Meta <=.
137006	`^addt>`	`( Meta2 Meta1 → Meta' )` Meta >.
139006	`^addt>=`	`( Meta2 Meta1 → Meta' )` Meta >=.
13B006	`^addt==`	`( Meta2 Meta1 → Meta' )` Meta ==.
13D006	`^addt!=`	`( Meta2 Meta1 → Meta' )` Meta !=.

13F006	`^addt%`	(`Meta2 Meta1` → `Meta'`) Meta %.
141006	`^addt%CH`	(`Meta2 Meta1` → `Meta'`) Meta %CH. Meta2*(1+Meta'/100)=Meta1.
143006	`^addt%T`	(`Meta2 Meta1` → `Meta'`) Meta %T.
145006	`^addtMOD`	(`Meta2 Meta1` → `Meta'`) Meta MOD.
147006	`^addtTRNC`	(`Meta2 Meta1` → `Meta'`) Meta TRNC.
149006	`^addtRND`	(`Meta2 Meta1` → `Meta'`) Meta RND.
14B006	`^addtCOMB`	(`Meta2 Meta1` → `Meta'`) Meta COMB.
14D006	`^addtPERM`	(`Meta2 Meta1` → `Meta'`) Meta PERM.
14F006	`^addtOR`	(`Meta2 Meta1` → `Meta'`) Meta OR.
151006	`^addtAND`	(`Meta2 Meta1` → `Meta'`) Meta AND.
153006	`^addtXOR`	(`Meta2 Meta1` → `Meta'`) Meta XOR.
506006	`^addtCONJ`	(`meta` → `meta'`) Meta complex conjugate.
523006	`^addtLN`	(`Meta` → `Meta'`) Meta LN.
535006	`^addtCOS`	(`Meta` → `Meta'`) 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].
53C006	`^addtASIN`	(`Meta` → `Meta'`) Meta ASIN.
53E006	`^addtACOS`	(`Meta` → `Meta'`) Meta ACOS.
540006	`^addtATAN`	(`Meta` → `Meta'`) Meta ATAN.
542006	`^addtSINH`	(`Meta` → `Meta'`) Meta SINH.
544006	`^addtCOSH`	(`Meta` → `Meta'`) Meta COSH.
546006	`^addtTANH`	(`Meta` → `Meta'`) Meta TANH.
549006	`^addtATANH`	(`Meta` → `Meta'`) Meta ATANH.

54C006	`^addtASINH`	(Meta → Meta') Meta ASINH.
54F006	`^addtACOSH`	(Meta → Meta') Meta ACOSH.
551006	`^addtSQRT`	(Meta → Meta') Meta SQRT.
554006	`^addtSQ`	(Meta → Meta') Meta SQ.
556006	`^addtINV`	(Meta → Meta') Meta INV.
558006	`^addtEXP`	(Meta → Meta') Meta EXP. Does not apply EXP[-..]=1/EXP[..].
559006	`^xSYMEXP`	(Meta → Meta') Meta EXP. Applies EXP[-..]=1/EXP[..].
55A006	`^addtD->R`	(Meta → Meta') Meta D→R.
55C006	`^addtR->D`	(Meta → Meta') Meta R→D.
55E006	`^addtFLOOR`	(Meta → Meta') Meta FLOOR.
560006	`^addtCEIL`	(Meta → Meta') Meta CEIL.
562006	`^addtIP`	(Meta → Meta') Meta IP.
564006	`^addtFP`	(Meta → Meta') Meta FP.
566006	`^addtXPON`	(Meta → Meta') Meta XPON.
568006	`^addtMANT`	(Meta → Meta') Meta MANT.
56A006	`^addtLNP1`	(meta → meta) Meta LNP1.
56C006	`^addtLOG`	(meta → 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.

5.5.3 Trigonometric and Exponential Operators

409006	`^cos2tan/2`	(meta → meta') x → (1-(tan(x/2))^2)/(1+(tan(x/2))^2)

40A006	`^1-x^2/1+x^2`	(meta → meta') x → (1-x^2)/(1+x^2)
40C006	`^sin2tan/2`	(meta → meta') x → 2 tan(x/2)/(1+(tan(x/2))^2)
40D006	`^2x/1+x^2`	(meta → meta') x → 2x/(1+x^2)
40F006	`^tan2tan/2`	(meta → meta') x → 2 tan(x/2)/(1-(tan(x/2))^2)
410006	`^addtTAN/2`	(meta → meta') x → tan(x/2)
413006	`^cos2tan`	(meta → meta') x → 1/sqrt(1+(tan(x))^2)
415006	`^sin2tan`	(meta → meta') x → tan(x)/sqrt(1+(tan(x))^2)
421006	`^tan2exp`	(meta → meta') x → (exp(i2x)-1)/(i*(exp(i2x)+1))
423006	`^asin2ln`	(meta → meta') x → = i*ln(x+sqrt(x^2-1))+π/2.
425006	`^acos2ln`	(meta → meta') x → ln(x+sqrt(x^2-1))/i
428006	`^sin/cos`	(meta → meta') x → sin(x)/cos(x)
42B006	`^cos*tan`	(meta → meta') x → cos(x)*tan(x)
42D006	`^sqrt1-sin^2`	(meta → meta') x → sqrt(1-(sin(x))^2).
42F006	`^sqrt1-cos^2`	(meta → meta') x → sqrt(1-(cos(x))^2).
432006	`^atan2asin`	(meta → meta') x → asin(x/sqrt(x^2+1))
435006	`^asin2atan`	(meta → meta') x → atan(x/sqrt(1-x^2))
438006	`^pi/2-acos`	(meta → meta') x → π/2-acos(x)
439006	`^pi/2-meta`	(meta → meta') x → π/2-x
43B006	`^pi/2-asin`	(meta → meta') x → π/2-asin(x)
43E006	`^atan2ln`	(meta → meta') x → i/2*ln((i+x)/(i-x))
441006	`^2*1-cos/sin`	(meta → meta') x → (1-cos(2x))/sin(2x)
443006	`^2*sin/1+cos`	(meta → meta') x → sin(2x)/(1+cos(2x))

445006	`^sin2exp`	`( meta → meta' )` x → (e^(i*x)-1/e^(i*x))/(2i)
447006	`^cos2exp`	`( meta → meta' )` x → (e^(i*x)+1/e^(i*x))/2
449006	`^sinh2exp`	`( meta → meta' )` x → (e^x-1/e^x)/2
44B006	`^cosh2exp`	`( meta → meta' )` x → (e^x+1/e^x)/2
44D006	`^tanh2exp`	`( meta → meta' )` x → (e^2x-1)/(e^2x+1)
44F006	`^asinh2ln`	`( meta → meta' )` x → ln(x+sqrt(x^2+1))
451006	`^acosh2ln`	`( meta → meta' )` x → ln(x+sqrt(x^2-1))
453006	`^atanh2ln`	`( meta → meta' )` x → ln((1+x)/(1-x))/2
455006	`^xroot2expln`	`( meta1 meta2 → meta' )` x y → exp(ln(y)/x)
458006	`^exp2sincos`	`( meta → meta' )` Returns EXP of meta as EXP[RE]*[COS+i*SIN].

5.5.4 Infinity and Undefs

3A1006	`^1metaundef#`	`( meta → meta # )` Tests presence of undef in meta. # is the position of undef.
3A0006	`^2metaundef#`	`( meta2 meta1 → meta2 meta1 # )` Tests presence of undef in meta2 and meta1. # is the position of undef.
3A2006	`^metaundef`	`( → meta )` Returns undef meta.
3A4006	`^1metainf#`	`( meta → meta # )` Finds position of infinity in meta. Metas of length>2 are considered as finite meta.
3A3006	`^2metainf#`	`( meta2 meta1 → meta2 meta1 # )` Finds position of infinity in meta 2 and meta1. Metas of length>2 are considered as finite meta.
3A5006	`^metainftype`	`( meta → # )` Returns infinity type: 1 for +infinity, 2 for -infinity or 0 for unsigned.
3A6006	`^unsignedinf`	`( → meta )` Returns unsigned infinty.
3A7006	`^plusinf`	`( → meta )` Returns plus infinty.

3A8006	^NDROPplusinf	(ob1..obn → meta) Replaces meta by plus infinty.
3A9006	^minusinf	(→ meta) Returns minus infinty.
3AA006	^NDROPminusinf	(ob1..obn → meta) Replace meta by minus infinty.

5.5.5 Expansion and Simplification

3BB006	^metasimp	(Meta → Meta) Simplifies a meta object. Non recursive rational simplification.
118007	^DISTRIB*	(meta → meta' T) (meta → meta F) Distribute *. Returns FALSE if no distribution done. First available in ROM 1.11.
3C2006	^DISTRIB/	(meta → meta' T) (meta → meta F) Distribute /. Returns FALSE if no distribution done.
304006	^METASINEXPA	(Meta → Meta') Expands SIN.
305006	^SINEXPA+	(Meta → Meta') Expands SIN(x+y).
306006	^SINEXPA-	(Meta → Meta') Expands SIN(x-y).
307006	^SINEXPA*	(Meta → Meta') Expands SIN(x*y). Expands if x or y is an integer.
308006	^SINEXPA*1	(Meta2 Meta1 → Meta') Expands SIN(x*y). Meta1 is assumed to be an integer.
30A006	^METACOSEXPA	(Meta → Meta') Expands COS.
30B006	^COSEXPA+	(Meta → Meta') Expands COS(x+y).
30C006	^COSEXPA-	(Meta → Meta') Expands COS(x-y).
30D006	^COSEXPA*	(Meta → Meta') Expands COS(x*y).
30E006	^COSEXPA*1	(meta2 meta1 → Meta') Expands COS(x*y). meta1 represents an integer.
310006	^METAEXPEXPA	(Meta → Meta') Expands EXP.
311006	^EXPEXPA+	(Meta → Meta') Expands EXP(x+y).

312006	`^EXPEXPA-`	`( Meta → Meta' )` Expands EXP(x-y).
313006	`^EXPEXPA*`	`( Meta → Meta' )` Expands EXP(x*y).
314006	`^EXPEXPANEG`	`( Meta → Meta' )` Expands EXP(-x).
315006	`^EXPEXPA*1`	`( Meta2 meta1 → Meta' )` Expands EXP(x*y). meta1 represents an integer.
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).
31E006	`^METATANEXPA`	`( meta → tan[meta] )` Expands tan[meta].

5.5.6 Tests

39A006	`^metafraction?`	`( Meta → Meta flag )` Tests if meta is a fraction of integers.
3BC006	`^metapi?`	`( Meta → Meta# )` Tests presence of π in a meta. # is the last occurence of π or 0.

3BD006	`^metaCOMPARE`	`( Meta2 Meta1 → Meta2 Meta1 # )`
		Comparison of 2 meta.
		`# =0 if undef`
		`# =1 if >`
		`# =2 if <`
		`# =3 if =`
		Assumes generic situation, e.g. X^2 > 0 in real mode. Look below `STRICTmetaCOMPARE` for a more careful comparison.

3BE006	`^STRICTmetaCOMPARE`	`( Meta2 Meta1 → Meta2 Meta1 # )`
		Comparison of 2 meta.
		`# =0 if undef`
		`# =1 if >`
		`# =2 if <`
		`# =3 if =`
		Unlike <REF>metaCOMPARE it does not assume generic situation.

3C3006	`^metareal?`	`( meta → meta flag )` Tests if IM[meta]==0.

5.6 Polynomials

5.6.1 Computation with Polynomials

118006	`^QAdd`	`( o1 → o2+o1 )` Adds two polynomials.
119006	`^RADDext`	`( o2 o1 → o2+o1 )` Internal `+`. This is the same entry as `^QAdd`.
117006	`^SWAPRADD`	`( o2 o1 → o1+o2 )` `SWAP`, then `QAdd`.
115006	`^QSub`	`( o2 o1 → o2-o1 )` Subtracts two polynomials.
116006	`^RSUBext`	`( o2 o1 → o2-o1 )` Internal `-`. This is the same entry as `^QSub`.
114006	`^SWAPRSUB`	`( o2 o1 → o1-o2 )` `SWAP`, then `QSub`.
111006	`^QMul`	`( Q1 Q2 → Q )` Multiplication of polynomials with extensions.
112006	`^RMULText`	`( Q1 Q2 → Q )` Multiplication of polynomials with extensions. This is the same entry as `^QMul`.
110006	`^SWAPRMULT`	`( Q1 Q2 → Q )` `SWAP`, then `^QMul`.
11C006	`^QDiv`	`( o2 o1 → o2/o1 )` Internal /.
11B006	`^RDIVext`	`( o2 o1 → o2/o1 )` Internal /. This is the same entry as `^QDiv`.
11A006	`^SWAPRDIV`	`( o2 o1 → o1/o2 )` `SWAP`, then `QDiv`.
0D9006	`^QMod`	`( Q, Z → Q mod Z )`
0DF006	`^QRoot`	Extracts Nth power factors from polynomial.
113006	`^RASOP`	`( n1/d1 n2/d2 → d1*d2 n1*d2 n2*d1 )` Used by `RADDext` and `RSUBext` for rational input.
11D006	`^R15SIMP`	
11E006	`^PPow#`	
11F006	`^RP#`	`( o2 # → o2^# )` Internal power (not for matrices).
120006	`^MPext`	`( ob # prg* → ob^# )` General power with a specified multiplication program.
123006	`^RPext`	`( o2 o1 → o2^o1 )` Tries to convert o1 to an integer to call `RP#`, otherwise `x^ext`.

122006	`^MPEXEC`	

108006	`^DISTDIVext`	`( P Q → quo mod T )`
		`( P Q → P Q F )`
		Euclidean division. Assumes P and Q have integer coefficientes. Returns `FALSE` if sparse short division fails.

3E5006	`^PTAYLext`	`( P, r → symb )` Taylor for polynomials.
15B006	`^CARCOMPext`	`( Q1/Q2 → Q1'/Q2' )` Extracts leading coefficients for the first variable from a rational polynomial.
3EE006	`^QDivRem`	`( ob2 ob1 → quo mod )` Polynomial Euclidean division of 2 objects. Dispatchs to `DIV2LISText` for list polynomials.
3EF006	`^DIV2LISText`	`( Z0 l1 l2 → div mod )` Euclidean division, l1 and l2 are list polynomials. Test first if l1=l2, then tries fast division, if it fails switch to SRPL division.
3F8006	`^PDIV2ext`	`( A B → Q R )` Step by step Euclidean division for univar poly.
3F9006	`^PSetSign`	`( P1 P2 → sign[P2]*P1 )` Sets sign of P1 according to leading coeff of P2.
3C4006	`^ModExpa`	`( Zn Fraction → Fraction modulo Zn )`
3C5006	`^ModAdd`	`( Q1 Q2 Zn → Z )` Modular addition. Z = Q1+Q2 (mod Zn).
3C6006	`^ModSub`	`( Q1 Q2 Zn → Z )` Modular subtraction. Z = Q1-Q2 (mod Zn).
3C7006	`^ModMul`	`( Q1 Q2 Zn → Z )` Modular multiplication. Z = Q1*Q2 (mod Zn).
3C8006	`^ModDiv`	`( Z1 Z2 Zn → Z )` Modular division. Z = Z1/Z2 (mod Zn).
3C9006	`^ModDiv2`	`( Q1 Q2 Zn → 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 → Z' )` Modular inversion. Z' = INV(Z) (mod Zn). `NONINTERR` if GCD[Z,Zn] $\neq$ 1 or if Z = 0 (otherwise the results would be unpredictable).
3CB006	`^ModGcd`	`( Q1 Q2 Zn → Q' )` Modular GCD.
3CC006	`^ModLGCD`	
3CD006	`^ModLOPD`	
3CE006	`^MODULOMODext`	
3CF006	`^MODULOMAText`	

3D1006	`^ModFctr`	

5.6.2 Factorization

08E006	`^BerlekampP`	`( P #prime → P F / P Lf #prime T )` Berlekamp's algorithm for finding modular factors 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$\leq$97.
0A8006	`^ALG48FCTR?`	`( P → [ meta cst_coeff TRUE | P FALSE ] )` Factorizes square-free polynomial in Erable format.
0A9006	`^MFactTriv`	`( P → meta-factor P' )` Extracts all trivial power factors of P.
0AA006	`^CheckPNoExt`	`( P → P flag )` Checks that P does not contain any `DOCOL` (i.e. extensions).
0AB006	`^PPP`	`( P → 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 → 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 → Lf) Heuristic factorization of primitive square-free non constant polynomial.
0D8007	`^P2P#`	(P → P' #) Extracts trivial power of poly. P must be a valid poly (if list, begin with a non zero coeff).
0B1006	`^DeCntMulti`	(R → L) Transforms list with count into simple list. `R = { {f1 #k1} ... {fn #kn} }` `L = { f1 f1 .. fn fn }.`
0B2006	`^DoLS`	(L S F → L') Applies program F(Li,S) to every elem of L.
0B3006	`^PNFctr`	(Z → Lf) Factorization of positive integer as polynomial. `Lf = {} if Z is 1` `Lf = { {Z1 #k1} ... {Zn #kn} } o/w.`
0B4006	`^PSQFF`	(P → Lsqff) Computes the square-free factorization of primitive P. The result is trimmed (provided P was).
0B5006	`^LiftZAdic`	(p z F → L) Lift n-1 z-adic factorization into n factorization.
0B6006	`^LFCProd`	(C L → C P) Calculates combination product.
0B7006	`^UFactor`	(P → Lf) Factorization of a square free primitive univariate polynomial.
0B8006	`^UFactor1`	(P → Lf) Factorization of a square free primitive univariate polynomial of degree > 2.
0B9006	`^MonicLf`	(Lfp p → Lfp') Converts true modular factorization to monic factorization by dividing by the leading coefficient of factor 1.
0BA006	`^DemonicLf`	(Lfp lc p → Lfp') Converts monic modular factorization to true modular factorization by multiplying factor1 by lcoeff.

0BB006	`^LiftLinear`	`( #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 and uses them for input and output.

0BC006	`^LiftGeneral`	`( → )` Lifts factorization mod p to factorization mod p^k where p^k exceeds the factor bound for succesful true factor extraction. Assumes `UFactor` lambda variables.
0BD006	`^UFactorDeg2`	`( P → Lf )` Factorization of a degree 2 polynomial. Polynomial is univariate, square free and primitive.
0BE006	`^CombineFac`	`( P Lfp p → Tf Tfp )` Combines modular factors to true factors. P is the polynomial to factor, Lfp is the list of modular factors, and p the modulo. The entry returns 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 → F )` Calculates modular combination.

0C0006	`^CombInit`	`( #r → 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 → P' Lf )` Extracts all trivial power factors of P.
0C4006	`^VarFactor`	`( P #var → P #n )` Calculates what power of the given variable is a factor in P.
0C5006	`^PFactPowCnt`	`( P → P Lk flag )` Calculates trivial power factors in P. flag is `TRUE` if any of the powers is nonzero.
0C6006	`^PDivLk`	`( P Lk → P' )` Divides polynomial by its trivial powers.
282006	`^FEVIDENText`	`( P → meta-fact cst coeff )` Real mode: full factorization over the integer Complex mode: find all 1st order factors of P.

5.6.3 General Polynomial Operations

09B006	`^ONE{}POLY`	(ob → {ob} ob1 → Q) Replaces `ONE{}N` for polynomial building.
09C006	`^TWO{}POLY`	(ob1 ob2 → Q) Replaces `TWO{}N` for polynomial building.
09D006	`^THREE{}POLY`	(ob1 ob2 ob3 → Q) Replaces `THREE{}N` for polynomial building.
09E006	`^TWO::POLY`	(ob1 ob2 → ::) Replaces `2Ob>Seco` for polynomial building.
09F006	`^::POLY`	(Meta → ::) Replaces `::N` for polynomial building. As opposed 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 → Q) Replaces `{}N` for polynomial building. As opposed 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 → Q) Builds polynomial.
0A1006	`^>TPOLY`	(P ob → P') Replaces `>TCOMP` for polynomial building.
0A2006	`^>HPOLY`	(P ob → P') Replaces `>HCOMP` for polynomial building.
0A3006	`^>TPOLYN`	(P ob1 .. obn #n → P') Improved `>TCOMP` for polynomial building.
0A4006	`^>HPOLYN`	(P ob1 .. obn #n → P') Improved `>HCOMP` for polynomial building.
0A5006	`^MKPOLY`	(#n #k → P) Makes polynomial of nth variable to the power k.
2AB006	`^MAKEPROFOND`	(ob # → {{{...{o}...}}}) Embedds ob in the given number of lists.
4F4006	`^TRIMext`	(Q → Q') Removes unnecessary zeros from polynomial.
4F5006	`^PTrim`	(ob → ob') Trims polynomial.
0A6006	`^ONE>POLY`	(Q → Q') Increases variable depth. Constants (Z,Irr,C) are not modified.
302006	`^TCHEBext`	(zint → P) Tchebycheff polynomial. If zint>0 then 1st kind, if <0 then second kind.

3DE006	^LRDMext	(P # → []) Left ReDiMension. Adds 0 to the left of polynomial to get a symbolic vector of lenght #+1.
3DF006	^RRDMext	({} # → {}) Right ReDiMension: like <REF>LRDMext but 0 at the right and {}.
3E0006	^DEGREext	({} → degre) Degree of a list-polynomial.
3E1006	^FHORNER	(P/d r → P[X]_div_[X-r]/d r P[r]/d) Horner scheme.
3E2006	^HORNext	(P r → P[X]_div_[X-r] r P[r]) Horner scheme.
3E3006	^HORN1	
3E4006	^MHORNext	(P r → P[X]_div_[X-r] r P[r]) Horner scheme for matrices.
3E6006	^LAGRANGEext	(M → 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 → P) Resultant of two polynomials. Depth of P is one less than depth of P1 and P2. First available in ROM 1.11.
110007	^RESULTANTLP	(res g h P1 P2 → +/-res g' h' P1' P2') Subresultant algorithm innerloop. First available in ROM 1.11.
111007	^RESPSHIFTQ	(P Q → 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. First available in ROM 1.11.
112007	^ADDONEVAR	(P → P') Adds one variable just below the main var. works for polynomial, not for fractions. First available in ROM 1.11.
0CF007	^SHRINKEVEN	(P → P') Changes var Y=X^2 in an even polynomial.
0D0007	^SINTEST	
0D1007	^SHRINK2SYM	(N D → N' D') Shrinks 2 polynomials using symmetry properties.
0D2007	^SHRINKSYM	(N → N') Shrinks 1 polynomial using symmetry properties. Degree of N must be even. If it is odd then N should be divided by X+1.
0D3007	^SHRINK2ASYM	(N D → N' D') Shrinks 2 polynomials using antisymmetry properties.

0D4007	^SHRINKASYM	(N → 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 → Z) Gets the coefficient of P with max norm.
161006	^SWAPNDXF	(Qden Qnom → symb) Builds a symbolic from rational polynomial.
162006	^NDXFext	(Qnom Qden → symb) Builds a symbolic from rational polynomial.
163006	^SWAPFXND	(symb ob → ob Qnom Qden) Converts symbolic to rational polynomial.
164006	^FXNDext	(symb → Qnom Qden) Converts symbolic to rational polynomial.
3D7006	^REGCDext	(a b → d u v au+bv=d)
3D8006	^EGCDext	(a b → d u v au+bv=d) Bezout identity for polynomials.
0EA006	^PEvalFast?	(Z Pn → Z Pn F / Pn[Z] T) Attempts to evaluate Pn at X1=Z using fast register arithmetic. Fails if any of the following is true: Pn is not sunivariate; Z is polynomial after all; Z size is too big for register; Any overflow occurs during Horner evaluation.
10E007	^FLAGRESULTANT	(symb1 symb2 → symb) Resultant of two polynomials in symbolic form. First available in ROM 1.11.

5.6.4 Tests

10B006	^Univar?	(P → P flag) Tests if polynomial is univariate.
10C006	^SUnivar?	(P → P flag) Tests if polynomial is univariate and the coefficients are bounded by register size.
0CC007	^POLYPARITY	(poly → 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 → Z) Tests symmetry of coefficients of polynomial. Z=1 for symmetric, -1 for anti, 0 otherwise.
0D7007	^POLYASYM	(P → Z) Tests "antisymmetry" of coef of polynomial. Z=1 for symmetric, -1 for anti, 0 otherwise.

5.7 Root Finding

5.7.1 Root Finding and Numerical Solvers

272006	`^MULMULText`	`( {} % → {}' )` Multiplies multiplicities in a factor list by coeff.
273006	`^METAMULMULT`	
274006	`^METAMM2`	`( meta % → meta' )` Multiplies by % all multiplicities of meta.
275006	`^COMPLISText`	`( {} → {}' )`
276006	`^METACOMPRIM`	`( Meta → Meta' )` Suppresses multiple occurrances of the same factor by adding corresponding multiplicities.
277006	`^METACOMP0`	
278006	`^METACOMP1`	`( f1...fk-1 mk-1 meta-res mk fk # → f1...fk-1 mk-1 meta-res )`
279006	`^ADDLISText`	`( {} %n ob → {}' )` Adds ob with multiplicity %n to the list. Checks if ob is in `{}`.
27A006	`^DIVISext`	`( ob → {divisors} )` Returns list of divisors of ob.
27B006	`^FACT1ext`	`( symb-poly → Lvar Q {} )` `{}` is the list of root/multiplicity of `sym` with respect to the current variable.
27C006	`^FACTOext`	`( symb → Lvar Q {} )` `{}` is the list of factors/multiplicity of symb.
27D006	`^ZFACTO`	`( C → {} C Lfact )`
27E006	`^SOLVext`	`( symb → {} )` Numeric solver for univariate polynomials. The list contains the roots without multiplicity.
27F006	`^FRND`	`( ob → ob') )` Float rounding for %%, C%% or list of either type. Used by `SOLVext` to reconstruct factors.
280006	`^BICARREE?`	`( P #5 → meta cst_coeff T )` `( P #5 → P #5 F )` `( P # → P # F )` Searches if P is a bisquared 4-th order equation. Returns a meta of factors and the multiplying coeff in that case.
281006	`^REALBICAR`	`( f1 #1 coef → meta rest T )`
113007	`^IROOTS`	`( P → list )` Finds integer roots of a polynomial. First available in ROM 1.11.

283006	`^EVIDENText`	`( P → meta cst_coeff )` Returns the roots of a polynomial P. Calls the 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`	`( P → {} )` Returns the roots of a 2nd order polynomial.
286006	`^METADEG2`	`( P → P meta )` Returns the roots of a 2nd order polynomial. P must be of order 1 or 2.
287006	`^METADEG1`	`( P → P meta )` Returns the roots of a 1st order polynomial. P must be of order 1.
288006	`^DEG1`	`( f → r )` Root of a first order factor. f is one level depth deeper than r.
289006	`^FDEG2ext`	`( P → meta-fact cst_coef )` Returns factors of a 2nd order polynomial and the corresponding multiplying coefficient. tests for 1st order polynomial.
28B006	`^RACTOFACext`	`( r → n d )` Converts root to factor. Factor is n/d, one level depth deeper than r.
28C006	`^FACTORACext`	`( f → 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 {} # → {} intob meta )` Like <REF>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 → 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,.
292006	^METASOLVOUT	
293006	^METASOLV2	(cst_coeff p → fr1 %m [fr2 %m] # cst_coeff) Returns roots/factors of 1st and 2nd order polynomials.
294006	^METASOLV4	(cst1 f1 ... fk #k cst2 → fr1 %m ... frn %m #2k cst_coeff) Returns factors or convert to roots if needed. #k=1,2 or 4, fk are of order 1 or 2.
295006	^ADDMULTIPL	(meta cst_coeff → 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.
29C006	^ID>DERext	(id → {} stripped_id)
093006	^ALG48MSOLV	(Lp → Lidnt Lsol) Calculates Groebner basis multivar solution. LAM3 must be bound to Lvar and LAM4 to Lidnt.
094006	^GMSOLV	(Lp → meta-sol) Calculates Groebner basis multivar solutions. LAM1 must be bound to the number 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 → G)
		Calculate Groebner basis.
		`G = { 1 }       if no solutions`
		`G = { 0 }       if identically true.`

096006	^GSOLVE	(Lp → Lg)
		Calculate factorized Groebner basis.
		`Lg = { Lg1 Lg2 .. Lgn }`
		`Lgi = independent solution (probably)`
		`Lg = {}          if no solutions`
		`Lg = { { 0 } }  if identically true.`

097006	^GFACTOR	(Lp fctr? → Lg) Calculate Groebner basis or factorized Groebner basis. Redundant bases are not removed.
098006	^GREDUCE	Interreduce basis. Lambda variables `{{ fctr? G k tmp todo Lg Irred }}`.
099006	^REDUCE	(p G → q) Reduces polynomial with respect to given basis.

09A006	`^FASTREDUCE`	(r P → q T / r P F) Assembly version of `REDUCE` for polynomials with short coefficients. Returns `FALSE` if an overflow occurs during the reduction. Assumes r is a genuine polynomial (not constant). Assumes G is not empty. Assumes G does not contain zeros (is trimmed).
37D006	`^ROOTM2ROOT`	(`{}/V` → `V'`) Transforms list of root/multiplicites to vector of roots.
0F2007	`^PASCAL_NEXTLINE`	(`{}` → `{}'`) Finds next line in the Pascal triangle.
0F3007	`^DELTAPSOLVE`	(`Q` → `P`) Solves P(x+1)-P(x)=Q(x). Internal polynomial function.

5.8 Calculus Operations

5.8.1 Limits and Series Expansion

46D006	`^LIMIText`	
46E006	`^REWRITEIFINF`	
46F006	`^SYMTAYLOR`	(`symb id %/z` → `symb`) Taylor series expansion around point 0 (McLaurin's series) with regard to given variable, and of the given order.
470006	`^SYMPAPRX`	
471006	`^TRUNCDL`	(`DL-l reste-l` → `truncated_DL`) Series expansion truncation.
472006	`^LIMSERIES!`	(`expression X=a|X %|zint` →) `a lim DL-l rest-l num-l/deno-l equiv-l lvar #` Series expansion. #=1 for X=a-h or X=-1/h.
473006	`^LIMITX!`	
474006	`^LIMITNOVX!`	
475006	`^LIMERR0!`	
476006	`^LIMERR1!`	
477006	`^LIMIT!`	(`symb` → `DL-l reste-l num-l/deno-l equiv.-l lim. lvar flag`) lim. = { symf direction }
478006	`^LIMSTEP1!`	(`symb` → `{ DL-l reste-l num-l/deno-l equiv.-l } flag`)
479006	`^LIMSTEP2!`	
47A006	`^LIMSTEP3!`	
47B006	`^LIMSTEP4!`	

47C006	^LIMLIM!	(# lvar equiv-l → lvar lim)
47D006	^n{}N	
47E006	^LIMLIM1!	
47F006	^LIMCMPL!	(reste-1-l reste-2-l → reste-l)
480006	^LIMEQUFR!	(n/d # → n/d-l equiv %)
481006	^LIMEQU!	({} # → {} / {}-equiv-l {}-equiv-l { # # # })
482006	^LIMEQU0!	
483006	^LIM+-!	(DL1...DLn #n op → DL flag) DL = { DL-l reste-l num-l/deno-l equiv-l }.
484006	^LIMERR10!	
485006	^LIMNEG!	
486006	^LIMRAC!	Racine carree, donc independant de x.
487006	^LIMINV!	
488006	^LIM/!	
489006	^LIMPOW!	
48A006	^LIMSQ!	
48B006	^LIM*!	
48C006	^LIMDIVPC!	(#ordre num-l deno-l → num-l deno-l)
48D006	^DIVPC!	
48E006	^LIMPROFEND!	(num deno #prof → num deno)
48F006	^LIMPROF!	
490006	^LIM%#!	(num-l deno-l {%...%} → num-l' deno-l' #prof {%...%})
491006	^LIMPROF0!	
492006	^LIMPROF1!	
493006	^LIMPROF2!	
494006	^LIMINVLN!	Operator INV[-LN].
495006	^LIMLN!	Operator LN.
496006	^LIMEXP!	
497006	^LIMSINCOS!	
498006	^LIMATAN!	
499006	^LIMASIN!	
49A006	^LIMSQRT!	
49B006	^LIMFLOOR!	
49C006	^LIMABS!	
49D006	^LPROF!	

49E006	`^LIM#VARX!`	(`lvar lvar` → `#varx`)
49F006	`^LIMBETA!`	
4A0006	`^LIMALPHA!`	
4A1006	`^HORNEXP!`	(`lim lvar X-l reste-l` → `lvar DL reste-l`)
4A2006	`^HORNCOS!`	
4A3006	`^HORNSIN!`	
4A4006	`^LIMSC0!`	
4A5006	`^LIMSC1!`	
4A6006	`^HORNATAN!`	
4A7006	`^LIMATAS!`	
4A8006	`^HORNASIN!`	
4A9006	`^HORNASIN1!`	
4AA006	`^HORNLN!`	
4AB006	`^LNOBJ!`	
4AC006	`^NEWLIMHORN`	
4AD006	`^LIMHORN!`	
4AE006	`^LRDM!`	
4AF006	`^LIMDL!`	
4B0006	`^LIMDLINF!`	
4B1006	`^LIMINFSIGN!`	
4B2006	`^LIMMAX!`	
4B3006	`^LIMCOMP!`	
4B4006	`^VARCOMP2!`	
4B5006	`^LIMSORT!`	
4B6006	`^VARCOMP!`	(`var1 var2` → `flag`)
4B7006	`^VARCOMPLN!`	
4B8006	`^VARCOMP3!`	
4B9006	`^VARCOMP31!`	
4BA006	`^VARCOMP32!`	(`var` → `0:`)
4BB006	`^VARCOMP33!`	
4BC006	`^LIMERR6!`	
4BD006	`^LIMVALOBJ!`	(`ob lvar` → `symb`)
4BE006	`^LIMVAL!`	(`ob` → `coeff val`)
4BF006	`^EQUIV!`	(`{} lequiv` → `equiv ordre`)
4C0006	`^LVARXNX2!`	(`ob` → `ob lvarx lvarnx`)
4C1006	`^SIMP1!`	
4C2006	`^FindCurVar`	(`symb` → `symb`) Sets a new current var if needed.

4C3006	`^LIMVAR!`	`( symb → symb lvar )`
4C4006	`^VAR%`	
15C006	`^RISCH13`	`( {}/{}' → {}'' )` Assuming `{}`' has length 1, divides all elements of `{}` by this element. Used by `RISCHext` and by SERIES to have a nicer output of series.

5.8.2 Derivatives

3DC006	`^PDer`	`( {} → der )`
19F006	`^ssSYMDER`	Algebraic derivative.
1A0006	`^SYMDER`	

1A1006	`^DERIVext`	`( ob id → ob' )`
		`( ob sym → ob' )`
		`( ob V → V' )`
		Calculates the derivative of the object. For a list argument calculates the gradient with respect to the variables in the list. If the variable is a symbolic, the first variable in it is used. Note that the gradient is a vector quantity, thus the result is returned as a list.

1A2006	`^siSYMDER`	
1A3006	`^DERIVIDNT`	`( ob id → ob' )` Main entry point for derivative with respect to a identifier.
1A4006	`^DERIVIDNT1`	`( ob → ob' )` Main entry point for derivative with respect to the identifier stored in LAM1.
1A5006	`^DERIV`	`( symb → symb' )` Derivative of symb with respect to the variable stored in LAM1.
1A6006	`^METADERIV`	`( Meta → Meta' )` Derivative of Meta object.
1BD006	`^METADER&NEG`	`( Meta → Meta' )` Meta derivative and negate.
1A8006	`^METADEROP`	Table of derivable functions and the respective derivative calculation subroutines.
1A9006	`^METADER+`	`( Meta&+ → Meta' )` Meta derivative of addition.
1AA006	`^METADER-`	`( Meta&- → 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') Meta derivative of power.
1AE006	^METADERFCN	(Meta → Meta') Meta derivative of a function.
1AF006	^METADERDER	(symb_id_; sym_fcn_; xDER #3 → Meta') Meta derivative of a derivative of a function.
1B0006	^METADERI4	(Meta → Meta') Meta derivative of a defined integral.
1B1006	^METADERI3	(Meta → Meta') Meta derivative of an undefined integral.
1B2006	^METADERIFTE	(Meta → Meta') Meta derivative of IFTE.
1B4006	^METADEREXP	(Meta → Meta') Meta derivative of EXP.
1B5006	^METADERLN	(Meta → Meta') Meta derivative of LN.
1B6006	^METADERLNP1	(Meta → Meta') Meta derivative of LNP1.
1B7006	^METADERLOG	(Meta → Meta') Meta derivative of LOG.
1B8006	^METADERALOG	(Meta → Meta') Meta derivative of ALOG.
1B9006	^METADERABS	(Meta → Meta') Meta derivative of ABS.
1BA006	^METADERINV	(Meta → Meta') Meta derivative of INV.
1BB006	^METADERNEG	(Meta → Meta') Meta derivative of NEG.
1BC006	^METADERSQRT	(Meta → Meta') Meta derivative of SQRT.
1BE006	^METADERSQ	(Meta → Meta') Meta derivative of SQ.
1BF006	^METADERSIN	(Meta → Meta') Meta derivative of SIN.
1C0006	^METADERCOS	(Meta → Meta') Meta derivative of COS.
1C1006	^METADERTAN	(Meta → Meta') Meta derivative of TAN.
1C2006	^METADERSINH	(Meta → Meta') Meta derivative of SINH.
1C3006	^METADERCOSH	(Meta → Meta') Meta derivative of COSH.
1C4006	^METADERTANH	(Meta → Meta') Meta derivative of TANH.
1C5006	^METADERASIN	(Meta → Meta') Meta derivative of ASIN.

1C6006	^METADERACOS	(Meta → Meta') Meta derivative of ACOS.
1C7006	^METADERATAN	(Meta → Meta') Meta derivative of ATAN.
1C8006	^METADERASH	(Meta → Meta') Meta derivative of ASINH.
1C9006	^METADERACH	(Meta → Meta') Meta derivative of ACOSH.
1CA006	^METADERATH	(Meta → Meta') Meta derivative of ATANH.
1B3006	^DERARG	(meta-symb → arg1 ... argk der1 ... derk #k op) Finds derivative of arguments.
1CB006	^pshder*	(Meta1 Meta2 → Meta2&Meta1'&*) Meta derivative utility.
1CC006	^SQRTINVpshd*	(Meta1 Meta2 → Meta2&SQRT&INV&Meta1'&*) Meta derivative utility.

5.8.3 Integration

07F007	^ODE_INT	(symb idnt → symb) Integration with addition of a constant.
2C5006	^IBP	(u'*v u → 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) IBP(ANS(1),sqrt(1-X^2)) IBP(ANS(1),C) the last step with an integral containing a cst C.
2D0006	^PREVALext	(symb inf sup x → 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 → symb inf sup vx) Warns user for singularity.
2D2006	^INText	(symb x → int[$,x, symb, xt]) Return unevaluated integral.
2D3006	^INT3	(f(x) x y → F(y) where F'=f) Undefined integration. No limit for underdetermined form.
3DD006	^INTEGRext	({} → prim)

5.8.4 Partial Fractions

3D2006	^PARTFRAC	(o → symb) Partial fraction expansion of o with respect to the current variable.
3D3006	^INPARTFRAC	(o list → 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 format.
3D4006	^PARTFRACRAT	
3D5006	^PFext	

5.8.5 Differential Equations

07E007	^DESOLVE	(list symb1 → list_sols) (symb symb1 → list_sols) Solves ordinary differential equation. For some ode's returned level2 is not symb1.
081007	^LDECSOLV	(2nd_member char_eq → solution) Linear differential equation with constant coefficients.
082007	^LDEGENE	(eq. carac → sol generale)
083007	^LDEPART	(2nd membre, eq carac → eq. carac, sol part)
084007	^LDSSOLVext	(V M → V') M is the matrix of the system. V is the vector of the 2nd members.
085007	^ODETYPESTO	(type →) Store ode type in variable ODETYPE.
086007	^ODE_SEPAR	(symb → symb symb-y symb-x T) (symb → symb F) Tries to separate symb as a product of a function of y and a function of x.

5.8.6 Laplace Transformation

087007	^LAPext	(symb → symb') Laplace transform for polynomial*exp/sin/cos. Returns LAP() for unknown transforms.
088007	^ILAPext	(symb → symb') Inverse Laplace transform for rational fractions. Delta functions for the integral part.
089007	^ILAPRAText	

08A007	`^ILAPDELTA`	
08B007	`^ILAPEXP`	`( ck rk → ck*exp[rk*x] )`
08C007	`^ILAPEXPSC`	

5.9 Summation

0F8007	`^QUOTExSIGMA`	
0F9007	`^SUM`	`( sym idnt → sym )` Internal `SUM`. The variable can be specified.
0FA007	`^FLAGSUM`	

0FB007	`^SUMVX`	`( sym → sym )`
		Internal `SUMVX`. Works always with respect to the current variable.
		`--`
		<REF>TEXT:Reserved|VX

0FC007	`^FLAGSUMVX`	
0FD007	`^RATSUM`	`( sym → sym )` Discrete rational sum.
0FE007	`^FTAYL`	`( f shift → f' )` Taylor shift for rational fractions.
0FF007	`^CSTFRACTION?`	`( ob → ob flag )` Taylor shift for rational fractions. Returns `TRUE` if ob is a cst fraction.
104007	`^HYPERGEO`	`( symb → symb )` Tests and does hypergeometric summation. First available in ROM 1.11.
100007	`^NONRATSUM`	`( z/symb → symb )` Discrete summation (hypergeometric case).
103007	`^meta_cst?`	`( meta → meta flag )` Tests for meta to be cst with respect to current var. First available in ROM 1.11.
105007	`^fk+1/fk`	First available in ROM 1.11.

108007	`^ZEILBERGER`	`( f(n,k) n k d → C T )`
		`( f(n,k) n k d → F )`
		Zeilberger algorithm * NOT IMPLEMENTED YET*. First available in ROM 1.11.

109007	`^SYMPSI`	`( sym → Psi(x) )` Digamma function. First available in ROM 1.11.
10A007	`^sympsi`	First available in ROM 1.11.
10B007	`^SYMPSIN`	`( sym int → Psi(x,n) )` Digamma function. First available in ROM 1.11.
10C007	`^sympsin`	First available in ROM 1.11.

11C007	^%%PSI	(%%x → %%) Digamma function. First available in ROM 1.11.
10D007	^IBERNOULLI	(#/zint → Q) Bernoulli numbers. First available in ROM 1.11.
0CD007	^PARITYTEST	
0CE007	^COSTEST	
0D9007	^NDEvalN/D	(num deno n d → num' deno') Evals list poly over a list fraction.
0DA007	^PEvalN/D	(P n d → num d #) Evals list poly over a list fraction.
3C1006	^vgerxssSYMSUM	(Meta2 Meta1 → meta) Symbolic sum with tests for two zints. lam'sumvar bound to 'id/lam' and lam'sumexpr to 'expr'.

5.10 Modular Operations

5.10.1 Modulo Operations

246006	^MAT*SCMOD	mat*scalar modulo.
247006	^SC*MATMOD	scalar*mat modulo.
248006	^MAT*MATMOD	mat*mat modulo.
249006	^DIVMOD	division modulo.
24A006	^GCD1MOD	GCD modulo.
24B006	^INVMOD	Inversion modulo for zint.
24C006	^MINVMOD	Inversion modulo for matrix of zint.
24D006	^FLAGDIV2MOD	Euclidean division modulo.
24E006	^FLAGPOWMOD	Power modulo.
24F006	^FLAGMPOWMOD	Matrix Power modulo.
250006	^EXPAMOD	expand modulo.
251006	^FLAGEXPAMOD	
252006	^FLAGFACTORMOD	(symb → symb) FACTOR modulo.
253006	^MFACTORMOD	(M → M') FACTOR modulo for amtrices.
254006	^RREFMOD	RREF modulo.

256006	`^LIFCext`	`( {contfrac} → fraction )` Converts continued fraction to rational.

5.10.2 Symmetric Modular Arithmetic

0E1006	`^PEvalMod`	`( Q Z Zn → Q' )` Computes value of polynomial mod Zn.
0E2006	`^QAddMod`	`( Q1 Q2 Zn → Q' )` Polynomial addition modulo Zn.
0E3006	`^QSubMod`	`( Q1 Q2 Zn → Q' )` Polynomial subtraction modulo Zn.
0E4006	`^QMulMod`	`( Q1 Q2 Zn → Q' )` Polynomial multiplication modulo Zn.
0E5006	`^QDivMod`	`( Q1 Q2 Zn → Qquo Qrem )` Polynomial division modulo Zn. In regular division the coefficients in the remainder can increase very quickly to tens of digits, thus it is important to normalize the coefficients whenever possible.
0E6006	`^QInvMod`	`( Q Zn → Q' )` Polynomial inversion modulo Zn.
0E7006	`^QGcdMod`	`( Q1 Q2 Zn → Q' )` Polynomial GCD modulo Zn for univariate polynomials. The result is made monic.
0E8006	`^QGcdExMod`	Extended polynomial GCD modulo Zn for univariate polynomials. The equation: Q1*Q1' + Q2*Q2' = 1 MOD Zn.
4C5006	`^ISOL1`	`( symb id → id symb' )`
4C6006	`^ISOLALL`	`( symb id → id {} )` Internal SOLVE.
4C7006	`^ISOL2ext`	`( symb id → symb' )` `( symb id → {} )` Like <REF>ISOL1 if isolflag is set. Otherwise returns the list of all found solutions.
4C8006	`^BEZOUTMSOLV`	`( Lpoly Lidnt → Lidnt sols )` If no extension in Lpoly, calls ALG48 `GSOLVE` Otherwise, solves by Bezout "Gaussian" elimination. 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 → list of roots )` Drops tagged roots.
4CA006	`^MHORNER`	`( poly-l {r1...rk} # → P[r1...rk] )` Top-level call. Poly-l might be a matrix.
4CB006	`^MHORNER1`	`( P { r } → P[..r..] )`

4CC006	`^SQFFext`	`( Q → { F1 mult1 .. Fn multn } )`
4CD006	`^MSQFF`	`( Q → F1 mult1 .. Fn multn #2n )` Full square-free factorization of object. The result is given as a Meta object.
4CE006	`^%1TWO`	`( ob → ob %1 #2 )` Square free factorization of unknown (?) object. See `MSQFF`.
4CF006	`^MZSQFF`	`( Z → Z1 mult1 .. Zn multn #2n )` Full factorization of an integer.
4D0006	`^MZSQFF1`	`( Meta curfac %n newfac T → Meta curfac %n+1 )` `( Meta curfac %n newfac F → Meta' newfac %1 )` Adds integer factor to factor list. If the factor is the same as the last time, only the multiplicity is increased.
4D2006	`^MLISTSQFF`	`( P → Meta )` Full square-free factorization of a polynomial with a recursive call on the GCD of all coefficients.
4D3006	`^METASQFFext`	`( P-list → S1 %1 ..Se-1 %e-1 %e ee Te Re )` Square-free factorization.
4DE006	`^LIDNText`	`( ob → {} )` 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?`	`( ob → flag )` Returns `TRUE` if symb is polynomial with respect to current variable.
4E1006	`^2POLYNOMIAL?`	`( symb1 symb2 → symb1 symb2 flag )` Returns `TRUE` if symb1 and symb2 are polynomial with respect to current variable.
4E2006	`^VXINDEP?`	`( symb → symb flag )` Returns `TRUE` if symb is independent of current variable.
4E3006	`^LVARXNX2ext`	
4E4006	`^RLVARext`	`( ob → {} )` Recursive search of all variables.
4E5006	`^LLVARDext`	`( o → #depth o lvar )`
4E6006	`^VXLVARext`	`( symb → symb lvar )`
4E7006	`^LVARext`	`( ob → ob {} )` List of variables. Square roots *are* included in the list of rational operators.

4E8006	^VX>LVARext	(ob → ob {}) Like <REF>LVARext but the current variable is added using >HCOMP. Square roots *are* included in the list of rational operators.
4E9006	^VX>	({} → {}') If VX is in the list then moves it to the beginning of the list. Otherwise does nothing. -- <REF>TEXT:Reserved|VX
4EA006	^VX!	({} → {}) If VX is in the list then moves it at the beginning. Otherwise VX is added to the beginning of the list. -- <REF>TEXT:Reserved|VX
4EC006	^LIDNTLVAR	(symb lidnt → symb lidnt lvar) lvar is the list of variables in symb, but elements of lidnt are moved to the beginning of lvar.
4ED006	^LISTOPRAC	(→ {}) Returns the list of rational operator with sqrt appended to the list.
4EE006	^LISTOPext	(→ {}) List of basic "rational" operators without square root.
4EF006	^LISTOPSQRT	(→ {}) 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" operators.
4F1006	^>VARLIST	
4F2006	^DEPTHext	(ob → #) Returns the max number of embedded lists in ob.
4F3006	^DEPTHOBJext	(objet # → depth)
4F6006	^TRIMOBJext	(ob → ob ') Trims object.
4F7006	^NEWTRIMext	(Q → Q) Recursively tests if Q is a list of one constant element. This is much faster than TRIMOBJext and sufficient for the output of programs which are trimmed on the fly.
4F8006	^>POLYTRIM	(meta → {}) Equivalent to {}POLY TRIMOBJext.
4F9006	^ELMGext	(ob → ob') Trims small numbers (less than epsilon).
51F006	^ZINTSQRT	
520006	^SHALT	

0E9006	^IsV>V?	(v1 v2 → flag) Returns TRUE if v1 is lexicographically after v2.
0EB006	^PZadic	(Q Z → Q')
104006	^LISTMAXext	(P → P Z T depth) (P → P ? F #0) Step 1 for gcdheu: Returns FALSE if gcdheu 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 → a b c pr[pgcd] A'/G' B'/G' flag) Heuristic GCD.

5.11 Sign Tables

237006	^SIGNE	(symb → sign) Compute the sign table of the expression with respect to the current variable. Internal version of the UserRPL command SIGNTAB.
0DC007	^SIGNE1ext	(expr → sign) Sign table of a polynomial or rational expression.
0DD007	^SIGNEext	
0DE007	^SIGNUNDEF	(→ sign) Returns undefined sign table.
0DF007	^SIGNPLUS	(→ sign) Returns always positive sign table.
0E0007	^SIGNMOINS	(→ sign) Returns always negative sign table.
0E1007	^SIGNELN	(sign → sign) Returns ln of a sign table.
0E2007	^SIGNEEXP	(sign → sign') Returns exp of a sign table.
0E3007	^SIGNESIN	(sign → sign') Returns sin of a sign table.
0E4007	^SIGNECOS	(sign → sign') Returns cos of a sign table.
0E5007	^SIGNETAN	(sign → sign') Returns tan of a sign table.
0E6007	^SIGNEATAN	(sign → sign') Returns atan of a sign table.
0E7007	^SIGNESQRT	(sign → sign') Returns sqrt of a sign table.
0E8007	^SUBSIGNE	(sign min max → sign') Truncates a sign table.
0E9007	^SIGNERIGHT	(sign ob → 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 → sign') Multiplies two sign tables.
0DB007	^POSITIFext	(ob → 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 → ob flag) Returns sign of an expression. Error if unable to find sign.
0F0007	^ZSIGNE	(ob → 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 → 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 → symb inf sup vx flag) Checks for singularities in expr.

5.12 Errors

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 expression" error.
581006	^NONUNARYERR	Error DE21h Generates "Non unary operator" error.

582006	^INTERNALERR	Error DE26h Generates "CAS internal error" error.
583006	^INVALIDOP	Error DE28h Generates "Operator not implemented (SERIES)" 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. dependent" 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" error.
58D006	^SWITCHNOTALLOWED	Error DE4Fh Generates "Mode switch not allowed here" error.
119007	^NONALGERR	Error DE50h Generates "Non algebraic in expression" error. First available in ROM 1.11.
58E006	^ERR$EVALext	(seco → action)
58F006	^Sys1IT	(ob →) Execute object if display flag is set.

5.13 CAS Configuration

08F007	^CFGDISPLAY	(→) Display current configuration of the CAS.
090007	^NEWVX	(→) Input new current variable from the user. -- <REF>TEXT:Reserved|VX
091007	^NEWMODULO	(→) Input new modulo from the user.

092007	^SWITCHON	(#flag →) Asks the user if a certain mode may be switched on by toggling system flag #flag. Errors if the user does not want to switch.
093007	^SWITCHOFF	(#flag →) Asks the user is a certain mode may be switched off by toggling system flag #flag. Error if the user does not want to switch.
094007	^FLAGNAME	(# → # $) Find the name of a flag.
1DC007	(^PUSHFLAGS)	(→) Internal version of User PUSH command: stores the current flag settings and path in the CASDIR/ENVSTK variable.
1DD007	(^POPFLAGS)	(→) Internal version of User POP command: pops the last pushed flag settings and path from the CASDIR/ENVSTK variable.
095007	^COMPLEXON	(→) Turns complex mode on. Depending on system flag 120, the user is asked first.
096007	^COMPLEXOFF	(→) Turns complex mode off. Depending on system flag 120, the user is asked first.
097007	^EXACTON	(→) Turns exact mode on. Depending on system flag 120, the user is asked first.
098007	^EXACTOFF	(→) Turns exact mode off. Depending on system flag 120, the user is asked first.
099007	^COMPLEXMODE	(→) Set complex mode, refresh configuration display.
09A007	^SETCOMPLEX	(→) Set complex mode.
09B007	^COMPLEX?	(→ flag) Test complex mode.
09C007	^REALMODE	(→) Set real mode, refresh configuration display.
09D007	^CLRCOMPLEX	(→) Set real mode.
09E007	^EXACTMODE	(→) Set exact mode, refresh configuration display.
09F007	^SETEXACT	(→) Set exact mode and gcd mode.
0A0007	^NUMMODE	(→) Set numeric mode, refresh configuration display.

0A1007	^CLREXACT	(→) Clear exact mode.
0A2007	^EXACT?	(→ flag) Test exact mode.
0A3007	^STEPBYSTEP	(→) Set step by step flag, refresh display.
0A4007	^NOSTEPBYSTEP	(→) Clear step by step flag, refresh display.
0A5007	^VERBOSEMODE	(→) Set verbose mode, refresh configuration display.
0A6007	^SILENTMODE	(→) Set silent mode, refresh configuration display.
0A7007	^RECURMODE	(→) Set recursive mode, refresh configuration display.
0A8007	^NONRECMODE	(→) Set nonrecursive mode, refresh configuration display.
0A9007	^PLUSAT0	(→) Set positive mode, refresh configuration display.
0AA007	^SETPLUSAT0	(→) Set positive mode.
0AB007	^PLUSATINFTY	(→) Set positive infinity mode, refresh configuration display.
0AC007	^CLRPLUSAT0	(→) Set positive infinity mode.
0AD007	^SPARSEDATA	(→) Set full data mode, refresh configuration display.
0AE007	^FULLDATA	(→) Set sparse mode, refresh configuration display.
0AF007	^RIGORMODE	(→) Set rigorous mode, refresh configuration display.
0B0007	^SLOPPYMODE	(→) Set sloppy mode, refresh configuration display.
0B1007	^SLOPPY?	(→ flag) Test sloppy mode.
1D2006	^SAVECASFLAGS	(→) Saves CAS flags and current var.
1D4006	^RESTORECASFLAGS	(→) Restore CAS flags and current var.
1D5006	^CASFLAGEVAL	(→) Execute next runstream object with flag protection.
0C2007	^RCLMODULO	(→ Z) Fetch MODULO from the home directory.

0C3007	^RCLPERIOD	(→ sym) Fetch PERIOD from the home directory.
0C4007	^RCLVX	(→ id) Fetch VX from home directory. -- <REF>TEXT:Reserved|VX
0C5007	^STOVX	(ob →) Store object in VX. -- <REF>TEXT:Reserved|VX
0C6007	^STOMODULO	(ob →) Store object in MODULO.
0C7007	^RCLEPS	(→ %) Fetch EPS from home directory.
0C8007	^ISIDREAL?	(id → id id T) (id → id F) Test if id is in the REALASSUME list.
0C9007	^ADDTOREAL	(id →) Add idnt to the list of real var.
0CA007	^RESETCASCFG	(→) Reset CAS config.
1D0006	^VERNUMext	(→ %version) CAS version number.

5.14 CAS Menus

1D1006	^MENUXYext	(#2 #1 → {}) Make list of Erable commands between the given numbers.
08D007	^MENUext	($6...$1 →) If the CAS quiet flag is not set, displays the six strings as menu keys. Otherwise does nothing.
0B2007	^MENUCHOOSE?	(→ prg flag) Return best CHOOSE command.
0B3007	^MENUCHOOSE	({} →) Offers a selection to the user. If Flag -117 is set, only installs a menu. If not, offer a CHOOSE box.
0B4007	^MENUGENE1	(→ {}) Menu for CAS.
0B5007	^MENUBASE1	(→ {}) Base algebra menu.
0B6007	^MENUCMPLX1	(→ {}) Complex operations menu.
0B7007	^MENUTRIG1	(→ {}) Trigonometric operations menu.

0B8007	^MENUMAT1	(→ {}) Matrix operations menu.
0B9007	^MENUARIT1	(→ {}) Arithmetic operations menu.
0BA007	^MENUSOLVE1	(→ {}) Solver menu.
0BB007	^MENUEXPLN1	(→ {}) Exponential and logarithmic operations menu.
0BC007	^MENUDIFF1	(→) Differential calculus menu.

5.15 Internal Version of UserRPL CAS Commands

218006	^ISPRIME	(z/% → %0/%1) Internal ISPRIME.
1D6006	^FLAGEXPAND	(symb → symb') Internal xEXPAND. Expands symbolic expression.
1D7006	^EXPANDBOTH	
1D8006	^FLAGFACTOR	(symb → symb') (z → symb) Internal xFACTOR. Factors symbolic or number.
1D9006	^FLAGLISTEXEC	(symb {} → symb') Internal xSUBST for the case that level 1 is an array or a matrix.
1DA006	^FLAGSYMBEXEC	(symb symb' → symb'') Internal xSUBST for the case that level 1 is a symbolic.
1DB006	^FLAGIDNTEXEC	(symb id → symb') Internal xSUBST for the case that level 1 is an id or a lam.
1DC006	^FLAGINTVX	(symb → symb') Internal xINTVX.
1DD006	^DERVX	(symb → symb') Internal xDERVX.
1DE006	^SOLVEXFLOAT	(% → {}) Internal xSOLVEVX for a float.
1DF006	^SYMLIMIT	(symb symb' → symb'') Internal xLIMIT for scalars.
1E0006	^FLAGMATRIXLIMIT	([] symb → []') Internal xLIMIT for matrices.
1E1006	^TAYLOR0	(symb → symb') Internal xTAYLOR0.
1E2006	^FLAGSERIES	(symb id z → {} symb') Internal xSERIES.
1E3006	^PLOTSTK	Internal PLOTSTK.

1E4006	^PLOTADD	(symb →) Internal xPLOTADD.
1E5006	^FLAGIBP	(symb1 symb2 → symb3 symb4) Internal xIBP.
1E6006	^FLAGPREVAL	(symb1 symb2 symb3 → symb4) Internal xPREVAL. Evaluates symb1 at the points symb2 and symb3 and takes the difference.
1E7006	^MATRIXRISCH	([] id → symb') Internal xRISCH for matrix arguments.
1E8006	^FLAGRISCH	(symb id → symb') Internal xRISCH for non-matrix argumetns.
1E9006	^FLAGDERIV	(symb id → symb') Internal xDERIV.
1EA006	^FLAGLAP	(symb → symb') Internal xLAP.
1EB006	^FLAGILAP	(symb → symb') Internal xILAP.
1EC006	^FLAGDESOLVE	(symb symb' → symb'') Internal xDESOLVE.
1ED006	^FLAGLDSSOLV	(symb1 symb2 → symb3) Internal xLDEC.
1EE006	^FLAGLDECSOLV	
1EF006	^FLAGTEXPAND	(symb → symb') Internal xTEXPAND.
1F0006	^FLAGLIN	(symb → symb') Internal xLIN.
1F1006	^FLAGTSIMP	(symb → symb') Internal xTSIMP.
1F2006	^FLAGLNCOLLECT	(symb → symb') Internal xLNCOLLECT.
1F3006	^FLAGEXPLN	(symb → symb') Internal xEXPLN.
1F4006	^FLAGSINCOS	(symb → symb') Internal xSINCOS.
1F5006	^FLAGTLIN	(symb → symb') Internal xTLIN.
1F6006	^FLAGTCOLLECT	(symb → symb') Internal TCOLLECT.
1F7006	^FLAGTRIG	(symb → symb') Internal xTRIG.
1F8006	^FLAGTRIGCOS	(symb → symb') Internal xTRIGCOS.
1F9006	^FLAGTRIGSIN	(symb → symb') Internal xTRIGSIN.
1FA006	^FLAGTRIGTAN	(symb → symb') Internal xTRIGTAN.
1FB006	^FLAGTAN2SC	(symb → symb') Internal xTAN2SC.

1FC006	^FLAGHALFTAN	(symb → symb') Internal xHALFTAN.
1FD006	^FLAGTAN2SC2	(symb → symb') Internal xTAN2SC2.
1FE006	^FLAGATAN2S	(symb → symb') Internal xATAN2S.
1FF006	^FLAGASIN2T	(symb → symb') Internal xASIN2T.
200006	^FLAGASIN2C	(symb → symb') Internal xASIN2C.
201006	^FLAGACOS2S	(symb → symb') Internal xACOS2S.
206006	^STEPIDIV2	(z1 z2 → z3 z4) Internal xIDIV2.
207006	^FLAGDIV2	(symb1 symb2 → symb3 symb4) Internal xDIV2.
208006	^FLAGGCD	(symb1 symb2 → symb3) Internal xGCD for the case with two symbolica arguments.
209006	^PEGCD	(symb1 symb2 → symb3 symb4 symb5) Internal xEGCD for polynomials.
20B006	^ABCUV	(symb1 symb2 symb3 → symb4 symb5) Internal polynomial xABCUV.
20C006	^IABCUV	(z1 z2 z3 → z4 z5) Internal integer xIABCUV.
20D006	^FLAGLGCD	({} → {} symb) Internal xLGCD.
20E006	^FLAGLCM	(symb1 symb2 → symb3) Internal xLCM.
20F006	^FLAGSIMP2	(symb1 symb2 → symb3 symb4) Internal xSIMP2.
210006	^FLAGPARTFRAC	(symb → symb') Internal xPARTFRAC.
211006	^FLAGPROPFRAC	(symb → symb') Internal xPROPFRAC.
212006	^FLAGPTAYL	(P(X) r → P(X+r)) Internal xPTAYL.
213006	^FLAGHORNER	(symb1 symb2 → symb3 symb4 symb5) Internal xHORNER.
214006	^EULER	(z → z') Internal xEULER.
216006	^FLAGCHINREM	(A1 A2 → A3) Internal xCHINREM.
217006	^ICHINREM	(A1 A2 → A3) Internal xICHINREM.
219006	^SOLVE1EQ	(symb id → {}) Internal xSOLVE for single equations.
21A006	^SOLVEMANYEQ	([] []' → {}'') Internal xSOLVE for arrays of equations.

21B006	^ZEROS1EQ	(symb id → {}) Internal xZEROS for single equations.
21C006	^ZEROSMANYEQ	([] []' → {}) Internal xZEROS for arrays of equations.
21D006	^FCOEF	([] → symb) Internal xFCOEF.
21E006	^FROOTS	(symb → []) Internal xFROOTS.
21F006	^FACTORS	(symb → {}) Internal xFACTORS.
220006	^DIVIS	(symb → {}) Internal xDIVIS.
221006	^STUDMULT	Internal xSTUDMULT.
222006	^STUDDIV	Internal xSTUDDIV.
223006	^rref	(M → A M') Internal xrref.
229006	^MADNOCK	(M → symb1 []' []'' symb3) Internal xMAD.
22A006	^SYSTEM	([] []' → []'' {} []''') Internal xLINSOLVE.
22B006	^VANDERMONDE	({} → M) Internal xVANDERMONDE.
22C006	^HILBERTNOCK	(z → M) Internal xHILBERT.
22E006	^CURL	([exprs] [vars] → []) Internal xCURL.
22F006	^DIVERGENCE	([exprs] [vars] → symb) Internal xDIV.
230006	^LAPLACIAN	([expr] [vars] → symb) Internal xLAPL.
231006	^HESSIAN	(symb A → M A' A'') Internal xHESS.
232006	^HERMITE	(z → symb) Internal xHERMITE.
233006	^TCHEBNOCK	(%degree → symb) Internal xTCHEBYCHEFF.
234006	^LEGENDRE	(z → symb) Internal xLEGENDRE.
235006	^LAGRANGE	(A → symb) Internal xLAGRANGE.
236006	^FOURIER	(symb z → C%) Internal xFOURIER.
238006	^TABVAR	(symb → symb {{}} grob) Internal xTABVAR.
239006	^FLAGDIVPC	(symb1 symb2 z → symb3) Internal xDIVPC.

23A006	^FLAGTRUNC	(symb1 symb2 → symb3) Internal xTRUNC.
23B006	^FLAGSEVAL	(symb → symb') Internal xSEVAL.
23C006	^XNUM	(symb → symb') Internal xXNUM.
23D006	^REORDER	(symb id → symb') Internal xREORDER.
23E006	^USERLVAR	(symb → symb []) Internal xLVAR.
23F006	^USERLIDNT	(symb → []) Internal xLNAME.
241006	^ADDTMOD	(symb1 symb2 → symb3) Internal xADDTMOD for scalars.
242006	^MADDTMOD	(M M' → M'') Internal xADDTMOD for matrices.
243006	^SUBTMOD	(symb1 symb2 → symb3) Internal xSUBTMOD for scalars.
244006	^MSUBTMOD	(M M' → M'') Internal xSUBTMOD for matrices.
245006	^MULTMOD	(symb1 symb2 → symb3) Internal xMULTMOD.

5.16 Miscellaneous

5.16.1 Verbose Mode Display Routines

579006	^Verbose1	($ →) 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	($ # →) Display message on given line if verbose mode on.

5.16.2 Evaluation

257006	^EvalNoCKx*	(ob ob' → ob'')
258006	^EvalNoCKx+	(ob ob' → ob'')
259006	^EvalNoCKx-	(ob ob' → ob'')
25A006	^EvalNoCKx/	(ob ob' → ob'')
25B006	^EvalNoCKx^	(ob ob' → ob'')
25C006	^EvalNoCKxCHS	(ob → ob')

25D006	`^EvalNoCKxINV`	(ob → ob')
25E006	`^EvalNoCKxMOD`	(ob ob' → ob'')
25F006	`^EvalNoCKxPERM`	(ob ob' → ob'')
260006	`^EvalNoCKxCOMB`	(ob ob' → ob'')
261006	`^EvalNoCKxOR`	(ob ob' → ob'')
262006	`^EvalNoCKxAND`	(ob ob' → ob'')
263006	`^EvalNoCKxXOR`	(ob ob' → ob'')
264006	`^EvalNoCKxXROOT`	(ob ob' → ob'')
265006	`^TABVALext`	(fnct x {} → {}') Table of values.

5.16.3 Conversion

266006	`^TOLISText`	(o1..on #n → Lvar Q1..Qn) Convert meta of symbolic objects to internal form.

267006	`^FROMLISText`	(Lvar Meta L → 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 } } }.`

5.16.4 Qpi

074007	`^QPI`	(ob → ob') Internal `xXQ`.
073007	`^QpiZ`	(ob → symb) Calls `^Qpi%` and converts the resulting (real) integers into zints.
075007	`^QpiSym`	(symb → symb') Internal `xXQ` for symbolics.
076007	`^QpiArry`	([] → []') Internal `xXQ` for arrays. Converts each element of the array.
077007	`^QpiList`	({} → {}') Internal `xXQ` for lists. Converts each element of the list.

078007	^Qpi	(%/C% → symb) Internal xXQ for real and complex numbers.
079007	^Qpi%	(% → 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 numbers 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. Returns numerator %' and denominator %". The accuracy of the approximation is determined by the current display format.

5.16.5 Infinity

2E2006	^INFINIext	(→ '∞')
2E3006	^MINUSINFext	(→ '-∞')
2E4006	^PLUSINFext	(→ '+∞')
2E5006	^?ext	(→ '?') Pushed the undefined symbolic.
2E6006	^POSINFext	(symb → symb #) Returns #1 if the symbolic contains '∞'.
2E1006	^TESTINFINI	(ob → ob flag) Test if object contains infinity.
2E7006	^POSUNDEFext	(symb → symb #) Returns #1 if the symbolic contains the undefined symbolic '?'.

5.16.6 Built-In Constants

2EA006	^pi	(→ 'π')
2EB006	^metapi	(→ π #1)
2F1006	^meta-pi	(→ π xNEG #2)
2E8006	^pisur2	(→ 'π/2')
2F2006	^metapi/2	(→ π 2 x/ #3)
2E9006	^pisur-2	(→ '-π/2')
2F4006	^meta-pi/2	(→ π 2 x/ xNEG #4)
2F3006	^metapi/4	(→ π 4 x/ #3)

2F5006	`^meta-pi/4`	(→ π 4 x/ xNEG #4)
2F6006	`^pifois2`	(→ '2*π')
2EC006	`^'xPI`	(→ xPI)
2F9006	`^base_ln`	(→ 'e')
2FA006	`^meta_e`	(→ e #1)
2EE006	`^'xi`	(→ xi)
2ED006	`^metai`	(→ i #1)
2EF006	`^ipi`	(→ 'i*π')
2F0006	`^metaipi`	(→ i π x* #3)
2F8006	`^metapi*2`	(→ π 2 x* #3)
2F7006	`^deuxipi`	(→ '2*i*π')

5.16.7 List Application

3F0006	`^DIVOBJext`	({o1...on} ob → {o1/ob...on/ob}) Division of all elements of a list by ob. Tests if ob=1.
3F2006	`^LOPDext`	({o1...on} ob → {o1/ob...on/ob}) `LOPDext` calls `QUOText` for the division, unlike DIVOBJ which calls `RDIVext`.
269006	`^LOP1ext`	({} ob binop → {}') Applies non-recursively << ob binop >> to the elements of the list.
26A006	`^LOPAext`	({} ob binop → {}') Applies recursively << op binop >> to the elements of the list (not the list elements themselves).
10F006	`^LOPMext`	(ob {} → {}') Multiplies each element of the list by the given object.
45F006	`^LISTEXEC`	(ob {} → ob') (ob {} → {}') 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`	({} prog → {}) Executes prog on each element of ob.
268006	`^PFEXECext`	(symb prg → symb)
26B006	`^LISTSECOext`	(composite → composite) Applies 1LAM non-recursively to all elements of the list.

26D006	^CK1TONOext	(ob → ob') Applies prg to ob, recursively for lists. prg is fetched from runstream.

5.16.8 Irrquads

167006	^TYPEIRRQ?	(ob → flag) Is ob an irrquad?
168006	^DTYPEIRRQ?	(ob → ob flag) DUP, then ^TYPEIRRQ?.
165006	^QXNDext	(irrq → a b c) b=0 and c=1 if stack level 1 is not an irrq.
166006	^NDXQext	(a b c → irrq)
2D8006	^IRRQ#ULTIMATE	(ob → # c) Finds ≪ depth and returns ultimate c of an irrq.
508006	^QCONJext	(irrq → irrq') irrq-conjugate of an irrq. This is *not* the complex conjugate.
509006	^QABSext	(irrq → irrq sign) Finds the sign of an irrq. Work always if irrq is made of Z.
51A006	^QNORMext	(Zirr → a^2-b*c^2) Irrq-norm of an irrquad. This is *not* the complex modulus.
4D4006	^SECOSQFFext	(:: x<< a b c x>> ; → { fact1 mult1 ... factn multn }) Factorization of irrquads and Gauss integers.
124006	^PREPARext	(o1 o2 → a1 b1 c1 a2 b2 c2) Returns irrquad decomposition of o1 and o2. with either c1=c2 or c1 and c2 have no factors in comon. c1<c2, ordering handled by LESSCOMPLEX? is made by type, then by CRC.
2DA006	^LISTIRRQ	(ob {} → {}') Add the C-part of all irrquads of object to the list.

5.16.9 Miscellaneous

0DA006	^QMODSYMext	
0DB006	^ModPow	
0DC006	^ZQUOText	
0DE006	^ZDIVext	
3E7006	^PSEUDOPREP	(o2 o1 → o2*a1.n^ o1 a1.n^)
3FA006	^PLCZ	

3FB006	`^HSECO2RCext`	`( ob → ob' )` Conversion of constants from internal to user form.
3FC006	`^SECO2CMPext`	`( seco → symb )` Back conversion of complex. polarflag should be disabled if not at the top level of rational expressions.
3FD006	`^SECO2CMPPOL`	Conversion of a complex in polar coordinates. should be used only at the top level of rational expr.
3FE006	`^SECO2CMPCART`	Conversion of a complex in cartesian coordinates.
3FF006	`^VALOBJext`	`( # {..{Q}..} {var1..varn} → {..{ob}..} )` Back conversion of objects embedded at depth # in lists. Simplifies var1..varn.
401006	`^VAL2ext`	`( # {..{Q}..} {var1..varn} → {..{ob}..} )` Back conversion of objects embedded at depth # in lists. Does not simplify var1..varn. Conversion is done in asc. power if positivfflag is set, which is useful for SERIES and LIMIT commands.
402006	`^INVAL2`	`( P # → symbpoly )` LAM2 must contain Lvar, # is the depth.
403006	`^METAVAL2`	`( # Meta_list → Meta_symb )` LMA2 must contain Lvar, LAM1 is modified.
404006	`^VAL1`	`( ob → ob )` LAM2 must contain Lvar, LAM1 is modified.
405006	`^VAL1M`	`( ob → Meta_symb )` LAM2 must contain Lvar, LAM1 is modified.
45C006	`^IDNTEXEC`	`( symb idnt → symb' )` Tries to find `idnt` such that symb=0. Return a solution as an equality 'idnt=..' in symb'.
45D006	`^SYMISOL`	
45E006	`^SYMQFORM`	
121006	`^MP0`	`( ob → ob 1 )` Returns number 1 of the selected type. The symbolic/ROMPTR one looks very strange it is used to avoid infinity^0/undef^0 to return 1.

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.
29D006	^SIMPIDNT	(idnt → ob) Evaluates idnt (looks recursively for its content if defined). Does not error for circular definition, but displays a warning.
29E006	^RCLALLIDNT	
29F006	^RCL1IDNT	(idnt/lam → ob) Recursive content of an idnt. LAM1 to LAM3 must be bound.
2A7006	^SWPSIMPNDXF	(ob2 ob1 → ob1/ob2) Simplified fraction (internal).
2A8006	^SIMPNDXFext	(ob2 ob1 → ob2/ob1) Simplified fraction (internal).
2B6006	^CMODext	(C2 C1 → C1 C2_mod_C1)
2BD006	^SQFF2ext	(l1...ln #n-1 → l1'...ln' #n-1)
2BE006	^PPZ	(p → 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 → ob zint) PPZ with further check to ensure returning a zint. First available in ROM 1.11.
2BF006	^PZHSTR	(a z → a mod z)
2C0006	^HORNER1ext	(P r → P[r])
2C1006	^PEval	(P r → P[r]) P must be a list polynomial.
2C2006	^RISCHext	
2C3006	^risch/	
2C4006	^rischABS	
2C6006	^SQRT_IN?	({} → {} flag) Returns TRUE if one element of {} is a symb containing a sqrt.
2C7006	^IS_SQRT?	(symb → flag)
2C8006	^XROOT_IN?	

2C9006	^IS_XROOT?	(symb → flag)
2CA006	^STOPRIMIT	(symb →) Stores antiderivative in PRIMIT variable.
2CB006	^CONTAINS_LN?	(symb → symb flag)
2CC006	^ISNT_IDNT?	
2CD006	^RISCHPF	
2CE006	^RISCHRAT	
2CF006	^rischlogpart	
2D4006	^FOURIERext	(symb n → cn) Computes n-th Fourier coefficient of a 2 π periodic function.
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.
2DB006	^LIST1i-1-i	Various constants. Caution: these constants are "covered" if you are using them be sure to return an uncovered result on the stack when exiting.
2DC006	^LIST10-10	
2DD006	^TABLECOSext	(→ {}) Table of special COS values (k*pi/12).
2DE006	^TABLETANext	(→ {}) Table of special TAN values (k*pi/12).
101007	^LINEARAPPLY	(symb nonrat_prg rat_prg → symb) Applies linearity. nonrat_prg is applied for a non rational part symb → symb. rat_prg is applied for a rational part symb → symb. Linearity is applied on symb.
102007	^linearapply	First available in ROM 1.11.
106007	^A/B2PQR	(A B → P Q R) Writes a fraction A/B as E[P]/P*Q/E[R]. Q and positive shifts of R are prime together. First available in ROM 1.11.
107007	^GOSPER?	(P Q R → P R Y T) (P Q R → F) Solves P = Q E[Y] - R Y for Y. First available in ROM 1.11.
0CB007	^FRACPARITY	(fr → 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 → N D %) Extract trivial power of fraction.
4D1006	^MSECOSQFF	(ob → Meta) Factorization of an extension.

6 Entries specific to the HP38/39/40

6.1 Topic Variables and the Topic Outer Loop

These entries are used for the implemtation of applets on the HP38G/39G/40G. On the HP49G, they are included for Hp38/39/40 compatibility, probably in order to allow applet devellopment on the HP49G.

2E2CD	(TopOuterLoop)	
2E3DE	(TOLSaveUI)	
2E451	(TOLSetTopicUI)	
2E46F	(TOLSetTopUI.1)	
2E4AB	(TOLSetViewUI)	
2E4C9	(TOLSetViUI.1)	
2E51E	(TOLKeyUI)	
2E573	(TOLErrorTrap)	
2E5A5	(TOLResUI&Err)	
2E5C3	(TOLRestoreUI)	
2E659	(?ExitThisTop)	
2E686	(BadTOLUI?)	
2E68B	(SetBadTOLUI)	
2E690	(ClrBadTOLUI)	
2E698	(CALCCXT!)	(ob →)
2E69D	(CALCCXT@)	(→ ob)
2E6A7	(PGMCXT!)	(ob →)
2E6AC	(PGMCXT@)	(→ ob)
2E6B6	(NOTESCXT!)	(ob →)
2E6BB	(NOTESCXT@)	(→ ob)
2E6C5	(apletPTR!)	(ob →)
2E6CA	(apletPTR@)	(→ ob)
2E6D4	(funcPTR!)	(ob →)
2E6D9	(funcPTR@)	(→ ob)
2E6E3	(polarPTR!)	(ob →)
2E6E8	(polarPTR@)	(→ ob)
2E6F2	(paramPTR!)	(ob →)
2E6F7	(paramPTR@)	(→ ob)
2E701	(seqPTR!)	(ob →)
2E706	(seqPTR@)	(→ ob)

2E710	(statPTR!)	(ob →)
2E715	(statPTR@)	(→ ob)
2E71F	(solvePTR!)	(ob →)
2E724	(solvePTR@)	(→ ob)
2E72E	(otherPTR!)	(ob →)
2E733	(otherPTR@)	(→ ob)
2E73D	(TopicDoN)	
2E76A	(TopicVar1!)	(ob →)
2E76B	(TopicVar1@)	(→ ob)
2E76C	(TopicVar2!)	(ob →)
2E76D	(TopicVar2@)	(→ ob)
2E76E	(TopicVar3!)	(ob →)
2E76F	(TopicVar3@)	(→ ob)
2E770	(TopicVar4!)	(ob →)
2E771	(TopicVar4@)	(→ ob)
2E772	(TopicVar5!)	(ob →)
2E773	(TopicVar5@)	(→ ob)
2E774	(TopicVar6!)	(ob →)
2E775	(TopicVar6@)	(→ ob)
2E776	(TopicVar7!)	(ob →)
2E777	(TopicVar7@)	(→ ob)
2E778	(TopicVar8!)	(ob →)
2E779	(TopicVar8@)	(→ ob)
2E77A	(TopicVar9!)	(ob →)
2E77B	(TopicVar9@)	(→ ob)
2E77C	(TopicVar10!)	(ob →)
2E77D	(TopicVar10@)	(→ ob)
2E77E	(TopicVar11!)	(ob →)
2E77F	(TopicVar11@)	(→ ob)
2E780	(TopicVar12!)	(ob →)
2E781	(TopicVar12@)	(→ ob)
2E782	(TopicVar13!)	(ob →)
2E783	(TopicVar13@)	(→ ob)
2E784	(TopicVar14!)	(ob →)
2E785	(TopicVar14@)	(→ ob)
2E786	(TopicVar15!)	(ob →)
2E787	(TopicVar15@)	(→ ob)
2E788	(TopicVar16!)	(ob →)

2E789	(TopicVar16@)	($\rightarrow$ ob)
2E78A	(TopicVar17!)	(ob $\rightarrow$)
2E78B	(TopicVar17@)	($\rightarrow$ ob)
2E78C	(TopicVar18!)	(ob $\rightarrow$)
2E78D	(TopicVar18@)	($\rightarrow$ ob)
2E78E	(TopicVar19!)	(ob $\rightarrow$)
2E78F	(TopicVar19@)	($\rightarrow$ ob)
2E790	(TopicVar20!)	(ob $\rightarrow$)
2E791	(TopicVar20@)	($\rightarrow$ ob)
2E792	(TopicVar21!)	(ob $\rightarrow$)
2E793	(TopicVar21@)	($\rightarrow$ ob)
2E794	(TopicVar22!)	(ob $\rightarrow$)
2E795	(TopicVar22@)	($\rightarrow$ ob)
2E796	(TopicVar23!)	(ob $\rightarrow$)
2E797	(TopicVar23@)	($\rightarrow$ ob)
2E798	(TopicVar24!)	(ob $\rightarrow$)
2E799	(TopicVar24@)	($\rightarrow$ ob)
2E79A	(TopicVar25!)	(ob $\rightarrow$)
2E79B	(TopicVar25@)	($\rightarrow$ ob)
2E79C	(TopicVar26!)	(ob $\rightarrow$)
2E79D	(TopicVar26@)	($\rightarrow$ ob)
2E79E	(TopicVar27!)	(ob $\rightarrow$)
2E79F	(TopicVar27@)	($\rightarrow$ ob)
2E7A0	(TopicVar28!)	(ob $\rightarrow$)
2E7A1	(TopicVar28@)	($\rightarrow$ ob)
2E7A2	(TopicVar29!)	(ob $\rightarrow$)
2E7A3	(TopicVar29@)	($\rightarrow$ ob)
2E7A4	(TopicVar30!)	(ob $\rightarrow$)
2E7A5	(TopicVar30@)	($\rightarrow$ ob)
2E7A6	(TopicVar31!)	(ob $\rightarrow$)
2E7A7	(TopicVar31@)	($\rightarrow$ ob)
2E7A8	(TopicVar32!)	(ob $\rightarrow$)
2E7A9	(TopicVar32@)	($\rightarrow$ ob)
2E7AA	(TopicVar33!)	(ob $\rightarrow$)
2E7AB	(TopicVar33@)	($\rightarrow$ ob)
2E7AC	(TopicVar34!)	(ob $\rightarrow$)
2E7AD	(TopicVar34@)	($\rightarrow$ ob)
2E7AE	(TopicVar35!)	(ob $\rightarrow$)

2E7AF	(TopicVar35@)	(→ ob)
2E7B0	(TopicVar36!)	(ob →)
2E7B1	(TopicVar36@)	(→ ob)
2E7B2	(TopicVar37!)	(ob →)
2E7B3	(TopicVar37@)	(→ ob)
2E7B4	(TopicVar38!)	(ob →)
2E7B5	(TopicVar38@)	(→ ob)
2E7B6	(TopicVar39!)	(ob →)
2E7B7	(TopicVar39@)	(→ ob)
2E7B8	(TopicVar40!)	(ob →)
2E7B9	(TopicVar40@)	(→ ob)
2E7BA	(TopicVar41!)	(ob →)
2E7BB	(TopicVar41@)	(→ ob)
2E7BC	(TopicVar42!)	(ob →)
2E7BD	(TopicVar42@)	(→ ob)
2E7BE	(TopicVar43!)	(ob →)
2E7BF	(TopicVar43@)	(→ ob)
2E7C0	(TopicVar44!)	(ob →)
2E7C1	(TopicVar44@)	(→ ob)
2E7C2	(TopicVar45!)	(ob →)
2E7C3	(TopicVar45@)	(→ ob)
2E7C4	(TopicVar46!)	(ob →)
2E7C5	(TopicVar46@)	(→ ob)
2E7C6	(TopicVar47!)	(ob →)
2E7C7	(TopicVar47@)	(→ ob)
2E7C8	(TopicVar48!)	(ob →)
2E7C9	(TopicVar48@)	(→ ob)
2E7CA	(TopicVar49!)	(ob →)
2E7CB	(TopicVar49@)	(→ ob)
2E7CC	(TopicVar50!)	(ob →)
2E7CD	(TopicVar50@)	(→ ob)
2E7CE	(TopicVar51!)	(ob →)
2E7CF	(TopicVar51@)	(→ ob)
2E7D0	(TopicVar52@)	(ob →)
2E7D1	(TopicVar52!)	(→ ob)
2E7D2	(TopicVar53@)	(ob →)
2E7D3	(TopicVar53!)	(→ ob)
2E7D4	(TopicVar54@)	(ob →)

2E7D5	(TopicVar54!)	($\rightarrow$ ob)
2E7D6	(TopicVar55@)	(ob $\rightarrow$)
2E7D7	(TopicVar55!)	($\rightarrow$ ob)
2E7D8	(TopicVar56@)	(ob $\rightarrow$)
2E7D9	(TopicVar56!)	($\rightarrow$ ob)
2E7DA	(TopicVar57@)	(ob $\rightarrow$)
2E7DB	(TopicVar57!)	($\rightarrow$ ob)
2E7DC	(TopicVar58@)	(ob $\rightarrow$)
2E7DD	(TopicVar58!)	($\rightarrow$ ob)
2E7DE	(TopicVar59@)	(ob $\rightarrow$)
2E7DF	(TopicVar59!)	($\rightarrow$ ob)
2E7E0	(TopicVar60@)	(ob $\rightarrow$)
2E7E1	(TopicVar60!)	($\rightarrow$ ob)
2E7E2	(TopicVar61@)	(ob $\rightarrow$)
2E7E3	(TopicVar61!)	($\rightarrow$ ob)
2E7E4	(TopicVar62@)	(ob $\rightarrow$)
2E7E5	(TopicVar62!)	($\rightarrow$ ob)
2E7E6	(TopicVar63@)	(ob $\rightarrow$)
2E7E7	(TopicVar63!)	($\rightarrow$ ob)
2E7E8	(TopicVar64@)	(ob $\rightarrow$)
2E7E9	(TopicVar64!)	($\rightarrow$ ob)
2E7EA	(TopicVar65@)	(ob $\rightarrow$)
2E7EB	(TopicVar65!)	($\rightarrow$ ob)
2E7EC	(TopicVar66@)	(ob $\rightarrow$)
2E7ED	(TopicVar66!)	($\rightarrow$ ob)
2E7EE	(TopicVar67@)	(ob $\rightarrow$)
2E7EF	(TopicVar67!)	($\rightarrow$ ob)
2E7F0	(TopicVar68@)	(ob $\rightarrow$)
2E7F1	(TopicVar68!)	($\rightarrow$ ob)
2E7F2	(TopicVar69@)	(ob $\rightarrow$)
2E7F3	(TopicVar69!)	($\rightarrow$ ob)
2E7F4	(TopicVar70@)	(ob $\rightarrow$)
2E7F5	(TopicVar70!)	($\rightarrow$ ob)
2E7F6	(TopicVar71@)	(ob $\rightarrow$)
2E7F7	(TopicVar71!)	($\rightarrow$ ob)
2E7F8	(TopicVar72@)	(ob $\rightarrow$)
2E7F9	(TopicVar72!)	($\rightarrow$ ob)
2E7FA	(TopicVar73@)	(ob $\rightarrow$)

2E7FB	(TopicVar73!)	(→ ob)
2E7FC	(TopicVar74@)	(ob →)
2E7FD	(TopicVar74!)	(→ ob)
2E7FE	(TopicVar75@)	(ob →)
2E7FF	(TopicVar75!)	(→ ob)
2E800	(TopicVar76@)	(ob →)
2E801	(TopicVar76!)	(→ ob)
2E802	(TopicVar77@)	(ob →)
2E803	(TopicVar77!)	(→ ob)
2E804	(TopicVar78@)	(ob →)
2E805	(TopicVar78!)	(→ ob)
2E806	(TopicVar79@)	(ob →)
2E807	(TopicVar79!)	(→ ob)
2E808	(TopicVar80@)	(ob →)
2E809	(TopicVar80!)	(→ ob)
2E80A	(TopicVar81@)	(ob →)
2E80B	(TopicVar81!)	(→ ob)
2E80C	(TopicVar82@)	(ob →)
2E80D	(TopicVar82!)	(→ ob)
2E80E	(TopicVar83@)	(ob →)
2E80F	(TopicVar83!)	(→ ob)
2E810	(TopicVar84@)	(ob →)
2E811	(TopicVar84!)	(→ ob)
2E812	(TopicVar85@)	(ob →)
2E813	(TopicVar85!)	(→ ob)
2E814	(TopicVar86@)	(ob →)
2E815	(TopicVar86!)	(→ ob)
2E816	(TopicVar87@)	(ob →)
2E817	(TopicVar87!)	(→ ob)
2E818	(TopicVar88@)	(ob →)
2E819	(TopicVar88!)	(→ ob)
2E81A	(TopicVar89@)	(ob →)
2E81B	(TopicVar89!)	(→ ob)
2E81C	(TopicVar90@)	(ob →)
2E81D	(TopicVar90!)	(→ ob)
2E81E	(TopicVar91!)	(ob →)
2E81F	(TopicVar91@)	(→ ob)
2E820	(TOLVar1!)	(ob →)

2E821	(TOLVar1@)	(→ ob)
2E822	(TOLVar2!)	(ob →)
2E823	(TOLVar2@)	(→ ob)
2E824	(TOLVar3!)	(ob →)
2E825	(TOLVar3@)	(→ ob)
2E826	(TOLVar4!)	(ob →)
2E827	(TOLVar4@)	(→ ob)
2E828	(TOLVar5!)	(ob →)
2E829	(TOLVar5@)	(→ ob)
2E82A	(TOLVar6!)	(ob →)
2E82B	(TOLVar6@)	(→ ob)
2E82C	(TOLVar7!)	(ob →)
2E82D	(TOLVar7@)	(→ ob)
2E82E	(TOLVar8!)	(ob →)
2E82F	(TOLVar8@)	(→ ob)
2E830	(TOLVar9!)	(ob →)
2E831	(TOLVar9@)	(→ ob)
2E832	(TOLVar10!)	(ob →)
2E833	(TOLVar10@)	(→ ob)
2E834	(TOLVar11!)	(ob →)
2E835	(TOLVar11@)	(→ ob)
2E836	(TOLVar12!)	(ob →)
2E837	(TOLVar12@)	(→ ob)
2E838	(TOLVar13!)	(ob →)
2E839	(TOLVar13@)	(→ ob)
2E83A	(TOLVar14!)	(ob →)
2E83B	(TOLVar14@)	(→ ob)
2E83C	(TOLVar15!)	(ob →)
2E83D	(TOLVar15@)	(→ ob)
2E83E	(TOLVar16!)	(ob →)
2E83F	(TOLVar16@)	(→ ob)
2E840	(TOLVar17!)	(ob →)
2E841	(TOLVar17@)	(→ ob)
2E842	(TOLVar18!)	(ob →)
2E843	(TOLVar18@)	(→ ob)
2E844	(TOLVar19!)	(ob →)
2E845	(TOLVar19@)	(→ ob)
2E846	(TOLVar20!)	(ob →)

2E847	(TOLVar20@)	($\rightarrow$ ob)
2E848	(TOLVar21!)	(ob $\rightarrow$)
2E849	(TOLVar21@)	($\rightarrow$ ob)
2E84A	(TOLVar22!)	(ob $\rightarrow$)
2E84B	(TOLVar22@)	($\rightarrow$ ob)
2E84C	(TOLVar23!)	(ob $\rightarrow$)
2E84D	(TOLVar23@)	($\rightarrow$ ob)
2E84E	(TOLVar24!)	(ob $\rightarrow$)
2E84F	(TOLVar24@)	($\rightarrow$ ob)
2E850	(TOLVar25!)	(ob $\rightarrow$)
2E851	(TOLVar25@)	($\rightarrow$ ob)
2E852	(TOLVar26!)	(ob $\rightarrow$)
2E853	(TOLVar26@)	($\rightarrow$ ob)
2E854	(TOLVar27!)	(ob $\rightarrow$)
2E855	(TOLVar27@)	($\rightarrow$ ob)
2E856	(TOLVar28!)	(ob $\rightarrow$)
2E857	(TOLVar28@)	($\rightarrow$ ob)
2E858	(TOLVar29!)	(ob $\rightarrow$)
2E859	(TOLVar29@)	($\rightarrow$ ob)
2E85A	(TOLVar30!)	(ob $\rightarrow$)
2E85B	(TOLVar30@)	($\rightarrow$ ob)
2E85C	(TOLVar31!)	(ob $\rightarrow$)
2E85D	(TOLVar31@)	($\rightarrow$ ob)
2E85E	(TOLVar32!)	(ob $\rightarrow$)
2E85F	(TOLVar32@)	($\rightarrow$ ob)
2E860	(TOLVar33!)	(ob $\rightarrow$)
2E861	(TOLVar33@)	($\rightarrow$ ob)
2E862	(TOLVar34!)	(ob $\rightarrow$)
2E863	(TOLVar34@)	($\rightarrow$ ob)
2E864	(TOLVar35!)	(ob $\rightarrow$)
2E865	(TOLVar35@)	($\rightarrow$ ob)
2E866	(TOLVar36!)	(ob $\rightarrow$)
2E867	(TOLVar36@)	($\rightarrow$ ob)
2E868	(TOLVar37!)	(ob $\rightarrow$)
2E869	(TOLVar37@)	($\rightarrow$ ob)
2E86A	(TOLVar38!)	(ob $\rightarrow$)
2E86B	(TOLVar38@)	($\rightarrow$ ob)
2E86C	(TOLVar39!)	(ob $\rightarrow$)

2E86D	(TOLVar39@)	(→ ob)
2E86E	(TOLVar40!)	(ob →)
2E86F	(TOLVar40@)	(→ ob)
2E870	(TOLVar41!)	(ob →)
2E871	(TOLVar41@)	(→ ob)
2E872	(TOLVar42!)	(ob →)
2E873	(TOLVar42@)	(→ ob)
2E874	(TOLVar43!)	(ob →)
2E875	(TOLVar43@)	(→ ob)
2E876	(TOLVar44!)	(ob →)
2E877	(TOLVar44@)	(→ ob)
2E878	(TOLVar45!)	(ob →)
2E879	(TOLVar45@)	(→ ob)
2E87A	(TOLVar46!)	(ob →)
2E87B	(TOLVar46@)	(→ ob)
2E87C	(TOLVar47!)	(ob →)
2E87D	(TOLVar47@)	(→ ob)
2E87E	(TOLVar48!)	(ob →)
2E87F	(TOLVar48@)	(→ ob)
2E880	(TOLVar49!)	(ob →)
2E881	(TOLVar49@)	(→ ob)
2E882	(TOLVar50!)	(ob →)
2E883	(TOLVar50@)	(→ ob)
2E884	(TOLVar51!)	(ob →)
2E885	(TOLVar51@)	(→ ob)
2E886	(TOLVar52!)	(ob →)
2E887	(TOLVar52@)	(→ ob)
2E888	(TOLVar53!)	(ob →)
2E889	(TOLVar53@)	(→ ob)
2E88A	(TOLVar54!)	(ob →)
2E88B	(TOLVar54@)	(→ ob)
2E88C	(TOLVar55!)	(ob →)
2E88D	(TOLVar55@)	(→ ob)
2E88E	(TOLVar56!)	(ob →)
2E88F	(TOLVar56@)	(→ ob)
2E890	(TOLVar57!)	(ob →)
2E891	(TOLVar57@)	(→ ob)
2E892	(TOLVar58!)	(ob →)

2E893	(TOLVar58@)	(→ ob)
2E894	(TOLVar59!)	(ob →)
2E895	(TOLVar59@)	(→ ob)
2E896	(TOLVar60!)	(ob →)
2E897	(TOLVar60@)	(→ ob)
2E898	(TOLVar61!)	(ob →)
2E899	(TOLVar61@)	(→ ob)
2E89A	(TOLVar62!)	(ob →)
2E89B	(TOLVar62@)	(→ ob)
2E89C	(TOLVar63!)	(ob →)
2E89D	(TOLVar63@)	(→ ob)
2E89E	(TOLVar64!)	(ob →)
2E89F	(TOLVar64@)	(→ ob)
2E8A0	(TOLVar65!)	(ob →)
2E8A1	(TOLVar65@)	(→ ob)
2E8A2	(TOLVar66!)	(ob →)
2E8A3	(TOLVar66@)	(→ ob)
2E8A4	(TOLVar67!)	(ob →)
2E8A5	(TOLVar67@)	(→ ob)
2E8A6	(TOLVar68!)	(ob →)
2E8A7	(TOLVar68@)	(→ ob)
2E8A8	(TOLVar69!)	(ob →)
2E8A9	(TOLVar69@)	(→ ob)
2E8AA	(TOLVar70!)	(ob →)
2E8AB	(TOLVar70@)	(→ ob)
2E8AC	(TOLVar71!)	(ob →)
2E8AD	(TOLVar71@)	(→ ob)
2E8AE	(TOLVar72!)	(ob →)
2E8AF	(TOLVar72@)	(→ ob)
2E8B0	(TOLVar73!)	(ob →)
2E8B1	(TOLVar73@)	(→ ob)
2E8B2	(TOLVar74!)	(ob →)
2E8B3	(TOLVar74@)	(→ ob)
2E8B4	(TOLVar75!)	(ob →)
2E8B5	(TOLVar75@)	(→ ob)
2E8B6	(TOLVar76!)	(ob →)
2E8B7	(TOLVar76@)	(→ ob)
2E8B8	(TOLVar77!)	(ob →)

2E8B9	(TOLVar77@)	(→ ob)
2E8BA	(TOLVar78!)	(ob →)
2E8BB	(TOLVar78@)	(→ ob)
2E8BC	(TOLVar79!)	(ob →)
2E8BD	(TOLVar79@)	(→ ob)
2E8BE	(TOLVar80!)	(ob →)
2E8BF	(TOLVar80@)	(→ ob)
2E8C0	(TOLVar81!)	(ob →)
2E8C1	(TOLVar81@)	(→ ob)
2E8C2	(TOLVar82!)	(ob →)
2E8C3	(TOLVar82@)	(→ ob)
2E8C4	(TOLVar83!)	(ob →)
2E8C5	(TOLVar83@)	(→ ob)
2E8C6	(TOLVar84!)	(ob →)
2E8C7	(TOLVar84@)	(→ ob)
2E8C8	(TOLVar85!)	(ob →)
2E8C9	(TOLVar85@)	(→ ob)
2E8CA	(TOLVar86!)	(ob →)
2E8CB	(TOLVar86@)	(→ ob)
2E8CC	(TOLVar87!)	(ob →)
2E8CD	(TOLVar87@)	(→ ob)
2E8CE	(TOLVar88!)	(ob →)
2E8CF	(TOLVar88@)	(→ ob)
2E8D0	(TOLVar89!)	(ob →)
2E8D1	(TOLVar89@)	(→ ob)
2E8D2	(TOLVar90!)	(ob →)
2E8D3	(TOLVar90@)	(→ ob)
2E8D4	(TOLVar91!)	(ob →)
2E8D5	(TOLVar91@)	(→ ob)
2E8D6	(TOLVar92!)	(ob →)
2E8D7	(TOLVar92@)	(→ ob)
2E8D8	(TOLVar93!)	(ob →)
2E8D9	(TOLVar93@)	(→ ob)
2E8DA	(TOLVar94!)	(ob →)
2E8DB	(TOLVar94@)	(→ ob)
2E8DC	(TOLVar95!)	(ob →)
2E8DD	(TOLVar95@)	(→ ob)
2E8DE	(TOLVar96!)	(ob →)

2E8DF	(TOLVar96@)	(→ ob)
2E8E0	(TOLVar97!)	(ob →)
2E8E1	(TOLVar97@)	(→ ob)
2E8E2	(TOLVar98!)	(ob →)
2E8E3	(TOLVar98@)	(→ ob)
2E8E4	(TOLVar99!)	(ob →)
2E8E5	(TOLVar99@)	(→ ob)
2E8E6	(TOLVar100!)	(ob →)
2E8E7	(TOLVar100@)	(→ ob)
2E8E8	(TOLVar101!)	(ob →)
2E8E9	(TOLVar101@)	(→ ob)
2E8EA	(TOLVar102!)	(ob →)
2E8EB	(TOLVar102@)	(→ ob)
2E8EC	(TOLVar103!)	(ob →)
2E8ED	(TOLVar103@)	(→ ob)
2E8EE	(TOLVar104!)	(ob →)
2E8EF	(TOLVar104@)	(→ ob)
2E8F0	(TOLVar105!)	(ob →)
2E8F1	(TOLVar105@)	(→ ob)
2E8F2	(TOLVar106!)	(ob →)
2E8F3	(TOLVar106@)	(→ ob)
2E8F4	(TOLVar107!)	(ob →)
2E8F5	(TOLVar107@)	(→ ob)
2E8F6	(TOLVar108!)	(ob →)
2E8F7	(TOLVar108@)	(→ ob)
2E8F8	(TOLVar109!)	(ob →)
2E8F9	(TOLVar109@)	(→ ob)
2E8FA	(TOLVar110!)	(ob →)
2E8FB	(TOLVar110@)	(→ ob)
2E8FC	(TOLVar111!)	(ob →)
2E8FD	(TOLVar111@)	(→ ob)
2E8FE	(TOLVar112!)	(ob →)
2E8FF	(TOLVar112@)	(→ ob)
2E900	(TOLVar113!)	(ob →)
2E901	(TOLVar113@)	(→ ob)
2E902	(TOLVar114!)	(ob →)
2E903	(TOLVar114@)	(→ ob)
2E904	(TOLVar115!)	(ob →)

2E905	(TOLVar115@)	(→ ob)
2E906	(TOLVar116!)	(ob →)
2E907	(TOLVar116@)	(→ ob)
2E908	(TOLVar117!)	(ob →)
2E909	(TOLVar117@)	(→ ob)
2E90A	(TOLVar118!)	(ob →)
2E90B	(TOLVar118@)	(→ ob)
2E90C	(TOLVar119!)	(ob →)
2E90D	(TOLVar119@)	(→ ob)
2E90E	(TOLVar120!)	(ob →)
2E90F	(TOLVar120@)	(→ ob)
2E910	(TOLVar121!)	(ob →)
2E911	(TOLVar121@)	(→ ob)
2E912	(TOLVar122!)	(ob →)
2E913	(TOLVar122@)	(→ ob)
2E914	(TOLVar123!)	(ob →)
2E915	(TOLVar123@)	(→ ob)
2E916	(TOLVar124!)	(ob →)
2E917	(TOLVar124@)	(→ ob)
2E918	(TOLVar125!)	(ob →)
2E919	(TOLVar125@)	(→ ob)
2E91A	(TOLVar126!)	(ob →)
2E91B	(TOLVar126@)	(→ ob)
2E91C	(TOLVar127!)	(ob →)
2E91D	(TOLVar127@)	(→ ob)
2E91E	(TOLVar128!)	(ob →)
2E91F	(TOLVar128@)	(→ ob)
2E920	(TOLVar129!)	(ob →)
2E921	(TOLVar129@)	(→ ob)
2E922	(TOLVar130!)	(ob →)
2E923	(TOLVar130@)	(→ ob)
2E924	(TOLVar131!)	(ob →)
2E925	(TOLVar131@)	(→ ob)
2E926	(TOLVar132!)	(ob →)
2E927	(TOLVar132@)	(→ ob)
2E928	(TOLVar133!)	(ob →)
2E929	(TOLVar133@)	(→ ob)
2E92A	(TOLVar134!)	(ob →)

2E92B	(TOLVar134@)	(→ ob)
2E92C	(TOLVar135!)	(ob →)
2E92D	(TOLVar135@)	(→ ob)
2E92E	(TOLVar136!)	(ob →)
2E92F	(TOLVar136@)	(→ ob)
2E930	(TOLVar137!)	(ob →)
2E931	(TOLVar137@)	(→ ob)
2E932	(TOLVar138!)	(ob →)
2E933	(TOLVar138@)	(→ ob)
2E934	(TOLVar139!)	(ob →)
2E935	(TOLVar139@)	(→ ob)
2E936	(TOLVar140!)	(ob →)
2E937	(TOLVar140@)	(→ ob)
2E938	(TOLVar141!)	(ob →)
2E939	(TOLVar141@)	(→ ob)
2E93A	(TOLVar142!)	(ob →)
2E93B	(TOLVar142@)	(→ ob)
2E93C	(TOLVar143!)	(ob →)
2E93D	(TOLVar143@)	(→ ob)
2E93E	(TOLVar144!)	(ob →)
2E93F	(TOLVar144@)	(→ ob)
2E940	(TOLVar145!)	(ob →)
2E941	(TOLVar145@)	(→ ob)
2E942	(TOLVar146!)	(ob →)
2E943	(TOLVar146@)	(→ ob)
2E944	(TOLVar147!)	(ob →)
2E945	(TOLVar147@)	(→ ob)
2E946	(TOLVar148!)	(ob →)
2E947	(TOLVar148@)	(→ ob)
2E948	(TOLVar149!)	(ob →)
2E949	(TOLVar149@)	(→ ob)
2E94A	(TOLVar150!)	(ob →)
2E94B	(TOLVar150@)	(→ ob)
2E94C	(TOLVar151!)	(ob →)
2E94D	(TOLVar151@)	(→ ob)
2E94E	(TOLVar152!)	(ob →)
2E94F	(TOLVar152@)	(→ ob)
2E950	(TOLVar153!)	(ob →)

2E951	(TOLVar153@)	(→ ob)
2E952	(TOLVar154!)	(ob →)
2E953	(TOLVar154@)	(→ ob)
2E954	(TOLVar155!)	(ob →)
2E955	(TOLVar155@)	(→ ob)
2E956	(TOLVar156!)	(ob →)
2E957	(TOLVar156@)	(→ ob)
2E958	(TOLVar157!)	(ob →)
2E959	(TOLVar157@)	(→ ob)
2E95A	(TOLVar158!)	(ob →)
2E95B	(TOLVar158@)	(→ ob)
2E95C	(TOLVar159!)	(ob →)
2E95D	(TOLVar159@)	(→ ob)
2E95E	(TOLVar160!)	(ob →)
2E95F	(TOLVar160@)	(→ ob)
2E960	(TOLVar161!)	(ob →)
2E961	(TOLVar161@)	(→ ob)
2E962	(TOLVar162!)	(ob →)
2E963	(TOLVar162@)	(→ ob)
2E964	(TOLVar163!)	(ob →)
2E965	(TOLVar163@)	(→ ob)
2E966	(TOLVar164!)	(ob →)
2E967	(TOLVar164@)	(→ ob)
2E968	(TOLVar165!)	(ob →)
2E969	(TOLVar165@)	(→ ob)
2E96A	(TOLVar166!)	(ob →)
2E96B	(TOLVar166@)	(→ ob)
2E96C	(TOLVar167!)	(ob →)
2E96D	(TOLVar167@)	(→ ob)
2E96E	(TOLVar168!)	(ob →)
2E96F	(TOLVar168@)	(→ ob)
2E970	(TOLVar169!)	(ob →)
2E971	(TOLVar169@)	(→ ob)
2E972	(TOLVar170!)	(ob →)
2E973	(TOLVar170@)	(→ ob)
2E974	(TOLVar171!)	(ob →)
2E975	(TOLVar171@)	(→ ob)
2E976	(TOLVar172!)	(ob →)

2E977	(TOLVar172@)	(→ ob)
2E978	(TOLVar173!)	(ob →)
2E979	(TOLVar173@)	(→ ob)
2E97A	(TOLVar174!)	(ob →)
2E97B	(TOLVar174@)	(→ ob)
2E97C	(TOLVar175!)	(ob →)
2E97D	(TOLVar175@)	(→ ob)
2E97E	(TOLVar176!)	(ob →)
2E97F	(TOLVar176@)	(→ ob)
2E980	(TOLVar177!)	(ob →)
2E981	(TOLVar177@)	(→ ob)
2E982	(TOLVar178!)	(ob →)
2E983	(TOLVar178@)	(→ ob)
2E984	(TOLVar179!)	(ob →)
2E985	(TOLVar179@)	(→ ob)
2E986	(TOLVar180!)	(ob →)
2E987	(TOLVar180@)	(→ ob)
2E988	(TOLVar181!)	(ob →)
2E989	(TOLVar181@)	(→ ob)
2E98A	(TOLVar182!)	(ob →)
2E98B	(TOLVar182@)	(→ ob)
2E98C	(TOLVar183!)	(ob →)
2E98D	(TOLVar183@)	(→ ob)
2E98E	(TOLVar184!)	(ob →)
2E98F	(TOLVar184@)	(→ ob)
2E990	(TOLVar185!)	(ob →)
2E991	(TOLVar185@)	(→ ob)
2E992	(TOLVar186!)	(ob →)
2E993	(TOLVar186@)	(→ ob)
2E994	(TOLVar187!)	(ob →)
2E995	(TOLVar187@)	(→ ob)
2E996	(TOLVar188!)	(ob →)
2E997	(TOLVar188@)	(→ ob)
2E998	(TOLVar189!)	(ob →)
2E999	(TOLVar189@)	(→ ob)
2E99A	(TOLVar190!)	(ob →)
2E99B	(TOLVar190@)	(→ ob)
2E99C	(TOLVar191!)	(ob →)

2E99D	(TOLVar191@)	(→ ob)
2E99E	(TOLVar192!)	(ob →)
2E99F	(TOLVar192@)	(→ ob)
2E9A0	(TOLVar193!)	(ob →)
2E9A1	(TOLVar193@)	(→ ob)
2E9A2	(TOLVar194!)	(ob →)
2E9A3	(TOLVar194@)	(→ ob)
2E9A4	(TOLVar195!)	(ob →)
2E9A5	(TOLVar195@)	(→ ob)
2E9A6	(TOLVar196!)	(ob →)
2E9A7	(TOLVar196@)	(→ ob)
2E9A8	(TOLVar197!)	(ob →)
2E9A9	(TOLVar197@)	(→ ob)
2E9AA	(TOLVar198!)	(ob →)
2E9AB	(TOLVar198@)	(→ ob)
2E9AC	(TOLVar199!)	(ob →)
2E9AD	(TOLVar199@)	(→ ob)
2E9AE	(TOLVar200!)	(ob →)
2E9AF	(TOLVar200@)	(→ ob)
2E9B0	(TOLVar201!)	(ob →)
2E9B1	(TOLVar201@)	(→ ob)
2E9B2	(TOLVar202!)	(ob →)
2E9B3	(TOLVar202@)	(→ ob)
2E9B4	(TOLVar203!)	(ob →)
2E9B5	(TOLVar203@)	(→ ob)
2E9B6	(TOLVar204!)	(ob →)
2E9B7	(TOLVar204@)	(→ ob)
2E9B8	(TOLVar205!)	(ob →)
2E9B9	(TOLVar205@)	(→ ob)
2E9BA	(TOLVar206!)	(ob →)
2E9BB	(TOLVar206@)	(→ ob)
2E9BC	(TOLVar207!)	(ob →)
2E9BD	(TOLVar207@)	(→ ob)
2E9BE	(TOLVar208!)	(ob →)
2E9BF	(TOLVar208@)	(→ ob)
2E9C0	(TOLVar209!)	(ob →)
2E9C1	(TOLVar209@)	(→ ob)
2E9C2	(TOLVar210!)	(ob →)

2E9C3	(TOLVar210@)	(→ ob)
2E9C4	(TOLVar211!)	(ob →)
2E9C5	(TOLVar211@)	(→ ob)
2E9C6	(TOLVar212!)	(ob →)
2E9C7	(TOLVar212@)	(→ ob)
2E9C8	(TOLVar213!)	(ob →)
2E9C9	(TOLVar213@)	(→ ob)
2E9CA	(TOLVar214!)	(ob →)
2E9CB	(TOLVar214@)	(→ ob)
2E9CC	(TOLVar215!)	(ob →)
2E9CD	(TOLVar215@)	(→ ob)
2E9CE	(TOLVar216!)	(ob →)
2E9CF	(TOLVar216@)	(→ ob)
2E9D4	(TOLVarN!)	(ob →)
2E9F8	(TOLVarN@)	(→ ob)
2EA1C	(ClrAllTVars)	
2EA52	(ClrAllTOLVs)	
2EA6E	(%0AllTopicVs)	
2EAA9	(%0AllTOLVars)	
2EAE4	(TOLVarSet!)	
2EB11	(SaveTOLVarSet)	
2EB66	(RestTOLVarSet)	
2EBB1	(%0TOLVarSet)	
2EC01	(1getcxt!)	
2EC15	(DoInCxt)	
2EC6F	(DoInCalcCxt)	
2EC88	(DoInAppCxt)	
2ECA1	(DoInFuncCxt)	
2ECBA	(DoInPolarCxt)	
2ECD3	(DoInParamCxt)	
2ECEC	(DoInSeqCxt)	
2ED05	(DoInStatCxt)	
2ED1E	(DoInSolveCxt)	
2ED37	(DoInOtherCxt)	
2ED91	(DoInOtherN)	
2EDD7	(DoInOtherU)	
2EE04	(otherNG?)	
2EE37	(GET@tTYPER)	

6.2 Rest

`0030E8`	`(~dontuple#)`	`( comp ob # → {} )` Takes objects from comp in groups of # and evals ob on them. The results are returned as a list.

7 UserRPL Commands

7.1 A-F

`030314` `~xABCUV`

(pa pb c → u v)
--
Related: LABCUV,EGCD

`39A07` `xABS`

(x → x')
Absolute Value Function
--
Returns the absolute value of its argument.

```
x          →  |x|
(x,y)      →  sqrt(x^2+y^2)
x_unit     →  |x|_unit
[ array ]  →  || array ||
'sym'      →  'ABS(sym)'
```

--
Flags: -3
--
Related: NEG,SIGN

`390E4` `xACK`

(→)
Acknowledge Alarm cmd
--
Acknowledges the oldest past due alarm.
--
Flags: -43 -44

```
Repeat Alarms Not Rescheduled -43
Acknowledge Alarms Saved      -44
```

--
Clears alert annunciator if

```
1. There are no other past-due
   alarms and
2. There are no other active
   alert sources - ie low batt.
```

Has no effect on control alarms Control alarms that come due are automatically acknowledged `AND` saved in the sys alarm list.
--
Related: ACKALL

`390C9` `xACKALL` (→)
Acknowledge All Alarms cmd
--
Acknowledges all past due alarms.
--
Flags: -43 -44
```
Repeat Alarms Not Rescheduled -43
Acknowledge Alarms Saved      -44
```
--
Clears alert annunciator if there are no other active alert sources, ie low batt. Has no effect on control alarms Control alarms that come due are automatically acknowledged `AND` saved in the sys alarm list.
--
Related: ACK

`3A7DC` `xACOS` (x → x')
Arc cos fn
--
Returns angle with given cos.
--
```
 z    → arc cos z
'sym' → 'ACOS(sym)'
```
--
Related: ASIN,ATAN,COS,ISOL,ACOSH

`025314` `~xACOS2S` (symb → symb')

`3A8D8` `xACOSH` (x → x')
Arc hyp cos fn
--
Returns val with given hyp cos.
--
```
 z    → arc hyp cos z
'sym' → 'ACOSH(sym)'
```
--
Related: ASINH,ATANH,COSH,ISOL

`05C0AB` `~xADD` ({} {}' → {}'')
({} ob → {}')
(ob {} → {}')
Add list cmd
--
Adds corresponding elems of 2 lists or adds a number to elem in a list.
--
Related: +,ΔLIST,ΠLIST,ΣLIST

`06E314` `~xADDTMOD` (symb1 symb2 → symb3)

0000DE	~xADDTOREAL	(var →) Make CAS assume that var is real. Add it to the list in CASDIR.
3AAE5	xALOG	(x → x') Common antilog fn -- ALOG x = 10^x -- Flags: -3 numeric result -- z → 10^z 'sym' → 'ALOG(sym)' -- Related: EXP,LN,LOG
04B0AB	~xAMORT	(n → princ intr bal) Amortize cmd -- Flags: -14 Fin pmt mode -14 -- Related: TVM,TVMBEG,TVMEND,TVMROOT
3CA07	xAND	(x1 x2 → x3) And fn -- Logical AND of 2 args. -- #n1 #n1 → #n3 "str1" "str2" → "str3" T/F1 T/F2 → 0/1 T/F 'sym' → 'T/F AND sym' 'sym' T/F → 'sym AND T/F' 'sym1' 'sym2' → 'sym1 AND sym2' -- Flags: -3 -5 Numeric res -3 Bin int wordsize -5 → -10 -- Related: NOT,OR,XOR
0140AB	~xANIMATE	(g1...gn n → same stack) (g1...gn {n {#X #Y} delay rep} → same stack) Animate cmd -- Displays grobs in sequence

3F033	xANS	`( n` $\rightarrow$ `ob )` Invokes results of previous calculations. -- Related: `LASTARG`
3D7AC	xAPPLY	`( {symb1 .. symbn} f` $\rightarrow$ `f(symb1...symbn) )` Apply to args fn -- Creates expr for specified fn name & args -- Related: QUOTE,|

3C8C6	xARC	`( c r` θ`1` θ`2` $\rightarrow$ `)`
		`( {#x #y} #r` θ`1` θ`2` $\rightarrow$ `)`
		Draw arc fn
		--
		Draws arc in PICT anticlockwise from $\theta 1$ to $\theta 2$ centred on coord specified on lev4 with radius on lev3
		--
		Flags: -17 -18
		`angle mode (-17 & -18)`
		--
		Related: BOX,LINE,TLINE

3EAC7	xARCHIVE	`( :port:name` $\rightarrow$ `)`
		`( :IO:name` $\rightarrow$ `)`
		Archive HOME cmd
		--
		Creates `backup` of HOME in RAM (including user key assignments & alarm catalog)
		--
		if :IO: is used `backup` transmitted through IO port via Kermit to `filename` 'name'
		--
		Flags: -33 -39
		`I/O Device -33`
		`I/O Messages -39 if :IO:name`
		--
		Related: RESTORE

3A390	xARG	`( c` $\rightarrow$ θ `)`
		Argument fn
		--
		Returns angle of a complex number
		--
		`(x,y)` $\rightarrow$ θ
		`'sym'` $\rightarrow$ `'ARG(sym)'`
		--
		Flags: -17 -18
		`Ang Mode -17,-18`

085314	~xARIT	(→) Display menu of arithmetic commands. -- Related: BASE,CMPLX,DIFF,EXP&LN, SOLVER,TRIGO
3BEC5	xARRY>	([] → x1...xn {n}) ([[]] → x11...xnm {n m}) Array to stack cmd -- Return elems of array to stack. OBJ→ includes this functionality. -- Related: →ARRY,DTAG,EQ→,LIST→, OBJ→,STR→ UserRPL: xARRY→
3BE9B	x>ARRY	(x1..xn n → []) (x11...xnm {n m} → [[]]) Stack to Array Cmd -- Returns a vector of n real or complex elements or a matrix of n m real or complex solutions. -- Related: ARRY→,LIST→,→LIST, OBJ→,STR→,→TAG,→UNIT UserRPL: x→ARRY
3A756	xASIN	(x → x') Arc sin fn -- Gives angle whose sin is given -- z → arc sin z 'sym' → 'ASIN(sym)' -- Flags: -1 -3 -17 -18 Principal soln -1 Numerical res -3 Angle mode -17,-18 -- Related: ACOS,ATAN,ISOL,SIN
024314	~xASIN2C	(symb → symb')
023314	~xASIN2T	(symb → symb')

3A88E	xASINH	`( x → x' )`
		Arc hyp sin fn
		`--`
		Gives Val whose hyp sin is given
		`--`
		`z     → arc hyp sin z`
		`'sym' → 'ASINH(sym)'`
		`--`
		Flags: -1 -3
		`Principal soln -1`
		`Numerical res  -3`
		`--`
		Related: ACOSH,ATANH,ISOL,SINH

3EEE7	xASN	`( obj key → )`
		`( 'SKEY' → )`
		Assign cmd
		`--`
		Defines single key on user kbd by assigning the given obj to the key x_key
		`--`
		Flags: -61 -62
		`User mode lock -61`
		`User mode      -62`
		`--`
		The arg x_key is a real number rc.p where r=row,c=col,p=plane as follows:
		`0,1 - unshifted`
		`2   - left shifted`
		`3   - right shifted`
		`4   -  shifted`
		`5   -  left shifted`
		`6   -  right shifted`
		Add 0.01 if the modifier is to be held pressed down.
		`--`
		After ASN, pressing the assigned in User or 1-User mode exeutes the assigned obj instead. Remains in effect until altered by ASN or STOKEYS or DELKEYS. If 'SKEY' is passed instead, the specified key is restored to std.
		`--`
		Related: DELKEYS,RCLKEYS,STOKEYS <REF>TEXT:Keycodes

`38DE1`	`xASR`	(# → #')
		Arithmetic shift right cmd
		--
		Shifts a bint 1 bit to the right except for the most significant bit which stays.
		--
		Flags: -5 -6 -7 -8 -9 -10 -11 -12
		`bint wordsize  -5 -> -10`
		`bint base      -11, -12`
		--
		Related: SL,SLB,SR,SRB

`0260DE`	`~xASSUME`	

`3A844`	`xATAN`	(x → x')
		Arc tan fn
		--
		Returns the angle having the tan
		--
		`z     → arc tan z`
		`'sym' → 'ATAN(sym)'`
		--
		Flags: -1 -3 -17 -18
		`Principle soln  -1`
		`Numeric results -3`
		`Angle mode      -17,-18`
		--
		Related: ACOS,ASIN,ISOL,TAN

`022314`	`~xATAN2S`	(symb → symb')

`3A94F`	`xATANH`	(x → x')
		Arc hyp tan fn
		--
		Returns the value with given hyp tan.
		--
		`z     → arc hyp tan z`
		`'sym' → 'ATANH(sym)'`
		--
		Flags: -1 -3 -22
		`Principle soln  -1`
		`Numeric results -3`
		`Infinite result exception -22`
		--
		Related: ACOSH,ASINH,ISOL,TANH

3EB64	xATTACH	(n →) (:nport:n →) Attach `library` cmd -- Attaches lib with given num to current directory. -- Related: DETACH,`LIBS`
0130DE	~xAUGMENT	

3C49F	xAUTO	(→)
		`Calculates a y-axis display range`
		`or an x- & y-axis display range.`
		--
		Action depends on plot type:

```
FUNCTION   sets range to max &
           min of y vals sampled
           at 40 equi-spaced x
           vals (excluding )
CONIC      sets y-axis scale = to
           x-axis scale
POLAR      same as FUNCTION
;
PARAMETRIC same as POLAR
;
TRUTH      no action
;
BAR        sets x-axis range from
           0 to # of elems in
           ΣDAT +1. sets y-range
           to min & max of the
           elts x-axis is always
           included.
HISTOGRAM  sets x-axis range to
           min & max of the elems
           in ΣDAT. sets y-range
           from 0 to # of rows in
           ΣDAT.
SCATTER    x-range is min & max
           of XCOL. y-range is
           min & max of YCOL
 --
```

Related: DRAW,SCALEH,SCALE,SCLΣ, SCALEW,XRNG,YRNG

3C3B2	xAXES	(c →) ({c tick $x $y } →) Axes cmd -- Specifies intersection coords of x- & y- axes, tick mark annotatn and x- & y- axes labels. stored in PPAR. -- <REF>TEXT:Reserved|PPAR -- Related: ATICK,DRAW,DRAX,LABEL
04A314	~xAXL	({} → []) ([] → ())
049314	~xAXM	([A] → [M])
04C314	~xAXQ	([nxn] [n] → [nxn]' [n])

3C9D3	xBAR	(→)
		Bar plot type cmd
		--
		Sets plot type to BAR When plot type is BAR, the DRAW Cmd plots a bar chart using data from 1 col of the stat matrix (ΣDAT). The col to be plotted is specified by the XCOL cmd & is stored in 1st param of ΣPAR. Plot params are specified in PPAR of ff form:
		`{ (xmin,ymin) (xmax,ymax) indep`
		`  res axes ptype depend }`
		For BAR they are used as follows:
		--
		(xmin,ymin) specifies lower left cnr of PICT (default: (-6.5,-3.1))
		--
		(xmax,ymax) specifies upper right cnr of PICT (default: (6.5,3.2))
		--
		indep name - specifies horiz axis label or list - { name x1 x2 } smaller of x1 & x2 is horiz location of 1st bar (default: X)
		--
		res real - bar width in user units or bint - bar width in pixels (default: 0 - 1 in user units)
		--
		axes list containing one or more of the ff in order: (xı,yı) - user unit origin pos a list specifying tick mark annotatn & 2 strings specifying horiz & vert axes labels (default: (0,0))
		--
		ptype plot type - BAR in this case
		--
		depend label for vert axis. (default: Y)
		--
		<REF>TEXT:Reserved|PPAR
		--
		Related: CONIC,DIFFEQ,FUNCTION,GRIDMAP, HISTOGRAM,PARAMETRIC,PARSURFACE,PCONTOUR, SCATTER,SLOPEFIELD,TRUTH,YSLICE

`3E196` `xBARPLOT` (→)
Draw bar plot cmd
--
Draws bar chart of specified col of stat matrix (ΣDAT) Col to be plotted is specified by `XCOL` & is stored as first param in ΣPAR. Default col is 1. data can be +ve or -ve giving bars above or below the axis. y-axis is autoscaled & plot type is `BAR`. When executed from a program, plot doesn't persist unless PICTURE,`PVIEW` (with empty list) or FREEZE is subsequently executed
--
Related: FREEZE,HISTPLOT,PICTURE, `PVIEW`,SCATRPLOT,`XCOL`

`080314` `~xBASE` (→)
Display menu of basic algebra commands.
--
Related: ARIT,CMPLX,DIFF,EXP&LN.SOLVER,TRIGO
aka: `xALGB`

`0110DE` `~xBASIS`

`3EDCC` `xBAUD` (n →)
Baud rate cmd
--
Specify bit transfer rate.
--
Related: CKSM,PARITY,TRANSIO

`39765` `xBEEP` (freq dur →)
Beep cmd
--
Sounds a tone of n Hz for x secs.
--
Flags: -56
```
Error Beep -56
Max Freq = 4400 Hz
Max Duration = 1048.575 secs.
```
--
Related: `HALT`,INPUT,PROMPT,`WAIT`

`3E2C1`	`xBESTFIT`	`( → )`
		Best fit model cmd
		`--`
		Executes LR with each of the 4 curve fitting models and selects the model giving the largest correlation coefficient.
		`--`
		Selected model stored in 5th param of ΣPAR & regression coeffs intercept & slope are stored in 3rd & 4th params.
		`--`
		Related: EXPFIT,LINFIT,LOGFIT,LR,PWRFIT

`3B655`	`xBIN`	`( → )`
		Binary mode cmd
		`--`
		Selects binary base for bint ops. (Default base is 10)
		`--`
		Flags: -5 -6 -7 -8 -9 -10 -11 -12
		`Bint wordsize -5 → -10`
		`Bint base     -11, -12`
		Bints require prefix #. Bints entered & returned in binary show the b suffix. If current base not binary, enter binary nums by using b suffix. The current base doesn't affect the internal representation of bints as unsigned bints.
		`--`
		Related: DEC,HEX,OCT,STWS,RCWS

`3E171`	`xBINS`	`( min width n → [[]] [] )`
		Sort Into Frequency Bins Cmd
		`--`
		Sorts the elements of the indep. col (`XCOL`) of the stat matrix (ΣDAT) into (nbins + 2) bins, where the left edge for bin 1 starts at value xmin and each bin has width xwidth.
		`--`
		`xmin xwidth nbins →`
		`[[ nbin1...nbinn ]]`
		`[ nbinL nbinR ]`
		`--`
		Related: BARPLOT,`XCOL`

3C70A	xBLANK	(#width #height → grob) Blank Graphics Obj Cmd -- Creates a blank graphics obj of the specified width and height. -- Related: →GROB,LCD→
3C6E0	xBOX	({#n1 #m1} {#n2 #m2} →) (c1 c2 →) Box Cmd -- Draws in PICT a box whose opposite corners are defined by the specified pixel or user-unit coords. -- Related: ARC,LINE,TLINE
3EE47	xBUFLEN	(→ nchars 0/1) Buffer Length Cmd -- Returns the number of characters in the HP 48's serial input buffer and a single digit indicating whether an error occurred during data reception. -- Related: CLOSEIO,OPENIO,SBRK,SRECV, STIME,XMIT
39480	xBYTES	(obj → chksum size) Bytes Size Cmd -- Returns the number of bytes & the checksum for the given obj. -- Related: MEM
38F21	xB>R	(# → R) Binary to Real Cmd -- Converts a binary integer to its floating-point equivalent. -- Related: R→B UserRPL: xB→R
01E0DE	~xC2P	({} → ?????)
07E314	~xCASCFG	(→)
0330DE	~xCASCMD	(→ ?)

`38B28` `xCASE`

`( → )`
CASE Conditional Structure Cmd
`--`
Starts CASE ... `END` conditional structure.
`--`
```
CASE  →
THEN T/F →
END  →
END  →
--
```
Related: `END`,IF,IFERR,THEN

`3AD1B` `xCEIL`

`( x → n )`
Ceiling Func
`--`
Returns the smallest integer greater than or equal to the argument.
`--`
```
x     → n
x_u   → n_u
'sym' → 'CEIL(sym)'
--
```
Flags: -3
`--`
Related: FLOOR,IP,RND,TRNC

`3C3DC` `xCENTR`

`( (x,y) → )`
`( x → )`
Centre Cmd
`--`
Adjusts the first two parameters in the reserved variable PPAR, (xmin, ymin) and (xmax,ymax), so that the point represented by the argument (x,y) is the plot centre.
`--`
<REF>TEXT:Reserved|PPAR
`--`
Related: SCALE

`3B4E9` `xCF`

`( n → )`
Clear Flag Cmd
`--`
Clears the specified user or system flag.
`--`
Related: FC?,FC?C,FS?,FS?C,SF

`03A314` `~xCHINREM` `( []1 []2 → []3 )`

`00B0DE` `~xCHOLESKY`

3BC19	xCHR	(n → $) Character Cmd -- Returns a string representing the HP 48 character corresponding to the character code n. -- Related: NUM,POS,REPL,SIZE,SUB

3B362	x%CH	(x1 x2 → x3)
		Percent Change Func
		--
		Returns the percent change from x (level 2) to y (level 1) as a percentage of x.
		--
		`x       y       → 100(y-x)/x`
		`x       'sym'   → '%CH(x,sym)'`
		`'sym'   x       → '%CH(sym,x)'`
		`'sym1'  'sym2'  → '%CH(sym1,sym2)'`
		`x_u     y_u     → 100(y_u-x_u)/x_u`
		`x_u     'sym'   → '%CH(x_u,sym)'`
		`'sym'   x_u     → '%CH(sym,x_u)'`
		--
		Flags: -3
		--
		Related: %,%T

01D0DE	~xCIRC	(prg {} → ?????)
3EDAC	xCKSM	(n_type →) Checksum Cmd -- Specifies the error-detection scheme. -- Related: BAUD,PARITY,TRANSIO ;
3DD4E	xCLEAR	(ob1 .. obn →) Clear Cmd -- Removes all objects from the stack. -- Related: CLVAR,PURGE
3DD8E	xCLSIGMA	(→) Clear Sigma Cmd -- Purges the current statistics matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: RCLΣ,STOΣ,Σ+,Σ- UserRPL: xCLΣ

39144	xCLKADJ	(ticks →) Adjust System Clock Cmd -- Adjusts the system time by x clock ticks, where 8192 clock ticks equal 1 second. -- Related: →TIME
39839	xCLLCD	(→) Clear LCD Cmd -- Clears (blanks) the stack display -- Related: DISP,FREEZE
3EC95	xCLOSEIO	(→) Close I/O Port Cmd -- Closes the serial port and the IR port, and clears the input buffer and any error messages for KERMIT. -- Related: BUFLEN,OPENIO
3E91A	xCLUSR	(→) Clear Variables Cmd -- Purges all variables and empty subdirectories in the current directory. -- Related: CLUSR,PGDIR,PURGE UserRPL: xCLVAR
081314	~xCMPLX	(→) Display a menu pertaining to complex numbers. -- Related: ARIT,BASE,DIFF,EXP&LN,SOLVER,TRIGO
3B193	xCNRM	([] → col_norm) Column Norm Cmd -- Returns the column norm (onenorm) of the array argument. -- Related: CROSS,DET,DOT,RNRM

0380AB ~x→COL

([[]] → [v1]...[vn] n)
([] → x1...xn n)
Matrix to Columns Cmd
--
Transforms a matrix into a series of column vectors and returns the vectors and a column count, or transforms a vector into its elements and returns the elements and an element count.
--
Related: COL→,→ROW,ROW→

0390AB ~xCOL→

([v1]...[vn] n → [[]])
(x1...xn n → [])
Columns to Matrix Cmd
--
Transforms a series of column vectors and a column count into a matrix containing those columns, or transforms a sequence of numbers and an element count into a vector with those numbers as elements.
--
Related: →COL,→ROW,ROW→

03F0AB ~xCOL+

([[]] [[]]' n → [[]]'')
([] x n → []')
Insert Column Cmd
--
Inserts an array (vector or matrix) into a matrix (one or more elements into a vector) at the position indicated by nindex, and returns the modified array.
--

```
[[mat]]1 [mat]2   nidx → [[mat]]3
[[mat]]1 [vec]col nidx → [[mat]]2
[vec]1   nelement nidx → [vec]2
```

--
Related: COL-,CSWP,ROW+,ROW-

03E0AB ~xCOL-

([] n → []' xn)
([[]] n → [[]]' [vn])
Delete Column Cmd
--
Deletes column n of a matrix (or element n of a vector), and returns the modified matrix (or vector) and the deleted column (or element).
--
Related: COL+,CSWP,ROW+,ROW-

3E5A0	xCOLCT	(symb → symb') Collect Like Terms Cmd -- Simplifies an algebraic expression or equation by "collecting" like terms. Does not modify numbers. -- Related: EXPAN,ISOL,QUAD,SHOW
0300DE	~xCOLLECT	(symb → symb')
3E0FD	xSIGMACOL	(x_col y_col →) Sigma Columns Cmd -- Specifies the independent variable and dependent-variable columns of the current stat matrix (the reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: BARPLOT,BESTFIT,CORR,COV, EXPFIT,HISTPLOT,LINFIT,LOGFIT, LR,PREDX,PREDY,PWRFIT,SCATRPLOT,XCOL,YCOL UserRPL: xCOLΣ
3B423	xCOMB	(n k → Cn,k) Combinations Func -- Returns the number of possible combinations ofn items taken m at a time. -- n m → Cn:m 'symn' m → 'COMB(symn,m)' n 'symm' → 'COMB(n,symm)' 'symn' 'symm' → 'COMB(symn,symm)' -- Related: PERM,!

```
3BF77     xCON
```

({ n } x → [])
({ n k } x → [[]])
([] x → []')
Constant Array Cmd
--
Returns a constant array, defined as an array whose elements all have the same value.
--
```
{ncols} zcnst → [[veccnst]]
{nrows mrows} zcnst → [[matcnst]]
[R-arr] xcnst → [R-arrcnst]
[C-arr] xcnst → [C-arrcnst]
'name'  zcnst →
```
--
Related: IDN

```
0260AB   ~xCOND
```

([[n*n]] → x)
Conditional Number Cmd
--
Returns the 1-norm (column norm) condition number of a square matrix.
--
Related: SNRM,SRAD,TRACE

```
3C967     xCONIC
```

(→)
Conic Plot Type Cmd
--
Sets the plot type to `CONIC`.
--
Related: `BAR`,DIFFEQ,`FUNCTION`,GRIDMAP, `HISTOGRAM`,`PARAMETRIC`,PARSURFACE,PCONTOUR, `POLAR`,`SCATTER`,SLOPEFIELD,`TRUTH`,WIREFRAME,YSLICE

```
39A6C     xCONJ
```

(x → x')
Conjugate Analytic Func
--
Conjugates a complex number or a complex array.
--
```
x           → x
(x,y)       → (x,-y)
[ R-arr ]   → [ R-arr ]
[ C-arr ]1  → [ C-arr ]2
'sym'       → 'CONJ(sym)'
```
--
Flags: -3
--
Related: ABS,IM,RE,SCONJ,SIGN

0180AB	~xCONLIB	(→) Open Constants Library Cmd -- Opens the Constants Library. -- Related: CONST
0190AB	~xCONST	(name → x) Constant Value Cmd -- Returns the value of a constant. -- Flags: +60 +61 -- Related: CONLIB
02A0DE	~xCONSTANTS	
3989C	xCONT	(→) Continue Program Execution Cmd -- Resumes execution of a halted program. -- Related: HALT,KILL,PROMPT
38F41	xCONVERT	(x1_u1 x2_u2 → x3_u2) Convert Units Cmd -- Converts a source unit object to the dimensions of a target object -- Related: UBASE,UFACT,→UNIT,UVAL
3DE24	xCORR	(→ x_correlation) Correlation Cmd -- Returns the correlation coefficient of the independent and dependent data columns in the current statistics matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: COLΣ,COV,PREDX,PREDY,XCOL,YCOL

3A5D0	xCOS	(x → x') Cos Func -- Returns the cos of the argument. -- z → cos z 'sym' → 'COS(sym)' x_uangular → cos(x_uangular) -- Flags: -3 -17 -18 -- Related: ACOS,SIN,TAN
3A6C2	xCOSH	(x → x') Hyp Cos Func -- Returns the hyp cos of the argument. -- z → cosh z 'sym' → 'COSH(sym)' -- Flags: -3 -- Related: ACOSH,SINH,TANH
3DE3F	xCOV	(→ x_covariance) Covariance Cmd -- Returns the sample covariance of the independent and dependent data columns in the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: COLΣ,CORR,PCOV,PREDX,PREDY, XCOL,YCOL
3D128	xCR	(→) Carriage Right Cmd -- Prints the contents, if any, of the printer buffer. -- Flags: -37 -34 -33 -- Related: DE-LAY,OLDPRT,PRLCD,PRST,PRSTC, PRVAR,PR1

393CA	xCRDIR	(name →) Create Directory Cmd -- Creates an empty subdirectory with the specified name within the current directory. -- Related: HOME,PATH,PGDIR,UPDIR
3B208	xCROSS	([1] [2] → [3]) Cross Product Cmd -- CROSS returns the cross product [3] = [1] x [2] of vectors [1] and [2]. -- Related: CNRM,DET,DOT,RNRM
0410AB	~xCSWP	([[]] n1 n2 → [[]]') ([] n1 n2 → []') Column Swap Cmd -- Swaps columns i and j of the argument matrix and returns the modified matrix, or swaps elements ments i and j of the argument vector and returns the modified vector. -- Related: COL+,COL-,RSWP
3C58E	xC>PX	((x,y) → {#n #m}) Complex to Pixel Cmd -- Converts the specifiec user-unit coordiates to pixel coordiates. -- (x,y) → { #n #m } -- Related: PX→C UserRPL: xC→PX
3BAF5	xC>R	((x,y) → x y) ([C] → [R] [I]) Complex to Real Cmd -- Separates the real and imaginary parts of a complex number or complex array. -- Related: R→C,RE,IM UserRPL: xC→R
057314	~xCURL	([func] [vars] → [])
0150DE	~xCYCLOTOMIC	

0120AB	~xCYLIN	(→) Cylindrical Mode Cmd -- Sets Cylindrical coordinate mode. -- Related: RECT,SPHERE
0610AB	~xDARCY	(xe/D yRe → xDarcy) Darcy Friction Factor Func -- Calculates the Darcy friction factor of certain fluid flows. -- Related: FANNING
39078	xDATE	(→ date) Returns the system date. -- Related: DATE+,DDAYS,TIME,TSTR
39104	xSETDATE	(date →) Set Date Cmd -- Sets the system date to date. -- Related: →TIME UserRPL: x→DATE
39238	xDATE+	(date ndays → date') Date Addition Cmd -- Returns a past or future date, given a date in level 2 and a nmber of days in level 1. -- Flags: -42 -- Related: DATE,DDAYS
0690AB	~xdB	(→ %1)
0150DD	~xDBUG	(prog →) (name →) Debug Operation -- Starts program execution, then suspends it as if HALT were the first program command. -- Related: HALT,NEXT

39218	xDDAYS	`( date1 date2 → days )` Delta Days Cmd -- Returns the number of days between two dates. -- Related: DATE,DATE+
3B670	xDEC	`( → )` Decimal Mode Cmd -- Selects decimal base for binary integer operations. (The default base is decimal.) -- Related: BIN,HEX,OCT,RCWS,STWS
3E576	xDECR	`( name → x_new )` Decrement Cmd -- Takes a variable on level 1, subtracts 1, stores the new value back into the original variable, and returns the new value to level 1. -- Related: INCR,STO+,STO-
0370DE	~xDEDICACE	Dedication message.
0250DE	~xDEF	
3E85C	xDEFINE	`( 'name=expr' → )` `( 'name(name1...)=expr(name1...) → )` Define Variable or Func Cmd -- Stores the expression on the right side of the = in the variable specified on the left side, or creates a user-defined function -- Related: STO
3B549	xDEG	`( → )` Degrees Cmd -- Sets Degrees angle mode. -- Related: GRAD,RAD
0360DE	~xDEGREE	

391D8	xDELALARM	(n →) Delete Alarm Cmd -- Deletes the alarm specified in level 1. -- Related: FINDALARM,RCLALARM,STOALARM
3D1C7	xDELAY	(x_delay →) Delay Cmd -- Specifies how many seconds the HP 48 waits between sending lines of information to the printer. -- Related: CR,OLDPRT,PRLCD,PRST,PRSTC,PRVAR,PR1
3EF3B	xDELKEYS	(rc.p →) ({ rc.p ... n } →) Delete Key Assignments Cmd -- Clears user-defined key assignments. -- Related: ASB,RCLKEYS,STOKEYS
3C51F	xDEPND	(name →) ({name} →) ({name y1 y2} →) ({y1 y2} →) (y1 y2 →) Dependent Variable Cmd -- Species the dependent variable (and its plotting range for TRUTH plots). -- Related: INDEP
3DCA7	xDEPTH	(→ n) Depth Cmd -- Returns a real number representing the number of objects present on the stack (before DEPTH was executed).
00E314	~xDERIV	(symb var → symb')
003314	~xDERVX	(symb → symb')
00F314	~xDESOLVE	(eq func → func')

3B1BA	xDET	([[]] → x) Determinant Func -- Returns the determinant of a square matrix. -- Related: CNRM,CROSS,DOT,RNRM
3EB84	xDETACH	(n →) (:port:n →) Detach Library Cmd -- Detaches the `library` with the specified number from the current directory. Each `library` has a unique number. If a port number is specified, it is ignored. -- Related: ATTACH,LIBS,PURGE
03A0AB	~x→DIAG	([[]] → vec) Matrix Diagonal to Array Cmd -- Returns a vector that contains the major diagonal elements of a matrix. -- Related: DIAG→
03B0AB	~xDIAG→	([] { dims } → [[]]) Array to Matrix Diagonal Cmd -- Takes an array and a specified dimension and returns a matrix whose major diagonal elements are the elements of the array. -- Related: →DIAG
00C0DE	~xDIAGMAP	
084314	~xDIFF	(→) Display a menu of calculus commands. -- Related: ARIT,BASE,CMPLX,EXP&LN,SOLVER,TRIGO
00E0AB	~xDIFFEQ	(→) Differential Eqn Plot Type Cmd -- Sets the plot type to DIFFEQ. -- Related: AXES,CONIC,FUNCTION,PARAMETRIC, POLAR,RKFSTEP,RRKSTEP,TRUTH
38BAE	xDIR	

39725	xDISP	(obj n_line →) Display Cmd -- Displays obj in the nth display line. -- Related: FREEZE,HALT,INPUT,PROMPT
0160DD	~xDISPXY	(ob {#x #y} %size →) Display ob (decompiled if nexessary) at the given display coordinates, using either the system font (%size=2) or the minifont (%size=1). First available in ROM 1.19-6.
0190DE	~xDISTRIB	
056314	~xDIV	([func] [vars] → func)
026314	~xDIV2	(symb1 symb2 → squot srem)
072314	~xDIV2MOD	(symb1 symb2 → squot srem)
044314	~xDIVIS	(symb → {})
071314	~xDIVMOD	(symb1 symb2 → sq)
062314	~xDIVPC	(symb1 symb2 n → symb3)
3816B	xDO	(→) DO Indefinite Loop Structure Cmd -- Starts DO ... UNTIL ... END indefinite loop structure. -- DO → UNTIL → END T/F → -- Related: END,UNTIL,WHILE
39527	xDOERR	(n →) (#n →) ($ →) (0 →) Do Error Cmd -- Executes a "user-specified" error, causing a program to behave exactly as if a normal error had occurred during program execution. -- Related: ERRM,ERRN,ERR0

`05B0AB` `~xDOLIST`

(`{1}...{n}` n prog → `{}`)
(`{1}...{n}` prog → `{}` `(n=1)`)
Do to List Cmd
`--`
Applies commands, programs, or user-defined functions to lists.
`--`

```
{lst}1 ...{lst}n n ≪prog≫ → {res}
{lst}1 ...{lst}n n cmd    → {res}
{lst}1 ...{lst}n n name   → {res}
{lst}1 ...{lst}n ≪prog≫   → {res}
{lst}1 ...{lst}n cmd      → {res}
{lst}1 ...{lst}n name     → {res}
--
```

Related: DOSUBS,ENDSUB,NSUB,STREAM

`0210DE` `~xDOMAIN`

`0540AB` `~xDOSUBS`

(`{}` n prog → `{}'`)
(`{}` prog → `{}' (n=1)`)
Do to Sublist Cmd
`--`
Applies a program or command to groups of elements in a list.
`--`

```
{list}1 n ≪prog≫  → {list}2
{list}1 n command → {list}2
{list}1 n name    → {list}2
{list}1 ≪prog≫    → {list}2
{list}1 command   → {list}2
{list}1 name      → {list}2
--
```

Related: `DOLIST`,ENDSUB,NSUB,STREAM

`3B1E1` `xDOT`

(`[1] [2]` → x)
Dot Product Cmd
`--`
Returns the dot product AoB of two arrays A and B, calculated as the sum of the products of the corresponding elements of the two arrays.
`--`
Related: CNRM,CROSS,DET,RNRM

3C484	xDRAW	(→) Draw Plot Cmd -- Plots the mathematical data in the reserved variable EQ or the statistical data in the reserved variable ΣDAT, using the specified x- and y-axis display ranges. -- <REF>TEXT:Reserved|EQ -- Related: AUTO,AXES,DRAX,ERASE,FREEZE, PICTURE,LABEL,PVIEW
06B0AB	~xDRAW3DMATRIX	([[]] v_min v_max →) -- Related: FAST3D
3C4BA	xDRAX	(→) Draw Axes Cmd -- Draws axes in PICT. -- Related: AXES,DRAW,LABEL
0230DE	~xDROITE	
3DC3B	xDROP	(ob →) Drop Object Cmd -- Removes the level 1 object from the stack. -- Related: CLEAR,DROPN,DROP2
3DC56	xDROP2	(ob1 ob2 →) Drop 2 Objects Cmd -- Removes the first two objects from the stack. -- Related: CLEAR,DROP,DROPN
3DCC7	xDROPN	(ob1...obn n →) Drop n Objects Cmd -- Removes the first n + 1 objects from the stack (the first n objects excluding the integer n itself). -- Related: CLEAR,DROP,DROP2

3EFEF	xDTAG	(tag:obj → obj) Delete Tag Cmd -- DTAG removes all tags (labels) from an object. -- Related: LIST→,→TAG
3DBEA	xDUP	(ob → ob ob) Duplicate Object Cmd -- DUP returns a copy to level 1 of the object in level 1. -- Related: DUPN,DUP2,PICK
3DC05	xDUP2	(1 2 → 1 2 1 2) Duplicate 2 Objects Cmd -- DUP2 returns copies of the objects in levels 1 and 2 of the stack. -- Related: DUP,DUPN,PICK
3F29A	xDUPDUP	(1 → 1 1) Duplicate 2 Objects Cmd -- DUP2 returns copies of the objects in levels 1 and 2 of the stack. -- Related: DUP,DUPN,NDUPN,DUP2
3DCE2	xDUPN	(1...n n → 1...n 1...n) Duplicate n Objects Cmd -- Takes an integer n from level 1 of the stack, and returns copies of the objects in stack levels 2n through n + 1. -- Related: DUP,DUP2,PICK
3B06E	xD>R	(x → (π/180)x) Degrees to Radians Func -- Converts a real number representing an angle in degrees to its equivalent in radians. -- x → (π/180) x 'sym' → 'D→R(sym)' -- Related: R→D UserRPL: xD→R

0070DD	~xEDIT	`( ob → ob' )` Move object to command line to edit it. -- Related: VISIT,EDITB,VISITB
0090DD	~xEDITB	`( ob → ob' )` Open the most suitable editor for object. For example, for a matrix, the matrix editor is opened. -- Related: VISIT,VISITB,EDIT
39B1E	xCONSTANTe	`( → e )` e Func -- Returns the symbolic constant e or its numerical representation, 2.71828182846. -- Related: EXP,EXPM,i,LN,LNP1,MAXR,MINR,π UserRPL: `xe`
02E314	~xEGCD	`( symb1 symb2 → symb3 symb4 symb5 )`
02C0AB	~xEGV	`( [[]] → [[evect]]' [evals] )` Eigenvalues and Eigenvectors Command -- Computes the eigenvalues and right eigenvectors for a square matrix. -- Related: EGVL
02D0AB	~xEGVL	`( [[]] → [egval] )` Eigenvalues Cmd -- Computes the eigenvalues of a square matrix. -- Related: EGV
3805D	xELSE	`( → )` ELSE Cmd -- Starts false clause in conditional or error-trapping structure. See the IF and IFERR keyword entries for syntax information. -- Related: IF,CASE,`DO`,ELSE,IFERR,`REPEAT`,THEN,`UNTIL`,`WHILE`

`38A54` `xENDDO` (1/0 →)
END Cmd
--
Ends conditional, error-trapping, and indefinite loop structures. ; See the IF, CASE, IFERR, DO, and WHILE keyword entries for syntax information.
--
Related: IF,CASE,DO,ELSE,IFERR,REPEAT, THEN,UNTIL,WHILE UserRPL: xEND

`3807D` `xIFEND`
END Cmd
--
Ends conditional, error-trapping, and indefinite loop structures.
--
See the IF, CASE, IFERR, DO, and WHILE keyword entries for syntax information.
--
Related: IF,CASE,DO,ELSE,IFERR,REPEAT, THEN,UNTIL,WHILE UserRPL: xEND

`38A2F` `xWHILEEND`
END Cmd
--
Ends conditional, error-trapping, and indefinite loop structures.
--
See the IF, CASE, IFERR, DO, and WHILE keyword entries for syntax information.
--
Related: IF,CASE,DO,ELSE,IFERR,REPEAT, THEN,UNTIL,WHILE UserRPL: xEND

`0570AB` `~xENDSUB` (→ x)
Ending Sublist Cmd
--
Provides a way to access the total number of sublists contained in the list used by DOSUBS.
--
Related: DOSUBS,NSUB

3B5DA	xENG	(n →) Engineering Mode Cmd -- Sets the number display format to Engineering mode, which displays one to three digits to the left of the fraction mark (decimal point) and an exponent that is a multiple of three. The total number of significant digits displayed is n + 1. -- Related: FIX,SCI,STD
088314	~xEPSX0	(symb1 → symb2)
00B0DD	~xEQW	(symb → symb') Open Equation Writer to edit an object. If the object is not symbolic, the object is placed into the command line. -- Related: EDIT,EDITB,VISIT,VISITB

3BDE6	xEQ>	('l=r' → l r)
		Equation to Stack Cmd
		--
		Separates an equation into its left and right sides.
		--
		`'sym1=sym2' → 'sym1' 'sym2'`
		`z           → z      0`
		`'name'      → 'name' 0`
		`x_u         → x_u    0`
		`'sym'       → 'sym'  0`
		--
		Related: ARRY→,DTAG,LIST→,OBJ→,STR→
		UserRPL: xEQ→

3C553	xERASE	(→) Erase PICT Cmd -- Erases PICT, leaving a blank PICT of the same dimensions. -- Related: DRAW
3955B	xERR0	(→) Clear Last Error Number Cmd -- Clears the last error number so that a subsequent execution of ERRN returns # 0h, and clears the last error message. -- Related: DOERR,ERRM,ERRN

`39591`	`xERRM`	`( → $msg )`
		Error Message Cmd
		`--`
		Returns a string containing the error message of the most recent calculator error.
		`--`
		Related: DOERR,`ERRN`,`ERR0`

`39576`	`xERRN`	`( → $nerr )`
		Error Number Cmd
		`--`
		Returns the error number of the most recent calculator error.
		`--`
		Related: DOERR,`ERRM`,`ERR0`

`038314`	`~xEULER`	`( z1 → z2 )`

`395AC`	`xEVAL`	`( ob → ? )`
		Evaluate Object Cmd
		`--`
		Evaluates the object.
		`--`
		`obj → (see below)`
		Obj. Type Effects of Evaluation Local Name Recalls the contents of the variable. Global Name Calls the contents of the variable: ; A name is evaluated. A program is evaluated. A directory becomes the current directory. Other objects are put on the stack. If no variable exists for a given name, evaluating the name returns the name to the stack. Program. Enters each object in the program: Names are evaluated (unless quoted). ed). Cmds are evaluated. Other objects are put on the stack. List Enters each object in the list: Names are evaluated. Cmds are evaluated. Programs are evaluated. Other objects are put on the stack. Tagged If the tag specifies a port, recalls and evaluates the specified object. Otherwise, puts the untagged object on the stack. Algebraic Enters each object in the algebraic expression: Names are evaluated. Cmds are evaluated. Other objects are put on the stack. Cmd, Func, XLIB Name Evaluates the specified object. Other Objects Puts the object on the stack.
		`--`
		Related: →`NUM`,SYSEVAL

3A9B7	xEXP	(x → x') Exponential Analytic Func -- Returns the exponential, or natural antilogarithm, of the argument; that is, e raised to the given power. -- z → ez 'sym' → 'EXP(sym)' -- Related: ALOG,EXPM,LN,LOG
06C314	~xEXLR	(symb → symb1 symb2)
01A0DE	~xEXP2POW	
3E5E9	xEXPAN	(symb1 → symb2) Expand Products Cmd -- Rewrites an algebraic expression or equation by expanding products and powers. -- Related: COLCT,EXPAND,ISOL,QUAD,SHOW
000314	~xEXPAND	(symb1 → symb2) ([symb1] → [symb2]) Expand Products Cmd -- Rewrites an algebraic expression or equation by expanding products and powers.
076314	~xEXPANDMOD	(symb1 → symb2)
3E25E	xEXPFIT	(→) Exponential Curve Fit Cmd -- Stores EXPFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the exponential curve futting model. -- <REF>TEXT:Reserved|ΣPAR -- Related: BESTFIT,LR,LINFIT,LOGFIT,PWRFIT
087314	~xEXP&LN	
017314	~xEXPLN	(symb1 → symb2)

3AB6F	xEXPM	(x → x') Exponential Minus 1 Analytic Func -- Returns e^x - 1. -- x → e^x - 1 'sym' → 'EXPM(sym)' -- Related: EXP,LNP1
0050AB	~xEYEPT	(xx xy xz →) Eye point command. -- Specifies the coordinates of the eye point in a perspective plot. The y coordinate must be 1 unit less than the volume's nearest point. These values are stored in reserved variable VPAR. -- <REF>TEXT:Reserved|VPAR -- Related: NUMX,NUMY,XVOL,XXRNG,YVOL, YYRNG,ZVOL
0620AB	~xF0λ	(y_lambda xT → x_power) Black Body Emissive Power Func -- Returns the fraction of total black-body emissive power.
001314	~xFACTOR	(symb → symb1*symb2...) (z → z1*z2...)
077314	~xFACTORMOD	(symb → symb1*symb2...) Eye Point Cmd -- Specifies the coordinates of the eye point in a perspective plot. -- xpoint ypoint zpoint →
043314	~xFACTORS	(z → {z1 m1...}) (symb → {symb1 m1...})

0600AB	~xFANNING	(x_x/D y_Re → x_fanning)
		Fanning Friction Factor Func
		--
		Calculates the Fanning friction factor of certain fluid flows.
		--
		`xx/D   yRe   → xfanning`
		`xx/D   'sym' → 'FANNING(xx/D,sym)'`
		`'sym'  yRe   → 'FANNING(sym,yRe)'`
		`'sym1' 'sym2'→ 'FANNING(sym1,sym2)'`
		--
		Related: DARCY

3F2DF	xFAST3D	(→) Fast 3D plot type command -- Set the plot type to FAST3D. -- Related: BAR,CONIC,DIFFEQ,FUNCTION,GRIDMAP, HISTOGRAM,PARAMETRIC,PARSURFACE,PCONTOUR, POLAR,SCATTER,SLOPEFIELD,TRUTH,WIREFRAME,YSLICE
3B529	xFC?	(n → 0/1) Flag Clear? Cmd -- Tests whether the system or user flag specified by nflag number is clear, and returns a corresponding test result: 1 (true) if the flag is clear or 0 (false) if the flag is set. -- Related: CF,FC?C,FS?,FS?C,SF
3B635	xFC?C	(n → 0/1) Flag Clear? Clear Cmd -- Tests whether the system or user flag specified by nflag number is clear, and returns a corresponding test result: 1 (true) if the flag is clear or 0 (false) if the flag is set. After testing, clears the flag. -- Related: CF,FC?,FS?,FS?C,SF
041314	~xFCOEF	([] → symb)
0180DE	~xFDISTRIB	

`01A0AB`	`~xFFT`	`( [] → []' )` Discrete Fourier Transform Cmd -- Computes the 1- or 2-dimensional discrete Fourier transform of an array. -- Related: IFFT
`00C0DD`	`~xFILER`	`( → )`

`391AE`	`xFINDALARM`	`( date → n )`
		`( {date time} → n )`
		`( 0 → n )`
		Find Alarm Cmd
		--
		Returns the alarm index nindex of the first alarm due after the specified time.
		--
		Related: `DELALARM`,RCLALARM,STOALARM

`3ED76`	`xFINISH`	`( → )` Finish Server Mode Cmd -- Terminates Kermit Server mode in a device connected to an HP 48. -- Related: BAUD,CKSM,KGET,PARITY,PKT, RECN,RECV,SEND,SERVER
`3B59A`	`xFIX`	`( n → )` Fix Mode Cmd -- Sets the number display format to Fix mode, which rounds the display to n display places. -- Related: `SCI`,`STD`,`ENG`
`0170AB`	`~xFLASHEVAL`	`( # → ? )` Evaluate flash command -- Evaluates unnamed flash functions. The number is of the form ffffbbbh, where bbb is the bank ID and ffff is the function number. -- Related: `EVAL`,LIBEVAL,SYSEVAL

`3ACD1` `xFLOOR`

(x → n)
Floor Func
--
Returns the greatest integer that is less than or equal to the argument.
--
```
x      → n
x_u    → n_u
'sym'  → 'FLOOR(sym)'
```
--
Related: CEIL,IP,RND,TRNC

`00F0DD` `~xFONT6`

(→ font)
Returns the system FONT6 object.
--
Related: FONT7,FONT8,→FONT,FONT→

`00E0DD` `~xFONT7`

(→ font)
Returns the system FONT7 object.
--
Related: FONT6,FONT8,→FONT,FONT→

`00D0DD` `~xFONT8`

(→ font)
Returns the system FONT8 object.
--
Related: FONT6,FONT7,→FONT,FONT→

`0030DD` `~xFONT→`

(→ font)
Returns the current system font.
--
Related: FONT6,FONT7,FONT8,→FONT

`0020DD` `~x→FONT`

(font →)
Set font function.
--
Sets the system font.
--
Related: FONT6,FONT7,FONT8,FONT→

38252	xSTARTVAR	(start finish →)
		FOR Definite Loop Structure Cmd
		--
		Starts
		`FOR ... NEXT and`
		`FOR ... STEP`
		definite loop structures.
		--
		`FOR  xstart xfinish        →`
		`           NEXT           →`
		`FOR  xstart xfinish        →`
		`    STEP   xincrement     →`
		`    STEP   'symincrement' →`
		--
		Related: NEXT,START,STEP UserRPL: `xFOR`

05E314	~xFOURIER	(symb z → c_z)

3AC87	xFP	(x → x')
		Fractional part Func
		--
		Returns the fractional part of an argument.
		--
		`x     → y`
		`x_u   → y_u`
		`'sym' → 'FP(sym)'`
		--
		Related: IP

3EB2C	xFREE	Not useful on the 49G. Free RAM Card Cmd
		--
		Frees (makes independent) the previously merged RAM in port 1. FREE is provided for compatibility with the HP 48SX, which could merge RAM in port 2 as well. See FREE1.
		--
		`{ }                       nport →`
		`{ namebackup ... nlib } nport →`
		`namebackup                nport →`
		`nlib                      nport →`

39745	xFREEZE	(n →) Freeze Display Cmd -- Freezes the part of the display specified by ndisplay area, so that it is not updated until a key is pressed. -- Related: CLLCD,DISP,`HALT`
042314	~xFROOTS	(symb → [])

3B615	xFS?C	(n → 0/1) Flag Set? Clear Cmd -- Tests whether the system or user flag specified by nflag number is clear, and returns a corresponding test result: 1 (true) if the flag is set or 0 (false) if the flag is clear. After testing, clears the flag -- Related: CF,FC?,FC?C,FS?C,SF
3B509	xFS?	(n → 0/1) Flag Set Cmd -- Tests whether the system or user flag specified by nflag number is set, and returns a corresponding test result: 1 (true) if the flag is set or 0 (false) if the flag is clear. -- Related: CF,FC?,FC?C,FS?,SF
3C955	xFUNCTION	(→) Function Plot Type Cmd -- Sets the plot type to `FUNCTION`. -- Related: `BAR`,`CONIC`,DIFFEQ,FASTEQ,FAST3D, GRIDMAP,`HISTOGRAM`,`PARAMETRIC`,PARSURFACE, PCONTOUR,`POLAR`,`SCATTER`,SLOPEFIELD,`TRUTH`, WIREFRAME,YSLICE
06B314	~xFXND	('x/y' → x y)

7.2 G-M

0070DE	~xGAMMA	(x → x')
04D314	~xGAUSS	(symb [vars] → [diag] [P] symb' [vars])
075314	~xGCDMOD	(x1 x2 → x3)
02C314	~xGCD	(x1 x2 → x3)

3C1C7	xGET	(ob n → elm) ob = [] or [[]] or {} or name pos = n or {n} or {n m} Get Element Command -- Returns from the level 2 array or list (or named array or list) the real or complex number zget or object objget whose position is specified in level 1.

```
--
[[ mat ]]    nposition      → zget
[[ mat ]]    { nrow mcol } → zget
'namematrix' nposition      → zget
'namematrix' { nrow mcol } → zget
[ vector ]   nposition      → zget
[ vector ]   { nposition  }→ zget
'namevector' nposition      → zget
'namevector' { nposition } → zget
{ list }     nposition   → objget
{ list }     {nposition} → objget
'namelist'   nposition   → objget
'namelist'   {nposition} → objget
--
```

Related: GETI,PUT,PUTI

3C22D xGETI

(ob pos → ob' pos' elm)
ob = [] or [[]] or {} or name
pos = n or {n} or {n m}
Get and Increment Index Command
--
Returns from the level 2 array or list (or named array or list) the real or complex number zget or object objget whose position is specified in level 1, along with the level 2 argument and the next position in that argument.
--
[[mat]] npos1
→ [[mat]] npos2 zget
[[mat]] { nr mc }1
→ [[mat]] { nr mc }2 zget
'namemat' npos1
→ 'namemat' npos2 zget
'namemat' { nr mc }1
→ 'namemat' { nr mc }2 zget
[vec] npos1
→ [vec] npos2 zget
[vec] { npos1 }
→ [vec] { npos2 } zget
'namevec' npos1
→ 'namevec' npos2 zget
'namevec' { npos1 }
→ 'namevec' { npos2 } zget
{ list } npos1
→ { list } npos2 objget
{ list } { npos1 }
→ { list } { npos2 } objget
'namelist' npos1
→ 'namelist' npos2 objget
'namelist' { npos1 }
→ 'namelist' { npos2 } objget
--
Related: GET,PUT,PUTI

0660AB ~xgmol

`3C74A` `xGOR`

(g_targ {#n #m} grob → g_targ')
(g_targ (x,y) grob → g_targ')
(PICT ... ... →)
Graphics OR Cmd
--
Superimposes grob1 onto grobtgt or PICT, with the upper left corner of grob1 positioned at the specified coordinate in grobtgt or PICT.
--
```
grobtgt {#n #m} grob1 → grob'
grobtgt (x,y)   grob1 → grob'
PICT    {#n #m} grob1 →
PICT    (x,y)   grob1 →
```
--
Related: GXOR,REPL,SUB

`3B57F` `xGRAD`

(→)
Grads Mode Cmd
--
Sets Grads angle mode.
--
Related: GRAD,RAD

`0090DE` `~xGRAMSCHMIDT`

`00A0AB` `~xGRIDMAP`

(→)
GRIDMAP Plot Type Cmd
--
Sets plot type to GRIDMAP.
--
Related: BAR,CONIC,DIFFEQ,FUNCTION, HISTOGRAM,PARAMETRIC,PARSURFACE, PCONTOUR,POLAR,SCATTER,SLOPEFIELD, TRUTH,WIREFRAME,YSLICE

`07C314` `~xGROBADD`

(gr1 gr2 → gr3)
Combines two grob objects.

`38C1B` `xGROB`

`3C8A1` `x>GROB`

(ob n_chrsize → grob)
Stack to Graphics Object Cmd
--
Creates a graphics object representing the level 2 object, where the argument nchar size specifies the character size of the representation.
--
Related: →LCD,LCD→ UserRPL: x→GROB

3C7D8	xGXOR	`( g_targ {#n #m} g_src → g_targ' )`
		`( g_targ (x,y) g_src → g_targ' )`
		`( PICT ... ... → )`
		Graphics Exclusive `OR` Cmd
		--
		Superimposes grob1 onto grobtgt or PICT, with the upper left corner of grob1 positioned at the specified coordinate in grobtgt or PICT.
		--
		`grobtgt {#n #m} grob1→ grobresult`
		`grobtgt (x,y)   grob1→ grobresult`
		`PICT    {#n #m} grob1→`
		`PICT    (x,y)   grob1→`
		--
		Related: GOR,REPL,SUB

046314	~xHADAMARD	`( [M1] [M2] → [M3] )`
020314	~xHALFTAN	`( symb → symb' )`
3880D	xHALT	`( → )` Halt Program Cmd -- Halts program execution. -- Related: CONT,KILL

0510AB	~xHEAD	`( {} → ob )`
		`( $ → $' )`
		First Listed Element Cmd
		--
		Returns the first element of a list or string.
		--
		Related: TAIL

0040DD	~x→HEADER	`( n → )` Set header size in lines: 0,1 or 2. -- Related: →HEADER
0050DD	~xHEADER→	`( → n )` Header size: Returns current header size in lines. -- Related: →HEADER
0320DE	~xHELP	
05C314	~xHERMITE	`( z → symb )`
059314	~xHESS	`( symb [vars] → [M] [grad] [vars] )`

3B68B	xHEX	(→) Hexadecimal Mode Cmd -- Selects hexadecimal base for binary integer operations. (The default base is decimal.) -- Related: BIN,OCT,DEC,RCWS,STWS
054314	~xHILBERT	(z → [M])
3C9C1	xHISTOGRAM	(→) Histogram Plot Type Cmd -- Sets the plot type to HISTOGRAM. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAP,PARAMETRIC,PARSURFACE,PCONTOUR, POLAR,SCATTER,SLOPEFIELD,TRUTH,WIREFRAME,YSLICE
3E1CA	xHISTPLOT	(→) Draw Histogram Plot Cmd -- Plots a frequency histogram of the specified column in the current stat matrix (reserved matrix ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: BARPLOT,BINS,FREESE,PICTURE, PVIEW,RES,SCATRPLOT,XCOL
3B14C	xHMS-	(hms1 hms2 → hms3) Hours-Minutes-Seconds Minus Cmd -- Returns the difference of two real number, where the arguments and the result are interpreted in hours-minutes-seconds format. -- Related: HMS→,→HMS,HMS+
3B12C	xHMS+	(hms1 hms2 → hms3) Hours-Minutes-Seconds Plus Cmd -- Returns the sum of two real number, where the arguments and the result are interpreted in hours-minutes-seconds format. -- Related: HMS→,→HMS,HMS-

3B0EC	x>HMS	(x → x') Decimal to Hours-Minutes-Seconds Cmd -- Converts a real number representing hours or degrees with a decimal fraction to hours-minutes-seconds format. -- Related: HMS→,HMS+,HMS- UserRPL: x→HMS
3B10C	xHMS>	(x → x') Hours-Min-Sec to Decimal Cmd -- Converts a real number in hours -minutes-seconds format to its decimal form (hours or degrees with a decimal fraction. -- Related: →HMS,HMS+,HMS- UserRPL: xHMS→
39405	xHOME	(→) HOME Directory Cmd -- Makes the HOME directory the current directory. -- Related: CRDIR,PATH,PGDIR,UPDIR
037314	~xHORNER	(symb1 x → symb2 x symb3)
02B0DE	~xHYPERBOLIC	
39B3B	xi	(→ i)
031314	~xIABCUV	(n1 n2 n3 → n4 n5)
0120DE	~xIBASIS	
0060DE	~xIBERNOULLI	(n → x)
00B314	~xIBP	(uv' v → uv -u'v)
03B314	~xICHINREM	([]1 []2 → []3)
027314	~xIDIV2	(n1 n2 → quot rem)
3C02E	xIDN	(n → [[]]) ([[]] → [[]]') (name → [[]]) Identity Matrix Cmd -- Returns an identity matrix; that is, a square matrix with its diagonal elements equal to 1 and its off-diagonal elements equal to 0. -- Related: CON
02F314	~xIEGCD	(n1 n2 → c b a)

37F48	xIF	(→) IF Conditional Structure Cmd -- Starts IF ... THEN ... `END` and IF ... THEN ... ELSE ... `END` conditional structures. -- `IF` → `THEN T/F` → `END` → → `IF` → `THEN T/F` → `ELSE` → `END` → -- Related: CASE,ELSE,`END`,IFERR,THEN
387AC	xIFERR	(→) If Error Conditional Struct Cmd -- Starts IFERR ... THEN ... `END` and IFERR ... THEN ... ELSE ... `END` error trapping structures. -- Related: CASE,ELSE,`END`,IF,THEN
01B0AB	~xIFFT	(`[]` → `[]’`) Inverse Discrete Fourier Tsfm Cmd -- Computes the 1D or 2D inverse discrete Fourier transform of an array. -- Related: FFT
396A4	xIFT	(`0/1 obj` → `?`) IF-THEN Cmd -- Executes obj if T/F is nonzero. Discards obj if T/F is zero. -- Related: IFTE
395F3	xIFTE	(`0/1 objT objF` → `?`) IF-THEN-ELSE Cmd -- Executes objT if T/F is nonzero. Discards objF if T/F is zero. -- Related: IFT
39B3B	xi	(→ `i`)
011314	~xILAP	(`symb` → `symb’`)

3B87E	xIM	((x,y) → y)
		([] → []')
		Imaginary Part Func
		--
		Returns the imaginary part of its (complex) argument.
		--
		`x          → 0`
		`(x,y)      → y`
		`[ R-arr ] → [ R-arr ]`
		`[ C-arr ] → [ R-arr ]`
		`'sym'      → 'IM(sym)'`
		--
		Related: C→R,RE,R→C

0100DE	~xIMAGE	
3E54C	xINCR	(name → x) Increment Cmd -- Takes a variable on level 1, adds 1, stores the new value back into the original variable, and returns the new value to level 1. -- Related: DECR
3C33E	xINDEP	(name →) ({name} →) Independent Variable Cmd -- Specifies the independent variable and its plotting range. -- Related: DEPND
08A314	~x∞	(→ '+∞') Infinity UserRPL: x∞

04C0AB	~xINFORM	($ {flds} fmt {rst} {init} → {} 1) ($ {flds} fmt {rst} {init} → 0) User-Defined Dialog Box Cmd -- Creates a user-defined input form (dialog box). -- 5: "title" 4: { s1 ... s2...sn } 3: format 2: { resets } 1: { init } ↓ ; 2: { vals } 1: 1 -- 5: "title" 4: { s1 ... s2...sn } 3: format 2: { resets } 1: { init } ↓ ; 1: 0 -- "title" -- `Title`. This appears at the top of the dialog box. -- { s1 ... s2...sn } -- Field definitions. A field definition (sx) can have two formats : "label", a field type, or { "label" "helpInfo" type0 type1 ... typen }, a field label with optional help text that appears near the bottom of the screen, and an optional list of valid object types for that field. If object types aren't specified, all object types are valid. For information about object types, see the `TYPE` command. When creating a multi-column dialog box, you can span columns by using an empty list as a field definition. A field that appears to the left of an empty space automatically expands to fill the empty space. -- { resets } -- Default values displayed when RESET is selected. Specify reset values in the list in the same order as the fields were specified. To specify no value, use the NOVAL command as a place holder. This list can be empty. -- { init } -- Initial values displayed when the dialog box appears. Specify initial values in the list in the same order as the fields were specified. To specify no value, use the

3EEBD	xINPUT	($prompt $ → $') ($prompt {specs} → $') Input Cmd -- Prompts for data input to the command line and prevents the user access to stack operations. -- Related: PROMPT,STR→
0290DE	~xINTEGER	
3F007	xINT	(f(var) var x0 → F(x0))
3A32B	xINV	(x → 1/x) ([[]] → [[]]') Inverse (1/x) Analytic Func -- Returns the reciprocal or the matrix inverse. -- Related: SINV,/
004314	~xINTVX	(f(x) → F(x))
074314	~xINVMOD	(x → x')
3AC3D	xIP	(x → n) Integer Part Func -- Returns the integer part of the argument. -- x → n x_u → n_u 'sym' → 'IP(sym)' -- Related: FP
029314	~xIQUOT	(n1 n2 → n3)
02B314	~xIREMAINDER	(n1 n2 → n3)
3F0B7	xI>R	(n → x) UserRPL: xI→R
3E648	xISOL	(symb var → symb') Isolate Variable Cmd -- Returns an algebraic symb' that rearranges symb to "isolate" the first occurrence of variable var. -- Related: COLCT,EXPAN,QUAD,SHOW,SOLVE
00D0DE	~xISOM	
03C314	~xISPRIME?	(n → 1) (n → 0)
3DB62	xFORMUNIT	UserRPL: x_

3F053	x;	
089314	~x?	
389EF	x'	
38A14	xENDTIC	UserRPL: x'
389B9	x<<	UserRPL: x≪
389D4	x>>	UserRPL: x≫
38999	x>>ABND	UserRPL: x≫
050314	~xJORDAN	([nxn] → minpol chrpol {} [])
00F0DE	~xKER	
3EE2C	xKERRM	(→ msg) Kermit Error Message Cmd -- Returns the text of the most recent Kermit error packet. -- Related: FINISH,KGET,PKT,RECN,RECV, SEND,SERVER
39854	xKEY	(→ rc 1) (→ 0) Key Cmd -- Returns to level 1 a test result, and if a key is pressed, returns to level 2 the row-column location xn m of that key. -- Related: WAIT,KEYEVAL
07B314	~xKEYEVAL	(rc.p → ?) Execute the action associated with the specified key. The number is row r, column c, plane p. If negative, force the default key action even in USER mode. -- <REF>TEXT:Keycodes
06C0AB	~x→KEYTIME	(ticks →) Set a new keytime value. This is the number of ticks which will be required between subsequent key presses. Keys pressed faster will not register. -- Related: KEYTIME→
06D0AB	~xKEYTIME→	(→ ticks) Return the current value of keytime. -- Related: →KEYTIME

3ECE4	xKGET	`( name → )`
		`( "name" → )`
		`( {names} → )`
		`( {{old new}...} → )`
		Kermit Get Cmd
		--
		Used by a local Kermit to get a Kermit server to transmit the named object(s).
		--
		Related: BAUD,CKSM,FINISH,PARITY,RECN, RECV,SEND,SERVER,TRANSIO

394F1	xKILL	`( → )` Cancel Halted Programs Cmd -- Cancels all currently halted programs. (Halted programs are typically canceled by pressing PRG NXT RUN KILL.) If KILL is executed within a program, that program is also canceled. -- Related: CONT,DOERR,HALT,PROMPT
3C5C9	xLABEL	`( → )` Label Axes Cmd -- Labels axes in PICT with x- and y-axis variable names and with the minimum and maximum values of the display ranges. -- Related: LABEL,DRAW,DRAX
05D314	~xLAGRANGE	`( [2xn] → pol )`

0000DD	~x→LANGUAGE	`( n → )`
		Set language for error messages etc.
		`0 English`
		`1 French`
		`2 Spanish`
		--
		Related: LANGUAGE→

0010DD	~xLANGUAGE→	`( → n )` Return the current language value. -- Related: →LANGUAGE
058314	~xLAPL	`( symb [vars] → symb' )`
010314	~xLAP	`( symb → symb' )`

397E5	xLAST	(→ ob1 .. obn) Last Arguments Cmd -- Returns copies of the arguments of the most recently executed command. UserRPL: xLASTARG
0670AB	~xlbmol	
3C881	x>LCD	(grob →) Graphics Object to LCD Cmd -- Displays the graphics object from level 1, with its upper left pixel in the upper left corner of the display. -- Related: LCD→,BLANK,→GROB UserRPL: x→LCD
3C866	xLCD>	(→ grob) LCD to Graphics Object Cmd -- Returns the current stack and menu display as a 131x64 graphics object. -- Related: →LCD,→GROB UserRPL: xLCD→
02D314	~xLCM	(symb1 symb2 → symb3)
055314	~xLCXM	(n1 n2 prog → [])
012314	~xLDEC	(symb1 symb2 → symb3)
05A314	~xLEGENDRE	(n → pol)
032314	~xLGCD	({symb...} → {} gcd)
0160AB	~xLIBEVAL	(# → ?) Evaluate Library Func Cmd -- Evaluates unnamed library functions. The number is of the form lllfffh where lll is a library number and fff a function number. -- Related: EVAL,SYSEVAL
3EB42	xLIBS	(→ {title nlib nport ...}) Libraries Cmd -- Lists the title, number, and port of each library attached to the current directory. -- Related: ATTACH,DETACH
005314	~xLIMIT	(func point → lim)
014314	~xLIN	(symb → symb')

3C68C	xLINE	((x1,y1) (x2,y2) →) ({#n1 #m1} {#n2 #m2} →) Draw Line Cmd -- Draws a line in PICT between the coordinates in levels 1 and 2. -- Related: ARC,BOX,TLINE
3E156	xSIGMALINE	(→ symb) Regression Model Formula Cmd -- Returns an expression representing the best fit line according to the current statistical model, using X as the independent variable name, and explicit values of the slope and intercept taken from the reserved variable ΣPAR. -- <REF>TEXT:Reserved|ΣPAR -- Related: BESTFIT,COLΣ,CORR,COV, EXPFIT,LINFIT,LOGFIT,LR,PREDX, PREDY,PWRFIT,XCOL,YCOL UserRPL: xΣLINE
3E214	xLINFIT	(→) Linear Curve Fit Cmd -- Stores LINFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the linear curve fitting model. -- <REF>TEXT:Reserved|ΣPAR -- Related: BESTFIT,EXPFIT,LOGFIT,LR, PWRFIT
0150AB	~xLININ	(symb var → 0/1) Linear Test Func -- Tests whether an algebraic is structurally linear for a given variable.
052314	~xLINSOLVE	([eqs] [vars] → [eqs] {pp} sol)

3BAC1	xLIST>	({} → ob1...obn n) List to Stack Cmd -- Takes a list of n objects and returns them in separate levels, and returns the total number of objects to level 1. -- Related: ARRY→,DTAG,EQ→,→LIST, OBJ→,STR→ UserRPL: xLIST→
3B7D2	x>LIST	(ob1 .. obn n → {}) Stack to List Cmd -- Takes n objects from level n+1 through level 2 and returns a list of those n objects. -- Related: →ARRY,LIST→,→STR, →TAG,→UNIT UserRPL: x→LIST
0550AB	~xΔLIST	({} → {}') List Differences Cmd -- Returns the first differences of the elements in a list. -- Related: ΣLIST,ΠLIST,STREAM
05A0AB	~xΠLIST	({} → x) List Product Cmd -- Returns the product of the elements in a list. -- Related: ΣLIST,ΔLIST,STREAM
0590AB	~xΣLIST	({} → x) List Sum Cmd -- Returns the sum of the elems in a list. -- Related: ΠLIST,STREAM
3AA01	xLN	(x → x') Natural Logarithm Analytic Func -- Returns the natural (base e) logarithm of the argument. -- z → ln z 'sym' → 'LN(sym)' -- Related: ALOG,EXP,ISOL,LNP1,LOG
06D314	~xLNAME	(symb → [vars])

016314	~xLNCOLLECT	`( symb → symb' )`

3AB2F	xLNP1	`( x → x' )`
		Natural Log of x+1 Analytic Func
		--
		Returns ln (x + 1).
		--
		`x      → ln(x+1)`
		`'sym' → 'LNP1(sym)'`
		--
		Related: EXPM,LN

3AA73	xLOG	`( x → x' )`
		Common Logarithm Analytic Func
		--
		Returns the common logarithm (base 10) of the argument.
		--
		`z      → log z`
		`'sym' → 'LOG(sym)'`
		--
		Related: ALOG,EXP,ISOL,LN

3E239	xLOGFIT	`( → )` Logarithmic Curve Fit Cmd -- Stores LOGFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the logarithmic curve-fitting model. -- <REF>TEXT:Reserved|ΣPAR -- Related: BESTFIT,EXPFIT,LINFIT,LR,PWRFIT
0320AB	~xLQ	`( [[]] → [[L]] [[Q]] [[P]] )` LQ Factorization of a Matrix Cmd -- Returns the LQ factorization of an nm matrix. -- Related: LSQ,QR

`3DF83` `xLR`

`( → Intercept Slope )`
Linear Regression Cmd
`--`
Uses the currently selected statistical model to calculate the linear regression coefficients (intercept and slope) for the selected dependent and independent variables in the current stat matrix (reserved variable ΣDAT).
`--`
<REF>TEXT:Reserved|ΣDAT
`--`
Related: BESTFIT,`COLΣ`,`CORR`,`COV`,EXPFIT, ΣLINE,LINFIT,LOGFIT,`PREDX`,`PREDY`, PWRFIT,`XCOL`,`YCOL`

`02B0AB` `~xLSQ`

`( [B] [[A]] → []' )`
`( [[B]] [[A]] → [[]]' )`
Least Squares Solution Cmd
`--`
Returns the minimum norm least squares solution to any system of linear equations where A X = B
`--`
Related: LQ,RANK,QR,/

`0300AB` `~xLU`

`( [[]] → [[L]] [[U]] [[P]] )`
LU Dec of a Sq. Matrix Cmd
`--`
Returns the LU decomposition of a square matrix.
`--`
Related: DET,INV,LSQ,/

`06A314` `~xLVAR`

`( symb → symb [vars] )`

`051314` `~xMAD`

`( [] → det inv coeff cpol )`

`07F314` `~xMAIN`

Show the main CAS menu.

`3B02E` `xMANT`

`( x → x' )`
Mantissa Func
`--`
Returns the mantissa of the argument.
`--`
```
x      → ymant
'sym'  → 'MANT(sym)'
--
```
Related: SIGN,XPON

`066314` `~xMAP`

`( {} prog → {}' )`

3DB04	xMATCHDN	`( symb {spat srepl} → symb' 0/1 )`
		`( symb {spat srepl scond} → symb' 0/1 )`
		Match Pattern Down Cmd
		--
		Rewrites an expression.
		--
		Related: X↑MATCH UserRPL: x↓MATCH

3DAD0	xMATCHUP	`( symb {spat srepl} → symb' 0/1 )`
		`( symb {spat srepl scond} → symb' 0/1 )`
		Bottom-Up Match and Replace Cmd
		--
		Rewrites an expression.
		--
		Related: X∇MATCH UserRPL: x↑MATCH

02F0DE	~xMATHS	Show the main MATH menu.
083314	~xMATR	Show the matrix menu. -- Related: ARIT,BASE,CMPLX,DIFF,EXP&LN, SOLVER,TRIGO

3ADA5	xMAX	`( x y → x' )`
		Maximum Func
		--
		Returns the greater (more positive) of the arguments.
		--
		`x       y       → max(x, y)`
		`x       'sym'   → 'MAX(x, sym)'`
		`'sym'   x       → 'MAX(sym, x)'`
		`'sym1'  'sym2'  → 'MAX(sym1, sym2)'`
		`x_u1    y_u2    → max(x_u1, y_u2)`
		--
		Related: MIN

39AE4	xMAXR	`( → MAXR )`
		Maximum Real Func
		--
		Returns the symbolic constant 'MAXR' or its numerical representation, 9.99999999999E499.
		--
		`→ 'MAXR'`
		`→ 9.99999999999E499`
		--
		Related: Ee,i,MINR,π

3DEE1	xMAXSIGMA	(→ xmax) (→ [x1...xn]) Maximum Sigma Cmd -- Finds the maximum coordinate value in each of the m columns of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: BINS,MEAN,MINΣ,SDEV,TOT,VAR UserRPL: xMAXΣ
0760AB	~xMCALC	(var →) ({vars} →) ("ALL" →) Make Calculated Value Cmd -- Designates a variable as a calculated value (not user-defined) for the Multiple-Equation Solver. -- Related: MUSER
3DEFC	xMEAN	(→ xmean) (→ [x1...xn]) Mean Cmd -- Returns the mean of each of the m columns of coordinate values in the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: BINS,MAXΣ,MINΣ,SDEV,TOT,VAR
3E8C1	xMEM	(→ x) Memory Available Cmd -- Returns the number of bytes of available RAM. -- Related: BYTES

3E9D4	xMENU	(% →) Display Menu Cmd -- Displays a built-in menu or a `library` menu, or displays a custom menu. -- `namemenu` → `{ listdefinition }` → `'namedefinition'` → `obj` → -- Related: RCLMENU,TMENU
07A314	~xMENUXY	(n1 n2 →) Menu of CAS commands.
3EB16	xMERGE	(1 →) Merge RAM Card Cmd Only useful on the 48. -- Merges the RAM from the card in port 1 with the rest of main user memory. Merged memory is no longer independent. -- Related: FREE,FREE1

3AE2B	xMIN	(x y → x')
		Minumum Func
		--
		Returns the lesser (more negative) of its two arguments.
		--
		`x y → min(x, y)`
		`x 'sym' → 'MIN(x, sym)'`
		`'sym' x → 'MIN(sym, x)'`
		`'sym1' 'sym2' → 'MIN(sym1, sym2)'`
		`x_u1 y_u2 → min(x_u1, y_u2)`
		--
		Related: MAX

0120DD	~xMINIFONT→	(→ font) Returns the current minifont. -- Related: →MINIFONT
0110DD	~x→MINIFONT	(font →) Sets the font as current minifont. -- Related: MINIFONT→

`0730AB`	`~xMINIT`	(→) Multiple Equation Menu Init Cmd -- Creates the reserved variable Mpar. -- <REF>TEXT:Reserved|Mpar -- Related: MITM,MROOT,MSOLVER
`39B01`	`xMINR`	(→ `MINR`) Minimum Real Func -- Returns the symbolic constant 'MINR' or its numerical representation, 1.00000000000E-499. -- → `'MAXR'` → `1.00000000000E-499` -- Related: e,i,MAXR,π
`3DF17`	`xMINSIGMA`	(→ `xmin`) (→ `[x1...xn]`) Minimum Sigma Cmd -- Finds the minimum coordinate value in each of the m current statistics matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: BINS,`MAXΣ`,`MEAN`,`SDEV`,`TOT`,`VAR` UserRPL: `xMINΣ`
`0740AB`	`~xMITM`	(`title {vars}` →) Multiple Equation Menu Item -- Order Cmd -- Changes multiple equation menu titles and order. -- Related: MINIT
`00E0DE`	`~xMKISOM`	

3AFCB	xMOD	(x y → x')
		Modulo Func
		--
		Returns a remainder defined by: x mod y = x - y floor (x/y)
		--
		`x      y      → x mod y`
		`x      'sym'  → 'MOD(x, sym)'`
		`'sym'  x      → 'MOD(sym, x)'`
		`'sym1' 'sym2' → 'MOD(sym1, sym2)'`
		--
		Related: FLOOR,/

079314	~xMODSTO	(mod →)
02C0DE	~xMODULAR	
0770AB	~xMROOT	(var → x) ("ALL" →) Multiple Roots Cmd -- Uses the Multiple-Equation Solver to solve for one or more variables using the equation set in Mpar -- Related: MCALC,MUSER
04E0AB	~xMSGBOX	($ →) Message Box Cmd -- Creates a user-defined message box. -- Related: CHOOSE,INFORM,PROMPT
0200DE	~xMSLV	
0720AB	~xMSOLVR	(→) Multiple-Equation Solver Cmd -- Gets the Multiple-Equation Solver variable menu for the set of equations defined by Mpar.
070314	~xMULTMOD	(symb1 symb2 → symb3)
0750AB	~xMUSER	(var →) ({vars} →) ("ALL" →) Make User-Defined Variable Cmd -- Designates a variable as user - defined for the Multiple-Equation Solver. -- Related: MCALC

7.3 N-S

0060DD	~x→NDISP	(n →) Set the number of program lines displayed on the screen.
01C0AB	~xNDIST	(xq v x → x') Normal Distribution Cmd -- Returns the normal probability distribution (bell curve) at x based on the mean m and variance v of the normal distribution. -- Related: UTPN
3F2B5	xNDUPN	(ob n → ob .. ob n) Duplicates object n times and returns n. -- Related: DUP,DUPDUP,DUPN,DUP2

39976	xNEG	(x → x')
		Negate Analytic Func
		--
		Changes the sign or negates an object.
		--
		`z         → -z`
		`#n1       → #n2`
		`[ arr ]   → [ -arr ]`
		`'sym'     → '-(sym)'`
		`x_u       → -x_u`
		`grob1     → grob2`
		`PICT1     → PICT2`
		--
		Related: ABS,CONJ,NOT,SIGN

394AA	xNEWOB	(ob → ob) New Object Cmd -- Creates a new copy of the specifiedfied object. -- Related: MEM,PURGE
3831C	xNEXT	(→) NEXT Cmd -- Ends definite loop structures. See the FOR and START command entries for syntax information. -- Related: FOR,START,STEP
03D314	~xNEXTPRIME	(n → n')

3F264	xNIP	(ob1 ob2 → ob2) -- Related: DUP,DUPDUP,DUPN,DUP2
3CB13	xNOT	(x → x') NOT Cmd -- Returns the one's complement or the logical inverse of the argument. -- #n1 → #n2 T/F → 0/1 "str1" → "str2" 'sym' → 'NOT sym' -- Related: AND,OR,XOR
3F0FC	xNOVAL	(→) INFORM Place Holder/Result Cmd -- Place holder for reset and initial values in user-defined dialog boxes. NOVAL is returned to the stack when a field is empty. -- Related: INFORM
3DE09	xNSIGMA	(→ nrows) Number of Rows Cmd -- Returns the number of rows in the current statistical matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: ΣX,ΣXY,ΣX2,ΣY,ΣY2 UserRPL: xNΣ
0560AB	~xNSUB	(→ npos) Number of Sublist Cmd -- Provides a way to access the current sublist position during an iteration of a program or command applied using DOSUBS. -- Related: DOSUBS,ENDSUB
3BBF9	xNUM	($ → n) Character Number Cmd -- Returns the character code n for the first character in the string. -- Related: CHR,POS,REPL,SIZE,SUB

0060AB ~xNUMX

(n →)
Number of X-Steps Cmd
--
Sets the number of x-steps for each y-step in 3D perspective plots.
--
Related: NUMY

0070AB ~xNUMY

(n →)
Number of Y-Steps Cmd
--
Sets the number of y-steps across the view volume in 3D perspective plots.
--
Related: NUMX

39785 x>NUM

(x → x')
Evaluate to Number Cmd
--
Evaluates a symbolic argument object and returns the numerical result.
--
`objsym → z`
--
Related: →Q,→Qpi UserRPL: `x→NUM`

3BE38 xOBJ>

(ob → ?)
Object to Stack Cmd
--
Separates an object into its components onto the stack. For some object types, the number of composites is returned to level 1.
--
```
(x,y)             →  x y
{obj1 ... objn}   → obj1 objn n
[x1 ... xn]       → x1 xn {n}
[[x11 ... xm n]]  → x11 xm n {m n}
"obj"             → evaluated-obj
'sym'             → obj1 ... objn n func
x_u               → x 1_u
:tag:obj          → obj "tag"
```
--
Related: `ARRY`→,C→R,DTAG,EQ→, R→C,STR→,→TAG UserRPL: `xOBJ→`

`3B6A6` `xOCT` (→)
Octal Mode Cmd
--
Selects octal base for binary integer operations. (The default base is decimal.)
--
Related: BIN,DEC,HEX,RCWS,STWS

`3950C` `xOFF` (→)
Off Cmd
--
Turns off the calculator.
--
Related: CONT,`HALT`,KILL

`3D0BC` `xOLDPRT`

Old Printer Cmd
--
Modifies the remapping string in the reserved variable PRTPAR so that the extended character set of the HP 48 matches that of the HP 82240A Infrared Printer. Not useful on the 49G.

`3EC75` `xOPENIO` (→)
Open I/O Port Cmd
--
Opens the serial port or the IR port using the I/O parameters in the reserved variable IOPAR.
--
<REF>TEXT:Reserved|IOPAR
--
Related: BUFLEN,CLOSEIO,SBRK,SRECV, STIME,XMIT

`3CA8D` `xOR` (x y → x')
`OR` Func
--
Returns the logical `OR` of two arguments.
--
```
#n1     #n2     → #n3
"str1" "str2" → "str3"
T/F1    T/F2    → 0/1
T/F     'sym'   → 'T/F OR sym'
'sym'   T/F     → 'sym OR T/F'
'sym1' 'sym2' → 'sym1 OR sym2'
```
--
Related: `AND`,`NOT`,`XOR`

3E8F0	xORDER	({names} →) Order Variables Cmd -- Reorders the variables in the current directory (shown in the VAR menu) to the order specified. -- Related: VARS
3DC8C	xOVER	(1 2 → 1 2 1) Over Cmd -- Returns a copy to stack level 1 of the object in level 2. -- Related: PICK,ROLL,ROLLD,ROT,SWAP
01F0DE	~xP2C	???
039314	~xPA2B2	(n → n')
3C98B	xPARAMETRIC	(→) Parametric Plot Type Cmd -- Sets the plot type to PARAMETRIC. -- Related: BAR,CONTOUR,DIFFEQ,FUNCTION, GRIDMAP,HISTOGRAM,PARSURFACE,PCONTOUR, POLAR,SCATTER,SLOPEFIELD,TRUTH, WIRE-FRAME,YSLICE
3EDEC	xPARITY	(n →) Parity Cmd -- Sets the parity value in the reserved variable IOPAR. -- <REF>TEXT:Reserved|IOPAR -- Related: BAUD,CKSM,TRANSIO
0090AB	~xPARSURFACE	(→) PARSURFACE Plot Type Cmd -- Sets the plot type to PARSURFACE. -- Related: BAR,CONIC,DIFFEQ,FAST3D, FUNCTION,GRIDMAP,HISTOGRAM,PARAMETRIC, PCONTOUR,POLAR,SCATTER,SLOPEFIELD,TRUTH, WIREFRAME,YSLICE
034314	~xPARTFRAC	(symb → symb')

393EA	xPATH	(→ {HOME dir1 .. dirn}) Current Path Cmd -- Returns a list specifying the path to the current directory. -- Related: CRDIR,HOME,PGDIR,UPDIR
04F314	~xPCAR	([nxn] → pol)
0450AB	~xPCOEF	([roots] → [coefs]) Monic Polynomial Coefficients Cmd -- Returns the coefficients of a monic polynomial (a polynomial with a leading coefficient of 1) having specific roots. -- Related: PEVAL,PROOT
00D0AB	~xPCONTOUR	(→) PCONTOUR Plot Type Cmd -- Sets the plot type to PCONTOUR. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE, POLAR,SCATTER,SLOPEFIELD,TRUTH, WIRE-FRAME,YSLICE
01F0AB	~xPCOV	(→ xpcovariance) Population Covariance Cmd -- Returns the population covariance of the independent and dependent data columns in the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: COLΣ,CORR,COV,PREDX,PREDY, XCOL,YCOL
3C4F5	xPDIM	((xmin,ymin) (xmax,ymax) →) (#width #height →) PICT Dimension Cmd -- Replaces PICT with a blank PICT of the specified dimensions. -- Related: PMAX,PMIN

3B477	xPERM	(n k → n')
		Permutations Func
		--
		Returns the number of possible permutations of n items taken m at a time.
		--
		`n      m      → Pn,m`
		`'symn' m      → 'PERM(symn,m)'`
		`n      'symm' → 'PERM(n,symm)'`
		`'symn' 'symm' → 'PERM(symn,symm)'`
		--
		Related: COMB,!

0460AB	~xPEVAL	([coefs] x → x') Polynomial Evaluation Cmd -- Evaluates an n-degree polynomial at x. -- Related: PCOEFF,PROOT
3EAA7	xPGDIR	(name →) Purge Directory Cmd -- Purges the named directory (whether empty or not). -- Related: CLVAR,CRDIR,HOME,PATH,PURGE, UPDIR
3DCFD	xPICK	(1...n n → 1..n 1) Pick Object Cmd -- Copies the contents of a specified level to level 1. -- Related: DUP,DUPN,DUP2,OVER,ROLL,ROLLD, ROT,SWAP
3F27F	xPICK3	(1 2 3 → 1 2 3 1) Duplicate the object on level 3 of the stack. -- Related: PICK,OVER,DUP
3C72A	xPICT	(→ PICT) PICT Cmd -- Puts the name PICT on the stack. -- Related: GOR,GCOR,NEG,PICTURE,PVIEW, RCL,REPL,SIZE,STO,SUB

3C5AE	xGRAPH	(→) Picture Environment Cmd -- Selects the Picture environment (selects the graphics display and activates the graphics cursor and Picture menu). -- Related: PVIEW,TEXT,PIC UserRPL: xPICTURE
06A0AB	~xPINIT	(→) Port Initialize Cmd -- Initializes all currently active ports. Does not affect data already stored in a port.
3C662	xPIX?	((x,y) → 1/0) ({#n #m} → 1/0) Pixel On? Cmd -- Tests whether the specified pixel in PICT is on; returns 1 (true) if the pixel is on, and 0 (false) if the pixel is off. -- Related: PIXON,PIXOFF
3C638	xPIXOFF	((x,y) →) ({#n #m} →) Pixel Off Cmd -- Turns off the pixel at the specified coordinate in PICT. -- Related: PIX?,PIXON
3C60E	xPIXON	((x,y) →) ({#n #m} →) Pixel On Cmd -- Turns on the pixel at the specified coordinate in PICT. -- Related: PIX?,PIXOFF
3EE9D	xPKT	($data $type → $response) Packet Cmd -- Used to send command "packets" (and receive requested data) to a Kermit server. -- Related: CLOSEIO,KERRM,SERVER
009314	~xPLOT	(f → f) Plots a function.

00A314	~xPLOTADD	(f →) Adds function to existing plot function list, and opens the Plot Setup screen.
3C392	xPMAX	((x,y) →) PICT Maximum Cmd -- Specifies (x,y) as the coordinates at the upper right corner of the display. -- Related: PDIM,PMIN,XRNG,YRNG
3C372	xPMIN	((x,y) →) PICT Minimum Cmd -- Specifies (x,y) as the coordinates at the lower left corner of the display. -- Related: PDIM,PMAX,XRNG,YRNG
0140DE	~xPMINI	
3C979	xPOLAR	(→) Polar Plot Type Cmd -- Sets the plot type to POLAR. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE, PCONTOUR,SCATTER,SLOPEFIELD,TRUTH, WIREFRAME,YSLICE
02D0DE	~xPOLYNOMIAL	(→) Display polynomial menu.
0350DE	~xPOP	(→) -- Related: PUSH
3BB94	xPOS	(str substring → n/0) ({} ob → n/0) Position Cmd -- Returns the position of a substring within a string or the position of an object within a list. -- Related: CHR,NUM,REPL,SIZE,SUB
0380DE	~xPOTENTIAL	
01B0DE	~xPOWEXPAND	
073314	~xPOWMOD	(symb exp → symb')

3D0D7	xPR1	(ob → ob) Print Level 1 Cmd -- Prints an object in multiline printer format. -- Related: CR,DELAY,OLDPRT,PRTLCD,PRST, PRSTC,PRVAR
3DFDD	xPREDV	(x → y) Predicted y-Value Cmd -- Returns the predicted dependent variable value ydepend, based on the independent-variable value xindep, the currently selected stat model, and the current regression coefficients in the reserved variable ΣPAR. -- <REF>TEXT:Reserved|ΣPAR -- Related: PREDX
3E01D	xPREDX	(y → x) Predicted x-Value Cmd -- Returns the predicted dependent variable value xindepend, based on the independent-variable value ydepend, the currently selected stat model, and the current regression coefficients in the reserved variable ΣPAR. -- <REF>TEXT:Reserved|ΣPAR -- Related: COLΣ,CORR,COV,EXPFIT, ΣLINE,LINFIT,LOGFIT,LR, PREDY,PWRFIT,XCOL,YCOL
3DFFD	xPREDY	(x → y) Predicted y-Value Cmd -- Returns the predicted dependent variable value ydepend, based on the independent-variable value xindepend, the currently selected stat model, and the current regression coefficients in the reserved variable ΣPAR. -- <REF>TEXT:Reserved|ΣPAR -- Related: COLΣ,CORR,COV,EXPFIT, ΣLINE,LINFIT,LOGFIT,LR, PREDX,PWRFIT,XCOL,YCOL

00C314	~xPREVAL	(f x1 x2 → symb) (f x1 x2 → x)
03E314	~xPREVPRIME	(n → n')
3D1E7	xPRLCD	(→) Print LCD Cmd -- Prints a pixel-by-pixel image of the current display (excluding the annunciators) -- Related: CR,DELAY,OLDPRT,PRST,PRSTC, PRVAR,PR1
38BBF	xPROMPT	($ →) Prompt Cmd -- Displays the contents of "prompt" in the status area, and halts program execution. -- Related: CONT,DISP,FREEZE,HALT,INFORM, IN-PUT,MSGBOX
08B314	~xPROMPTSTO	(var →) Creates a variable and promts for a value to store there. -- Related: PROMPT,STO
0440AB	~xPROOT	([coefs] → [roots]) Polynomial Roots Cmd -- Returns all roots of an n-degree polynomial having real or complex roots. -- Related: PCOEFF,PEVAL
035314	~xPROPFRAC	(x → symb')
3D10D	xPRST	(→) Print Stack Cmd -- Prints all objects in the stack, starting with the object in the highest level. -- Related: CR,DELAY,OLDPRT,PRLCD,PRSTC, PRVAR,PR1

`3D0F2`	`xPRSTC`	Print Stack (Compact) Cmd -- Prints in compact form all objects in the stack, starting with the object in the highest level. -- Related: PR,DELAY,OLDPRT,PRLCD,PRST, PRVAR,PR1

`3D143`	`xPRVAR`	`( name` $\rightarrow$ `)`
		`( {names}` $\rightarrow$ `)`
		`( :port:name` $\rightarrow$ `)`
		Print Variable Cmd
		--
		Searches the current directory path or port for the specified variables and prints the name and contents of each variable.
		--
		Related: CP,DELAY,OLDPRT,PR1,PRLCD, PRST,PRSTC

`01D0AB`	`~xPSDEV`	`(` $\rightarrow$ `xpsdev )`
		`(` $\rightarrow$ `{x1...xn} )`
		Population Standard Deviation Cmd
		--
		Calculates the population standard deviation of each of the m columns of coordiate values in the current stastics matrix (reserved variable ΣDAT).
		--
		<REF>TEXT:Reserved|ΣDAT
		--
		Related: `MEAN`,PCOV,PVAR,`SDEV`,`TOT`,`VAR`

`0040DE`	`~xPSI`	`( symb` $\rightarrow$ `symb' )`
`0030DE`	`~xPsi`	`( symb n` $\rightarrow$ `symb' )`
`036314`	`~xPTAYL`	`( pol x` $\rightarrow$ `pol' )`

`3E87C`	`xPURGE`	`( name` $\rightarrow$ `)`
		`( {names}` $\rightarrow$ `)`
		`( PICT` $\rightarrow$ `)`
		Purge Cmd
		--
		Purges the named variables or empty subdirectories from the current directory.
		--
		Related: CLEAR,CLVAR,NEWOB,PGDIR

`0340DE`	`~xPUSH`	`(` $\rightarrow$ `)`

3C0BF	xPUT	`( ob pos obj → ob' )`
		`ob = [] or [[]] or {} or name`
		`pos = n or {n} or {n m}`
		Put Element Cmd
		--
		In the level 3 array or list, PUT replaces with zput or objput the object whose position is specified in level 2; if the array or list is unnamed, returns the new array or list.
		--
		Related: GET,GETI,PUTI

3C139	xPUTI	`( ob pos obj → [] pos' )`
		`ob = [] or [[]] or {} or name`
		`pos = n or {n} or {n m}`
		Put and Increment Index Cmd
		--
		In the level 3 array or list, replaces with zput or objput the object whose position is specified in level 2, returning the new array or list and the next position in that array or list.
		--
		Related: GET,GETI,PUT

01E0AB	~xPVAR	`( → xpvariance )`
		`( → [x1...xn] )`
		Poplulation Variance Cmd
		--
		Calculates the population variance of the coordinate values in each of the m columns in the current stat matrix (ΣDAT).
		--
		Related: MEAN,PCOV,PSDEV,SDEV,VAR

3EA49	xPVARS	`( nport → {} mem )` Port-Variables Cmd -- Returns a list of the `backup` objects (:nport:name) and the lib- rary objects (:nport:nlibrary) in the specified port. Also returns the available memory size (if RAM) or the memory type. -- Related: VARS

3C5E4	xPVIEW	`( (x,y) → )`
		`( {#n #m} → )`
		PICT View Cmd
		--
		Displays PICT with the specified coordinate at the upper left corner of the graphics display.
		--
		Related: FREEZE,PICTURE,PICT,TEXT

3E283	xPWRFIT	Power Curve Fit Cmd -- Stores PWRFIT as the fifth parameter in the reserved variable ΣPAR, indicating that subsequent executions of LR are to use the power curve fitting model. -- <REF>TEXT:Reserved|ΣPAR -- Related: BESTFIT,EXPFIT,LINFIT, LOGFIT,LR
3C56E	xPX>C	`( {#m #n} → (x,y) )` Pixel to Complex Cmd -- Converts the specified pixel coordinates to user-unit coordinates. -- Related: C→PX UserRPL: `xPX→C`

3DA3E	x->Q	`( x → a/b )`
		To Quotient Cmd
		--
		Returns a rational form of the argument.
		--
		`x       → 'a/b'`
		`(x,y)   → 'a/b+c/d*i`
		`'sym1'  → 'sym2'`
		--
		Related: →Qπ,/ UserRPL: `x→Q`

3DA63	x->QPI	(x → symb) To Quotient Times π Cmd -- Returns a rational form of the argument, or a rational form of the argument with π factored out, whichever yields the smaller denominator. -- x → 'a/b*π' x → 'a/b' 'sym1' → 'symb2' (x,y) → 'a/b*π+c/d*π*i (x,y) → 'a/b+c/d*i -- Related: →Q,/,π UserRPL: x→Qπ
0080DE	~xqr	
0310AB	~xQR	([[]] → [[Q]] [[R]] [[P]]) QR Factorization of a Matrix Cmd -- Returns the QR factorization of an nm matrix. -- Related: LQ,LSQ
3E66F	xQUAD	(symb var → symb') Solve Quadratic Equation Cmd -- Solves an algebraic object symb for the variable var, and returns an expression symb' representing the solution. -- Related: COLCT,EXPAN,ISOL,SHOW,SOLVE
3D6F6	xQUOTE	(ob → 'ob) Quote Argument Func -- Returns its argument unevaluated. -- 'sym' → 'sym' obj → obj -- Related: APPLY,|
028314	~xQUOT	(p1 p2 → p3)
04B314	~xQXA	(symb [vars] → [[]] [vars])
3B564	xRAD	(→) Radians Mode Cmd -- Sets Radians angle mode. -- Related: DEG,RAD

3B3E6	xRAND	(→ x) Random Number Cmd -- Returns a pseudo-random number generated using a seed value, and updates the seed value. -- Related: COMB,PERM,RDZ,!
02A0AB	~xRANK	([[]] → n) Matrix Rank Cmd -- Returns the rank of a rectangular matrix. -- Related: LQ,LSQ,QR

0350AB	~xRANM	({m n} → [[]])
		([[]] → [[]]')
		Random Matrix Cmd
		--
		Returns a matrix of specified dimensions that contains random integers in the range -9 through 9.
		--
		`{ m n }       → [[ rand mat ]]mn`
		`[[ mat ]]mn → [[ rand mat ]]mn`
		--
		Related: RAND,RDZ

3DBCA	xPREDIV	(x y → x/y)
		Prefix Divide Func
		--
		Prefix form of / (divide) generated by the Equation Writer Application.
		--
		`z1     z2      → z1/z2`
		`[arr]  [[mat]]  → [[arrmat^-1]]`
		`[arr]  z       → [arr/z]`
		`z      'sym'   → 'z/sym'`
		`'sym'  z       → 'sym/z'`
		`'sym1' 'sym2'  → 'sym1/sym2'`
		`#n1    n2      → #n3`
		`n1     #n2     → #n3`
		`#n1    #n2     → #n3`
		`x_u1   y_u2    → (x/y)_u1/u2`
		`x      y_u     → (x/y)_1/u`
		`x_u    y       → (x/y)_u`
		`'sym'  x_u     → 'sym/x_u'`
		`x_u    'sym'   → 'x_u/sym'`
		--
		Related: / UserRPL: xRATIO

3D393	xRCEQ	(→ EQ) Recall from EQ Cmd -- Returns the unevaluated contents of the reserved variable EQ from the current directory. -- <REF>TEXT:Reserved|EQ -- Related: STEQ
0420AB	~xRCI	([[]] x nrow → [[]]') [] x n []' Multiply Row by Constant Cmd -- Multiplies row n of a matrix (or element n of a vector) by a const x, and returns the modified matrix. -- Related: RCIJ
0430AB	~xRCIJ	([[]] x n* n+ → [[]]') ([] x n* n+ → []') Add Multiplied Row Cmd -- Multiplies row n* of a matrix by a constant x, adds this product to row n+ of the matrix, and returns the modified matrix. Or, multiplies element n* of a vector by a constant x, adds this product to element n+ of the vector, and returns the modified vector. -- Related: RCI
3E6F1	xRCL	(var → x) (:port:nlib → lib) (:port:name → ob) (:port:{path} → ob) Recall Cmd -- Returns the unevaluated contents of a specified variable or plug -in object. -- Related: STO
3918E	xRCLALARM	(n → {date time action rep}) Recall Alarm Cmd -- Recalls a specified alarm. -- Related: DELALARM,FINDALARM,STOALARM

3B715	xRCLF	(→ {#s1 #u1 #s2 #u2}) Recall Flags Cmd -- Returns a list containing four 64 bit binary integers representing the states of the 64 system and user flags, respectively. -- Related: STOF
3EF79	xRCLKEYS	(→ {ob ... key ...}) (→ {S ob ... key ...}) Recall Key Assignments Cmd -- Returns the current user key assignments. This includes an S if the standard key definitions are active (not suppressed) for those keys without user key assignments. -- Related: ASN,DELKEYS,STOKEYS
3EA2E	xRCLMENU	(→ x) Recall Menu Number Cmd -- Returns the menu number of the currently displayed menu. -- Related: MENU,TMENU
3DDA9	xRCLSIGMA	(→ [[]]) Recall Sigma Cmd -- Returns the current stat matrix (the contents of reserved var ΣDAT) from the current directory. -- <REF>TEXT:Reserved|ΣDAT -- Related: CLΣ,STOΣ,Σ+,Σ- UserRPL: xRCLΣ
03F0DE	~xRCLVX	(→ name) Recall the current content of the reserved -- CAS variable VX. -- <REF>TEXT:Reserved|VX First available in ROM 1.19-6.
3B6FA	xRCWS	(→ n) Recall Wordsize Cmd -- Returns the current wordsize in bits (1 through 64). -- Related: BIN,DEC,HEX,OCT,STWS

3BEEC	xRDM	`( ob size → ob' )`
		`( name size → )`
		`ob= [] or [[]]`
		`size = {n} or {n m}`
		Redimension Array Cmd
		--
		Rearranges the elements of the argument according to the specified dimensions.
		--
		Related: TRN

3B401	xRDZ	`( x → )` Randomize Cmd -- Uses a real number xseed as a seed for the RAND command. -- Related: COMB,PERM,RAND,!

3B819	xRE	`( (x,y) → x )`
		`( [] → []' )`
		Real Part Func
		--
		Returns the real part of the argument.
		--
		`x          → x`
		`x_u        → x`
		`(x,y)      → x`
		`[ R-arr ]  → [ R-arr ]`
		`[ C-arr ]  → [ R-arr ]`
		`'sym'      → 'RE(sym)'`
		--
		Related: C→R,IM,R→C

3ED22	xRECN	`( name → )`
		`( $name → )`
		Receive Renamed Object Cmd
		--
		Prepares the HP 48 to receive a file from another Kermit device, and to store the file in a specified variable.
		--
		Related: BAUD,CKSM,CLOSEIO,FINISH, KERRM,KGET,PARITY,RECV,SEND, SERVER,TRANSIO

0110AB	~xRECT	(→) Rectangular Mode Cmd -- Sets Rectangular coordinate mode. -- Related: CYLIN,SPHERE
3ED56	xRECV	(→) Receive Object Cmd -- Instructs the HP 48 to look for a named file from another Kermit device. The received file is stored in a variable named by the sender. -- Related: BAUD,CKSM,FINISH,KGET,PARITY, RECN,SEND,SERVER,TRANSIO
048314	~xREF	([[]] → [[]]')
02A314	~xREMAINDER	(p1 p2 → p3)
0130DD	~xRENAME	(name name' →) -- Related: COPY
069314	~xREORDER	(pol var → pol')
38105	xREPEAT	(1/0 →) REPEAT Cmd -- Starts loop clause in `WHILE ... REPEAT ... END` indefinite loop structure. -- Related: END,WHILE
3B9D2	xREPL	(ob pos new → ob') ob= [[]] or [] or {} or $ or PICT pos= N or {n m} or (n,m) Replace Cmd -- Replaces a portion of the level 3 target object with the level 1 object, beginning at a position specified in level 2. -- Related: CHR,GOR,GXOR,NUM,POS,SIZE,SUB

3C41A	xRES	(n_int →) (#n_int →) Resolution Cmd -- Specifies the resolution of mathematical and statistical plots, where the resolution is the interval between values of the independent variable used to generate the plots. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAP,HISTOGRAM,PARAMETRIC,PARSURFACE, PCONTOUR,POLAR,SCATTER,SLOPEFIELD, TRUTH,WIREFRAME,YSLICE
3EAE7	xRESTORE	(:port:name →) (ob →) Restore HOME Cmd -- Replaces the current HOME directory with the specified backup copy. -- :nport:namebackup → obj backup → -- Related: ARCHIVE
0050DE	~xRESULTANT	(p1 p2 → res)
05D0AB	~xREVLIST	({1...n} → {n...1}') Reverse List Cmd -- Reverses the order of the elements in a list. -- Related: SORT
0280DE	~xREWRITE	
00D314	~xRISCH	(f var → F)
0200AB	~xRKF	({} xtol xTf → {} xtol) ({} {xtol step} xTf → {} xtol) Runge-Kutta-Fehlberg) Cmd -- Computes the solution to an initial value problem for a differential equation, using the Runge-Kutta-Fehlberg (4,5) method. -- Related: RKFERR,RKFSTEP,RRK,RRKSTEP,RBSERR

0220AB	~xRKFERR	({} h → {} h dy err) Error Estimates for <REF>RKF -- Returns the absolute error estimate for a given step h when solving an initial value problem for a differential equation. -- Related: RKF,RKFSTEP,RRK,RRKSTEP,RSBERR
0210AB	~xRKFSTEP	({} tol h → {} tol h') Next Solution Step for RKF Cmd -- Computes the next solution step (hnext) to an initial value problem for a differential equation. -- Related: RKF,RKFERR,RRK,RRKSTEP,RSBERR
38E01	xRL	(# → #') Rotate Left Cmd -- Rotates a binary integer one bit to the left. -- Related: RLB,RR,RRB
38E21	xRLB	(# → #') Rotate Left Byte Cmd -- Rotates a binary integer one byte to the left. -- Related: RL,RR,RRB
3AEB1	xRND	(x n → x') Round Func -- Rounds an object to a specified number of decimal places or significant digits, or to fit the current display format. --

```
z1      nrnd     → z2
z       'symrnd' → 'RND(z,symrnd)'
'sym'   nrnd     → 'RND(symb,nrnd)'
'sym1' 'symrnd' → 'RND(sym1,symrnd)'
[arr1] nrnd     → [arr2]
x_u     nrnd     → y_u
x_u     'symrnd' → 'RND(x_u,symrnd)'
```

--
Related: TRNC

3B16C	xRNRM	`( [] → x )` Row Norm Cmd -- Returns the row norm (infinity norm) of its argument array. -- Related: CNRM,CROSS,DET,DOT
3DD18	xROLL	`( 1...n n → 2...n 1 )` Roll Objects Cmd -- Moves the contents of a specified level to level 1, and rolls upwards the portion of the stack beneath the specified level. -- Related: OVER,PICK,ROLLD,ROT,SWAP
3DD33	xROLLD	`( n ... 1 n → 1 n...2 )` Roll Down Cmd -- Moves the contents of level 1 to a specified level, and rolls downwards the portion of the stack beneath the specified level -- Related: OVER,PICK,ROLL,ROT,SWAP
06F0AB	~xROMUPLOAD	`( → )` Upload the rom to another calculator -- 1. On the sending calculator, enter ROMUPLOAD and press ENTER. On the receving calc, hold down ON and press F4. On the receiving calc, hold down ON and +, and press ENTER. On the receiving calc, press 4 to select Download option. On the sending calc, press any key to start the process (takes about 20 min).

3D3CE	xROOT	`( prog/s var guess → x )`
		`( prog/s var {guesses} → x )`
		Root-Finder Cmd
		--
		Returns a real number xroot that is a value of the specified variable var for which the specified program or algebraic object most nearly evaluates to zero or a local extremum.

3DC71 xROT

(1 2 3 → 2 3 1)
Rotate Objects Cmd
--
Rotates the first three objects on the stack, moving the object in level 3 to level 1.
--
Related: OVER,PICK,ROLL,ROLLD,SWAP,UNROT

03C0AB ~xROW-

([[]] nrow → [[]]' [])
([] n → []' elt)
Delete Row Cmd
--
Deletes row n of a matrix (or element n of a vector), and returns the modified matrix (or vector) and the deleted row (or element).
--
Related: COL-,COL+,ROW-,RSWP

03D0AB ~xROW+

([[]] [[]]' n → [[]]'')
([[]] [] n → [[]]')
([] n n' → [])
Insert Row Cmd
--
Inserts an array into a matrix (or one or more numbers into a vector) at a position indicated by nindex, and returns the modified matrix (or vector).
--
Related: COL-,COL+,ROW-,RSWP

0370AB ~xROW→

([1]...[n] n → [])
(x1...xn → [])
Rows to Matrix Cmd
--
Transforms a series of row vectors and a row count into a matrix rix containing those rows, or transforms a sequence of numbers and an element count into a vector with those numbers as elements.
--
Related: →COL,COL→,→ROW

0360AB ~x→ROW

([[]] → [1]...[n] n)
([] → x1...xn n)
Matrix to Rows Cmd
--
Transforms a matrix into a series of row vectors and returns the vectors and a row count, or transforms a vector into its elements and returns the elements and an element count.
--
Related: →COL,COL→,ROW→

3F218	xRPL>	
0680AB	~xrpm	
38E41	xRR	`( # → x' )` Rotate Right Cmd -- Rotates a binary integer one bit to the right. -- Related: RL,RLB,RRB
38E61	xRRB	`( # → x' )` Rotate Right Byte Cmd -- Rotates a binary integer one byte to the right. -- Related: RL,RLB,RR
0340AB	~xRREF	`( [[]] → [[]]' )` Reduced Row Echelon Form Cmd -- Converts a rectangular matrix to a reduced row echelon form.
047314	~xrref	`( [[]] → [pp] [[]]' )`
078314	~xRREFMOD	`( [[]] → [[]]' )`
0230AB	~xRRK	`( {} xtol xTfinal → {} xtol )` Solve for Initial Values (Rosenbrock, Runge-Kutta) Cmd -- Computes the solution to an initial value problem for a differential equation with known partial derivatives. -- Related: RKF,RKFERR,RKFSTEP,RRKSTEP,RSBERR
0240AB	~xRRKSTEP	`( {} xtol h last → {} xtol h' cur )` Next Solution Step and Method (RKF or RRK) Cmd -- Computes the next solution step (hnext) to an initial value problem for a differential equation, and displays the method used to arrive at that result. -- Related: RKF,RKFERR,RKFSTEP,RRK,RSBERR

0250AB	~xRSBERR	`( {} h → {} h dy err )`
		Error Estimate for Rosenbrock Method Cmd
		`--`
		Returns an error estimate for a given step h when solving an initial value problem for a differential equation.
		`--`
		Related: RKF,RKFERR,RKFSTEP,RRK,RRKSTEP

3B22F	xRSD	`( [B] [[A]] [Z] → []' )`
		`( [[B]] [[A]] [[Z]] → [[]]' )`
		Residual Cmd
		`--`
		Computes the residual B - AZ of the arrays B, A, and Z.

0400AB	~xRSWP	`( []/[[]] i j → []/[[]] )`
		Row Swap Cmd
		`--`
		Swaps rows i and j of a matrix and returns the modified matrix, or swaps elements i and j of a vector and returns the modified vector.
		`--`
		Related: CSWP,ROW+,ROW-

3E632	xRULES	

38F01	xR>B	`( x → # )`
		Real to Binary Cmd
		`--`
		Converts a positive real integer to its binary integer equivalent.
		`--`
		Related: B→R UserRPL: `xR→B`

3B7ED	xR>C	`( x y → (x,y) )`
		`( [X] [Y] → [(x,y)] )`
		Real to Complex Cmd
		`--`
		Combines two real numbers or real arrays into a single complex number or array.
		`--`
		Related: C→R,IM,RE UserRPL: `xR→C`

3B0AE	xR>D	(x → (180/π)x) Radians to Degrees Func -- Converts a real number expressed in radians to its equivalent in degrees. -- x → (180/π)x 'sym' → 'R→D(sym)' -- Related: D→R UserRPL: xR→D
3F070	xR>I	(x → n) UserRPL: xR→I
3C9E5	xSAME	(ob1 ob2 → 1/0) Display information about the makers of the calculator. Same Object Cmd -- Compares two objects, and returns a true result (1) if they are identical, and a false result (0) if they are not. -- Related: TYPE,==
3EE82	xSBRK	(→) Serial Break Cmd -- Interrupts serial transmission or reception. -- Related: BUFLEN,SRECV,STIME,XMIT
3C4D5	xSCALE	(xs ys →) Scale Plot Cmd -- Adjusts the first two parameters in PPAR, (xmin, ymin) and (xmax, ymax), so that xscale and yscale are the new plot horizontal and vertical scales, and the center point doesn't change. -- <REF>TEXT:Reserved|PPAR -- Related: AUTO,CENTR,SCALEH,SCALEW
3C444	x*H	(xf →) Multiply Height Cmd -- Multiplies the vertical plot scale by xfactor. -- Related: AUTO,SCALEW,YRING UserRPL: xSCALEH

3C464	`x*W`	(yf →) Multiply Width Cmd -- Multiplies a plot's horizontal scale by xfactor. -- Related: `AUTO`,SCALEH,YRING UserRPL: `xSCALEW`
3E1EF	`xSCATRPLOT`	(→) Draw Scatter Plot Cmd -- Draws a scatter plot of (x, y) data points from the specified columns of the current statistics matrix (reserved variable ΣDAT). -- Related: BARPLOT,PICTURE,HISTPLOT, `PVIEW`,SCLΣ,`XCOL`,`YCOL`
3C9AF	`xSCATTER`	Scatter Plot Type Cmd -- Sets the plot type to `SCATTER`. -- Related: `BAR`,`CONIC`,DIFFEQ,`FUNCTION`, GRIDMAP,`HISTOGRAM`,`PARAMETRIC`,PARSURFACE, PCONTOUR,`POLAR`,SLOPEFIELD,`TRUTH`, WIRE-FRAME,YSLICE
0330AB	`~xSCHUR`	(`[[]]` → `[[Q]]` `[[T]]`) Schur Decomp. of Squ. Matrix Cmd -- Returns the Schur decomposition of a square matrix. -- Related: LQ,LU,QR,SVD,SVL,TRN
3B5BA	`xSCI`	(n →) Scientific Mode Cmd -- Sets the number display format to Scientific mode, which displays one digit to the left of the fraction mark and n significant digits to the right. -- Related: `ENG`,`FIX`,`STD`

3E127	xSCLSIGMA	(→) Scale Sigma Cmd -- Adjusts (xmin,ymin) and (xmax, ymax) in PPAR so that a subsequent scatter plot exactly fills PICT. -- <REF>TEXT:Reserved|PPAR -- Related: `AUTO`,SCATRPLOT UserRPL: `xSCL`Σ
3E385	xSCONJ	(`name` →) Store Conjugate Cmd -- Conjugates the contents of a named object. -- Related: CONJ,SINV,SNEG
07D314	~xSCROLL	(`ob` →)
3DF32	xSDEV	(→ `xsdev`) (→ `[x1...xn]`) Standard Deviation Cmd -- Calculates the sample standard deviation of each of the m columns of coordinate values in the current stat matrix (reserved var ΣDAT). -- Related: `MAX`Σ,`MEAN`,`MIN`Σ,PSDEV, PVAR,`TOT`,`VAR`
3ECB0	xSEND	(`name` →) (`{names}` →) (`{{old new}...}` →) Send Object Cmd -- Sends a copy of the named object to a Kermit device. -- Related: BAUD,CLOSEIO,CKSM,FINISH, KERRM,KGET,PARITY,RECN, RECV,SERVER,TRANSIO
0530AB	~xSEQ	(`prog var start end incr` → `{}`) Sequential Calculation Cmd -- Returns a list of results generated ated by repeatedly executing prog using index var over the range start to end, in increments of incr. -- Related: DOSUBS,STREAM
007314	~xSERIES	(`func var order` → `{} symb'`)

3ED91 xSERVER (→)
Server Mode Cmd
--
Selects Kermit Server mode.
--
Related: BAUD,CKSM,FINISH,KERRM, KGET,PARITY,PKT,RECN,RECV, SEND,TRANSIO

064314 ~xSEVAL (symb → symb')

3B4C9 xSF (n →)
Set Flag Cmd
--
Sets a specified user or system flag.
--
Related: CF,FC?,FC?C,FS?,FS?C

3E696 xSHOW (symb name → symb')
(symb {names} → symb')
Show Variable Cmd
--
Returns symb' which is equivalent to symb except that all implicit references to a variable name are made explicit.
--
Related: COLCT,EXPAN,ISOL,QUAD

0630AB ~xSIDENS (x → x')
Silicon Intrinsic Density Cmd
--
Calculates the intrinsic density of silicon as a function of temperature, xT.
--

```
xT     → xdensity
x_u    → x_1/cm3
'sym'  → 'SIDENS(symb)'
```

0020DE ~xSIGMA (f var → F)

0010DE ~xSIGMAVX (f(x) → F(x))

3A3EE xSIGN (x → x')
Sign Func
--
Returns the sign of a real number argument, the sign of the numerical part of a unit object argument, or the unit vector in the direction of a complex number argument.
--
Related: ABS,MANT,XPON

05F314 ~xSIGNTAB (symb → {})

033314 ~xSIMP2 (x y → x/gcd y/gcd)

0220DE ~xSIMPLIFY (symb → symb')

018314 ~xSINCOS (symb → symb')

3A57C xSIN (x → x')
Sine Analytic Func
--
```
z            → sin z
'sym'        → 'SIN(sym)'
x_uangular   → sin(x_uangular)
```
--
Related: ASIN,COS,TAN

3A678 xSINH (x → x')
Hyperbolic Sine Analytic Func
--
Returns the hyperbolic sine of the argument.
--
```
z      → sinh z
'sym'  → 'SINH(sym)'
```
--
Related: ANUSH,COSH,TANH

3E331 xSINV (name →)
Store Inverse Cmd
--
Replaces the contents of the named variable with its inverse.
--
Related: INV,SCONJ,SNEG

3BB1F	xSIZE	(ob → n) (ob → {N m}) (ob → #nw #nh) Size Cmd -- Returns the number of characters in a string, the number of elements in a list, the dimensions of an array, the number of objects in a unit object or algebraic object, or the dimensions of a graphics object. -- "str" → n { list } → n [vector] → { n } [[mat]] → { n m } 'sym' → n grob → #nwidth #mheight PICT → #nwidth #mheight x_u → n -- Related: CHR,NUM,POS,REPL,SUB
38E81	xSL	(# → #') Shift Left Cmd -- Shifts a binary integer one bit to the left. -- Related: ASR,SLB,SR,SRB
38EA1	xSLB	(# → #') Shift Left Byte Cmd -- Shifts a binary integer one byte to the left. -- Related: ASR,SL,SR,SRB
00C0AB	~xSLOPEFIELD	(→) SLOPEFIELD Plot Type Cmd -- Sets the plot type to SLOPEFIELD. -- Related: BAR,CONIC,DIFFEQ,FUNCTION, GRIDMAX,HISTOGRAM,PARAMETRIC, PARSURFACE,PCONTOUR,POLAR,SCATTER, TRUTH,WIREFRAME,YSLICE

```
3E35B     xSNEG         ( name → )
```
Store Negate Cmd
--
Replaces the contents of a variable with its negative.
--
Related: NEG,SCONJ,SINV

```
0290AB   ~xSNRM         ( [] → x )
```
Spectral Norm Cmd
--
Returns the spectral norm of an array.
--
Related:
ABS,CNRM,COND,RNRM,SRAD,TRACE

```
03F314   ~xSOLVE        ( symb var → {zeros} )
086314   ~xSOLVER       ( → )
```
Displays a menu of commands used in solving equations.

```
008314   ~xSOLVEVX      ( symb → {zeros} )
05E0AB   ~xSORT         ( {} → {}' )
```
Ascending Order Sort Cmd
--
Sorts the elements in a list in ascending order.
--
Related: REVLIST

```
0130AB   ~xSPHERE       ( → )
```
Spherical Mode Cmd
--
Sets Spherical coordinate mode.
--
Related: CYLIN,RECT

```
3A4EF     xSQ           ( x → x' )
```
Square Analytic Func
--
Returns the square of the argument.
--
```
z          → z2
x_u        → x2_u2
[[ mat ]]  → [[ mat  mat ]]
'sym'      → 'SQ(sym)'
```
--
Related: $\sqrt{a}$,^

`38EC1`	`xSR`	`( # → #' )` Shift Right Cmd -- Shifts a binary integer one bit to the right. -- Related: ASR,SL,SLB,SRB
`0280AB`	`~xSRAD`	`( [[]] → x )` Spectral Radius Cmd -- Returns the spectral radius of a square matrix. -- Related: COND,SNRM,TRACE
`38EE1`	`xSRB`	`( # → #' )` Shift Right Byte Cmd -- Shifts a binary integer one byte to the right. -- Related: ASR,SL,SLB,SR
`3EC55`	`xSRECV`	`( n → $ 0/1 )` Serial Receive Cmd -- Reads up to n characters from the serial input buffer and returns them as a string, along with a digit indicating whether errors occurred. -- Related: BUFFLEN,CLOSEIO,OPENIO, SBRK,STIME,XMIT
`0100DD`	`~xSREPL`	`( str find repl → str' n )` Globally replace find with repl in `str`. n is the number of matches. Efficient ML implementation.

`381AB`	`xSTART`	`( start finish → )`
		START Definite `Loop` Structure Cmd
		--
		`START xstart xfinish →`
		`NEXT xstart xfinish →`
		`STEP xincrement →`
		`STEP 'symbincrement' →`
		--
		Related: FOR,NEXT,STEP

`3B5FA`	`xSTD`	`( → )` Standard Mode Cmd -- Sets the number display format to Standard mode. -- Related: `ENG`,`FIX`,`SCI`

3851F	xSTEP	`( n → )`
		`( symb → )`
		STEP Cmd
		--
		Defines the increment (step) value, and ends definite loop struct See the FOR and START command entries for syntax information.
		--
		Related: FOR,BEXT,START

3D3AE	xSTEQ	`( ob → )` Store in `EQ` Cmd -- Stores an object into the reserved variable `EQ` in the current directory. -- <REF>TEXT:Reserved|EQ -- Related: RCEQ
3EE62	xSTIME	`( x/0 → )` Serial Time-Out Cmd -- Specifies the period that SRECV (serial reception) and XMIT (serial transmission) `wait` before timing out. -- Related: BUFLEN,CLOSEIO,SBRK,SRECV,XMIT

3E739	xSTO	`( ob name → )`
		`( ob :port:name → )`
		`( lib port → )`
		`( bup port → )`
		`( ob 'name(i)' → )`
		Store Cmd
		--
		Stores an object into a specified variable or object.
		--
		Related: DEFINE,RCL,→

3E406	xSTO-	`( ob name → )`
		`( name ob → )`
		Store Minus Cmd
		--
		Calculates the difference between a number (or other object) and the contents of a specified variable, and stores the new value to the specified variable.
		--
		Related: STO+,STO*,STO/,-

3E4D2	xSTO*	`( ob name → )`
		`( name ob → )`
		Store Times Cmd
		--
		Multiplies the contents of a specified variable by a number or other object.
		--
		Related: STO+,STO-,STO/,*

3E46C	xSTO/	`( ob name → )`
		`( name ob → )`
		Store Divide Cmd
		--
		Calculates the quotient of a number (or other object) and the contents of a specified variable, and stores the new value to the specified variable.
		--
		Related: STO+,STO-,STO*,/

3E3AF	xSTO+	`( ob name → )`
		`( name ob → )`
		Store Plus Cmd
		--
		Adds a number or other object to the contents of a specified variable.
		--
		Related: STO-,STO*,STO/,+

39164	xSTOALARM	`( time → n )`
		`( {date time act rep} → n )`
		Store Alarm Cmd
		--
		Stores an alarm in the system alarm list and returns its alarm index number. act and rep arguments are optional.
		--
		Related: DELALARM,FINDALARM,RCLALARM

3B749	xSTOF	`( {#s1 #u1 #s2 #u2} → )` Store Flags Cmd -- Sets the states of the system flags or the system and user flags. -- Related: RCLF,STWS,RCWS

3EF07	xSTOKEYS	({ob key ...} →) ({'S' ob key ...} →) ('S' →) Store Key Assignments Cmd -- Defines multiple keys on the user keyboard by assigning objects to specified keys. -- Related: ASN,DELKEYS,RCLKEYS
3DD6E	xSTOSIGMA	(ob →) Store Sigma Cmd -- Stores obj in the reserved variable ΣDAT. -- Related: CLΣ,RCLΣ,Σ+,Σ- UserRPL: xSTOΣ
0400DE	~xSTOVX	(name →) Store object into the reserved CAS variable VX. -- <REF>TEXT:Reserved|VX First available in ROM 1.19-6.
3E823	xSTO>	(ob id →) (ob symb →) Like <REF>xSTO, but if the level 1 argument is symbolic, use the first element of it as the variable to write to.
0240DE	~xSTORE	
3BBD9	xSTR>	($ → ob) Evaluate String Cmd -- Evaluates the text of a string as if the text were entered from the command line. -- Related: ARRY→,DTAG,EQ→,LIST→, OBJ→,→STR UserRPL: xSTR→
3BBBE	x>STR	(ob → $) Object to String Cmd -- Converts any object to string form. -- Related: →ARRY,→LIST,STR→, →TAG,→UNIT UserRPL: x→STR

`0580AB` `~xSTREAM` `( {} prog → x )`
Stream Execution Cmd
--
Moves the first two elements from the list onto the stack, and executes prog. The moves the next element (if any) onto the stack, and executes obj again using the previous result and the new element. Repeats this until the list is exhausted, and returns the final result.
--
Related: DOSUBS

`0170DE` `~xSTURMAB`

`0160DE` `~xSTURM`

`3B6C1` `xSTWS` `( n → )`
`( #n → )`
Set Wordsize Cmd
--
Sets the current binary integer wordsize to n bits, where n is a value from 1 through 64 (the default is 64).
--
Related: BIN,DEC,HEX,OCT,RCWS

`3B8D7` `xSUB` `( ob start end → ob' )`
`ob= [[]], $, {}, grob, PICT`
`start,end = n, {n m}, (n,m)`
Subset Cmd
--
Returns the portion of a string or list defined by specified positions, or returns the rectangular portion of a graphics object or PICT defined by two corner pixel coordinates.
--
Related: CHR,GOR,GXOR,NUM,POS,REPL,SIZE

`002314` `~xSUBST` `( symb var=s1 → symb' )`

`06F314` `~xSUBTMOD` `( x1 x2 → x3 )`

`02E0AB` `~xSVD` `( [[]] → [[U]] [[V]] [S] )`
Singular Value Decomposition Cmd
--
Returns the sigular value decomposition of an mn matrix.
--
Related: DIAG→,MIN,SVL

02F0AB	~xSVL	([[]] → []) Singular Values Cmd -- Returns the singular values of an mn matrix. -- Related: MIN,SVD
3DC20	xSWAP	(ob1 ob2 → ob2 ob1) Swap Objects Cmd -- Interchanges the first two objects on the stack. -- Related: DUP,DUPN,DUP2,OVER,PICK,ROLL,ROLLD,ROT
04E314	~xSYLVESTER	([[]] → [D] [P])
39705	xSYSEVAL	(# → ?) Evaluate System Object Cmd -- Evaluates unnamed operating system objects specified by their memory addresses. -- Related: EVAL,LIBEVAL,FLASHEVAL
00A0DE	~xSYST2MAT	

7.4 T-Z

3B2DC x%T

(x y → 100y/x)
Percent of Total Function
--
Returns the percent of the level 2 argument that is represented by the level 1 argument.
--

```
x       y       →  100y/x
x       'sym'   →  '%T(x,sym)'
'sym'   x       →  '%T(sym,x)'
'sym1'  'sym2'  →  '%T(sym1,sym2)'
x_u1    y_u2    →  100y_u2/x_u1
x_u     'sym'   →  '%T(x_u,sym)'
'sym'   x_u     →  '%T(sym,x_u)'
```

--
Related: %,%ch

061314	~xTABVAL	(symb(x) {vals} → symb(x) {{vals} {res}})
060314	~xTABVAR	(symb(x) → symb(x) {{}{}} grob)

3EFB1	x->TAG	(ob tag → :tag:ob) Stack to Tag Cmd -- Combines objects in levels 1 and 2 to created tagged (labeled) object. Tag may be any object. It will eb converted to a string. -- Related: →ARRY,DTAG,→LIST,OBJ→, →STR,→UNIT UserRPL: x→TAG
0520AB	~xTAIL	({} → {}') ($ → $') Last Listed Elements Cmd -- Returns all but the first element of a list or string. -- Related: HEAD
01C0DE	~xTAN2CS2	(symb → symb')
021314	~xTAN2SC2	(symb → symb')
01F314	~xTAN2SC	(symb → symb')

3A624	xTAN	(x → x')
		Tangent Analytic Func
		--
		Returns the tangent of the argument.
		--
		`z → tan z`
		`'sym' → 'TAN(symb)'`
		`x_unitang → tan(x_unitang)`
		-
		Related: ATAN,COS,SIN

3A70C	xTANH	(x → x')
		Hyperbolic Tangent Analytic Func
		--
		Returns the hyperbolic tangent of the argument.
		--
		`z → tanh z`
		`'sym' → 'TANH(sym)'`
		-
		Related: ATANH,COSH,SINH

006314	~xTAYLOR0	(symb → symb')
3E6CA	xTAYLR	(symb var n → symb') Taylor's Polynomial Cmd -- Calculates the nth order Taylor's polynomial of 'symb' in the variable var. -- Related: ∂,∫,Σ

05B314	~xTCHEBYCHEFF	(n → pol)
01A314	~xTCOLLECT	(symb → symb')
0640AB	~xTDELTA	(x y → x') Temperature Delta Func -- Calculates a temperature change. -- x y → x x_u1 y_u2 → x_u1 x_u 'sym' → 'TDELTA(x_u,sym)' 'sym' y_u → 'TDELTA(sym,y_u)' 'sym1' 'sym2' → 'TDELTA(sym1,sym2)' -- Related: TINC
02E0DE	~xTESTS	
065314	~xTEVAL	(ob → ? time) Execute ob and return how long it took.
013314	~xTEXPAND	(symb → symb')
3C8FA	xTEXT	(→) Show Stack Display Cmd -- Displays the stack display. -- Related: PICTURE,PVIEW
37F7F	xTHEN	(0/1 →) THEN Cmd -- Starts the true-clause in conditional or error-trapping structure -- Related: CASE,ELSE,END,IFERR
38B43	xTHENCASE	THEN in a CASE statement. -- Related: CASE,ELSE,END,IFERR UserRPL: xTHEN
38ABA	xERRTHEN	THEN in an ON ERROR construct. -- Related: CASE,ELSE,END,IFERR UserRPL: xTHEN

39093	xTICKS	(→ #) Ticks Cmd -- Returns the system time as a binary integer, in units of 1/8192 second. -- Related: TIME
3905D	xTIME	(→ time) Time Cmd -- Returns the system time in the form HH.MMSSs. -- Related: DATE,TICKS,TSTR
39124	xSETTIME	(time →) Set System Time Cmd -- Sets the system time. -- Related: CLKADJ,→DATE UserRPL: x→TIME

0650AB	~xTINC	(x y → x')
		Temperature Increment Cmd
		--
		Calculates a temperature increment.
		--
		`xinit  y      → xfinal`
		`x_u1   y_u2    → x_u1final`
		`x_u    'sym'   → 'TINC(x_u,sym)'`
		`'sym'  y_u    → 'TINC(sym,y_u)'`
		`'sym1' 'sym2' → 'TINC(sym1,sym2)'`
		--
		Related: TDELTA

3C6B6	xTLINE	((x1,y1) (x2,y2) →) ({#n1 #m1} {#n2 #m2} →) Toggle Line Cmd -- For each pixel along the line in PICT defined by the specified coordinates, TLINE turns off every pixel that is on, and turns on every pixel that is off. -- Related: ARC,BOX,LINE
019314	~xTLIN	(symb → symb')

3E97B	xTMENU	(% → [InitMenu%])
		({} →)
		(name →)
		(Ob → [@LIST InitMenu])
		Temporary Menu Cmd
		--
		Displays a built-in menu, libary menu, or a user-defined menu.
		--
		Related: MENU,RCLMENU
3DF4D	xTOT	(→ xsum)
		(→ {x1...xn})
		Total Cmd
		--
		Computes the sum of each of the m columns of coordinate values in the current stat matrix (reserved variable ΣDAT).
		--
		<REF>TEXT:Reserved|ΣDAT
		--
		Related: MAXΣ,MINΣ,MEANMPSDEV, PVAR,SDEV,VAR
0270AB	~xTRACE	([[]] → x)
		Matrix Trace Cmd
		--
		Returns the trace of a square matrix.
045314	~xTRAN	([[]] → [[]]')
		(name →)
		--
		Related: CONJ,TRN
3EE0C	xTRANSIO	(n →)
		I/O Translation Cmd
		--
		Specifies the character translation option. These translations affect only ASCII Kermit transfer and files printed to the serial port.
		--
		Related: BAUD,CKSM,PARITY
01B314	~xTRIG	(symb → symb')
01C314	~xTRIGCOS	(symb → symb')
082314	~xTRIGO	(→)
01D314	~xTRIGSIN	(symb → symb')
01E314	~xTRIGTAN	(symb → symb')

`3C084` `xTRN`

`( [[]] → [[]]' )`
`( name → )`
Transpose Matrix Cmd
--
Returns the (conjugate) transpose of a matrix.
--
Related: CONJ

`3AF3E` `xTRNC`

`( x n → )`
Truncate Func
--
Truncates an object to a specified number of decimal places or significant digits, or to fit the current display format.
--

```
z1       ntrnc     → z2
z1       'symtrnc' →
'TRNC(z1,symtrnc)'
'sym1'   ntrnc     →
'TRNC(sym1,ntrnc)'
'sym1'   'symtrnc' →
'TRNC(sym1,symtrnc)'
[ arr ]1 ntrnc     → [ arr ]2
x_u      ntrnc     → y_u
x_u      'symtrnc' →
'TRNC(x_u,symtrnc)'
```

--
Related: RND

`063314` `~xTRUNC`

`( symb1 symb2 → symb3 )`

`3C99D` `xTRUTH`

`( → )`
Truth Plot Type Cmd
--
Sets the plot type to `TRUTH`.
--
Related: `BAR`,`CONIC`,DIFFEQ,`FUNCTION`, GRIDMAP,`HISTOGRAM`,`PARAMETRIC`,PARSURFACE, PCONTOUR,`POLAR`,`SCATTER`,SLOPEFIELD,WIREFRAME,YSLIC

`015314` `~xTSIMP`

`( symb → symb' )`

`391F8` `xTSTR`

`( date time → $ )`
Date and Time String Cmd
--
Returns a string derived from the date and time.
--
Related: `DATE`,TICKS,TIME

`39456`	`xTVARS`	`( ntype → {} )`
		`( {n...} → {} )`
		Typed Variables Cmd
		`--`
		Lists all global variables in the current directory that contain objects of the specified types.
		`--`
		Related: PVARS,`TYPE`,VARS

`0470AB`	`~xTVM`	`( → )`
		TVM Menu Cmd
		`--`
		Displays the TVM Solver menu.
		`--`
		Related: AMORT

`0480AB`	`~xTVMBEG`	`( → )`
		Payment at Start of Period Cmd
		`--`
		Specifies that TVM calculations treat payments as being made at the beginnign of the compounding periods.
		`--`
		Related: AMORT,TVM,TVMEND,TVMROOT

`0490AB`	`~xTVMEND`	`( → )`
		Payment at End of Period Cmd
		`--`
		Specifies that TVM calculations treat payments as being made at the end of the compounding periods.
		`--`
		Related: AMORT,TVM,TVMBEG,TVMROOT

`04A0AB`	`~xTVMROOT`	`( var → x )`
		TVM Root Cmd
		`--`
		Solves for the specified TVM variable using values from the re- maining TVM variables.
		`--`
		Related: AMORT,TVM,TVMBEG,TVMEND

3BC39 xTYPE (ob → %type)

Type Cmd

--

Returns the type number of an object.

--

User Objects:

--

```
Object    Type   Number
------    ----   ------
Real      number      0
Complex   number      1
Character string      2
Real      Array       3
Complex   Array       4
List                  5
Global    name        6
Local     name        7
Program               8
Algebraic Object      9
Binary    Integer    10
Graphics  object     11
Tagged    object     12
Unit      object     13
XLIB      name       14
Directory            15
Library              16
Backup    object     17
--
Built-in Cmds:
 --
Object    Type     Number
------    ----     ------
Built-in  function     18
Built-in  command      19
 --
System Objects:
 --
Object    Type    Number
------    ----    ------
System    binary      20
Extended  real        21
Extended  complex     22
Linked    array       23
Character             24
Code      object      25
Library   data        26
External  object   26-31
--
```

Related: SAME,TVARS,VTYPE

38FD7	xUBASE	(u → u')
		Convert to SI Base Units Func
		--
		Converts a unit object to SI base units.
		--
		`x_u   → y_base-units`
		`'sym' → 'UBASE(symb)'`
		--
		Related: CONVERT,UFACT,→UNIT,UVAL

3900B	xUFACT	(u1 u2 → u3) Factor Unit Cmd -- Factors the level 1 unit from the unit expression of the level 2 unit object. -- Related: CONVERT,UBASE,→UNIT,UVAL
0140DD	~xUFL1→MINIF	(ob n → font)
0310DE	~xUNASSIGN	
0270DE	~xUNASSUME	
38FB5	x>UNIT	(x u → u') Stack to Unit Object Cmd -- Creates a unit object from a real number and the unit part of a unit object. -- Related: →ARRY,→LIST,→STR,→TAG UserRPL: x→UNIT
3F249	xUNPICK	(obn...ob1 ob n → ob...ob2) Replaces the object at level n+2 with the object at level 2 and deletes the objects at level 1 and level 2. -- Related: OVER,PICK,ROLL,ROLLD,SWAP,ROT
3F22E	xUNROT	(1 2 3 → 3 1 2) Changes the order of the first three objects on the stack, in the opposite way compared to ROT. -- Related: OVER,PICK,ROLL,ROLLD,SWAP,ROT

38195	`xUNTIL`	(→) `UNTIL` Cmd -- Starts test-clause in `DO ... UNTIL ... END` indefinite loop structure. -- See the `DO` entry for syntax info. -- Related: `DO`,`END`
39420	`xUPDIR`	(→) Up Directory Cmd -- Makes the parent of the current directory the new current directory. -- Related: CRDIR,HOME,PATH,PGDIR
3E07D	`xUTPC`	(n x → x') Upper Chi-Square Distribution Cmd -- Returns the probability utpc(n,x) that a chi-square random variable is greater than x, where n is the number of degrees of freedom of the distribution. -- Related: UTPF,UTPN,UTPT
3E0BD	`xUTPF`	(n1 n2 x → x') Upper Snedecor's F Distrib. Cmd -- Returns the probability utpf(n1,n2,x) that a Snedecor's F random variable is greater than x, where n1 and n2 are the numerator and denominator degrees of freedom of the F distribution. -- Related: UTPC,UTPN,UTPT
3E09D	`xUTPN`	(n v x → x') Upper Normal Distribution Cmd -- Returns the probability utpn(m,v,x) that a normal random variable is greater than x, where m and v are the mean and variance, respectively, of the normal distribution. -- Related: UTPC,UTPF,UTPT

3E0DD	xUTPT	(n x → x') Upper Student's t Distrib. Cmd -- Returns the probability utpt(n,x) that a Student's t random variable is greater than x, where n is the number of degrees of freedom of the distribution. -- Related: UTPC,UTPF,UTPN

38F81	xUVAL	(u → x)
		Unit Value Func
		--
		Returns the numerical part of a unit object.
		--
		`x_u    → x`
		`'sym' → 'UVAL(sym)'`
		--
		Related: CONVERT,UBASE,UFACT,→UNIT

3C2AC	xV>	([]/() → x y)
		([]/() → x y z)
		(in current co-system)
		Vector/Complex Num to Stack Cmd
		--
		`[ x y ]             → x y`
		`[ xr ANGyθ ]        → xr yθ`
		`[ x1 x2 x3 ]        → x1 x2 x3`
		`[ x1 ANGxθ xz ]     → x1 xθ xz`
		`[ x1 ANGxθ ANGx] → x1 xθ x`
		`[ x1 x2 ... xn ]    → x1 ... xn`
		`(x,y)               → x y`
		`(xr ANGyθ)          → xr yθ`
		--
		Related: →V2,→V3 UserRPL: xV→

3C2D6	x>V2	(x y → []) (x y → ()) Stack to Vector/Complex Num Cmd -- Converts two numbers from the stack into a 2-element vector or complex number. -- Related: V→,→V3 UserRPL: x→V2
3C30A	x>V3	(x y z → []) Stack to 3-Element Vector Cmd -- Converts three numbers into a 3-element vector. -- Related: V→,→V2 UserRPL: x→V3

`053314`	`~xVANDERMONDE`	`( {} → [[]] )`

`3DF68`	`xVAR`	`( → x )`
		`( → [x1...xn] )`
		Variance Cmd
		--
		Calculates the sample variance of the coordinate values in each of the m columns in the current stat matrix (ΣDAT).
		--
		Related: `MAXΣ`,`MEAN`,`MINΣ`,PSDEV,PVAR, `SDEV`,`TOT`

`3943B`	`xVARS`	`( → {} )` Variables Cmd -- Returns a list of all variables' names in the `VAR` menu (the current directory). -- Related: ORDER,PVARS,TVARS
`08C314`	`~xVER`	`( → $ )`
`00F0AB`	`~xVERSION`	`( → $ $ )` Software Version Cmd -- Displays the software version and copyright message.
`0080DD`	`~xVISIT`	`( name → )` For a specified variable, opens the content in the command-line editor. -- Related: VISITB,EDIT,EDITB
`00A0DD`	`~xVISITB`	`( name → )` For a specified variable, opens the contents in the most suitable editor for the object type. For example, if the variable holds an equation, the equation writer is used. -- Related: VISIT,EDIT,EDITB
`0390DE`	`~xVPOTENTIAL`	

3BDB2	xVTYPE	`( name → n )`
		Variable Type Cmd
		--
		Returns the type number of the object contained in the named variable.
		--
		`'name'             → ntype`
		`:nport:namebackup → ntype`
		`:nport:nlibrary    → ntype`
		--
		Related: `TYPE`

39819	xWAIT	`( sec → )`
		`( 0 → rc.p )`
		Wait Cmd
		--
		Suspends program execution for specified time, or until a key is pressed.
		--
		Related: KEY

380DB	xWHILE	`( → )`
		`WHILE` Indefinite `Loop` Struct Cmd
		--
		Starts the
		`WHILE ... REPEAT ... END`
		indefinite loop structure.
		--
		Related: `DO`,`END`,`REPEAT`

0080AB	~xWIREFRAME	`( → )` WIREFRAME Plot Type Cmd -- Sets the plot type to WIREFRAME. -- Related: `BAR`,`CONIC`,DIFFEQ,`FUNCTION`, GRIDMAP,`HISTOGRAM`,`PARAMETRIC`,PARSURFACE, PCONTOUR,`POLAR`,`SCATTER`,SLOPEFIELD,`TRUTH`,YSLICE
390AE	xWSLOG	`( → $ $ $ $ )` `Warmstart` Log Cmd -- Returns four strings recording the date, time, and cause of the four most recent warmstart events

`3DE90`	`xSUMX2`	(→ `xsum`) Sum of Squares of x-Values Cmd -- Sums the squares of the values in the independent-variable column of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: `NΣ`,`XCOL`,`ΣX`,`ΣXY`,`ΣX2`,`ΣY`,`ΣY2` UserRPL: `xΣX2`
`3E03D`	`xXCOL`	(`n` →) Independent Column Cmd -- Specifies the independent variable column of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT
`0700AB`	`~xXGET`	(`name` →) Xmodem get command: Retrieves a specified `filename` via XMODEM. The other calculator needs to be in server mode. -- Related: BAUD,RECN,RECV,SEND,XRECV,XSERV,XPUT
`3EC35`	`xXMIT`	(`$` → `1`) (`$` → `$rest 0`) Serial Transmit Cmd -- Sends a string serially without using Kermit protocol, and returns a single digit that indicates whether the transmission was successful. -- Related: BUFLEN,SBRK,SRECV,STIME
`067314`	`~xXNUM`	(`x` → `x'`)

3CB7A xXOR (# #’ → #’’)
($ $’ → $’’)
(1/0 1/0 → 1/0)
Exclusive OR Cmd
--
Returns the logical exclusive OR of two arguments.

```
#n1     #n2     → #n3
"str1"  "str2"  → "str3"
T/F1    T/F2    → 0/1
T/F     'sym'   → 'T/F XOR sym'
'sym'   T/F     → 'sym XOR T/F'
'sym1'  'sym2'  → 'sym1 XOR sym2'
```

--
Related: AND,OR,NOT

3AD65 xXPON (% → n)
Exponent Func
--
Returns the exponent of the arg.
--
Related: MANT,SIGN

0710AB ~xXPUT (name →)
Xmodem command: Sends a specified filename via XMODEM to a claculator. The receiving calculator needs to be in server mode.
--
Related:
BAUD,RECN,RECV,SEND,XRECV,XSERV,XGET

068314 ~xXQ (x → x’)

0500AB ~xXRECV (name →)
XModem Receive Cmd
--
Prepares the HP 48 to receive an object via XModem. The received object is stored in the given name.
--
Related: BAUD,RECV,RECN,SEND,XSEND

3C915 xXRNG (x1 x2 →)
x-Axis Display Range Cmd
--
Specifies the x-axis display range.
--
Related: AUTO,PDIM,PMAX,PMIN,YRNG

3A278	xXROOT	(y x → Y')
		xth Root of y Cmd
		--
		Computes the xth root of a real number.
		`y      x      → x ROOT y`
		`'sym1' 'sym2' → 'XROOT(sym2,sym1)'`
		`'sym'  x      → 'XROOT(x,sym)'`
		`y      'sym'  → 'XROOT(sym,y)'`
		`y_u    x      → x ROOT y_u1/x`
		`y_u    'sym'  → 'XROOT(sym,y_u)'`

04F0AB	~xXSEND	(name →) XModem Send Cmd -- Sends a copy of the named object via XModem. -- Related: BAUD,RECN,RECV,SEND,XRECV

06E0AB	~xXSERV	(→)
		Xmodem server command: Puts the calculator in XMODEM server mode. When in server mode, the following commands are available:
		`P: Put a file in calc`
		`G: Get a file from calc`
		`E: Execute a cmd line`
		`M: Get the calc memory`
		`L: List files in current dir`
		--
		Related: BAUD,RECN,RECV,SEND,XRECV, XGET,XPUT

0000AB	~xXVOL	(x1 x2 →) X Volume Coordinates Cmd -- Sets the width of the view volume in the reserved variable VPAR. -- <REF>TEXT:Reserved|VPAR -- Related: EYEPT,XXRNG,YVOL,YYRNG,ZVOL
0030AB	~xXXRNG	(x1 x2 →) X Range of an Input Plane Cmd -- Specifies the x range of an input plane (domain) for GRIDMAP and PARSURFACE plots. -- Related: EYEPT,NUMX,NUMY,XVOL,YVOL, YYRNG,ZVOL

3DEC6	xSUMXY	(→ xsum) Sum of x Times Y Cmd -- Sums the products of the corresponding values in the independent and dependent variable columns of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: NΣ,XCOL,ΣX,ΣX2,ΣY,ΣY2 UserRPL: xΣXY
3DE75	xSUMY	(→ xsum) Sum of y-Values Cmd -- Sums the values in the dependent variable column of the current stat matrix (reserved var ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: NΣ,XCOL,ΣX,ΣXY,ΣX2,ΣY2 UserRPL: xΣY
3DEAB	xSUMY2	(→ xsum) Sum of Squares of y-Values Cmd -- Sums the squares of the values in the dependent-variable column of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: NΣ,XCOL,ΣX,ΣXY,ΣX2,ΣY UserRPL: xΣY2
3E05D	xYCOL	(n →) Dependent Column Cmd -- Specifies the dependent-variable column of the current stat matrix (reserved variable ΣDAT). -- <REF>TEXT:Reserved|ΣDAT -- Related: BARPLOT,BESTFIT,COLΣ,CORR, COV,EXPFIT,HISTPLOT,LINFIT,LOGFIT, LR,PREDX,PREFY,PWRFIT,SCATRPLOT,XCOL
3C935	xYRNG	(y1 y2 →) y-Axis Display Range Cmd -- Specifies the y-axis display range. -- Related: AUTO,PDIM,PMAX,PMIN,XRNG

`00B0AB`	`~xYSLICE`	(→) Y-Slice Plot Cmd -- Sets the plot type to YSLICE. -- Related: `BAR`,`CONIC`,DIFFEQ,`FUNCTION`, GRIDMAP,`HISTOGRAM`,`PARAMETRIC`,PARSURFACE, PCONTOUR,`POLAR`,`SCATTER`,SLOPEFIELD,`TRUTH`,WIREFRAME
`0010AB`	`~xYVOL`	(`y1 y2` →) Y Volume Coordinates Cmd -- Sets the depth of the view volume in the reserved variable VPAR. `ynear yfar` → -- <REF>TEXT:Reserved|VPAR -- Related: EYEPT,XVOL,XXRNG,YYRNG,ZVOL
`0040AB`	`~xYYRNG`	(`y1 y2` →) Y Range of an Input Plane Cmd -- Specifies the y range of an input plane (domain) for GRIDMAP and PARSURFACE p lots. -- Related: EYEPT,XVOL,XXRNG,YYRNG,ZVOL
`040314`	`~xZEROS`	(`symb var` → `{zeros}`)
`05F0AB`	`~xZFACTOR`	(`xTr yPr` → `xZf`) Gas Compressibility Z Factor Func -- Calculates the gas compressibility ity correction factor for non-ideal behavior of a hydro-carbon gas.
`0020AB`	`~xZVOL`	(`x1 x2` →) Z Volume Coordinates Cmd -- Sets the height of the view volume in the reserved variable VPAR. -- <REF>TEXT:Reserved|VPAR -- Related: EYEPT,XVOL,XXRNG,YVOL,YYRNG

7.5 Non A-Z

`3A097`	`x^`	(y x → y^x)
		Power Analytic Func
		--
		Returns the value of the level 2 object raised to the power of the level 1 object.
		`w      z      → w^z`
		`z      'sym'  → 'z^sym'`
		`'sym'  z      → '(sym)^z'`
		`'sym1' 'sym2' → 'sym1^(sym2)'`
		`x_u    y      → xy_uy`
		`x_u    'sym'  → '(x_u)^(sym)'`
		--
		Flags: -1 -3
		`Principal soln  -1`
		`Numeric results -3`
		--
		Related: EXP,ISOL,LN,XROOT

`3D56B`	`x|`	(symb {var val ...} → x') Where Func -- Substitutes values for names in an expression. -- 2: 'symold' 1: { name1 'sym1' name2 'sym2' ... } ↓ ; 1: 'symnew' -- 2: x 1: { name1 'sym1' name2 'sym2' ... } ↓ ; 1: x -- 2: (x, y) 1: { name1 'sym1' name2 'sym2' ... } ↓ ; 1: (x, y) -- Flags: -3 `Numeric results -3` -- Related: APPLY,QUOTE

3A442	xSQRT	`( x → x' )`
		Square Root Analytic Func
		--
		Returns the (+ve) square root of the argument.
		--
		`z` → $\sqrt{a}$`z`
		`x_u` → $\sqrt{a}$`(x)_u`
		`'sym'` → `'SQRT(sym)'`
		--
		Flags: -1 -3
		--
		Related: SQ,^,ISOL UserRPL: `x`$\sqrt{a}$

3D434	x$\int$	`( x1 x2 symb var → symb' )` Integral Func -- Integrates symb from lower limit x1 to upper limit x2 respect to a variable var of integration. -- Flags: -3 -45 -46 -47 -48 -49 -50 -- Related: TAYLR,∂,Σ

3DDC4	xSIGMA+	`( x → )`
		`( x1...xn → )`
		`( []/[[]] → )`
		Sigma Plus Cmd
		--
		Adds one or more data points to the current stat matrix (reserved variable ΣDAT).
		--
		<REF>TEXT:Reserved|ΣDAT
		--
		Related: CLΣ,RCLΣ,STOΣ,Σ- UserRPL: `xΣ+`

3DDEE	xSIGMA-	`( → x )`
		`( → [] )`
		Sigma Minus Cmd
		--
		Returns a vector of m real numbers (or one number x if m = 1) corresponding to the coordinate values of the last data point entered by Σ+ into the current stat matrix (reserved variable ΣDAT).
		--
		<REF>TEXT:Reserved|ΣDAT
		--
		Related: CLΣ,RCLΣ,STOΣ,Σ+ UserRPL: `xΣ-`

`39AC7` `xPI` (→ π)
PI Func
--
Returns the symbolic constant 'π' or its numerical representation, 3.14159265359.
```
→ 'π'
→ 3.14159265359
```
--
Flags: -2 -3
--
Related: e,i,MAXR,MINR,→Qπ UserRPL: `xπ`

`3D202` `x∂` (`symb var` → `symb'`)
Derivative Func
--
Takes the derivative of an expression, number, or unit object with respect to a specified variable of differentiation.
--
```
'sym1' 'name' → 'sym2'
z      'name' → 0
x_unit 'name' → 0
```
--
Flags: -3
--
Related: TAYLR,∫,Σ

`3CF80` `x<=?` (x y → 1)
(x y → 0)
Less Than or Equal Func
--
Tests whether one object is less than or equal to another object.
--
```
x      y      → 0/1
#n1    #n2    → 0/1
"str1" "str2" → 0/1
x      'sym'  → 'x<=sym'
'sym'  z      → 'sym<=z'
'sym1' 'sym2' → 'sym1<=sym2'
x_u1   y_u2   → 0/1
x_u    'sym'  → 'x_unit<=sym'
'sym'  x_u    → 'sym<=x_unit'
```
--
Flags: -3
--
Related: <,>,≥,==,≠ UserRPL: `x≤`

`3D01F` `x>=?`

(x y → 1)
(x y → 0)
Greater Than or Equal Func
--

```
x       y       →  0/1
#n1     #n2     →  0/1
"str1"  "str2"  →  0/1
x       'sym'   →  'x≥sym'
'sym'   z       →  'sym≥z'
'sym1'  'sym2'  →  'sym1≥sym2'
x_u1    y_u2    →  0/1
x_u     'sym'   →  'x_u≥sym'
'sym' x_u       →  'sym≥x_u'
```

--
Flags: -3
--
Related: <,≤,>,==,≠ UserRPL: x≥

`3CD21` `x#?`

(x y → 1)
(x y → 0)
Not Equal Func
--
Tests if two objects are equal.

```
obj1    obj2    →  0/1
(x,0)   x       →  0/1
x       (x,0)   →  0/1
z       'sym'   →  'z≠sym'
'sym'   z       →  'sym≠z'
'sym1'  'sym2'  →  'sym1≠sym2'
```

--
Flags: -3
--
Related: SAME,TYPE,<,≤,>,≥, == UserRPL: x≠

`3885C` `xRPN->`

(ob1 .. obn →)
Create Local Variables Cmd
--
Creates local variables.

```
obj1 ... objn →
```

--
Syntax:

```
→ name1 name2 ... nameN ≪ prog ≫
→ name1 name2 ... nameN 'Expr'
```

--
Related: DEFINE,STO UserRPL: x→

`38093` `xALG->`

Create local variable comand. <REF>xRPN-> UserRPL: x→

3ABAF	xFACT	(x → x')
		Factorial (Gamma) Func
		--
		Returns the factorial n! of a positive integer argument n, or the gamma function (x+1) of a non-integer argument x.
		`n      → n!`
		`x      → (x+1)`
		`'sym'  → '(sym!)'`
		--
		Flags: -3 -20 -21
		`Numerical Results   -3`
		`Underflow exception -20`
		`Overflow exception  -21`
		--
		Related: COMB,PERM UserRPL: `x!`

3B251	x%	(x y → xy/100)
		Percent Func
		--
		Returns x (level 2) percent of y (level 1).
		`x      y      → xy/100`
		`x      'sym'  → '%(x,sym)'`
		`'sym'  x      → '%(sym,x)'`
		`'sym1' 'sym2' → '%(sym1,sym2)'`
		`x      y_unit → (xy/100)_unit`
		`x_unit y      → (xy/100)_unit`
		`'sym'  x_unit → '%(sym,x_unit)'`
		`x_unit 'sym'  → '%(x_unit,sym)'`
		--
		Flags:
		`Numerical Results -3`
		--
		Related: `%CH`,`%T`

39DE8	x*	(x y → x*y)

Multiply Analytic Func

--

Returns the product of the args.

```
z1         z2       → z1z2
[[ mat ]]  [ arr ]  → [[ matarr ]]
z          [ arr ]  → [ z  array ]
[ arr ]    z        → [ arr  z ]
z          'sym'    → 'z * sym'
'sym'      z        → 'sym * z'
'sym1'     'sym2'   → 'sym1 * sym2'
#n1        n2       → #n'
n1         #n2      → #n'
#n1        #n2      → #n'
x_u        y_u      → xy_ux  unity
x          y_u      → xy_u
x_u        y        → xy_u
'sym'      x_u      → 'sym * x_u'
x_u        'sym'    → 'x_u * sym'
```

--

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

```
Numeric results -3
bint wordsize   -5 → -10
```

--

Related: +,-,/,=

39B58 x+

(x y → x+y)
Add Analytic Func
--
Returns the sum of the arguments. Addition. If one arg is list, insert element in list or concatenate lists.
<REF>xADD

```
z1        z2        → z1+z2
[ arr ]1 [ arr ]2 → [ arr ]1+2
z         'sym'     → 'z+(sym)'
'symb'    z         → 'sym+z'
'sym1'    'sym2'    → 'sym1 + sym2'
{ lst1 } { lst2 } → { lst1 lst2 }
obj       { o... } → { obj o... }
{ o... } o         → { o... obj }
"str1"    "str2"    → "str1str2"
obj       "str"     → "obj str"
"str"     obj       → "str obj"
#n1       n2        → #n'
n1        #n2       → #n'
#n1       #n2       → #n'
x1_u1     y_u2      → (x2+y)_u2
'sym'     x_u       → 'sym+x_u'
x_u       'sym'     → 'x_u+sym'
grob1     grob2     → grob'
```

--
Flags: -3 -4 -5 -6 -7 -8 -9 -10
```
Numeric results -3
Bint wordsize   -5 → -10
```
--
Related: -,*,/,=

39CFC x- (x y → x-y)
Subtract Analytic Func
--
Returns the difference of the arguments: the object in level 1 is subtracted from the object in level 2.

```
z1        z2        → z1-z2
[ arr ]1  [ arr ]2  → [ arr ]1_2
z         'sym'     → 'z-sym'
'sym'     z         → 'sym-z'
'sym1'    'sym2'    → 'sym1 - sym2'
#n1       n2        → #n'
n1        #n2       → #n'
#n1       #n2       → #n'
x1_u1     y_u2      → (x2-y)_u2
'sym'     x_u       → 'sym-x_u'
x_u       'sym'     → 'x_u-sym'
```

--
Flags: -3
`Numeric results -3`
--
Related: +,*,/,=

39F49 x/ (x y → x/y)
Divide Analytic Func
--
Returns the quotient of the arguments: the level 2 object divided by the level 1 object. (Abbrev. _u = _unit)

```
z1       z2        → z1 / z2
[ arr ]  [[ mat ]] → [[mat^-1arr]]
[ arr ]  z         → [ arr / z ]
z        'sym'     → 'z / sym'
'sym'    z         → 'sym / z'
'sym1'   'sym2'    → 'sym1 / sym2'
#n1      n2        → #n'
n1       #n2       → #n'
#n1      #n2       → #n'
x_u1     y_u2      → (x/y)_u1/u2
x        y_u       → (x/y)_1/u
x_u      y         → (x/y)_u
'sym'    x_u       → 'sym/x_u'
x_u      'sym'     → 'x_u/sym'
```

--
Related: +,-,*,=,RATIO

3CE42	x<	(x y → 1)
		(x y → 0)
		Less Than Func
		--
		Tests whether one object is less than another object.
		`x       y       →  0/1`
		`#n1     #n2     →  0/1`
		`"str1"  "str2"  →  0/1`
		`x       'sym'   →  'x<sym'`
		`'sym'   x       →  'sym<z'`
		`'sym1'  'sym2'  →  'sym1<sym2'`
		`x_u1    y_u2    →  0/1`
		`x_u     'sym'   →  'x_u<sym'`
		`'sym'   x_u     →  'sym<x_u'`
		--
		Flags: -3
		`Numeric results -3`

398B9	x=	(x y → x=y)
		Makes equation out of two expressions. Equals Analytic Func
		--
		Returns an equation formed from the two arguments.
		`z1      z2      →  'z1=z2'`
		`z       'sym'   →  'z=sym'`
		`'sym'   z       →  'sym=z'`
		`'sym1'  'sym2'  →  'sym1=sym2'`
		`y       x_u     →  'y=x_u'`
		`y_u     x       →  'y_u=x'`
		`y_u     x_u     →  'y_u=x_u'`
		`'sym'   x_u     →  'sym=x_u'`
		`x_u     'sym'   →  'x_u=sym'`
		--
		Flags: -3
		`Numeric results -3`
		--
		Related: DEFINE,`EVAL`,-

3CBF6	x==	(x y → 1)
		(x y → 0)
		Logical Equality Func
		--
		Tests if two objects are equal.
		`obj1   obj2   → 0/1`
		`(x,0)  x      → 0/1`
		`x      (x,0)  → 0/1`
		`z      'sym'  → 'z==sym'`
		`'sym'  z      → 'sym==z'`
		`'sym1' 'sym2' → 'sym1==sym2'`
		--
		Flags: -3
		`Numeric results -3`
		--
		Related: SAME,TYPE,<,≤,>,≥,≠

3CEE1	x>	(x y → 1)
		(x y → 0)
		Greater Than Func
		--
		Tests whether one object is greater than another object.
		`x       y       → 0/1`
		`#n1     #n2     → 0/1`
		`"str1"  "str2"  → 0/1`
		`x       'sym'   → 'x>sym'`
		`'sym'   z       → 'sym>z'`
		`'sym1'  'sym2'  → 'sym1>sym2'`
		`x_u1    y_u2    → 0/1`
		`x_u     'sym'   → 'x_u>sym'`
		`'sym'   x_u     → 'sym>x_u'`
		--
		Flags: -3
		`Numeric results -3`
		--
		Related: <,≤,≥,==,≠ ;

7.6 The Development Library 256

000100	~x→H	(ob → $hex)
001100	~xH→	($hex → ob)
002100	~x→A	(ob → hxs)
003100	~xA→	(hxs → ob)
004100	~xA→H	(hxs → $hex)

005100	~xH→A	($hex → hxs)
006100	~x→CD	($hex → code)
007100	~xCD→	(code → $hex)
008100	~xS→H	($ → $hex)
009100	~xH→S	($hex → $)
00A100	~x→LST	(comp → {}) (ob1..obn %n → {})
00B100	~x→ALG	(comp → symb) (ob1..obn %n → symb)
00C100	~x→PRG	(comp → ::) (ob1..obn %n → ::)
00D100	~xCOMP→	(comp → ob1...obn %n)
00E100	~x→RAM	(ob → ob)
00F100	~xSREV	($ → $')
010100	~xPOKE	(hxs $hex →)
011100	~xPEEK	(hxs1 hxs2 → $hex)
012100	~xAPEEK	(hxs → hxs')
013100	~xR~SB	(% → #) (# → %)
014100	~xSB~B	(# → hxs) (hxs → #)
015100	~xLR~R	(%% → %) (% → %%)
016100	~xS~N	($ → ID) (ID → $)
017100	~xLC~C	(%%C → %C) (%C → %%C)
018100	~xASM→	(Code → $)
019100	~xBetaTesting	(→ $)
01A100	~xCRLIB	(→ lib)
01B100	~xCRC	($ → #crc)
01C100	~xMAKESTR	(xlen → $)
01D100	~xSERIAL	(→ $)
01E100	~xASM	($ → ob)
01F100	~xER	($ {errors} → $')
020100	~x→S2	(ob → $)
021100	~xXLIB~	(xlib xn → ROMPTR) (ROMPTR → xlib xn)

7.7 The EXTABLE Library

001102	`~xGETADR`	`( $ → hxs )` Get the address of an entry name.
002102	`~xGETNAME`	`( hxs → $ )` Get the entry name corresponding to an address.
003102	`~xGETNAMES`	`( $start → {} )` Get all entry names which start with the given string.
004102	`~xGETNEAR`	`( $sub → {} )` Get all entry names which contain the given string.

8 ML Entry Points

8.1 General Purpose

0679B	SAVPTR	D0 to RPLTOP D1 to DSKTOP B to RETTOP D to FREETOP Clear carry
067D2	GETPTR	<see>SAVPTR in reverse Clears Carry.
05143	GETPTRLOOP	<see>GETPTR , Loop to RPL
36897	D0=DSKTOP	Get new D0 from DSKTOP, uses A
368A6	D1=DSKTOP	Get new D1 from DSKTOP, uses C
26767	AllowIntr	Allow interrupts.
26791	DisableIntr	Disable interrupts.
0020A	AINRTN	A=IN see also <see>CINRTN For hardware reasons (bug) A=IN must be at even addr
00212	CINRTN	C=IN see also <see>AINRTN For hardware reasons (bug) C=IN must be at even addr

8.2 Errors

8.2.1 Generating Errors

04FBB	DOMEMERR	Insufficient Memory error
26CA7	DOSIZEERR	Bad Argument Value error
05023	Errjmp	Error exit A.A = error number
266C6	ErrjmpC	A=C.A <see>Errjmp
266DB	GPErrjmpC	A=C.A <see>GETPTR <see>Errjmp
065AA	GPMEMERR	<see>GETPTR <see>DOMEMERR

8.2.2 Error Number Constants

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

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

8.3 Hexadecimal Math

26A2A	ADIV3	A.A = A.A/3 Uses A.6 C.6 P
26A23	ADIV6	A.A = A.A/6 Uses A.6 C.6 P
26A15	ADivC	B.A=A.A/C.A Uses A.A C.A
269F2	AMULT34	A.A=A.A*34 Uses C.A
26A1C	BMULT34	B.A=B.A*34 Uses C.A
269F9	CMULT34	C.A=A.A*34 Uses A.A
26F00	DCHXW	Converts BCD in C.W to hex in A.W B.W C.W. See <see>HXDCW Uses P CRY
06A8E	DIV5	C.A = C.A/5 Uses A.10 C.10 D.10 P
26A0E	HEXTODEC	Converts hex in C.A to BCD in A.A Uses A.6 B.6 P
2DEAA	HXDCW	Converts hex in A.W to BCD in A.W B.W C.W. See <see>DCHXW Uses P CRY Note that HXDCW wants the input in A but DCHXW wants it in C
03F24	IntDiv	A.A/C.A -> A.A=remainder, C.A=quotient, uses D.A P SB
2709E	MPY	Multiply A.W and C.W (-> A.W=C.W) Uses D.W, SB. Returns carry clear
03991	MUL#	B.A = A.A*C.A
26A07	MULTB+A*C	B.A=B.A+(A.A*C.A)

26A00	MULTBAC	B=0.A <see>MULTB+A*C

8.4 Long Reals

8.4.1 Storage Handling

31348	STAB0	A.W -> R0 B.W -> R1
31356	STAB2	A.W -> R2 B.W -> R3
31364	STCD0	C.W -> R0 B.W -> R1
31372	STCD2	C.W -> R2 B.W -> R3
3139C	RCAB0	R0 -> A.W R1 -> B.W
313A7	RCAB2	R2 -> A.W R3 -> B.W
313B2	RCCD0	R0 -> C.W R1 -> D.W
313BD	RCCD2	R2 -> C.W R3 -> D.W

31380	EXAB0	A.W <-> R0
		B.W <-> R1

3138E	EXAB2	A.W <-> R2
		B.W <-> R3

3133A	XYEX	A:B <-> C:D

8.4.2 Calculating

31756	DIVF	x=x/y
316FD	MULTF	x=x*y
3158F	RADD1	x=x+1 see <see>RADDF
315A9	RADDF	x=x+y
31586	RSUB1	x=x-1 see <see>RADDF

8.4.3 Conversion

2F4A2	PACK	(x -> A) <see>PACKSB without rounding

2F47D	PACKSB	(x -> A) Converts %% to %.If SB is clear uses roundup, if set uses lowest nibble in % field to determine rounding direction. Obeys and sets flow flags/indicators
31131	SPLITA	(A -> x) Convert % to %%
31193	(SPLITC)	(C -> y) Convert % to %%
31187	SPLTAC	(A,C -> x, y) Convert 2 reals to long reals

8.5 Memory Handling

8.5.1 General Memory Handling Routines

069F7	ADJMEM	D= @FREETOP=<see>ROOM / 5 Uses A.10 B.10 C.10 D.10 <see>DIV5
0554C	DOGARBAGE	If ST=1 10 then <see>GPMEMERR else <see>GARBAGECOL and <see>GETPTR
0613E	GARBAGECOL	Garbage collection does not use R1..R4
06806	ROOM	-> C.A = @DSKTOP-@RETTOP Uses A.A D0
03019	SKIPOB	Skip object in D0, clears ST1, clears carry, P=0 --> D0 = addr past object Uses: A.A C.A P ST1 RSTK2

8.5.2 Moving and Swapping Memory Areas

2682B	BLKSWAP+	<see>SWAPMEM_D0D1C and adjusts all refs
26871	EndTempOb	Moves TEMPOB zone at D0 to top of TEMPOB area -> D0=new addr Note that (1) the object must be skippable and (2) it must be a TEMPOB zone of its own (not embedded). This entry is however safe to use from TEMPOB because it keeps track of one RSTK address as well. aka: NEWADR

0670C	MOVEDOWN	Copy downwards C.A nibbles from D0 to D1, D0 and D1 will point to the next locations Used: A.W C.A P Use this to move upwards
06992	MOVERSD	Delete a block below RSK A.A=end C.A=nibbles Adjusts all refs, then <see>ADJMEM Uses A.W B.A C.W D.10 D0 D1 P
06A53	MOVERSU	Open a block below RSK A.A=start C.A=nibbles Adjusts all refs, then <see>ADJMEM Uses A.W B.A C.10 D.10 D0 D1 P
06A1D	MOVEDSD	Open a block above stack A.A=end C.A=nibbles Adjusts all refs, then <see>ADJMEM Uses A.W B.A C.10 D.10 D0 D1 P
069C5	MOVEDSU	Delete a block above stack A.A=start C.A=nibbles Adjusts all refs, then <see>ADJMEM Uses A.W B.A C.10 D.10 D0 D1 P
066B9	MOVEUP	Copy upwards C.A nibbles from D0 to D1 D0 D1 will point to start of area Used: A.W C.A P Use this to move downwards
269B3	SWAPMEM	Swaps two memory areas Area 1: R1.A to R2.A Area 2: R2.A to R3.A Uses <see>SWAPMEM_D0D1C
269DD	SWAPMEMEQ	Swaps two memory areas of the same size <see>SWAPMEMEQ_D0D1C R1.A->Area1 R2.A->Area2
269E4	SWAPMEMEQ_D0D1C	Swaps two memory areas of the same size D0->Area1 D1->Area2 C=(D1-D0) Uses A.W B.A C.W P CRY
269BA	SWAPMEM_D0D1C	D=C.A <see>SWAPMEM_D0D1D
269C1	SWAPMEM_D0D1C_nofree	D=C.A <see>SWAPMEM_D0D1D_nofree
269C8	SWAPMEM_D0D1D	Swaps two memory areas Area 1: D0 to D1 Area 2: D1 to (D1+D.A) Uses A.W B.A C.W D.W P CRY
269CF	SWAPMEM_D0D1D_nofree	<see>SWAPMEM_D0D1D but does not alter the memory @RSKTOP

269D6	SWAPMEM_nofree	<see>SWAPMEM but does not alter the memory @RSKTOP

8.5.3 Allocating Memory in TEMPOB

06AD8	CREATETEMP	Allocates C.A nibbles carry if not enough memory -> D0=bottom, D1=top of area -> B.A = C.A = @D1 = offset to previous tempob = #nibbles+6
039BE	GETTEMP	<see>CREATETEMP with <see>GARBAGECOL if necessary <see>GPMEMERR if not enough memory
268CC	GETBOTTEMP	Allocates C.A nibbles at the bottom of TEMPOB, errors if not enough memory Returns D0=top D1=bottom of area C.A=nibbles Uses A.W B.A C.W D.10 R1.A Bottom of TEMPOB means two things: 1. VERY dangerous if called from TEMPOB 2. The allocated string will not be moved by GC
05B79	MAKE$	Creates character string in tempob area Does SETHEX, C=C+C.A and then <see>MAKE$N
05B7D	MAKE$N	Creates character string in tempob area If not enough mem even after GC then memerr C.A = nibbles -> A=nibbles+5, B=nibbles+16 C=D1=addr of stack D0 = addr of body of $ R0 = addr of $ Not used: R1-R4
26919	MAKEBOT$N	Creates a C.A nibs long string at the bottom of TEMPOB D0->body R0.A->string R1.A=len Uses A.W B.A C.W D.10 See <see>WIPEOUT <see>GETBOTTEMP
26920	MAKERAM$	Allocates all free mem in a str, leaves 5 nibbles for pushing See <see>MAKE$N <see>ROOM

8.5.4 Resizing TEMPOB Areas

26840	Clean$	Shrink strobj in top of TEMPOB R1=addr of length field A.A=new end address Uses A.W B.A C.W D.A D0 D1
26847	Clean$R0	R1=R0+5 <see>Clean$
26721	Shrink$	Shrinks a strobj R0.A=->$ D0=end of $ Uses A.W B.A C.W D.10 D0 D1
26990	Stretch$	Expands a strobj R0.A=->$ D0=end of $ Uses A.W B.A C.10 D.10 D0 D1 aka: SIZEPLUS

8.5.5 CRC Routines

05981	DoCRC	Calculates the CRC of A.A nibs at D0. Returns CRC in A.A Uses C.W P Turns interrupts off and on
0597E	DoCRCc	D0=C <see>DoCRC
266B8	CKLBCRC	Check CRC of library at D0 CC: Ok CS: CRC is wrong Uses A.A C.W D0 P Disables and re-enables interrupts

8.5.6 Working with Memory

26C53	CompareACbBytes	Compares A.B=C.B bytes at D0 and D1 CC: Equal CS: Not equal D0/D1 always point past the end Uses A.M A.A C.M C.B P
2690B	INV.ZONE	Inverts (bitwise NOT) C.A nibbles at D0 Uses A.W C.A P
0675C	WIPEOUT	Zeroes C.A nibbles at D1 Uses A.W C.A P
269EB	WIPESPACE	Inits C.A nibbles at D1 with spaces (#20h) see <see>WIPEOUT

8.5.7 Other Routines

26808	aBZU	Decompress a BZ-compressed string D0->compressed D1->room to decompress to Uses A-D R0-R2

083D1	GETRRP	Returns the RRP in which the object at A.A lies. If ob is SysRRP, returns CS and leaves A.A unchanged; else CC and A.A->RRP B.A->RAM-WORD Uses A.A B.A C.A D.A D0 An RRP is a directory, the returned address points to the last-object-offset inside the directory. The SysRRP is the same as HOME.
26C68	RclAssembly	Recalls an object from the current directory D1->Name (ID etc) Returns object at D0 Uses A.W B.A C.W D.A D0 D1 ST P

8.6 Bank Switching

26BB9	ACCESSBank0	P=0: Switch to bank 0 P=1: Switch back Uses D0 C.A P
26BC0	ACCESSBank1	Bank 1, see <see>ACCESSBank0
26BC7	ACCESSBank2	Bank 2, see <see>ACCESSBank0
26BCE	ACCESSBank3	Bank 3, see <see>ACCESSBank0
26BD5	ACCESSBank4	Bank 4, see <see>ACCESSBank0
26BDC	ACCESSBank5	Bank 5, see <see>ACCESSBank0
26BE3	ACCESSBank6	Bank 6, see <see>ACCESSBank0
26BEA	ACCESSBank7	Bank 7, see <see>ACCESSBank0
26BF1	ACCESSBank8	Bank 8, see <see>ACCESSBank0
26BF8	ACCESSBank9	Bank 9, see <see>ACCESSBank0
26BFF	ACCESSBank10	Bank 10, see <see>ACCESSBank0
26C06	ACCESSBank11	Bank 11, see <see>ACCESSBank0
26C0D	ACCESSBank12	Bank 12, see <see>ACCESSBank0
26C14	ACCESSBank13	Bank 13, see <see>ACCESSBank0
26C1B	ACCESSBank14	Bank 14, see <see>ACCESSBank0
26C22	ACCESSBank15	Bank 15, see <see>ACCESSBank0

8.7 Memory Addresses

0010B	ANNCTRL	Annunciator control [LA4 LA3 LA2 LA1] (alarm alpha -> <-)

00104	CRC	4 nibbles for CRC. Every memory fetch updates CRC.
00137	TIMER1	1 nibble timer decremented 16 times/s
00138	TIMER2	8 nibble timer decremented 8192 times/s

8.8 Display

266B1	$5x7	(D.A B.A C.A D0 D1 -->) Displays string body at D1 in grob at D0 C.A = chars B.A = xlocation D.A = row length in nibbles -> D1 = addr after $ D0 = location of next char D.A = row length
2677C	D0->Row1	(--> D0) Gets addr of current display
26783	D0->Sft1	(--> D0) Gets address of menu grob
26A38	DISP_DEC	Displays hex in C.A as dec D0->GROB Uses A.6 B.W C.W CRY RSTK2 ST see <see>MINI_DISP_AWP If C.A > #99999h, it displays —— instead of the actual number
2679F	DispOn	Turns display on <see>Dispoff
26798	DispOff	Turns display off <see>Dispon
26894	GET_HEADER	<see>GET_HEADERTYPE and inits ST9 (normal/minifont)
2689B	GET_HEADERTYPE	Returns the header type in A.A Uses D0 The header type is the header height in pixels, including the black separator line
2687F	GET_@FONTE	Returns the address of the system font in A.A Currently LA 84D82 RTN
268A2	GET_HFONTE	Returns the heigth of the system font in A.A, uses D0
268A9	GET_HFONTECMD	Returns the heigth of the command line font, uses D0 -> A.A=height ST9=normal/minifont
268B0	GET_HFONTESTK	Returns the height of the stack font, uses D0 -> A.A=height ST9=normal/minifont

268B7	GET_HFONTESTKD1C	Returns the height of the stack font, uses D1 -> C.A=height ST9=normal/minifont
2674B	makegrob	R0.A = x, R1.A = y --> D0 = body Makes a grob of size x,y Prolog is in D0-20
26927	MINI_DISP	Display string in minifont D1->string D0->GROB C.A=chrs ST11=normal/inverted Only works on a 34 nibble wide screen, at a nibble aligned position Advances D0 and D1 to next character
2692E	MINI_DISP_AWP	Display A.WP in minifont D0->GROB, ST11=normal/inverted ST10=show/hide starting zeros Uses A.A B.W C.W CRY RSTK2
2693C	MINI_DISP_VAL	Display C.A digits of B.W in minifont, D0->GROB ST/Uses see <see>MINI_DISP_AWP
26974	SCREEN.MARGIN	ST9=0 -> C.A=00016 C.A is the number of characters which can be displayed with MINI_DISP (ST9=1) or $5x7 (ST9=0)
2696D	SCREEN.MARGIN2	Zeroes R0.A then does <see>SCREEN.MARGIN Uses R0.W

269AC	STYLE.MINIFONT	Changes minifont character data in A.6, uses P
		ST1=1 -> italic
		ST2=2 -> underline
		ST3=3 -> invert

26760	w->W	Calculates GROB width A.A=width in pixels -> A.A=width in nibbles Basically the same as 8 / CEIL 2 * since the width must be an even number of nibbles

8.9 Graphical Toolbox

26B7A	Arrows	Draws arrows to signal that further scrolling is possible D0->GROB ST4-7=arrows: 4=up 5=down 6=left 7=right ST9=normal/minifont Uses D1 A.A B.A C.A D.A ST0-7 P RSTK2 ST9 actually selects big or small arrow
26AB6	aCircleB	Draws black circle on GROB at D0 A.A = cx, B.A = cy, C.A = r Uses: RSTK2 D0 D1 R3.A R4.A A.S C.S
26AC4	aCircleG1	Draws light gray circle. <see>aCircleB
26ACB	aCircleG2	Draws dark gray circle. <see>aCircleB
26ABD	aCircleW	Draws white circle. <see>aCircleB
26AD2	aCircleXor	Inverts circle. <see>aCircleB
26B0A	aDistance	C.A=sqrt(A.A^2+B.A^2) Uses A.6 B.6 C.6 D.6 CRY SB P
26B34	aFBoxB	Draws a black filled box D0->GROB A.A=x1 B.A=y1 C.A=x2 D.A=y2 Uses RSTK2 A.W B.W C.W D.A D.S D0 D1 R3.A R4.A
26B42	aFBoxG1	Draws a light gray filled box <see>aFBoxB
26B49	aFBoxG2	Draws a dark gray filled box <see>aFBoxB
26B3B	aFBoxW	Draws a white filled box <see>aFBoxB
26B50	aFBoxXor	Inverts a filled box <see>aFBoxB
26AF5	aGrey?	Returns info about GROB at D0 ST0: 0=B&W 1=Gray R4.A= Plane len R3.A= Line len
26AFC	aGNeg	Inverts GROB at D0 Uses RSTK2 A.W B.A C.A D0 R3.A R4.A
26B57	aLBoxB	Draws a black rectangle D0->GROB A.A=x1 B.A=x2 C.A=y1 D.A=y2 Uses same as <see>aFBoxB
26B65	aLBoxG1	Draws light gray rectangle <see>aLBoxB
26B6C	aLBoxG2	Draws dark gray rectangle <see>aLBoxB

26B5E	aLBoxW	Draws white rectangle `<see>aLBoxB`
26B73	aLBoxXor	Inverts a rectangle `<see>aLBoxB`
26A93	aLineB	Draws black line on GROB at D0 A.A=x1, B.A=x2, C.A=y1, D.A=y2 Uses: RSTK2 D0 D1 R3.A R4.A A.A A.S B.A B.S C D.A
26AA1	aLineG1	Draws light gray line. `<see>aLineB`
26AA8	aLineG2	Draws dark gray line. `<see>aLineB`
26A9A	aLineW	Draws white line. `<see>aLineB`
26AAF	aLineXor	Inverts a line. `<see>aLineB`
26B18	aPixonB	Draws black pixel on GROB at D0 A.A = x, B.A = y Uses RSTK2 C.W D0 D1 R3.A R4.A
26B1F	aPixonG1	Draws light gray pixel. `<see>aPixonB`
26B26	aPixonG2	Draws dark gray pixel. `<see>aPixonB`
26B11	aPixonW	Draws white pixel. `<see>aPixonB`
26B2D	aPixonXor	Inverts pixel. `<see>aPixonB`
26B03	aScroolVGrob	Scroll GROB at D0 R0.A=h R1.A=Ys R2.A=Yd R3.A=X R4.A=w Uses A.A B.A B.S C.W D.A D.S RSTK2 R3.A R4.A D0 D1
26AE0	aSubReplGor	
26AE7	aSubReplGxor	
26AD9	aSubReplRepl	

8.10 Popping and Pushing

8.10.1 Pointers

03249	DropLoop	Pop stack, Loop
34202	4DropLoop	Pop 4, Loop
03672	GPOverWrALp	`<see>GETPTR` , OverWr A, Loop
0366F	GPOverWrR0Lp	`<see>GETPTR` , OverWr R0, Loop
266E2	GPPushA	`<see>GETPTR` , Push A, Clear Carry
268EF	GPPushALp	`<see>GETPTR` , Push A, Loop
268E8	GPPushR0Lp	`<see>GETPTR` , Push R0, Loop
26705	PopASavptr	Pop to A.A, `<see>SAVPTR`
2670C	PopSavptr	Pop `<see>SAVPTR`

03A86	PUSHA	Push A, Loop

8.10.2 TRUE and FALSE

266CD	GETPTRFALSE	<see>GETPTR , Do FALSE
266D4	GETPTRTRUE	<see>GETPTR , Do TRUE
35213	GPOverWrFLp	<see>GETPTR , OverWr FALSE, Loop
351F3	GPOverWrTLp	<see>GETPTR , OverWr TRUE, Loop
351F0	GPOverWrT/FL	<see>GETPTR , OverWr TRUE/FALSE, Loop
3524F	GPPushFLoop	<see>GETPTR , Push FALSE, Loop
35236	GPPushTLoop	<see>GETPTR , Push TRUE, Loop
35233	GPPushT/FLp	<see>GETPTR , Push TRUE/FALSE, Loop
3521D	OverWrFLoop	OverWr FALSE, Loop
351FD	OverWrTLoop	OverWr TRUE, Loop
3521A	OverWrT/FLp	OverWr TRUE/FALSE, Loop
34A68	popflag	Pop to A.A, if TRUE then set carry
35259	PushFLoop	Push FALSE, Loop
3523D	PushF/TLoop	Push FALSE (CRY)/TRUE, Loop
35240	PushTLoop	Push TRUE, Loop
35256	PushT/FLoop	Push TRUE (CRY)/FALSE, Loop aka: PushT/F

8.10.3 System Binary Integers (BINT)

06641	POP#	Pop # to A.A
03F5D	POP2#	(#1 #2 -->) Pop #1 to A.A and #2 to C.A
06537	PUSH#	<see>GETPTR , Push R0 as #
03DC7	#PUSHA-	<see>SAVPTR , R0=A, <see>PUSH# , Loop
06529	PUSH2#	<see>GETPTR , Push R0 & R1 as #
0357F	PUSH#LOOP	<see>GETPTR , Push R0 as #, Loop
0357C	PUSH#ALOOP	<see>GETPTR , Push A as #, Loop
03F14	Push2#Loop	<see>GETPTR , Push R0 & R1 as #, Loop
35812	Push2#aLoop	<see>GETPTR , Push R0 & A as #, Loop
036F7	Push#TLoop	<see>GETPTR , Push R0 as #, Do TRUE

283A3	Push#FLoop	<see>GETPTR , Push R0 as #, Do FALSE

8.10.4 HXS and ZINTs

266FE	PUSHhxs	Push A.WP as hxs
0596D	PUSHhxsLoop	Push A.WP as hxs, Loop
26951	PUSHzint	Push A.WP as ZINT
26958	PUSHzintLoop	Push A.WP as ZINT, Loop

8.10.5 Real and Complex Numbers

2F62C	POP1%SPLITA	(%pop -> x) Pop %, convert to %%, <see>SAVPTR
2F636	POP1%	(%pop -> A) Pop %, <see>SAVPTR
2F65E	POP2%	(%pop1 %pop2 -> A,C) Pop 2 reals, <see>SAVPTR
2F7E4	PUSH%	(A -> %push) Push A as %, <see>GETPTR
2F899	PUSH%LOOP	(A -> %push) Push A as %, <see>GETPTRLOOP
26A62	POPC%	(C%pop -> A:C) Pop C% (<see>SETDEC)
26A70	POPC%%	(C%%pop -> A:B C:D) Pop C%% (<see>SETDEC)
26A69	PUSHC%	(A:C -> C%push) Push C%
26A77	PUSHC%%	(A:B:C:D -> C%%push) Push C%%

8.11 Keyboard Handling

2A4AA	ATTNchk	ATTN exit check with restoreiram

2678A	Debounce	Scans keyboard until no more instabilities detected returns a map of the pressed keys in A.W 48G[X+] Keymap nibbles: (Nibble: [Bit1 Bit2 Bit3 Bit4]) 0: [ON + SPC .] 1: [0 ' - 3] 2: [2 1 A RS] 3: [* 6 5 4] 4: [MTH LS / 9] 5: [8 7 SIN alpha] 6: [BackSp DEL EEX +/-] 7: [ENTER 1/x y^x SQRT] 8: [TAN COS right down] 9: [left EVAL STO NXT] A: [up VAR CST PRG] B: [F E D C] C: [B none none none] 49G Keymap nibbles: 0: [ON RS LS alpha] 1: 2: [right down left up] 3: 4: [A B C D] 5: [E F none APPS] 6: 7: [EEX y^x HIST MODE] 8: [0 1 4 7] 9: [+/- SQRT CAT TOOL] A: [. 2 5 8] B: [1/x SIN EQW VAR] C: [SPC 3 6 9] D: [X COS SYMB STO] E: [ENTER + - *] F: [/ TAN BackSp NXT]
04999	KeyInBuff?	Carry if true
267C2	OnKeyDown?	Carry if true
267C9	OnKeyStable?	Carry if true
267A6	Flush	Flushes key buffer.
267AD	FlushAttn	Flushes attn counter.
04840	POPKEY	(-> C.A) Sets carry if buffer is empty.Else returns key in C.B (and in @KEYSTORE) Uses: A.S C.S C.A D1 (sets P=0)

267DE	SrvcKbdAB	(A.W ->) Sets KEYSTATE and KEYBUFFER
26D1E	(ThisKeyDn?)	CS if key in A.B is down Uses: A.A C.A D1 P OR
26D17	(ThisKeyDnCb?)	A=C.B <see>ThisKeyDn?

8.12 Various ML Entries

26E60	ASRW5	ASR.W 5 times
26E71	ASLW5	ASL.W 5 times
313C8	CCSB1	Uses D.S to set SB, clears carry
26832	CHANGE_FLAG	Change ST flag # A.B (1-4) If A.B > 10, A.B-11 is stored into R0.B. Clears carry if ok See <see>CHANGE_FLAG2
26839	CHANGE_FLAG2	Change ST flag # A.B (1-4) Does some strange magic if A.B > 10. Sets ST7
267EC	clkspd	Measure CPU clock speed Interrupts off on entry and exit -> A.A=spd/16 B.A=loops/16s Uses C.A D0 P CRY
26E82	CSRW5	CSR.W 5 times
26E93	CSLW5	CSL.W 5 times
04292	DeepSleep	Puts calc into "deep sleep" Low power mode, display off Wakeup on ON key or interrupt
266F7	GetStrLenStk	Pop $ -> C.A = length, D1 = body
266F0	GetStrLenC	D1 = C, <see>GetStrLen

266E9	GetStrLen	D1=$ -> C.A = length,
		D1 = body

268D3	GetStrLenL	D1=$ -> C.A = length in chars
267F3	makebeep	C = msec, D = Hz Checks BEEP flag.
04929	liteslp	Puts calc into "lite sleep" Low power mode with display on Wakeup on any key or interrupt

8.13 Debugging

2685C	DBUG	Displays the contents of all registers. Uses one RSTK level and #8190C to save them. <see>DBUG.TOUCHE
26863	DBUG.TOUCHE	<see>DBUG then freezes display until keypress

8.14 Object Types

029E8	DOARRY	Array prologue 5 size 5 prologue of objects 5 # of dimensions 5n dimensions .. objects (content only)
02B62	DOBAK	Backup prologue 5 size 2 # of chars in name .. name .. object 5 DOBINT 5 CRC Apparently unused on the 49
02911	DOBINT	BINT prologue 5 number (hex)
029BF	DOCHAR	Character prologue 2 character
02977	DOCMP	Complex number prologue 3 real exponent 12 real mantissa 1 real sign 3 complex exponent 12 complex mantissa 1 complex sign
02DCC	DOCODE	Code prologue 5 length .. machine code
02D9D	DOCOL	Secondary prologue .. objects 5 SEMI
02A2C	DOCSTR	String prologue 5 length .. characters

0299D	DOECMP	Long complex prologue 5 real exponent 15 real mantissa 1 real sign 5 complex exponent 15 complex mantissa 1 complex sign
02955	DOEREL	Long real prologue 5 exponent 15 mantissa 1 sign
02ADA	DOEXT	Unit object prologue .. object (usually a real) .. unit 5 SEMI
026AC	DOFLASHP	Flash pointer prologue 3 flash bank # 4 command #
02B1E	DOGROB	GROB prologue 5 size 5 height 5 width
02A4E	DOHSTR	HXS prologue 5 length .. hex digits, reverse order aka: DOHXS
02E48	DOIDNT	Global name (ID) prologue 2 # of characters .. characters
02614	DOINT	ZINT prologue 5 length .. BCD digits, reverse order 1 sign
02E6D	DOLAM	Local name (LAM) prologue see <see>DOIDNT
02A0A	DOLNKARRY	Linked array prologue Not used by the system.

02B40	DOLIB	Library prologue 5 size 2 # of characters .. name 2 # of characters (unless 0) 3 library ID 5 hash table offset 5 message table offset 5 link table offset 5 config object offset .. contents 4 CRC ; XLIBs: 1 or 3: kind 3 library ID 3 command ID .. object -- <REF>TEXT:Libraries
02A74	DOLIST	List prologue see <see>DOCOL
02686	DOMATRIX	Matrix prologue .. objects 5 SEMI Nested DOMATRIX objects build a multi-dimensional matrix
02933	DOREAL	Real number prologue 3 exponent 12 mantissa 1 sign
02E92	DOROMP	XLIB prologue 3 library ID 3 command #

02A96	DORRP	Directory prologue Home directory: 3 # of attached libs n*[3 library ID 5 address of hash table 5 address of message table] 5 offset of last object *[5 offset to previous object 00000 for the first one 2 # of characters .. name of object 2 # of characters .. object] ; Subdirectories: 3 # of attached library 7FF if none 5 offset of last object .. same as above
02AB8	DOSYMB	Symbolic prologue .. objects 5 SEMI
02AFC	DOTAG	Tagged object prologue 2 # of chars in tag .. tag .. object
026D5	DOAPLET	
02B88	DOEXT0	
02BAA	DOEXT1	aka: DOACPTR
02BCC	DOEXT2	
02BEE	DOEXT3	
02C10	DOEXT4	
02660	DOLNGCMP	
0263A	DOLNGREAL	

9 RAM entries

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

9.1 RPL pointers

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

80E9B	AVMEM	Free mem / 5 (5)
806F8	DSKTOP	->Data stack
806F3	RSKTOP	->Return stack
8076B	INTRPPTR	->RPL runstream aka: OBUPSTART

9.2 Memory management pointers

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

9.3 Screen related

806D5	ADISP	->Stack grob
806E4	GDISP	->Blackboard grob
8229E	GROBSCR1	<see>SCREEN1 with GROB header
82B32	GROBSCR2	<see>SCREEN2 with GROB header
833C6	GROBSCR3	<see>SCREEN3 with GROB header
83C5A	GROBSCR4	<see>SCREEN4 with GROB header
844EE	GROBSCR5	<see>SCREEN5 with GROB header
8069C	GreyOn?	Zero if greyscale on (1) If this is set to zero the interrupt system will display in greyscale, by showing each of GreyScrN/GreySoftN for one screen refresh. Note that the entries for PrintLCD use the same memory area!
8069D	GreyScr1	->1st greyscale screen
806A7	GreyScr2	->2nd greyscale screen
806B1	GreyScr3	->3rd greyscale screen
806A2	GreySoft1	->1st greyscale menu
806AC	GreySoft2	->2nd greyscale menu

806B6	GreySoft4	->3rd greyscale menu
822B2	SCREEN1	Space for one screen (2176) aka: ECRAN
82B46	SCREEN2	<see>SCREEN1
833DA	SCREEN3	<see>SCREEN1
83C6E	SCREEN4	<see>SCREEN1
84502	SCREEN5	Extra screen used by <see>DBUG (2176)
806DA	VDISP	->Display grob aka: VDISP1, SYSUPSTART
806D0	VDISP2	->Menu grob
806DF	VDISP3	->Not displayed grob <see>VDISP

9.4 Annunciators

80F00	ANNUNCIATORS	Annunciator flags (2)

9.5 Save areas

805DB	INTRAM	Save area for the interrupt sys (16)
806C0	R1[A]save	Used by PrintLCD inside the interrupt system (5)
806BA	R2[A]save	<see>R1[A]save (5)
806BF	R2[S]save	<see>R1[A]save (1)
81269	SAUV_80702	Backup of <see>TEMPENV aka: SavTEMPENV
8126E	SAUV_80865	Backup of <see>FIRSTCHAR aka: SavFIRSTCHAR
818CF	SAUV_CHARS	Used by CHARS (31) aka: SavChars
8221D	SAUV_DIVERS	Free area (128) aka: SavMisc
81278	SAUV_MATRIX	Used by MTRW (40) aka: SavMatrix
818F3	SAUV_REGA	Used by <see>DBUG (5) aka: SavRegA
818F8	SAUV_REGB	Used by <see>DBUG (5) aka: SavRegB
818FD	SAUV_REGC	Used by <see>DBUG (5) aka: SavRegC
81902	SAUV_REGD	Used by <see>DBUG (5) aka: SavRegD

81907	SAUV_REGD1	Used by <see>DBUG (5) aka: SavRegD1
8190C	SAUV_REGISTR	Used by <see>DBUG (101) aka: SavRegisters
80EF0	SAVECLK	Save of CLKON state (1)
80FB7	SAVECROSS	cursor moves in plotting (10)
805F5	SAVE_A	<see>INTRAM (16)
80608	SAVE_B	<see>INTRAM (16)
805F0	SAVE_C[A]	<see>INTRAM (5)
806C5	SAVE_BO	Save BitOffset (1)
80618	SAVE_D	<see>INTRAM (16)
8063D	SAVE_D0	<see>INTRAM (5)
806C6	SAVE_LC	Save LineCount (2)
806C8	SAVE_LN	Save LineNibs (3)
805EB	SAVE_MODES	<see>INTRAM (5)
806CB	SAVE_OFFSET	Save Window Offset (5)
80638	SAVE_PC	<see>INTRAM (5)
80628	SAVE_R0	<see>INTRAM (16)
80605	SAVE_ST	<see>INTRAM (3)
8069C	Stk0save	RSTK0 used by PrintLCD inside the interrupt sys (5)
806A1	Stk1save	RSTK1 <see>Stk0save (5)
806A6	Stk2save	RSTK2 <see>Stk0save (5)
806AB	Stk3save	RSTK3 <see>Stk0save (5)
806B0	Stk4save	RSTK4 <see>Stk0save (5)
806B5	Stk5save	RSTK5 <see>Stk0save (5)

9.6 System and User Flags

80F12	FLAG_SYSTEM2	Metakernel system flags (16) For compatibility only.
80F32	FLAG_USER2	Metakernel system flags (16) Dito.
80F02	SystemFlags	128 System flags (16)
80F22	UserFlags	128 User Flags (16)

9.7 Internal System Flags

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

80EC0	SysNib1	ISysFlags 1
001C0	NoRolDA2MASK	DA2 can't be rolled up to become valid <see>SysNib1
002C0	AbbrStkMASK	Display obj types only <see>SysNib1
004C0	DA2bIsEdMASK	DA2b shows the edit line <see>SysNib1
008C0	IgnorAlmMASK	Ignore <see>ALARMSDUE in <see>GETKEY <see>SysNib1
80EC1	SysNib2	ISysFlags 2
001C1	ReqClkOnMASK	Flag for System Request of CLKON state <see>SysNib2
002C1	ServModeMASK	Server mode on <see>SysNib2
004C1	TrackMASK	New context needs to be compared with old <see>SysNib2
008C1	BadMenuMASK	Menu system corrupt <see>SysNib2
80EC2	SysNib3	ISysFlags 3
001C2	UNDOMASK	Automatic stack save <see>SysNib3
002C2	INSERTMASK	Insert/replace mode <see>SysNib3
004C2	ALGMASK	Algebraic entry mode <see>SysNib3
008C2	PRINTINGMASK	<see>SysNib3
80EC3	SysNib4	ISysFlags 4
001C3	DA2aTempMASK	DA2a temporarily valid <see>SysNib4
002C3	DA2bTempMASK	DA2b temporarily valid <see>SysNib4
004C3	DA3TempMASK	DA3 temporarily valid <see>SysNib4
008C3	RebuildMASK	Menu requires TOUCHTAB rebuild each time it is redisplayed <see>SysNib4
80EC4	SysNib5	ISysFlags 5
001C4	COMMANDMASK	CMD history enabled <see>SysNib5
002C4	BLINKMASK	Active Timer1 Int's <see>SysNib5
004C4	LOWERMASK	Lowercase keys <see>SysNib5
008C4	STKDCMASK	Decompilation for stack display (not editing) <see>SysNib5
80EC5	SysNib6	ISysFlags 6
001C5	Do1UserMASK	One-key user mode <see>SysNib6
002C5	ASuspOKMASK	Suspending current environment is allowed <see>SysNib6
004C5	BadPOLUIMASK	POL UI possibly corrupt <see>SysNib6
008C5	DA1TempMASK	DA1 temporarily valid <see>SysNib6

80EC6	SysNib7	ISysFlags 7
001C6	DA1ValidMASK	DA1 known to be valid <see>SysNib7
002C6	DA2aValdMASK	DA2a known to be valid <see>SysNib7
004C6	DA2bValdMASK	DA2b known to be valid <see>SysNib7
008C6	DA3ValidMASK	DA3 known to be valid <see>SysNib7
80EC7	SysNib8	ISysFlags 8
001C7	DA1NoChMASK	DA1 not changed <see>SysNib8
002C7	DA2aNoChMASK	DA2a not changed <see>SysNib8
004C7	DA2bNoChMASK	DA2b not changed <see>SysNib8
008C7	DA3NoChMASK	DA3 not changed <see>SysNib8
80EC8	SysNib9	ISysFlags 9
001C8	DA1BadMASK	DA1 invalid <see>SysNib9
002C8	DA2aBadMASK	DA2a invalid <see>SysNib9
004C8	DA2bBadMASK	DA2b invalid <see>SysNib9
008C8	DA3BadMASK	DA3 invalid <see>SysNib9
80EC9	SysNib10	ISysFlags 10 aka: EDITFLAG, EDITLFLAG
001C9	EDITLMASK	Edit line exists <see>SysNib10
002C9	NAppKeyMASK	Non-app keys allowed in POL <see>SysNib10
004C9	NUsrKeyMASK	Non-user keys allowed in USR mode <see>SysNib10
008C9	AppModeMASK	POL application running <see>SysNib10
80ECA	SysNib11	ISysFlags 11 aka: ParenModFLAG
001CA	ParenModMASK	Implicit parenthesized "/", "^", and "SQRT" in EQW <see>SysNib11
002CA	1PDCMASK	Partial DeCompile info will not be saved <see>SysNib11
004CA	NewEditLMASK	New one-line edit line has been created <see>SysNib11
008CA	DoStdKeyMASK	Do only standard keys <see>SysNib11
80ECB	SysNib12	ISysFlags 12
001CB	DispTimeMASK	Status bar clock may be displayed <see>SysNib12
002CB	NOP2MASK12	unused <see>SysNib12
004CB	CaseSensitiv	unused <see>SysNib12
008CB	SpeedMASK	Metakernel repeat speed <see>SysNib12
80ECC	SysNib13	ISysFlags 13
001CC	InApletMASK	Aplet running <see>SysNib13
002CC	SplitMASK	<see>SysNib13
004CC	RightMASK	<see>SysNib13

008CC	CurTknMASK	<see>SysNib13
80ECD	SizeMLDisp	aka: SysNib14
80ECE	SysNib15	ISysFlags 15
001CE	BadTOLUIMASK	TOL UI potentially corrupt <see>SysNib15 aka: NOP1MASK15
002CE	NoAlgProcess	EVAL-> will not create a list nor return NOVAL <see>SysNib15 aka: NOP2MASK15
004CE	InSimplyExpr	<see>SysNib15 aka: NOP4MASK15
008CE	DoCreateMenu	<see>SysNib15 aka: NOP8MASK15
80ECF	SysNib16	ISysFlags 16 (unused)
001CF	NOP1MASK16	<see>SysNib16
002CF	NOP2MASK16	<see>SysNib16
004CF	NOP4MASK16	<see>SysNib16
008CF	NOP8MASK16	<see>SysNib16
80ED0	SysNib17	ISysFlags 17 (unused)
001D0	NOP1MASK17	<see>SysNib17
002D0	NOP2MASK17	<see>SysNib17
004D0	NOP4MASK17	<see>SysNib17
008D0	NOP8MASK17	<see>SysNib17
80ED1	SysNib18	ISysFlags 18 (unused)
001D1	NOP1MASK18	<see>SysNib18
002D1	NOP2MASK18	<see>SysNib18
004D1	NOP4MASK18	<see>SysNib18
008D1	NOP8MASK18	<see>SysNib18
80ED2	SysNib19	ISysFlags 19 (unused)
001D2	NOP1MASK19	<see>SysNib19
002D2	NOP2MASK19	<see>SysNib19
004D2	NOP4MASK19	<see>SysNib19
008D2	NOP8MASK19	<see>SysNib19
80ED3	SysNib20	ISysFlags 20 (unused)
001D3	NOP1MASK20	<see>SysNib20
002D3	NOP2MASK20	<see>SysNib20
004D3	NOP4MASK20	<see>SysNib20
008D3	NOP8MASK20	<see>SysNib20

9.8 Warmstart log

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

9.9 Command line management

810B6	BEG	Absolute BEGIN in CommandLine (5)
810A2	BEGIN_REL	Relative BEGIN in CommandLine (5)
810AC	BEGX	X position of BEGIN (5)
81273	CHECK_TEXTE	Checksum of cmd line (5) aka: CheckCLE
8125F	CHECK_VAL	Backup of the size of the cmd line (5)
81264	CHECK_VAL2	Checksum of the key cmd line definition (5)
80F49	CR_COUNT	# of newlines in editline (5)
80F61	CURSOR	Cursor editline position (5) aka: CURSOREPOSN
80F6E	CURSORCHR	Char under Cursor (2)
80F70	CURSORGROB	Cursor Grob Data (40)
80F6B	CURSOROFFSET	Cursor position from left of screen (2) aka: CURSORPOSN
80F66	CURSORPART	Cursor display row (5) aka: CURSORROW
80F6D	CURSORSTATE	Show cursor/char underneath (1)
80F98	CURSORX	Pxl X-Coord of Cursor (5)
80F9D	CURSORY	Pxl Y-Coord of Cursor (5)
806FD	EDITLINE	->Command line
810BB	END	Absolute END in CmdLine
810B1	ENDX	Y Position of END
810A7	END_REL	Relative END in CmdLine
810C0	SizeCLScreen	Size of CmdLine screen aka: T_ECRAN

9.10 POL variables

80ED4	AppCount	# of nested POLs (2)
807DE	AppCursor	->App cursor sub-programs
807C0	AppDisplay	->App display object
807E3	AppDoKeyOb	->App DoKeyOb procedure for POL
807CF	AppError	->App error handler
807CA	AppExitCond	->App exit condition
807C5	AppKeys	->App key assignments
807D9	AppResume	->App resume procedure of POL
807D4	AppSuspend	->App suspend procedure of POL

9.11 Topic/TOL variables

8086A	TopicVar1	->generic topic var 1
8086F	TopicVar2	->generic topic var 2
80874	TopicVar3	->generic topic var 3
80879	TopicVar4	->generic topic var 4
8087E	TopicVar5	->generic topic var 5
80883	TopicVar6	->generic topic var 6
80888	TopicVar7	->generic topic var 7
8088D	TopicVar8	->generic topic var 8
80892	TopicVar9	->generic topic var 9
80897	TopicVar10	->generic topic var 10
8089C	TopicVar11	->generic topic var 11
808A1	TopicVar12	->generic topic var 12
808A6	TopicVar13	->generic topic var 13
808AB	TopicVar14	->generic topic var 14
808B0	TopicVar15	->generic topic var 15
808B5	TopicVar16	->generic topic var 16
808BA	TopicVar17	->generic topic var 17
808BF	TopicVar18	->generic topic var 18
808C4	TopicVar19	->generic topic var 19
808C9	TopicVar20	->generic topic var 20
808CE	TopicVar21	->generic topic var 21
808D3	TopicVar22	->generic topic var 22
808D8	TopicVar23	->generic topic var 23

808DD	TopicVar24	->generic topic var 24
808E2	TopicVar25	->generic topic var 25
808E7	TopicVar26	->generic topic var 26
808EC	TopicVar27	->generic topic var 27
808F1	TopicVar28	->generic topic var 28
808F6	TopicVar29	->generic topic var 29
808FB	TopicVar30	->generic topic var 30
80900	TopicVar31	->generic topic var 31
80905	TopicVar32	->generic topic var 32
8090A	TopicVar33	->generic topic var 33
8090F	TopicVar34	->generic topic var 34
80914	TopicVar35	->generic topic var 35
80919	TopicVar36	->generic topic var 36
8091E	TopicVar37	->generic topic var 37
80923	TopicVar38	->generic topic var 38
80928	TopicVar39	->generic topic var 39
8092D	TopicVar40	->generic topic var 40
80932	TopicVar41	->generic topic var 41
80937	TopicVar42	->generic topic var 42
8093C	TopicVar43	->generic topic var 43
80941	TopicVar44	->generic topic var 44
80946	TopicVar45	->generic topic var 45
8094B	TopicVar46	->generic topic var 46
80950	TopicVar47	->generic topic var 47
80955	TopicVar48	->generic topic var 48
8095A	TopicVar49	->generic topic var 49
8095F	TopicVar50	->generic topic var 50
80964	TopicVar51	->generic topic var 51
80969	TopicVar52	->generic topic var 52
8096E	TopicVar53	->generic topic var 53
80973	TopicVar54	->generic topic var 54
80978	TopicVar55	->generic topic var 55
8097D	TopicVar56	->generic topic var 56
80982	TopicVar57	->generic topic var 57
80987	TopicVar58	->generic topic var 58
8098C	TopicVar59	->generic topic var 59
80991	TopicVar60	->generic topic var 60
80996	TopicVar61	->generic topic var 61

8099B	TopicVar62	->generic topic var 62
809A0	TopicVar63	->generic topic var 63
809A5	TopicVar64	->generic topic var 64
809AA	TopicVar65	->generic topic var 65
809AF	TopicVar66	->generic topic var 66
809B4	TopicVar67	->generic topic var 67
809B9	TopicVar68	->generic topic var 68
809BE	TopicVar69	->generic topic var 69
809C3	TopicVar70	->generic topic var 70
809C8	TopicVar71	->generic topic var 71
809CD	TopicVar72	->generic topic var 72
809D2	TopicVar73	->generic topic var 73
809D7	TopicVar74	->generic topic var 74
809DC	TopicVar75	->generic topic var 75
809E1	TopicVar76	->generic topic var 76
809E6	TopicVar77	->generic topic var 77
809EB	TopicVar78	->generic topic var 78
809F0	TopicVar79	->generic topic var 79
809F5	TopicVar80	->generic topic var 80
809FA	TopicVar81	->generic topic var 81
809FF	TopicVar82	->generic topic var 82
80A04	TopicVar83	->generic topic var 83
80A09	TopicVar84	->generic topic var 84
80A0E	TopicVar85	->generic topic var 85
80A13	TopicVar86	->generic topic var 86
80A18	TopicVar87	->generic topic var 87
80A1D	TopicVar88	->generic topic var 88
80A22	TopicVar89	->generic topic var 89
80A27	TopicVar90	->generic topic var 90
80A2C	TopicVar91	->generic topic var 91
0005B	TopicVarNum	Number of TopicVars
80A31	TOLVar1	->TOL var 1
80A36	TOLVar2	->TOL var 2
80A3B	TOLVar3	->TOL var 3
80A40	TOLVar4	->TOL var 4
80A45	TOLVar5	->TOL var 5
80A4A	TOLVar6	->TOL var 6
80A4F	TOLVar7	->TOL var 7

80A54	TOLVar8	->TOL var 8
80A59	TOLVar9	->TOL var 9
80A5E	TOLVar10	->TOL var 10
80A63	TOLVar11	->TOL var 11
80A68	TOLVar12	->TOL var 12
80A6D	TOLVar13	->TOL var 13
80A72	TOLVar14	->TOL var 14
80A77	TOLVar15	->TOL var 15
80A7C	TOLVar16	->TOL var 16
80A81	TOLVar17	->TOL var 17
80A86	TOLVar18	->TOL var 18
80A8B	TOLVar19	->TOL var 19
80A90	TOLVar20	->TOL var 20
80A95	TOLVar21	->TOL var 21
80A9A	TOLVar22	->TOL var 22
80A9F	TOLVar23	->TOL var 23
80AA4	TOLVar24	->TOL var 24
80AA9	TOLVar25	->TOL var 25
80AAE	TOLVar26	->TOL var 26
80AB3	TOLVar27	->TOL var 27
80AB8	TOLVar28	->TOL var 28
80ABD	TOLVar29	->TOL var 29
80AC2	TOLVar30	->TOL var 30
80AC7	TOLVar31	->TOL var 31
80ACC	TOLVar32	->TOL var 32
80AD1	TOLVar33	->TOL var 33
80AD6	TOLVar34	->TOL var 34
80ADB	TOLVar35	->TOL var 35
80AE0	TOLVar36	->TOL var 36
80AE5	TOLVar37	->TOL var 37
80AEA	TOLVar38	->TOL var 38
80AEF	TOLVar39	->TOL var 39
80AF4	TOLVar40	->TOL var 40
80AF9	TOLVar41	->TOL var 41
80AFE	TOLVar42	->TOL var 42
80B03	TOLVar43	->TOL var 43
80B08	TOLVar44	->TOL var 44
80B0D	TOLVar45	->TOL var 45

80B12	TOLVar46	->TOL var 46
80B17	TOLVar47	->TOL var 47
80B1C	TOLVar48	->TOL var 48
80B21	TOLVar49	->TOL var 49
80B26	TOLVar50	->TOL var 50
80B2B	TOLVar51	->TOL var 51
80B30	TOLVar52	->TOL var 52
80B35	TOLVar53	->TOL var 53
80B3A	TOLVar54	->TOL var 54
80B3F	TOLVar55	->TOL var 55
80B44	TOLVar56	->TOL var 56
80B49	TOLVar57	->TOL var 57
80B4E	TOLVar58	->TOL var 58
80B53	TOLVar59	->TOL var 59
80B58	TOLVar60	->TOL var 60
80B5D	TOLVar61	->TOL var 61
80B62	TOLVar62	->TOL var 62
80B67	TOLVar63	->TOL var 63
80B6C	TOLVar64	->TOL var 64
80B71	TOLVar65	->TOL var 65
80B76	TOLVar66	->TOL var 66
80B7B	TOLVar67	->TOL var 67
80B80	TOLVar68	->TOL var 68
80B85	TOLVar69	->TOL var 69
80B8A	TOLVar70	->TOL var 70
80B8F	TOLVar71	->TOL var 71
80B94	TOLVar72	->TOL var 72
80B99	TOLVar73	->TOL var 73
80B9E	TOLVar74	->TOL var 74
80BA3	TOLVar75	->TOL var 75
80BA8	TOLVar76	->TOL var 76
80BAD	TOLVar77	->TOL var 77
80BB2	TOLVar78	->TOL var 78
80BB7	TOLVar79	->TOL var 79
80BBC	TOLVar80	->TOL var 80
80BC1	TOLVar81	->TOL var 81
80BC6	TOLVar82	->TOL var 82
80BCB	TOLVar83	->TOL var 83

80BD0	TOLVar84	->TOL var 84
80BD5	TOLVar85	->TOL var 85
80BDA	TOLVar86	->TOL var 86
80BDF	TOLVar87	->TOL var 87
80BE4	TOLVar88	->TOL var 88
80BE9	TOLVar89	->TOL var 89
80BEE	TOLVar90	->TOL var 90
80BF3	TOLVar91	->TOL var 91
80BF8	TOLVar92	->TOL var 92
80BFD	TOLVar93	->TOL var 93
80C02	TOLVar94	->TOL var 94
80C07	TOLVar95	->TOL var 95
80C0C	TOLVar96	->TOL var 96
80C11	TOLVar97	->TOL var 97
80C16	TOLVar98	->TOL var 98
80C1B	TOLVar99	->TOL var 99
80C20	TOLVar100	->TOL var 100
80C25	TOLVar101	->TOL var 101
80C2A	TOLVar102	->TOL var 102
80C2F	TOLVar103	->TOL var 103
80C34	TOLVar104	->TOL var 104
80C39	TOLVar105	->TOL var 105
80C3E	TOLVar106	->TOL var 106
80C43	TOLVar107	->TOL var 107
80C48	TOLVar108	->TOL var 108
80C4D	TOLVar109	->TOL var 109
80C52	TOLVar110	->TOL var 110
80C57	TOLVar111	->TOL var 111
80C5C	TOLVar112	->TOL var 112
80C61	TOLVar113	->TOL var 113
80C66	TOLVar114	->TOL var 114
80C6B	TOLVar115	->TOL var 115
80C70	TOLVar116	->TOL var 116
80C75	TOLVar117	->TOL var 117
80C7A	TOLVar118	->TOL var 118
80C7F	TOLVar119	->TOL var 119
80C84	TOLVar120	->TOL var 120
80C89	TOLVar121	->TOL var 121

80C8E	TOLVar122	->TOL var 122
80C93	TOLVar123	->TOL var 123
80C98	TOLVar124	->TOL var 124
80C9D	TOLVar125	->TOL var 125
80CA2	TOLVar126	->TOL var 126
80CA7	TOLVar127	->TOL var 127
80CAC	TOLVar128	->TOL var 128
80CB1	TOLVar129	->TOL var 129
80CB6	TOLVar130	->TOL var 130
80CBB	TOLVar131	->TOL var 131
80CC0	TOLVar132	->TOL var 132
80CC5	TOLVar133	->TOL var 133
80CCA	TOLVar134	->TOL var 134
80CCF	TOLVar135	->TOL var 135
80CD4	TOLVar136	->TOL var 136
80CD9	TOLVar137	->TOL var 137
80CDE	TOLVar138	->TOL var 138
80CE3	TOLVar139	->TOL var 139
80CE8	TOLVar140	->TOL var 140
80CED	TOLVar141	->TOL var 141
80CF2	TOLVar142	->TOL var 142
80CF7	TOLVar143	->TOL var 143
80CFC	TOLVar144	->TOL var 144
80D01	TOLVar145	->TOL var 145
80D06	TOLVar146	->TOL var 146
80D0B	TOLVar147	->TOL var 147
80D10	TOLVar148	->TOL var 148
80D15	TOLVar149	->TOL var 149
80D1A	TOLVar150	->TOL var 150
80D1F	TOLVar151	->TOL var 151
80D24	TOLVar152	->TOL var 152
80D29	TOLVar153	->TOL var 153
80D2E	TOLVar154	->TOL var 154
80D33	TOLVar155	->TOL var 155
80D38	TOLVar156	->TOL var 156
80D3D	TOLVar157	->TOL var 157
80D42	TOLVar158	->TOL var 158
80D47	TOLVar159	->TOL var 159

80D4C	TOLVar160	->TOL var 160
80D51	TOLVar161	->TOL var 161
80D56	TOLVar162	->TOL var 162
80D5B	TOLVar163	->TOL var 163
80D60	TOLVar164	->TOL var 164
80D65	TOLVar165	->TOL var 165
80D6A	TOLVar166	->TOL var 166
80D6F	TOLVar167	->TOL var 167
80D74	TOLVar168	->TOL var 168
80D79	TOLVar169	->TOL var 169
80D7E	TOLVar170	->TOL var 170
80D83	TOLVar171	->TOL var 171
80D88	TOLVar172	->TOL var 172
80D8D	TOLVar173	->TOL var 173
80D92	TOLVar174	->TOL var 174
80D97	TOLVar175	->TOL var 175
80D9C	TOLVar176	->TOL var 176
80DA1	TOLVar177	->TOL var 177
80DA6	TOLVar178	->TOL var 178
80DAB	TOLVar179	->TOL var 179
80DB0	TOLVar180	->TOL var 180
80DB5	TOLVar181	->TOL var 181
80DBA	TOLVar182	->TOL var 182
80DBF	TOLVar183	->TOL var 183
80DC4	TOLVar184	->TOL var 184
80DC9	TOLVar185	->TOL var 185
80DCE	TOLVar186	->TOL var 186
80DD3	TOLVar187	->TOL var 187
80DD8	TOLVar188	->TOL var 188
80DDD	TOLVar189	->TOL var 189
80DE2	TOLVar190	->TOL var 190
80DE7	TOLVar191	->TOL var 191
80DEC	TOLVar192	->TOL var 192
80DF1	TOLVar193	->TOL var 193
80DF6	TOLVar194	->TOL var 194
80DFB	TOLVar195	->TOL var 195
80E00	TOLVar196	->TOL var 196
80E05	TOLVar197	->TOL var 197

80E0A	TOLVar198	->TOL var 198
80E0F	TOLVar199	->TOL var 199
80E14	TOLVar200	->TOL var 200
80E19	TOLVar201	->TOL var 201
80E1E	TOLVar202	->TOL var 202
80E23	TOLVar203	->TOL var 203
80E28	TOLVar204	->TOL var 204
80E2D	TOLVar205	->TOL var 205
80E32	TOLVar206	->TOL var 206
80E37	TOLVar207	->TOL var 207
80E3C	TOLVar208	->TOL var 208
80E41	TOLVar209	->TOL var 209
80E46	TOLVar210	->TOL var 210
80E4B	TOLVar211	->TOL var 211
80E50	TOLVar212	->TOL var 212
80E55	TOLVar213	->TOL var 213
80E5A	TOLVar214	->TOL var 214
80E5F	TOLVar215	->TOL var 215
80E64	TOLVar216	->TOL var 216
000D8	TOLVarNum	number of TOLVars

9.12 User interrupts

8600D	UserInt1	->User interrupt routine 1 This interrupt handler is called *before* the normal one. Only D1, P, Hex/Dec, CRY, SB, C.W and A.W are saved at that point.
86017	UserInt1g	Copy of <see>UserInt1 If this address is not equal to the one in `UserInt1`, none of the two will be called.
86012	UserInt2	->User interrupt routine 2 This interrupt handler is called *after* the normal one, before RESTORECPU. All registers are still saved.
8601C	UserInt2g	Copy of <see>UserInt2 If this address is not equal to the one in `UserInt2`, none of the two will be called.

9.13 UART buffering

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

9.14 ROM Part Tables

8605E	FROMPTAB0_15	Bank switcher addresses (16*5)
860AE	FROMPTABPTR	-> <see>FROMPTAB0_15
8611D	ROMPTAB	Library table (3+n*16) Header: 3 number of libraries For each library: 3 library ID 5 address 5 switch routine (0 if none) 3 000 aka: RESRAMEND, FlashROMPTAB
860CC	FlashROMTAB2	Bank switcher addresses (16*5) sorted by physical bank number

9.15 Fonts

81971	ArryFont	Array of used fonts (1708) aka: @FONTE
84D82	FONTE_SYSTEM	Big system font (4626) aka: SystemFont
81098	FontHeight	Height of the current font (5) aka: H_FONTE
8201D	HashArryFont	Font hash table (512) aka: TAB_FONTE
812CF	MINI_FONT	Minifont (1536) aka: MiniFont
812C3	MINI_FONT.OBJ	<see>MINI_FONT with font header aka: MiniFontObj
812AA	NB_FONTE	Number of detected fonts (5) aka: NbFont

9.16 Constants

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

00008	IRAMHOMEmsn	MSN of the IRAM base address
0001D	LOCUPSIZE	Number of variables between <see>SYSUPSTART and <see>OBUPSTART
000F4	NBMAXFONT	Maximum number of fonts
0016F	OBUPSIZE	Number of variables between <see>OBUPSTART and <see>OBUPEND
00001	mEditLExists	aka: ParenModmask
0018C	SYSUPSIZE	<see>OBUPSIZE + <see>LOCUPSIZE

9.17 Other/Uncategorized

80FF1	ACCUM	(1)
8072A	ALARMS	->System Alarm List (5)
80EF1	ALARMSDUE	Flags Alarm Due (1)
80EAB	ATTNFLG	Counts ON presses (5)
800E6	AccessInit	Saved value of INITEN & sALLOWINTR (2)
86051	BounceTiming	Minimum time between 2 same key press for key validation (8)
80734	CALCCXT	->Calculator variables dir (5)
80000	CMOS	Quick RAM corrupt check (5) aka: HARDROMEND, RAMSTART
81001	COLCOUNT	Dot Cols on line (2)
80FF3	COLWIDTH	(2)
80524	CONFRAM	RAM configuration (7) Port1: 1 Status [r w s 0] 1 Size/Address Code Port2: 1 Status [r w s 0] 1 Size/Address Code where r=readable, w=writable, s=system RAM 2 #banks 1 ID
8052B	CONFTAB	RAM configuration with CRC (11) 4 nibbles for CRC 7 nibbles as in CONFRAM
8071B	CONTEXT	->Current dir
800EB	COVERsave	Save area for G/DoCovered (10)
800E8	COVERstate	Iram state before uncovering (3)
80076	TIMEOUTCLK	ScratchPad (4)
80655	CSPEED	CPU speed (16hz units) (5)
80FA2	CURRENTMENU	Menu ID of current menu (2)
80E69	CatalogCache	->CAT list
86059	CatalogEntry	->Last CAT item selected
80E6E	Clipboard	->Clipboard
80FFA	ClkOnNib	Clock display on/off (1)

85FBE	CplxX	Complex number used by plotter (37)
85FE3	CplxY	<see>CplxX
807E8	CtlAlarm	->Control alarm data
860BD	CurRAMBank1	Backup of current RAM view 1 (5)
860C2	CurRAMBank2	Backup of current RAM view 2 (5)
860C7	CurRAMBank3	Backup of current RAM view 3 (5)
860B3	CurROMBank1	Backup of current ROM view 1 (5)
860B8	CurROMBank2	Backup of current ROM view 2 (5)
80EDC	DEPTHSAVE	Saved user stack depth (5)
86008	DIGITS	Infinite precision digits (5)
8065B	DISABLE_KBD	Keyboard handshake (1) aka: HANDSHK
8068D	DISP1CTLg	Ghost for DISP1CTL (5)
80695	DISP2CTLg	Ghost for DISP2CTL (5)
80707	DOLPENV	->DO LOOP environments
80EF3	DOUSEALARM	Flags Deactivate Curr Alarm (1)
8064A	DREND	Display Refresh Hi Bound (5)
80645	DRSTART	Display Refresh Lo Bound (5)
80FCD	DcompWidth	String Decomp Width (2)
80FFD	DelayCt	REDEYE Print time/line (2)
80F42	ELEMENT	decompile obj depth counter (2)
80FF5	ENTRWISE	(1)
80EA5	ERROR	(5)
807BB	EXITMSG	->msg set by user in EXIT word
8102B	EqPtr	Points to Curr Eqn in EqList (5)
80F44	FIRSTCHAR	offset to 1st visible (5)
80EB0	FIRSTPROC	->StartupProc Secondary (5)
80FD1	FONTCOUNT	counter (3)
80FCF	FONTHEIGHT	font-height selector (1)
80FD0	FONTWIDTH	font-width selector (1)
8072F	FSTVGERPTR	aka: VSTACK
80085	FailTime	SelfTest Fail Time (Ticks) (13)
81009	FifoByteCt	Sum of FIFO Line Counts (2)
80E73	FindPattern	->Find Pattern address
80833	FlagMBox	->Flag mailbox
81082	FlashPtrBkp	Space to create a FPTR (12)
818EE	FreeRoom	DSKTOP-RSKTOP, used by SWAPMEM (5)
80FAD	GARBSCRATCH1	Saves 1 RSTK level in G.C. (5)

80FB2	GARBSCRATCH2	Saves counter in G.C. (5)
80FFF	GCOLCOUNT	Graphics #Cols (2)
8085B	GraphContext	->Graphic Context
8030E	GraphPrtHook	(11) aka: IRAMBEND
90000	HARDRAMEND	IRAM Home ends at #7FFFF Appears to be an obsolete constant from the 48G, where IRAM was only 32kB big and thus ranged from #80000 to #8FFFF. The description even seems to come from the 48S!
80798	HISTORY1	-> $ with the most recent CMD history entry
8079D	HISTORY2	->2nd entry <see>HISTORY1
807A2	HISTORY3	->3rd entry <see>HISTORY1
807A7	HISTORY4	->4th (oldest) entry <see>HISTORY1
80F59	HISTORYLEVEL	which stack level is next (1)
8000A	HOMEMASK	Home Size of RAM (mask) (5)
8000F	HRAMEND	M.S.N. of size of RAM chip (1)
80851	HStackPtr	->Highlight in stack
80856	HStackTop	->How many items on stack
810E8	HashCLE	Command line hash table (360) aka: TAB_CMD
8108E	HeaderHeight	Header size in lines (5) aka: T_HEADER
80847	HiLitePtr	->Highlight in window
8065A	INITEN	Warmstart Enable flag (1)
80669	INPUTSTREAM	Key Buffer (max 15 keys). (34) aka: KEYBUFFER
80523	IOCNIB	Saves IOC in OUTUART (1)
81006	IOCsave	Save of IOC before change (1)
80654	IOSAVE	Saves HiNib of ANNCTRL (1)
00219	IRAMBSIZE	Size of <see>IRAMBUFF
800F5	IRAMBUFF	Exec Buff (code under IRAM) (537)
80127	IRAMBUFF2	<see>IRAMBUFF +50
80005	IRAMMASK	IRAM Size Config Mask (5)
8064F	IREG	Saves Interrupt History (3)
80ED6	ITEM1LINES	# display lines currently (1)
80793	ITEM1STATE	->list of lists describing stack level 1
807B1	KERMERRM	->Kermit error message aka: PDCSYMB
80FCC	KERMMODE	Kermit Mode information (1)

80FEB	KEYLIST	(5)
80FF0	KEYLOCK	(1)
8065C	KEYSTATE	location of kbd state (16)
86037	KSTATEVGER	KeyState for Vger Keyboard From rammap.a: "(we didn't use the previous \ KEYSTATE to maintain the entry \ points)"
8082E	KeyOb	->Pending key-object
81030	KeyRomPtr0	RomPtr for KeyOb (11)
8103B	KeyRomPtr1	RomPtr for MenuKey 1 (11)
81046	KeyRomPtr2	RomPtr for MenuKey 2 (11)
81051	KeyRomPtr3	RomPtr for MenuKey 3 (11)
8105C	KeyRomPtr4	RomPtr for MenuKey 4 (11)
81067	KeyRomPtr5	RomPtr for MenuKey 5 (11)
81072	KeyRomPtr6	RomPtr for MenuKey 6 (11)
80EA0	LANGUAGE	(5)
80775	LASTARG	->1st argument saved in CK<n> aka: LASTARG1
8077A	LASTARG2	->2nd <see>LASTARG
8077F	LASTARG3	->3rd <see>LASTARG
80784	LASTARG4	->4th <see>LASTARG
80789	LASTARG5	->5th <see>LASTARG
80F5A	LASTARGCOUNT	# of args saved by CK<n> (1)
80F5B	LASTARGf	Flag #Args>3 (1)
80F5C	LASTERROR	Save area for error number (5)
80FDA	LASTOP	3-state encoding of operand/ unary/binary (1)
80829	LASTROMWDOB	->Last user-level ROM-WORD evaluated (set by CK<n>)
80FDB	LEFTTREE	(3)
8069A	LINECOUNTg	Ghost for LINECOUNT (2)
80692	LINENIBSg	Ghost for LINENIBS (3)
80EFF	LPD_HIST	Low Power Detect History (1)
80801	LabelDef	->How to make menu labels
8081A	LastContext	->RRP saved for CheckContext
86047	LastKey	Last key press (2)
86049	LastKeyTime	Last key press time (8)
807F2	LastMenuDef	->Last menu definition
8107D	LastMenuRow	(5)
8100B	LastPrntTime	Time (Upper 11 nibs) (11)
81007	LineByteCt	Line Byte Counter (2)

80077	LoBatTime	Flag periodic ((*)) updates (1)
80FA4	MENULEVEL	User-menu level (5)
807F7	MenuData	->Menu data for touch table
807ED	MenuDef	->Current menu definition
80824	MenuExitAct	->Menu exit action definition
8080B	MenuKeyLS	->Left-shift menu key handler
80806	MenuKeyNS	->No-shift menu key handler
80810	MenuKeyRS	->Right-shift menu key handler
81026	MenuRow	(5)
807FC	MenuRowAct	->Prev/Next action definition
81093	NB_LIGNE	Size of the stack's screen in lines (5) aka: StackHeight
80058	NEXTIRQ	Time at next Timer2 int. (13)
80EF4	NOALARMSRV	Flags Disable Alarm Service (1)
80FD4	NODECOUNT	expr-tree node count (3)
8073E	NOTESCXT	->"notes" directory (5)
80FD7	OBTREELEN	object length (3)
80FA9	OLDMENU	Saves previous menu number (2)
80642	SAVE_OR	aka: ORghost
80770	OSAVE	
80E7D	ObjectU1	->Updatable object 1
80E82	ObjectU2	->Updatable object 2
80E87	ObjectU3	->Updatable object 3
80E8C	ObjectU4	->Updatable object 4
80E91	ObjectU5	->Updatable object 5 aka: OBUPEND
80FAC	PADCOUNT	Indentation count for decomp (1)
80FC1	PADJSAVE1	Status save in PTRADJUST (1)
80FC2	PADJSAVE2	RSTK save in PTRADJUST (10)
807B6	PAINTTREE	->hxs of "textbook-mode" graphics
80FF6	PARENCOUNT	(2)
80FE1	PARENTTREE	(3)
80EF2	PASTDUE	Flags Past Due Alarm (1)
807AC	PDCHXS	->hxs map of outermost symbolic
81016	PFIFO	FIFO Buffer (16)
80739	PGMCXT	->programming dir (5)
8068B	POPPEDKEY	Last Key from POPKEY (2)

80536	PORT0EOS	(5)
8053B	PORT1EOS	(5)
80540	PORT2EOS	(5)
80FE4	PRECSTACK	Op Precedence textbook entry (7)
800E2	Port1CRC	CRC for Device in Port1 (4)
800E1	PortStat	Copy of CARDSTAT Nib (1)
8083D	ProgMBox	->Program mailbox
81003	PrtStatus	CPU Status Bits et al. (3)
80E96	RAMEND	->End of RAM aka: SYSNOUPSTART
8611C	RESRAMEND0	End of statically reserved RAM
80FDE	RIGHTTREE	(3)
80EE1	RNSEED	Random number seed (15)
80716	ROMPARTS	->RomParts Area
85F94	RealX	Real number used by plotter (21)
85FA9	RealY	<see>RealX
80E78	ReplacePatte	->Replace pattern
80815	ReviewKey	->Review-key definition
80652	SEMAPH	Saves control byte for IREG (2)
80F4E	STACKNUM	ref. number of 1st visible (5)
80720	STOPSIGN	(5)
80FF8	STRETCHCOUNT	(2)
812B4	SWITCH	Used by the Memory Manager (15)
800D4	SW_ETime	Stopwatch Elapsed Time Ticks (13)
800BE	SW_Image	"HH:MM:SS:ss" Stopwatch (22)
812A0	SizeLine	Size of one line of text aka: T_LIGNE
80078	StartTime	SelfTest Start Time (Ticks) (13)
80FAB	T1COUNT	Decremented by srvc_timer1 (1)
80702	TEMPENV	->LAM environments (5)
80092	TESTMSG	SelfTest Msg Buffer (44)
80065	TIMECRC	CRC CheckSum for NEXTIRQ (4)
80069	TIMExmit	Time at scheduled timeout (13) aka: TIMEOUT
80F53	TOPLINE	Editline-segment which appears first on the screen (5)
8070C	TOUCHTAB	(5)
8109D	TYPE_HEADER	Type of header (5)
8125A	T_BLOC	Size of a HashCLE block (5)

812A5	T_LARGEUR	Width of the current screen in nibbles (5) aka: WidthScreen
80842	Title	->Home Title
8081F	TrackAct	->Action when CONTEXT changes
80725	UserKeys	->User key assignments (5)
812AF	VERIF_CARD	
0000C	VGERPTRCT	
80ED7	VIEWLEVEL	stack element currently viewed (5)
80838	ViewMBox	->View mailbox
8084C	WindowPtr	->Item at bottom of window
80FFB	XmitSrcvTOut	XMIT/SRECV timeout (2)
80743	apletPTR	->current aplet (5)
80748	funcPTR	->current func instance (5)
86026	has_font_f_s	Tells if the Decompiler has found a special font character (2)
8078E	leeway	->hxs which will be GC'ed in a very-low-memory condition
86028	misc1_f_s	(5)
8602D	misc2_f_s	(5)
86032	misc3_f_s	(5)
86021	nb_line_f_s	Number of line created during decompilation (FSTR3) (5)
80766	otherPTR	->current "other" instance (5)
80752	paramPTR	->current param instance (5)
8074D	polarPTR	->current polar instance (5)
80757	seqPTR	->current sequence instance (5)
80761	solvePTR	->current solve instance (5)
8075C	statPTR	->current stat instance (5)

10 Miscellaneous Entries

10.1 Various Matrix operations

00F004	^algunwrap
06C003	^laDELROW
06E003	^laGPROW
06D003	^laINSROW
2F205	laMGET0

10.2 Undescribed Entry Points

38D83	x<STRUCT
3F11C	xCMDAPPLY
3D258	xDER
38C2C	xEVAL>
3D81D	xFCNAPPLY
3D47E	xINTEGRAL
38D2F	xNOEVAL>
38D94	xSTRUCT->
38D72	xSTRUCT>
3D605	xWHERE
2F390	xssgeneral
2F315	!#1+IF<dim-1
2F316	!#1-IF>0
263D2	!MATTRNnc
25F68	!REDIMTEMP
25F63	!REDIMUSER
31568	1/X15
37C06	>LASTRAM-WORD
25F9F	?ACCPTR>
26C37	ACCESSERAM1
26C3E	ACCESSERAM2
26B81	ACCESSID1
26B88	ACCESSID2
26B8F	ACCESSID3
26B96	ACCESSID4

26B9D	ACCESSID5
26BA4	ACCESSID6
26BAB	ACCESSID7
26C29	ACCESSIDn
26C30	ACCESSRAM0
315BB	ADDF
26CD8	addrADISP
26CDF	addrATTNFLG
2B7CC	addrClkOnNib
00A0E	addrKEYSTATE
26CE6	addrLINECNTg
01661	addrORghost
04E66	addrTEMPENV
2ACA9	addrTEMPTOP
26CED	addrVDISP
26CF4	addrVDISP2
2619D	addtics
2F179	AdjEdModes
047CF	adrDISABLE_K
047DD	adrKEYBUFFER
26CFB	adrTIMEOUTCLK
2680F	AFFICHE.REG
26816	AFFICHE.SBR
2681D	AFFICHEPIX.SBR
31123	aH>HMS
25E7A	ALARMxcp
25E7B	ALGeq?
000FF	allkeys
31066	aMODF
2EEEE	APPprompt1!
2F17A	APPprompt2
068004	^Arbo
25E7D	ATTNxcp
2676E	BITMAP
2F31E	BUILDKPACKET
2AA70	CASEVAL
0BE002	^ChangeFocus
26D10	(ChkGrHook)

2BF1C	CkEQUtil
2A7A7	CkSecoType
2684E	CleanVirtualStack
2F153	CLKADJ*
2EF68	ClrDouseAlm
319C1	CLRFRC
26736	clrtimeout
2BAB3	COLAthexFCN
26775	Coldstart
266BF	COMPCONFCRC
26AEE	ComputePixel
2F327	convertbase
2C393	COPYVAR
2673D	corner
25EA3	CRUNCHNoBlame
2597B	CtlAlarm!
25980	CtlAlarm@
25971	(CtlAlarm0)
25976	(CtlAlarm0?)
2EEFE	CURRENTMARK?
2658A	CURSOR+
26A31	D0=ALoop
2EEA6	DA2bTemp?
29EE9	DaDGNTc
2DEBB	DAY#
2DD27	Day>Date
00C007	^DEB.MATRIX
00D007	^DEB.MATRIXTYPE
29D6A	delimcase
2C0ED	derprod1
2C0A7	derquot
004007	^DIMS
25EBD	DispVarsUtil
25F16	DISP_LINE
31994	DIV2
25EC0	DoCAlarmKey
0AF002	^DoKeyCancel
0B5002	^DoKeyEdit

0B4002	^DoKeyOK
0AE002	^DoMKeyOK
25ECA	DoPlotMenu
2EECC	DOPRLCD
2DE4A	dowutil
2F32D	drax
2F32F	DropSysErr$
26062	DropSysObs
37258	DupAndThen
00003	DZP
2C121	easyabs
25ED1	Echo2Macros
039EF	ECUSER
2F1A9	EDITF
2EEEC	EDITPARTS
2F332	EQCURSOR?
2F1A1	ErrorHandled?
25ED0	EVALCRUNCH
2EF69	EvalParsed
27C33	ExitFcn
2F334	Extobcode
2F335	FcnUtilEnd
26C5A	FindInDir
2F337	FixRRP
2DCB5	FLOAT
26878	GET.FONT
314E4	GETAB0
314CA	GETAB1
26BB2	GetBankAccess
2DDD5	getBPOFF
31518	GETCD0
0BB002	^GetFieldVals
2EF6D	GetLastEdit
2F108	GETRHS
267B4	GetTimChk
267BB	GetTime++
268DA	GETX.VISIBLE
268E1	GETX.VISIBLE.STR

26886	GET_@TAB
2688D	GET_ATTRIBN.REAL
268BE	GET_NBLIGNE
268C5	GET_NBLIGNESTK
0C80B0	~gFldVal
2F341	GraphicExit
2608A	GsstFIN
25636	HISTON?
2563B	(HISTON)
0BC002	^IFEDispField
04B004	^IfTet
092DB	InitEnab
2F075	InitSysUI
268F6	INIT_AFFICHELIGNE
268FD	INIT_ AFFICHELIGNENORM
26912	InverseParcelle
00110	IOC
0011F	IRAM@
0011A	IRC
04E004	^KeyLookup
25F2A	Keyword?
2F351	LASTPT?
33A5D	(lbrac)
2F21C	Lift
2F353	LINECHANGE
2F354	List
05149	Loop
35AE2	MACRODCMP
2639B	MATATLOOP
376C1	matchob?Lp
0120E4	~MESRclEqn
26943	MiniFontCmd?
2694A	MiniFontStk?
2DE26	mpop1%
2C2CB	nCOLCTQUOTE
2AC72	need'case
26C45	NEWACCESSRAM
2F357	newBASE

2F0D5	NEWINDEP
2F358	NEWMARK
37702	nextpos
2F359	NEXTRRPOB
2F35A	NEXTSTEP
26201	nextsym'R
29E29	ngsizecase
257E2	NoIgnoreAlm
267FA	norecCSseq
2F35B	NUMSOLVE
2C044	nWHEREDER
2C039	nWHEREIFTE
2C04F	nWHEREINTG
2C05A	nWHERESUM
2C065	nWHEREWHERE
2F35C	OB>BAKcode
2F19B	OngoingText?
0020F	OUTCINRTN
351FA	OverWrF/TLp
35B46	PALPTRDCMP
02E0E7	~PCunpack
2B682	POLErrorTrap
3ABFD	preFACT
2F360	PREMARKON
028FC	PRLG
2F363	PtoR
2C37D	PTYPE>PINFO
31532	PUTAB0
00114	RBR
267D0	RCKBp
26C4C	RclCompareNames
26274	RCL_NB_AFF_LGN
26279	RCL_NB_AFF_LGNSTK
00111	RCS
25F6D	realPAcode
2F369	RECORDX&YC%
069004	^RENAME
2579A	REPLACE_MODE

313D3	RNDC[B]	
34FE6	Rom-Word?	
3A200	rpnXROOT	
26713	SAFESKIPOB	
0000F	sALLOWINTR	
34D51	SAVELAM	
267D7	SavPtrTime*	
00008	sBEG	
00004	sBPOFF	
26966	SCAN.FONTE	
26C61	ScanEveryObjects	
07661	SET	
25683	SetBadPOLUI	
26752	setflag	
2671A	SetISysFlag	
2F37C	SETLOOPENV	
2F25D	SETROMPART	
26759	settimeout	
2697B	SET_HEADER	
0D80B0	~sFldVal	
26982	Shrink$Any	
26989	Shrink$AnySafe	
26A4D	Shrink$List	
2AAE0	SimplifyExpression	
25EFA	SLEEPxcp	
00002	sNEGATE	aka: sFLUSH
2C2D6	SPLITWHERE	
317EE	SQRF	
26801	srvc_timer2	
261B1	stackitw	
2B74F	StartupProc	
2F066	STOAPPLDATA	
26997	STOFONT	
2699E	STOMINIFONT	
2628D	STO_ML_DISP_SIZE	
269A5	Stretch$Any	
00001	sTRUNC	
261B6	subpdcdptch	

2EFEC	symbn	
2EED9	SYMBNUMSOLVE	
2EE5E	SysErrorTrap	
2F1A3	SysErrorTrapAction	
2EE5F	SysErrorTrapConfirm	
08D66	SysPtr@	
26157	SystemLevel?	
00116	TBR	
00112	TCS	
26161	TIMEOUT?	
0012E	TIMERCTRL.1	
0012F	TIMERCTRL.2	
25F2D	TogInsertKey	
3125D	TST15	
25F05	TurnOffKey	
02F0E7	~UTTYPEEXT0?	
0110E7	~UTVUNS1Arg	
26C6F	ValidPortTag?	
25F0A	VLM	
2A4FC	WaitTbz0	
267E5	Warmstart	aka: norecPWLseq
26728	WindowXY	
31219	Y<=X	
255A6	ZoomX	
255AB	ZoomY	

11 Entries sorted by address

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

00001	sTRUNC
00001	mEditLExists
00001	ParenModmask
00002	sFLUSH
00002	sNEGATE
00003	DZP
00004	sBPOFF
00008	IRAMHOMEmsn
00008	sBEG
0000A	portnotaverr
0000C	VGERPTRCT
0000F	sALLOWINTR
0001D	LOCUPSIZE
0005B	TopicVarNum
000D8	TOLVarNum
000F4	NBMAXFONT
000FF	allkeys
00104	CRC
0010B	ANNCTRL
00110	IOC
00111	RCS
00112	TCS
00113	CRER
00114	RBR
00116	TBR
0011A	IRC
0011F	IRAM@
0012E	TIMERCTRL.1
0012F	TIMERCTRL.2
00137	TIMER1
00138	TIMER2
0016F	OBUPSIZE
0018C	SYSUPSIZE
001C0	NoRolDA2MASK
001C1	ReqClkOnMASK
001C2	UNDOMASK
001C3	DA2aTempMASK
001C4	COMMANDMASK
001C5	Do1UserMASK
001C6	DA1ValidMASK
001C7	DA1NoChMASK
001C8	DA1BadMASK
001C9	EDITLMASK
001CA	ParenModMASK
001CB	DispTimeMASK
001CC	InApletMASK
001CE	NOP1MASK15
001CE	BadTOLUIMASK
001CF	NOP1MASK16
001D0	NOP1MASK17
001D1	NOP1MASK18
001D2	NOP1MASK19
001D3	NOP1MASK20
00202	argtypeerr
00203	argvalerr
0020A	AINRTN
0020F	OUTCINRTN
00212	CINRTN
00219	IRAMBSIZE
002C0	AbbrStkMASK
002C1	ServModeMASK
002C2	INSERTMASK
002C3	DA2bTempMASK
002C4	BLINKMASK
002C5	ASuspOKMASK
002C6	DA2aValdMASK
002C7	DA2aNoChMASK
002C8	DA2aBadMASK
002C9	NAppKeyMASK
002CA	1PDCMASK
002CB	NOP2MASK12
002CC	SplitMASK
002CE	NOP2MASK15
002CE	NoAlgProcess
002CF	NOP2MASK16
002D0	NOP2MASK17
002D1	NOP2MASK18
002D2	NOP2MASK19
002D3	NOP2MASK20
00301	posunferr
00302	negunferr
00303	ofloerr
00305	infreserr
004C0	DA2bIsEdMASK
004C1	TrackMASK
004C2	ALGMASK
004C3	DA3TempMASK
004C4	LOWERMASK
004C5	BadPOLUIMASK
004C6	DA2bValdMASK
004C7	DA2bNoChMASK
004C8	DA2bBadMASK
004C9	NUsrKeyMASK
004CA	NewEditLMASK
004CB	CaseSensitiv
004CC	RightMASK

004CE	NOP4MASK15	01661	addrORghost	02E92	DOROMP
004CE	InSimplyExpr	02614	DOINT	02FD6	DoLam
004CF	NOP4MASK16	0263A	DOLNGREAL	02FEF	ROMSEC
004D0	NOP4MASK17	02660	DOLNGCMP	03019	SKIPOB
004D1	NOP4MASK18	02686	DOMATRIX	0312B	SEMI
004D2	NOP4MASK19	026AC	DOFLASHP	0314C	DEPTH
004D3	NOP4MASK20	026D5	DOAPLET	03188	DUP
008C0	IgnorAlmMASK	026FE	DOMINIFONT	031AC	2DUP
008C1	BadMenuMASK	028FC	PRLG	031D9	NDUP
008C2	PRINTINGMASK	02911	DOBINT	03223	SWAP
008C3	RebuildMASK	02933	DOREAL	03244	DROP
008C4	STKDCMASK	02955	DOEREL	03249	DropLoop
008C5	DA1TempMASK	02977	DOCMP	03258	2DROP
008C6	DA3ValidMASK	0299D	DOECMP	0326E	NDROP
008C7	DA3NoChMASK	029BF	DOCHAR	03295	ROT
008C8	DA3BadMASK	029E8	DOARRY	032C2	OVER
008C9	AppModeMASK	02A0A	DOLNKARRY	032E2	PICK
008CA	DoStdKeyMASK	02A2C	DOCSTR	03325	ROLL
008CB	SpeedMASK	02A4E	DOHSTR	0339E	UNROLL
008CC	CurTknMASK	02A4E	DOHXS	03442	MAKEARRY
008CE	NOP8MASK15	02A74	DOLIST	03562	ARSIZE
008CE	DoCreateMenu	02A96	DORRP	0357C	PUSH#ALOOP
008CF	NOP8MASK16	02AB8	DOSYMB	0357F	PUSH#LOOP
008D0	NOP8MASK17	02ADA	DOEXT	035A9	DIMLIMITS
008D1	NOP8MASK18	02AFC	DOTAG	0366F	GPOverWrR0Lp
008D2	NOP8MASK19	02B1E	DOGROB	03672	GPOverWrALp
008D3	NOP8MASK20	02B40	DOLIB	03685	ARRYEL?
00A03	intrptderr	02B62	DOBAK	03685	FINDELN
00A0E	addrKEYSTATE	02B88	DOEXT0	036F7	Push#TLoop
00B02	constuniterr	02BAA	DOEXT1	0371D	GETATELN
00C02	timeouterr	02BAA	DOACPTR	03826	#A8241
00C06	xferfailerr	02BCC	DOEXT2	03880	#102A8
00C0D	kermsendmsg	02BEE	DOEXT3	038DC	#E13A8
00C0E	kermrecvmsg	02C10	DOEXT4	03991	MUL#
00C10	kermpktmsg	02D9D	DOCOL	039BE	GETTEMP
00C13	prtparerr	02DCC	DOCODE	039EF	ECUSER
00C14	lowbaterr	02E48	DOIDNT	03A81	TRUE
01118	LowBat?	02E6D	DOLAM	03A86	PUSHA

03AC0	FALSE	03FE5	TYPEEXT	04ED1	ERRJMP
03ADA	XOR	03FEF	TYPEINT	04FAA	SETLBERR
03AF2	NOT	03FF9	TYPEMATRIX	04FB6	SETMEMERR
03B2E	EQ	041A7	TurnOff	04FBB	DOMEMERR
03B46	AND	041ED	DEEPSLEEP	04FC2	SETDIRRECUR
03B75	OR	0426A	ShowInvRomp	04FCE	SETLAMERR
03B97	EQUAL	04292	DeepSleep	04FDA	SETCORPORT
03C64	TYPE	04544	AlertStatus	04FE6	SETOBINUSE
03CA6	#0=	04575	Alert$	04FF2	SETPORTNOTAV
03CC7	#0<>	04708	CHECKKEY	04FFE	SETNOROOM
03CE4	#<	04714	GETTOUCH	0500A	SETXNONEXT
03D19	#=	047C7	REPKEY?	05016	SETROMPERR
03D4E	#<>	047CF	adrDISABLE_K	05023	Errjmp
03D83	#>	047DD	adrKEYBUFFER	05040	ATTNFLG@
03DBC	#+	04840	POPKEY	05068	ATTNFLGCLR
03DC7	#PUSHA-	048F9	ShowClk?	05089	CARCOMP
03DE0	#-	04912	LiteSlp	050ED	CAR$
03DEF	#1+	04929	liteslp	05143	GETPTRLOOP
03E0E	#1-	04999	KeyInBuff?	05149	Loop
03E2D	#2+	04A0B	GETPROC	05153	CDRCOMP
03E4E	#2-	04A41	GETDF	0516C	CDR$
03E6F	#2*	04A4C	SETDF	0518A	&HXS
03E8E	#2/	04A57	SETPROC	05193	&$
03EB1	#AND	04CE6	ERROR@	0521F	&COMP
03EC2	#*	04D0E	ERRORSTO	0525B	>H$
03EF7	#/	04D33	ERRORCLR	052C6	>HCOMP
03F14	Push2#Loop	04D3E	DROPNULL$	052EE	>T$
03F24	IntDiv	04D57	TWODROPNULL$	052FA	>TCOMP
03F5D	POP2#	04D64	GETTHEMESG	05331	COMPN
03F8B	TYPEREAL	04D87	JstGETTHEMSG	05445	::N
03F95	TYPECMP	04D87	JstGetTHEMESG	05459	{}N
03F9F	TYPELIST	04DD7	SPLITmsg	0546D	SYMBN
03FA9	TYPEIDNT	04E07	GETEXITMSG	05481	EXTN
03FB3	TYPECOL	04E37	EXITMSGSTO	054AF	INNERCOMP
03FBD	TYPESYMB	04E5E	ERRSET	0554C	DOGARBAGE
03FC7	TYPERRP	04E66	addrTEMPENV	05566	NULLHXS?
03FD1	TYPELAM	04EA4	ABORT	0556F	NULL$?
03FDB	TYPEEREL	04EB8	ERRTRAP	055B7	NULLCOMP?

055D5	NULLHXS	05F61	MEM	07012	R@
055DF	NULL$	0613E	GARBAGECOL	0701F	R>
055E9	NULL{}	064BD	TOTEMPOBADJ	070C3	RPITE
055F3	NULLSYMB	064D6	DOADJ1	070FD	RPIT
055FD	NULL::	064E2	DOADJ	0712A	?SKIP
05616	LENHXS	06529	PUSH2#	0712A	NOT_IT
05622	OVERLEN$	06537	PUSH#	0714D	SKIP
05636	LEN$	0657E	#61441	0715C	2SKIP
0567B	LENCOMP	065AA	GPMEMERR	0716B	IDUP
056B6	NTHELCOMP	065D9	PTRREFD?	071A2	BEGIN
05733	SUB$	065E5	REFERENCED?	071AB	AGAIN
05815	SUBHXS	06641	POP#	071C8	UNTIL
05821	SUBCOMP	06657	TOTEMPOB	071E5	REPEAT
05902	OSIZE	066B9	MOVEUP	071EE	WHILE
05944	OCRC	0670C	MOVEDOWN	07221	INDEX@
0596D	PUSHhxsLoop	0675C	WIPEOUT	07249	ISTOP@
0597E	DoCRCc	0679B	SAVPTR	07258	JINDEX@
05981	DoCRC	067D2	GETPTR	07264	JSTOP@
059CC	#>HXS	06806	ROOM	07270	INDEXSTO
05A03	HXS>#	06992	MOVERSD	07295	ISTOPSTO
05A51	CHR>#	069C5	MOVEDSU	072AD	JINDEXSTO
05A75	#>CHR	069F7	ADJMEM	072C2	JSTOPSTO
05AB3	CHANGETYPE	06A1D	MOVEDSD	07321	STOPLOOP
05AED	ID>LAM	06A53	MOVERSU	07334	LOOP
05B01	LAM>ID	06A8E	DIV5	073A5	+LOOP
05B15	$>ID	06AD8	CREATETEMP	073C3	ZERO_DO
05B79	MAKE$	06B3E	FREEINTEMP?	073CE	ONE_DO
05B7D	MAKE$N	06B4E	INTEMNOTREF?	073DB	#1+_ONE_DO
05BE9	ID>$	06BC2	NOTREF?	073F7	DO
05C27	%>C%	06DDE	>TOPTEMP	07497	ABND
05C72	%%>C%%	06E8E	NOP	074D0	BIND
05D2C	C%>%	06E97	'	074E4	DOBIND
05DBC	C%%>%%	06EEB	'R	075A5	GETLAM
05E81	>TAG	06F66	'REVAL	075E9	PUTLAM
05E9F	{}>TAG	06F8E	EVAL	07638	SETHASH
05EC9	TAG>	06F9F	>R	0764E	SETMESG
05F2E	ID>TAG	06FB7	RDROP	07661	SET
05F42	GARBAGE	06FD1	COLA	076AE	OFFSRRP

07709	TOSRRP	08D66	SysPtr@	255D3	FBoxW
0778D	ONSRRP?	08D82	STOPSIGN@	255D3	FBoxG1
077E4	MAKERRP	08D92	SYSCONTEXT	255D8	FBoxG2
078E9	FIRST@LAM	08D92	HOMEDIR	255DD	FBoxB
078F5	NTH@LAM	08DC4	SYSSTOPSIGN	255E2	FBoxXor
07943	@LAM	08DD4	SYSRRP?	255E7	LBoxW
0797B	@	08DF2	CREATERRP	255EC	LBoxG1
07C18	COMPILEID	08DF7	#7FF	255F1	LBoxG2
07D1B	STOLAM	08E33	#>TCOMP+1	255F6	LBoxB
07D27	STO	08ECE	#536A8	255FB	LBoxXor
07E50	#>ROMPTR	08F1F	#D6A8	25600	Do1User?
07E76	PTR>ROMPTR	0905F	BAK>OB	25605	SetDo1User
07E99	ROMPTR@	091B4	#2D541	2560A	ClrDo1User
07F86	ROMPART	092DB	InitEnab	25617	SetNUsrKeyOK
0803F	#414C1	09378	TRUESWAP	2561C	ClrNUsrKeyOK
08081	ROMPART>ADDR	0B954	RunInNewContext	25621	NonUsrKeyOK?
080BF	ROMPARTSIZE	25565	LineW	25636	HISTON?
080DA	NEXTROMPID	2556A	LineB	2563B	HISTON
08112	GETHASH	2556F	LineG1	2564D	SetNAppKeyOK
08130	GETMSG	25574	LineG2	25652	ClrNAppKeyOK
0813C	GETLINK	25579	LineXor	2565A	DoStdKeys?
08157	GETCONFIG	2557E	Sub	2565F	SetDoStdKeys
08199	ROMPARTNAME	25583	Repl	25664	ClrDoStdKeys
081D9	BAKNAME	25588	Gor	2566C	AppSuspOK?
081DE	LIB>#	2558D	Gxor	25671	SetAppSuspOK
081FB	ROMPTRDECOMP	25592	SubRepl	25676	ClrAppSuspOK
082E3	RAM-WORDNAME	25597	SubGor	25683	SetBadPOLUI
08309	MYRAMROMPAIR	2559C	SubGxor	25690	AppMode?
08326	LASTRAM-WORD	255A1	Grey?	25695	SetAppMode
08376	PREVRAM-WORD	255A6	ZoomX	2569A	ClrAppMode
083D1	GETRRP	255AB	ZoomY	256A2	UNDO_ON?
085D3	REPLACE	255B0	ScrollVGrob	256A7	UNDO_ON
08696	CREATE	255B5	Distance	256AC	UNDO_OFF
08C27	PURGE	255BA	PixonW	256BE	NOBLINK
08CCC	ROMPTR>#	255BF	PixonB	256EA	AlgEntry?
08D08	CONTEXT!	255C4	PixonG1	25719	SetAlgEntry
08D4A	STOPSIGN!	255C9	PixonG2	2571E	ClrAlgEntry
08D5A	CONTEXT@	255CE	PixonXor	25726	LOWERCASE?

2572B	SETLOWERCASE	258A4	MenuKeyLS@	25E6E	2DropBadKey
25730	CLRLOWERCASE	258AE	DoMenuKeyLS	25E6F	>Review$
25738	TOGLOWERCASE	258B3	MenuKeyRS!	25E70	?ATTNQUIT
2576D	CaseSensitive?	258B8	MenuKeyRS@	25E70	?ATTN_QUIT
25772	SetCaseSensitive	258C2	DoMenuKeyRS	25E71	?BlinkCursor
25777	ClrCaseSensitive	258C7	ReviewKey!	25E72	?CaseKeyDef
2577F	TOGGLE_I/R	258CC	ReviewKey@	25E73	?CaseRomptr@
25790	INSERT?	258D6	DoReview	25E74	?ClrAlg
25795	INSERT_MODE	258DB	TrackAct!	25E75	?ClrAlgSetPr
2579A	REPLACE_MODE	258E0	TrackAct@	25E76	?Key>UKeyOb
257A2	EditLExists?	258EA	DoTrack	25E77	?OKINALG
257BE	ClrNewEditL	258EF	MenuExitAct!	25E78	?PURGE_HERE
257E2	NoIgnoreAlm	258F4	MenuExitAct@	25E79	?STO_HERE
257F7	Track?	258FE	DoMenuExit	25E79	XEQSTOID
257FC	SetTrack	25903	LastMenuDef?	25E7A	ALARMxcp
25801	ClrTrack	25908	LastMenuDef!	25E7B	ALGeq?
25809	Rebuild?	2590D	LastMenuDef@	25E7C	AND$
2580E	SetRebuild	25917	LastContext!	25E7D	ATTNxcp
25813	ClrRebuild	2591C	LastContext@	25E7E	BLANKIT
2581B	BadMenu?	2593F	KeyOb0	25E7F	Box/StdLabel
25820	SetBadMenu	25944	KeyOb0?	25E80	Box/StdLbl:
25825	ClrBadMenu	25949	KeyOb!	25E81	C%1/
2582D	MenuDef?	2594E	KeyOb@	25E82	C%>%%
25840	MenuDef!	25958	UserKeys0	25E83	C%>%%SWAP
25845	MenuDef@	2595D	UserKeys0?	25E84	C%ABS
2584F	MenuData!	25962	UserKeys!	25E85	C%ACOS
25854	MenuData@	25967	GetUserKeys	25E86	C%ACOSH
2585E	GetMenuData	25971	CtlAlarm0	25E87	C%ALOG
25863	MenuRowAct!	25976	CtlAlarm0?	25E88	C%ARG
25868	MenuRowAct@	2597B	CtlAlarm!	25E89	C%ASIN
25872	DoMenuRowAct	25980	CtlAlarm@	25E8A	C%ASINH
25877	LabelDef!	25E67	!*triand	25E8B	C%ATAN
2587C	LabelDef@	25E68	!*trior	25E8C	C%ATANH
25886	DoLabel	25E69	%+SWAP	25E8D	C%COS
2588B	MenuKeyNS!	25E6A	'DoBadKey	25E8E	C%COSH
25890	MenuKeyNS@	25E6B	'DoBadKeyT	25E8F	C%C^C
2589A	DoMenuKeyNS	25E6C	1A/LockA	25E90	C%C^R
2589F	MenuKeyLS!	25E6D	1stkdecomp$w	25E91	C%EXP

25E92	C%LN	25EBA	DPRADIX?	25EE0	InitMenu
25E93	C%LOG	25EBB	DUPGROBDIM	25EE1	InitMenu%
25E94	C%R^C	25EBC	Disp5x7	25EE2	InitTrack:
25E95	C%SGN	25EBD	DispVarsUtil	25EE3	KEYINBUFFER?
25E96	C%SIN	25EBE	Do1st/2nd+:	25EE4	KeepUnit
25E97	C%SINH	25EBF	DoBadKey	25EE5	Key>StdKeyOb
25E98	C%SQRT	25EC0	DoCAlarmKey	25EE6	Key>U/SKeyOb
25E99	C%TAN	25EC1	DoDelim	25EE7	LastNonNull
25E9A	C%TANH	25EC2	DoDelims	25EE8	LoadTouchTbl
25E9B	C>Im%	25EC3	DoFirstRow	25EE9	LockAlpha
25E9C	C>Re%	25EC4	DoHere:	25EEA	ModifierKey?
25E9D	CK0ATTNABORT	25EC5	DoKeyOb	25EEB	NEXTLIBBAK
25E9E	CK1NoBlame	25EC6	DoMenuKey	25EEC	NULL$TEMP
25E9F	CKREF	25EC7	DoNameKeyLRS	25EED	NoAttn?Semi
25EA0	COERCE$22	25EC8	DoNameKeyRS	25EEE	NoEdit?case
25EA1	CREATEDIR	25EC9	DoNextRow	25EEF	NoExitAction
25EA2	CRUNCH	25ECA	DoPlotMenu	25EF0	OR$
25EA3	CRUNCHNoBlame	25ECB	DoPrevRow	25EF1	PATHDIR
25EA6	CheckMenuRow	25ECC	DoSolvrMenu	25EF2	PrevNonNull
25EA7	Ck&DecKeyLoc	25ECD	DropBadKey	25EF3	RAD?
25EA8	Ck&Freeze	25ECE	EDITDECOMP$	25EF4	RECLAIMDISP
25EA9	CodePl>%rc.p	25ECF	EQUATION	25EF5	REPEATER
25EAA	DECOMP$	25ED0	EVALCRUNCH	25EF6	REPEATERCH
25EAB	DISPROW1*	25ED1	Echo2Macros	25EF7	SAFE@_HERE
25EAC	DISPROW2*	25ED2	EditMenu	25EF8	SEP$NL
25EAD	DISPSTATUS2	25ED3	EqList?	25EFA	SLEEPxcp
25EAE	DO#EXIT	25ED4	FlashMsg	25EFB	SaveLastMenu
25EAF	DO$EXIT	25ED5	GBUFFGROBDIM	25EFC	SetKeysNS
25EB0	DO%EXIT	25ED6	GETKEY	25EFD	SetSomeRow
25EB1	DO>STR	25ED7	GETKEY*	25EFE	SetThisRow
25EB2	DOBEEP	25ED8	GROB>GDISP	25EFF	SolvMenuInit
25EB3	DOCHR	25ED9	GetKeyOb	25F00	StartMenu
25EB4	DODISP	25EDA	GetMenu%	25F01	Std/BoxLabel
25EB5	DORCLE	25EDB	GetNextToken	25F02	StdMenuKeyLS
25EB6	DOSTOE	25EDC	H/W>KeyCode	25F03	StdMenuKeyNS
25EB7	DOSTR>	25EDD	H/WKey>KeyOb	25F04	SuspendOK?
25EB8	DOTVARS%	25EDE	HARDHEIGHT	25F05	TurnOffKey
25EB9	DOVARS	25EDF	ImmedEntry?	25F06	UART?

25F07	UARTxcp	25F2E	ExecGetLibsExten..	25FEA	DISPLASTROWBUT1
25F08	UnLockAlpha	25F63	!REDIMUSER	25FEF	CENTER$3x5
25F09	UserKeys?	25F68	!REDIMTEMP	25FF4	CENTER$5x7
25F0A	VLM	25F6D	realPAcode	25FF9	LEFT$3x5
25F0B	WaitForKey	25F72	#:>$	25FFE	LEFT$5x7
25F0C	XEQStoKey	25F77	#>$	26003	LEFT$5x7Arrow
25F0D	XOR$	25F7C	$>GROB	26008	LEFT$3x5Arrow
25F0E	XYGROBDISP	25F81	$>grob	2600D	LEFT$5x7CRArrow
25F0F	a%>$	25F86	$>GROBCR	26012	LEFT$3x5CRArrow
25F0F	a%>$,	25F8B	$>grobCR	26017	LEFT$5x7CR
25F10	ederr	25F90	>LANGUAGE	2601C	LEFT$3x5CR
25F11	editdecomp$w	25F95	LANGUAGE>	26021	CLEARVDISP
25F12	sstDISP	25F9A	OLASTOWDOB!	2602B	COERCEFLAG
25F13	stkdecomp$w	25F9A	OLastRomWrd!	26030	CURSOR_OFF
25F14	XEQPGDIR	25F9F	?ACCPTR>	26035	ClrAlphaAnn
25F15	>FONT	25FA4	ABUFF	2603A	ClrLeftAnn
25F16	DISP_LINE	25FA9	ALARM?	2603F	ClrRightAnn
25F17	GetMetaVStackDROP	25FAE	ATTN?	26044	ClrSysFlag
25F18	GetVStack	25FB3	DISPN	26049	ClrUserFlag
25F19	PopMetaVStack	25FB3	BIGDISPN	2604E	DOENG
25F1A	PopMetaVStackDROP	25FB8	DISPROW1	26053	DOFIX
25F1B	PopVStack	25FB8	DISP@01	26058	DOSCI
25F1C	PopVStackAbove	25FB8	BIGDISPROW1	2605D	DOSTD
25F1D	PushMetaVStack&D..	25FBD	DISPROW2	26062	DropSysObs
25F1E	PushVStack	25FBD	DISP@09	26067	ERRBEEP
25F1F	PushVStack&Clear	25FBD	BIGDISPROW2	2606C	ERASE&LEFT$5x7
25F20	PushVStack&Keep	25FC2	BIGDISPROW3	26071	ERASE&LEFT$3x5
25F21	PushVStack&KeepD..	25FC2	DISPROW3	26076	GBUFF
25F22	RestoreSysFlags	25FC2	DISP@17	2607B	GROB!
25F23	SaveSysFlags	25FC7	BIGDISPROW4	26080	GROB!ZERO
25F24	RIGHT$3x6	25FC7	DISP@25	26085	GROBDIM
25F25	CKNNOLASTWD	25FC7	DISPROW4	2608A	GsstFIN
25F29	'EvalNoCK:_sup	25FCC	DISPROW5	2608F	HARDBUFF
25F29	EvalNoCK:	25FD1	DISPROW7	26094	HARDBUFF2
25F2A	Keyword?	25FD6	DISPROW8	26099	HEIGHTENGROB
25F2B	CHECKMENU	25FDB	DISPROW9	2609E	INVGROB
25F2C	Find1stT.1	25FE0	DISPROW10	260A3	KILLGDISP
25F2D	TogInsertKey	25FE5	DISPLASTROW	260A8	LastMenuRow!

260AD	LastMenuRow@	26170	TestSysFlag	2622E	SetVStackProtect..
260B2	MAKEGROB	26175	TestUserFlag	26233	GetVStackProtect..
260B7	MenuRow!	2617F	WINDOWCORNER	26238	GetFontCmdHeight
260BC	MenuRow@	26184	WINDOWDOWN	2623D	GetFontHeight
260C1	NOHALTERR	26189	WINDOWLEFT	26242	StackFontHeight
260C6	NOTLISTcase	2618E	WINDOWRIGHT	26242	GetFontStkHeight
260CB	NOTROMPcase	26193	WINDOWUP	26247	GetHeader
260D0	NOTSECOcase	26198	WINDOWXY	2624C	GetMetaVStack
260D5	PIXOFF3	2619D	addtics	26251	InitVirtualStack
260DA	PIXON3	261A2	rstfmt1	26256	INITMKFONT
260DF	PIXOFF	261A7	savefmt1	2625B	MINIFONT>
260E4	PIXON	261AC	setbeep	26260	nDISPSTACK
260E9	PIXON?3	261B1	stackitw	26265	PushMetaVStack
260EE	PIXON?	261B6	subpdcdptch	2626A	PutElemBotVStack
260F3	PULLCMPEL	261BB	tok8trior	2626F	PutElemTopVStack
260F8	PULLREALEL	261C0	CLCD10	26274	RCL_NB_AFF_LGN
260FD	PUTCMPEL	261C5	CLEARLCD	26279	RCL_NB_AFF_LGNSTK
26102	PUTEL	261CA	FLUSHKEYS	2627E	SCANFONT
26107	PUTREALEL	261CA	FLUSH	26283	SetHeader
2610C	PrgmEntry?	261CF	%%>C%	26288	StackLineHeight
26111	RDUP	261D4	C%%0=	2628D	STO_ML_DISP_SIZE
26116	SETCIRCERR	261D9	C%%>C%	26292	CK0NOLASTWD
2611B	SETCURSOR	261DE	C%%CHS	26297	CK1NOLASTWD
26120	SLOW	261E3	C%%CONJ	2629C	CK2NOLASTWD
26125	VERYSLOW	261E8	C%0=	262A1	CK3NOLASTWD
2612A	VERYVERYSLOW	261ED	C%CHS	262A6	CK4NOLASTWD
2612F	SUBGROB	261F2	C%CONJ	262AB	CK5NOLASTWD
26134	SYNTAXERR	261F7	DISPROW6	262B0	CK0
26139	SetAlphaAnn	261FC	Re>C%	262B5	CK1
2613E	SetLeftAnn	26201	nextsym'R	262BA	CK2
26143	SetPrgmEntry	26206	tok8cktrior	262BF	CK3
26148	SetRightAnn	2620B	>MINIFONT	262C4	CK4
2614D	SetSysFlag	26210	CHECK_SCAN_FONT	262C9	CK5
26152	SetUserFlag	26215	DropVStack	262CE	CKN
26157	SystemLevel?	2621A	FONT>	262D3	CKN+1
26161	TIMEOUT?	2621F	FSCANFONT	262D8	SETSIZEERR
26166	TOADISP	26224	GetElemBotVStack	262DD	SETTYPEERR
2616B	TOGDISP	26229	GetElemTopVStack	262E2	SETSTACKERR

262E7	SETNONEXTERR	26562	CURSORPLUS	26721	Shrink$
262EC	%ABSCOERCE	26567	CURSORMINUS	26728	WindowXY
262F1	COERCE	26571	?CURSOR+	26736	clrtimeout
262F6	UNCOERCE	26576	CURSOR_OFF!	2673D	corner
262FB	COMPEVAL	2657B	CURSOR_OFF0	2674B	makegrob
26300	CK1&Dispatch	26580	CURSOR_OFF+	26752	setflag
26305	CK2&Dispatch	26585	CURSOR@	26759	settimeout
2630A	CK3&Dispatch	2658A	CURSOR+	26760	w->W
2630F	CK4&Dispatch	2658F	CURSOR-	26767	AllowIntr
26314	CK5&Dispatch	26594	CURSOR_PART	2676E	BITMAP
26319	EvalNoCK	26599	CURSOR_PART+	26775	Coldstart
2631E	CK&DISPATCH0	2659E	CURSOR_PART-	2677C	D0->Row1
26323	CK&DISPATCH2	265A3	CURPART->1	26783	D0->Sft1
26328	CK&DISPATCH1	265A8	CURPART->CR+	2678A	Debounce
2632D	#*OVF	265B7	Z>	26791	DisableIntr
2639B	MATATLOOP	265BC	Z<	26798	DispOff
263D2	!MATTRNnc	265C1	Z=	2679F	DispOn
26436	$>$?	265C6	Z<>	267A6	Flush
2644A	RROLL	265CB	Z>=	267AD	FlushAttn
26459	setStdWid	265D0	Z<=	267B4	GetTimChk
2645E	setStdEditWid	265D5	SetMetaVStack	267BB	GetTime++
2646D	!AND$	265DA	GetLibExt	267C2	OnKeyDown?
26472	!OR$	266B1	$5x7	267C9	OnKeyStable?
26477	!XOR$	266B8	CKLBCRC	267D0	RCKBp
2647C	!NOT$	266BF	COMPCONFCRC	267D7	SavPtrTime*
2649F	ClrBusyAnn	266C6	ErrjmpC	267DE	SrvcKbdAB
264A4	ClrI/OAnn	266CD	GETPTRFALSE	267E5	norecPWLseq
264B3	TOPLINE!	266D4	GETPTRTRUE	267E5	Warmstart
264B8	TOPLINE@	266DB	GPErrjmpC	267EC	clkspd
264BD	TOPLINE+	266E2	GPPushA	267F3	makebeep
264C2	TOPLINE-	266E9	GetStrLen	267FA	norecCSseq
264CC	FIRSTC@	266F0	GetStrLenC	26801	srvc_timer2
264D1	SETFIRSTC_0	266F7	GetStrLenStk	26808	aBZU
264D6	FIRSTC-	266FE	PUSHhxs	2680F	AFFICHE.REG
264DB	FIRSTC+	26705	PopASavptr	26816	AFFICHE.SBR
264F4	ClrPrgmEntry	2670C	PopSavptr	2681D	AFFICHEPIX.SBR
26508	NOEQERR	26713	SAFESKIPOB	2682B	BLKSWAP+
26521	SETUNDOERR	2671A	SetISysFlag	26832	CHANGE_FLAG

26839	CHANGE_FLAG2	2694A	MiniFontStk?	26A70	POPC%%
26840	Clean$	26951	PUSHzint	26A77	PUSHC%%
26847	Clean$R0	26958	PUSHzintLoop	26A93	aLineB
2684E	CleanVirtualStack	26966	SCAN.FONTE	26A9A	aLineW
26855	CMDSIZE	2696D	SCREEN.MARGIN2	26AA1	aLineG1
2685C	DBUG	26974	SCREEN.MARGIN	26AA8	aLineG2
26863	DBUG.TOUCHE	2697B	SET_HEADER	26AAF	aLineXor
26871	EndTempOb	26982	Shrink$Any	26AB6	aCircleB
26871	NEWADR	26989	Shrink$AnySafe	26ABD	aCircleW
26878	GET.FONT	26990	SIZEPLUS	26AC4	aCircleG1
2687F	GET_@FONTE	26990	Stretch$	26ACB	aCircleG2
26886	GET_@TAB	26997	STOFONT	26AD2	aCircleXor
2688D	GET_ATTRIBN.REAL	2699E	STOMINIFONT	26AD9	aSubReplRepl
26894	GET_HEADER	269A5	Stretch$Any	26AE0	aSubReplGor
2689B	GET_HEADERTYPE	269AC	STYLE.MINIFONT	26AE7	aSubReplGxor
268A2	GET_HFONTE	269B3	SWAPMEM	26AEE	ComputePixel
268A9	GET_HFONTECMD	269BA	SWAPMEM_D0D1C	26AF5	aGrey?
268B0	GET_HFONTESTK	269C1	SWAPMEM_D0D1C_no..	26AFC	aGNeg
268B7	GET_HFONTESTKD1C	269C8	SWAPMEM_D0D1D	26B03	aScroolVGrob
268BE	GET_NBLIGNE	269CF	SWAPMEM_D0D1D_no..	26B0A	aDistance
268C5	GET_NBLIGNESTK	269D6	SWAPMEM_nofree	26B11	aPixonW
268CC	GETBOTTEMP	269DD	SWAPMEMEQ	26B18	aPixonB
268D3	GetStrLenL	269E4	SWAPMEMEQ_D0D1C	26B1F	aPixonG1
268DA	GETX.VISIBLE	269EB	WIPESPACE	26B26	aPixonG2
268E1	GETX.VISIBLE.STR	269F2	AMULT34	26B2D	aPixonXor
268E8	GPPushR0Lp	269F9	CMULT34	26B34	aFBoxB
268EF	GPPushALp	26A00	MULTBAC	26B3B	aFBoxW
268F6	INIT_AFFICHELIGNE	26A07	MULTB+A*C	26B42	aFBoxG1
268FD	INIT_AFFICHELIGN..	26A0E	HEXTODEC	26B49	aFBoxG2
26904	Init_MetaKernelF..	26A15	ADivC	26B50	aFBoxXor
2690B	INV.ZONE	26A1C	BMULT34	26B57	aLBoxB
26912	InverseParcelle	26A23	ADIV6	26B5E	aLBoxW
26919	MAKEBOT$N	26A2A	ADIV3	26B65	aLBoxG1
26920	MAKERAM$	26A31	D0=ALoop	26B6C	aLBoxG2
26927	MINI_DISP	26A38	DISP_DEC	26B73	aLBoxXor
2692E	MINI_DISP_AWP	26A4D	Shrink$List	26B7A	Arrows
2693C	MINI_DISP_VAL	26A62	POPC%	26B81	ACCESSID1
26943	MiniFontCmd?	26A69	PUSHC%	26B88	ACCESSID2

26B8F	ACCESSID3	26CF4	addrVDISP2	2722F	tokDIR
26B96	ACCESSID4	26CFB	adrTIMEOUTCLK	2723F	tok:
26B9D	ACCESSID5	26D10	ChkGrHook	2724B	tok‘
26BA4	ACCESSID6	26D17	ThisKeyDnCb?	27257	tokELSE
26BAB	ACCESSID7	26D1E	ThisKeyDn?	27269	tokEND
26BB2	GetBankAccess	26DF7	%14400	27279	tokUNTIL
26BB9	ACCESSBank0	26E21	%38400	2728D	tokREPEAT
26BC0	ACCESSBank1	26E36	%57600	272A3	tokNEXT
26BC7	ACCESSBank2	26E4B	%115200	272B5	tokSTEP
26BCE	ACCESSBank3	26E60	ASRW5	272C7	tokTHEN
26BD5	ACCESSBank4	26E71	ASLW5	272D9	tok->
26BDC	ACCESSBank5	26E82	CSRW5	272E5	MARKED
26BE3	ACCESSBank6	26E93	CSLW5	272F3	CUREQ
26BEA	ACCESSBank7	26F00	DCHXW	272FE	NULLID
26BF1	ACCESSBank8	26F36	%1+	27308	NULLID!
26BF8	ACCESSBank9	26F4A	%1-	27308	NULLID1
26BFF	ACCESSBank10	27012	%%1+	27308	EvalNULLID
26C06	ACCESSBank11	2708A	DUP%%0=	2733F	Z-9
26C0D	ACCESSBank12	2709E	MPY	2734B	Z-8
26C14	ACCESSBank13	270BF	#5*	27357	Z-7
26C1B	ACCESSBank14	270DA	#3*	27363	Z-6
26C22	ACCESSBank15	270EE	%1.8	2736F	Z-5
26C29	ACCESSIDn	27103	%80	2737B	Z-4
26C30	ACCESSRAM0	27118	%.1	27387	Z-3
26C37	ACCESSERAM1	2712D	%.15	27393	Z-2
26C3E	ACCESSERAM2	27142	LAMLNAME	2739F	Z-1
26C45	NEWACCESSRAM	27155	’IDX	273AB	Z0
26C4C	RclCompareNames	2715F	ID_X	273B6	Z1
26C53	CompareACbBytes	2716D	StdIOPAR	273C2	Z2
26C5A	FindInDir	27195	CRLF$	273CE	Z3
26C61	ScanEveryObjects	271A3	IDIOPAR	273DA	Z4
26C68	RclAssembly	271B1	ListSTARTUP	273E6	Z5
26C6F	ValidPortTag?	271B9	ID_STARTUP	273F2	Z6
26CA7	DOSIZEERR	271D3	IDSTARTERR	273FE	Z7
26CD8	addrADISP	271D8	ID_STARTERR	2740A	Z8
26CDF	addrATTNFLG	271F4	2NULLLAM{}	27416	Z9
26CE6	addrLINECNTg	27208	3NULLLAM{}	27422	Z10
26CED	addrVDISP	27221	tokTO	2742F	Z12

2743C	Z24	2779C	Acknowledge#	27B2F	ID_TPAR
27449	Z100	277A6	KeyInAlrm#	27B43	'IDFUNCTION
274A4	INTERNALiX	277B0	SelectRpt#	27B57	'IDCONIC
274A9	Z1Z0	277BA	IOSetupMenu#	27B6B	'IDPOLAR
274A9	ZINT1_0	277C4	PlotType#	27B7F	'IDPARAMETER
27516	Z0Z1	277CE	NoExecAct#	27B93	'IDTRUTH
2754B	Z-1Z0	277D8	OffScreen#	27BA7	'IDSCATTER
2756C	Z1Z1	277E2	OnlyPtypes#	27BBB	'IDHISTOGRAM
275C6	TakeOver	277EC	StatName#	27BCF	'IDBAR
275EE	Modifier	277F6	LN_0	27BE3	'IDFAST3D
275FD	MenuKey	27800	LN_Neg	27C0B	$1:_
27620	MenuMaker	2780A	InvalidEQ	27C33	ExitFcn
2768E	#60E	27814	Cureq#	27C70	Z0ONE
27698	NoStatPlot#	2781E	NoCureq#	27D3F	CROSSGROB
276AC	SolvingFor#	27828	EnterEq#	27D5D	MARKGROB
276B6	NoCurrent#	27832	EnterName#	27D7B	NullMenuLbl
276C0	PressSig+#	2783C	SelPtype#	27DBF	C%-1
276CA	SelectModl#	27846	EmptyCat#	27DE4	C%0
276D4	NoAlarms#	27878	BINT800h	27E09	C%1
276DE	PressALRM#	27882	Attn#	27E2E	C%%1
276E8	NextALRM#	27937	ID_SIGMADAT	27E5D	%100
276F2	ZoomPrompt#	27946	ID_SIGMAPAR	27E72	nohalt
276FC	CatToStack#	2795A	ID_N	27E87	TrueTrue
27706	XAutoZoom#	27963	ID_I%YR	27E9B	failed
27710	IR/wire#	27972	ID_PV	27EAF	<SkipKey
2771A	ASCII/bin#	2797D	ID_PMT	27EB4	<Skip$
27724	#62A	2798A	ID_FV	27EFB	>SkipKey
2772E	#62B	2799A	ID_PYR	27F00	>Skip$
27738	#62C	2799A	ID_PPAR	27F47	<DelKey
27742	#62D	279F6	StdBaseLabel	27F4C	<Del$
2774C	Lackint#	27A3A	StdPRTPAR	27F9A	>DelKey
27756	Constant#	27A89	%%2PI	27F9F	>Del$
27760	Zero#	27AA3	NULLPAINT	27FED	NullMenuKey
2776A	RevSgn#	27AB7	8NULLLAM{}	28001	ROT#1+UNROT
27774	Extremum#	27AE9	'IDPAR	28071	#1-SWAP
2777E	#12F	27B07	'IDVPAR	28071	pull
27788	EnterMatrix#	27B11	ID_VPAR	28085	pullrev
27792	PastDue#	27B25	'IDTPAR	28099	SWAP#1+SWAP

280AD	SWAP#1-SWAP	295BA	!PTR>HCOMP	2A095	Clipboard@
280C1	ORDERXY#	29616	RDROPTRUE	2A0A5	Clipboard0
280F8	ORDERXY%	2962A	RDROPFALSE	2A0B5	Clipboard?
2812F	SWAPTRUE	2963E	psh&	2A0C5	FindPattern!
28143	NDUPN	29693	psh1top&	2A0D5	FindPattern@
28187	reversym	296A7	top&	2A0E5	FindPattern0
281D5	#1-UNROT	29722	top&top&	2A0F5	FindPattern?
281E9	DROPCOLA	2973B	pshtop&	2A105	ReplacePattern!
281FD	DUPROLLSWAP	29754	pullpsh1&	2A115	ReplacePattern@
28211	NDROPFALSE	29786	'RSWP1+	2A125	ReplacePattern0
28225	9UNROLL	2979A	'R'RROT2+	2A135	ReplacePattern?
28239	SWAPDROPFALSE	297EF	INNERtop&	2A145	AppError!
2825E	TWONTHCOMPDROP	29808	'Rswapop	2A158	AppError@
28286	OVER#1-	29821	psh1&	2A4AA	ATTNchk
282CC	DROP%0	298C0	psh1&rev	2A4FC	WaitTbz0
28335	KEEP	29972	pshzer	2A5CA	SUB$1#
283A3	Push#FLoop	29986	pshzerpsharg	2A7A7	CkSecoType
283E8	FalseFalse	29A35	dup	2A7CF	ABNDTrue
283FC	ISTOP-INDEX	29A5D	psh	2A7E3	ABNDFalse
286E7	symcomp	29A8F	roll2ND	2A7F7	DispTimeReq?
286F6	ONESYMBN	29B12	unroll2ND	2A842	Decomp1Line
287E6	3PICK#2+	29CB9	uncrunch	2A85D	DecompEdit
28804	3PICK#1+	29D18	ONE{}N	2A878	Decomp#Line
28989	SWAPDROP#1-	29D6A	delimcase	2A893	Decomp#Disp
2899D	2pull2DROP	29E29	ngsizecase	2A8AE	DecompEcho
28ACE	DROP?symcomp	29E67	nultrior	2A8C9	DecompStd1Line
28D38	4DROPFALSE	29E99	tok=casedrop	2A8E4	DecompStd1Line32
28DAB	3DROPTRUE	29ED0	'Rapndit	2A904	RPNDecomp1Line
28E05	5DROPFALSE	29EE9	DaDGNTc	2A924	RPNDecompEdit
29137	pulldroppull	29F0C	DEL_END$	2A944	RPNDecomp#Line
29362	DUP#2+PICK	29F25	AppDisplay!	2A964	RPNDecomp#Disp
293A3	?symcomp	29F35	AppDisplay@	2A984	RPNDecompEcho
293C1	PSYMBN	29F55	AppKeys!	2A9A4	RPNDecompStd1Line
293F8	P{}N	29F65	AppKeys@	2A9C4	RPNDecompStd1Lin..
2942F	P::N	29F75	AppKeys0	2A9E9	RunRPN:
2949D	!>HCOMP	2A055	AppExitCond!	2AA43	AlgDecomp
294CF	!>HCOMPcopy	2A065	AppExitCond@	2AA70	CASEVAL
29501	!&HCOMP	2A085	Clipboard!	2AAE0	SimplifyExpression

2AB69	RunInApprox	2B3FD	%IP>#	2C10B	D/D=
2ABD7	RunSafeFlagsNoEr..	2B42A	PUTLIST	2C116	D/DABS
2ABF0	RunSafeFlags	2B475	ParOuterLoop	2C121	easyabs
2AC0E	DoRunSafe	2B4AC	POLSaveUI	2C13A	D/DACOS
2AC72	need'case	2B542	POLSetUI	2C145	D/DACOSH
2ACA9	addrTEMPTOP	2B628	POLKeyUI	2C150	D/DALOG
2ACB0	?TogU/LCase	2B682	POLErrorTrap	2C15B	D/DARG
2AD81	EQUALcasedrop	2B6B4	POLResUI&Err	2C166	D/DASIN
2ADBD	nonopcase	2B6CD	POLRestoreUI	2C171	D/DASINH
2ADE0	numb1stcase	2B709	InitPOLVars	2C17C	D/DATAN
2AE32	M-1stcasechs	2B74F	StartupProc	2C187	D/DATANH
2AF37	AEQ1stcase	2B7CC	addrClkOnNib	2C192	D/DCHS
2AFFB	MEQ1stcase	2B8BE	OBJ>R	2C1B0	D/DCONJ
2B01B	MEQopscase	2B8E6	R>OBJ	2C1CE	D/DCOS
2B06A	AEQopscase	2B90B	'DROPFALSE	2C1D9	D/DCOSH
2B083	Mid1stcase	2BAB3	COLAthexFCN	2C1E4	D/DEXP
2B0CC	idntcase	2BB21	sscknum2	2C1EF	D/DINV
2B0EF	idntlamcase	2BB3A	sncknum2	2C1FA	D/DLN
2B11C	num0=case	2BB53	nscknum2	2C205	D/DLNP1
2B149	%0=case	2BCA2	cknumdsptch1	2C210	D/DLOG
2B15D	C%0=case	2BD8C	Cr	2C21B	D/DIFTE
2B176	num1=case	2BE36	AlgebraicModecase	2C226	D/DSIN
2B1A3	%1=case	2BF1C	CkEQUtil	2C231	D/DSINH
2B1C1	C%1=case	2BF3A	DA1OK?NOTIT	2C23C	D/DSQ
2B1DF	num2=case	2BF53	DA2aOK?NOTIT	2C247	D/DSQRT
2B20C	%2=case	2BF6C	DA2bOK?NOTIT	2C252	D/DTAN
2B22A	C%2=case	2BF85	DA3OK?NOTIT	2C25D	D/DTANH
2B25C	num-1=case	2C039	nWHEREIFTE	2C268	D/D^
2B289	%-1=case	2C044	nWHEREDER	2C273	D/D^X
2B2A7	C%-1=case	2C04F	nWHEREINTG	2C27E	D/D^Y
2B2C5	NOTcaseFALSE	2C05A	nWHERESUM	2C289	D/DDER
2B2F2	DoLevel1:	2C065	nWHEREWHERE	2C294	D/DWHERE
2B31A	Roll&Do:	2C07B	D/D*	2C29F	D/DINTEGRAL
2B351	Rcl&Do:	2C086	D/D+	2C2AA	D/DSUM
2B3A6	1NULLLAM{}	2C091	D/D-	2C2B5	D/DAPPLY
2B3AB	NULLLAM	2C09C	D/D/	2C2C0	nCustomMenu
2B3B7	4NULLLAM{}	2C0A7	derquot	2C2CB	nCOLCTQUOTE
2B3D5	@DROP	2C0ED	derprod1	2C2D6	SPLITWHERE

2C2F9	DispStsBound	2D90F	tokmol	2E451	TOLSetTopicUI
2C305	DispStatus	2D929	unit_?	2E46F	TOLSetTopUI.1
2C311	?DispStatus	2D933	tok?	2E4AB	TOLSetViewUI
2C341	?DispStack	2D949	UMSIGN	2E4C9	TOLSetViUI.1
2C371	DoInputForm	2D95D	UMIP	2E51E	TOLKeyUI
2C37D	PTYPE>PINFO	2D971	UMFP	2E573	TOLErrorTrap
2C388	MOVEVAR	2D985	UMFLOOR	2E5A5	TOLResUI&Err
2C393	COPYVAR	2D999	UMCEIL	2E5C3	TOLRestoreUI
2C3FA	DOTVARS	2D9CB	UMRND	2E659	?ExitThisTop
2C4AA	2DROP%0	2D9EE	UMTRC	2E686	BadTOLUI?
2C4D2	SYMSYMSYMANY	2DA11	cfF	2E68B	SetBadTOLUI
2C53B	DUP%ABS	2DA2B	cfC	2E690	ClrBadTOLUI
2D74F	um*	2DCB5	FLOAT	2E698	CALCCXT!
2D759	um/	2DD27	Day>Date	2E69D	CALCCXT@
2D763	um^	2DDD5	getBPOFF	2E6A7	PGMCXT!
2D76D	umP	2DE26	mpop1%	2E6AC	PGMCXT@
2D777	umEND	2DE4A	dowutil	2E6B6	NOTESCXT!
2D781	SIbasis	2DEAA	HXDCW	2E6BB	NOTESCXT@
2D7A9	unit_r	2DEBB	DAY#	2E6C5	apletPTR!
2D7B3	tokr	2DFCC	?DispMenu	2E6CA	apletPTR@
2D7C9	unit_sr	2DFE0	DispMenu	2E6D4	funcPTR!
2D7D3	toksr	2DFF4	DispMenu.1	2E6D9	funcPTR@
2D7F5	unit_R	2E094	StdLabelDef	2E6E3	polarPTR!
2D7FF	tokdegR	2E0D5	Grob>Menu	2E6E8	polarPTR@
2D817	%%TANDEG	2E0F3	Str>Menu	2E6F2	paramPTR!
2D837	unit_kg	2E107	Seco>Menu	2E6F7	paramPTR@
2D848	tok_g	2E11B	Id>Menu	2E701	seqPTR!
2D863	unit_m	2E139	MakeDir/StdLabel	2E706	seqPTR@
2D86D	tok_m	2E166	MakeStdLabel	2E710	statPTR!
2D883	unit_A	2E189	MakeBoxLabel	2E715	statPTR@
2D88D	tokA	2E198	BoxLabelGrobInv	2E71F	solvePTR!
2D8A3	unit_s	2E1EB	MakeDirLabel	2E724	solvePTR@
2D8AD	tok_s	2E1FA	DirLabelGrobInv	2E72E	otherPTR!
2D8C3	unit_K	2E24D	MakeInvLabel	2E733	otherPTR@
2D8CD	tokK	2E25C	InvLabelGrob	2E73D	TopicDoN
2D8E3	unit_cd	2E2AA	MakeLabel	2E76A	TopicVar1!
2D8ED	tokcd	2E2CD	TopOuterLoop	2E76B	TopicVar1@
2D905	unit_mol	2E3DE	TOLSaveUI	2E76C	TopicVar2!

2E76D	TopicVar2@	2E793	TopicVar21@	2E7B9	TopicVar40@
2E76E	TopicVar3!	2E794	TopicVar22!	2E7BA	TopicVar41!
2E76F	TopicVar3@	2E795	TopicVar22@	2E7BB	TopicVar41@
2E770	TopicVar4!	2E796	TopicVar23!	2E7BC	TopicVar42!
2E771	TopicVar4@	2E797	TopicVar23@	2E7BD	TopicVar42@
2E772	TopicVar5!	2E798	TopicVar24!	2E7BE	TopicVar43!
2E773	TopicVar5@	2E799	TopicVar24@	2E7BF	TopicVar43@
2E774	TopicVar6!	2E79A	TopicVar25!	2E7C0	TopicVar44!
2E775	TopicVar6@	2E79B	TopicVar25@	2E7C1	TopicVar44@
2E776	TopicVar7!	2E79C	TopicVar26!	2E7C2	TopicVar45!
2E777	TopicVar7@	2E79D	TopicVar26@	2E7C3	TopicVar45@
2E778	TopicVar8!	2E79E	TopicVar27!	2E7C4	TopicVar46!
2E779	TopicVar8@	2E79F	TopicVar27@	2E7C5	TopicVar46@
2E77A	TopicVar9!	2E7A0	TopicVar28!	2E7C6	TopicVar47!
2E77B	TopicVar9@	2E7A1	TopicVar28@	2E7C7	TopicVar47@
2E77C	TopicVar10!	2E7A2	TopicVar29!	2E7C8	TopicVar48!
2E77D	TopicVar10@	2E7A3	TopicVar29@	2E7C9	TopicVar48@
2E77E	TopicVar11!	2E7A4	TopicVar30!	2E7CA	TopicVar49!
2E77F	TopicVar11@	2E7A5	TopicVar30@	2E7CB	TopicVar49@
2E780	TopicVar12!	2E7A6	TopicVar31!	2E7CC	TopicVar50!
2E781	TopicVar12@	2E7A7	TopicVar31@	2E7CD	TopicVar50@
2E782	TopicVar13!	2E7A8	TopicVar32!	2E7CE	TopicVar51!
2E783	TopicVar13@	2E7A9	TopicVar32@	2E7CF	TopicVar51@
2E784	TopicVar14!	2E7AA	TopicVar33!	2E7D0	TopicVar52@
2E785	TopicVar14@	2E7AB	TopicVar33@	2E7D1	TopicVar52!
2E786	TopicVar15!	2E7AC	TopicVar34!	2E7D2	TopicVar53@
2E787	TopicVar15@	2E7AD	TopicVar34@	2E7D3	TopicVar53!
2E788	TopicVar16!	2E7AE	TopicVar35!	2E7D4	TopicVar54@
2E789	TopicVar16@	2E7AF	TopicVar35@	2E7D5	TopicVar54!
2E78A	TopicVar17!	2E7B0	TopicVar36!	2E7D6	TopicVar55@
2E78B	TopicVar17@	2E7B1	TopicVar36@	2E7D7	TopicVar55!
2E78C	TopicVar18!	2E7B2	TopicVar37!	2E7D8	TopicVar56@
2E78D	TopicVar18@	2E7B3	TopicVar37@	2E7D9	TopicVar56!
2E78E	TopicVar19!	2E7B4	TopicVar38!	2E7DA	TopicVar57@
2E78F	TopicVar19@	2E7B5	TopicVar38@	2E7DB	TopicVar57!
2E790	TopicVar20!	2E7B6	TopicVar39!	2E7DC	TopicVar58@
2E791	TopicVar20@	2E7B7	TopicVar39@	2E7DD	TopicVar58!
2E792	TopicVar21!	2E7B8	TopicVar40!	2E7DE	TopicVar59@

2E7DF	TopicVar59!	2E805	TopicVar78!	2E82B	TOLVar6@
2E7E0	TopicVar60@	2E806	TopicVar79@	2E82C	TOLVar7!
2E7E1	TopicVar60!	2E807	TopicVar79!	2E82D	TOLVar7@
2E7E2	TopicVar61@	2E808	TopicVar80@	2E82E	TOLVar8!
2E7E3	TopicVar61!	2E809	TopicVar80!	2E82F	TOLVar8@
2E7E4	TopicVar62@	2E80A	TopicVar81@	2E830	TOLVar9!
2E7E5	TopicVar62!	2E80B	TopicVar81!	2E831	TOLVar9@
2E7E6	TopicVar63@	2E80C	TopicVar82@	2E832	TOLVar10!
2E7E7	TopicVar63!	2E80D	TopicVar82!	2E833	TOLVar10@
2E7E8	TopicVar64@	2E80E	TopicVar83@	2E834	TOLVar11!
2E7E9	TopicVar64!	2E80F	TopicVar83!	2E835	TOLVar11@
2E7EA	TopicVar65@	2E810	TopicVar84@	2E836	TOLVar12!
2E7EB	TopicVar65!	2E811	TopicVar84!	2E837	TOLVar12@
2E7EC	TopicVar66@	2E812	TopicVar85@	2E838	TOLVar13!
2E7ED	TopicVar66!	2E813	TopicVar85!	2E839	TOLVar13@
2E7EE	TopicVar67@	2E814	TopicVar86@	2E83A	TOLVar14!
2E7EF	TopicVar67!	2E815	TopicVar86!	2E83B	TOLVar14@
2E7F0	TopicVar68@	2E816	TopicVar87@	2E83C	TOLVar15!
2E7F1	TopicVar68!	2E817	TopicVar87!	2E83D	TOLVar15@
2E7F2	TopicVar69@	2E818	TopicVar88@	2E83E	TOLVar16!
2E7F3	TopicVar69!	2E819	TopicVar88!	2E83F	TOLVar16@
2E7F4	TopicVar70@	2E81A	TopicVar89@	2E840	TOLVar17!
2E7F5	TopicVar70!	2E81B	TopicVar89!	2E841	TOLVar17@
2E7F6	TopicVar71@	2E81C	TopicVar90@	2E842	TOLVar18!
2E7F7	TopicVar71!	2E81D	TopicVar90!	2E843	TOLVar18@
2E7F8	TopicVar72@	2E81E	TopicVar91!	2E844	TOLVar19!
2E7F9	TopicVar72!	2E81F	TopicVar91@	2E845	TOLVar19@
2E7FA	TopicVar73@	2E820	TOLVar1!	2E846	TOLVar20!
2E7FB	TopicVar73!	2E821	TOLVar1@	2E847	TOLVar20@
2E7FC	TopicVar74@	2E822	TOLVar2!	2E848	TOLVar21!
2E7FD	TopicVar74!	2E823	TOLVar2@	2E849	TOLVar21@
2E7FE	TopicVar75@	2E824	TOLVar3!	2E84A	TOLVar22!
2E7FF	TopicVar75!	2E825	TOLVar3@	2E84B	TOLVar22@
2E800	TopicVar76@	2E826	TOLVar4!	2E84C	TOLVar23!
2E801	TopicVar76!	2E827	TOLVar4@	2E84D	TOLVar23@
2E802	TopicVar77@	2E828	TOLVar5!	2E84E	TOLVar24!
2E803	TopicVar77!	2E829	TOLVar5@	2E84F	TOLVar24@
2E804	TopicVar78@	2E82A	TOLVar6!	2E850	TOLVar25!

2E851	TOLVar25@	2E877	TOLVar44@	2E89D	TOLVar63@
2E852	TOLVar26!	2E878	TOLVar45!	2E89E	TOLVar64!
2E853	TOLVar26@	2E879	TOLVar45@	2E89F	TOLVar64@
2E854	TOLVar27!	2E87A	TOLVar46!	2E8A0	TOLVar65!
2E855	TOLVar27@	2E87B	TOLVar46@	2E8A1	TOLVar65@
2E856	TOLVar28!	2E87C	TOLVar47!	2E8A2	TOLVar66!
2E857	TOLVar28@	2E87D	TOLVar47@	2E8A3	TOLVar66@
2E858	TOLVar29!	2E87E	TOLVar48!	2E8A4	TOLVar67!
2E859	TOLVar29@	2E87F	TOLVar48@	2E8A5	TOLVar67@
2E85A	TOLVar30!	2E880	TOLVar49!	2E8A6	TOLVar68!
2E85B	TOLVar30@	2E881	TOLVar49@	2E8A7	TOLVar68@
2E85C	TOLVar31!	2E882	TOLVar50!	2E8A8	TOLVar69!
2E85D	TOLVar31@	2E883	TOLVar50@	2E8A9	TOLVar69@
2E85E	TOLVar32!	2E884	TOLVar51!	2E8AA	TOLVar70!
2E85F	TOLVar32@	2E885	TOLVar51@	2E8AB	TOLVar70@
2E860	TOLVar33!	2E886	TOLVar52!	2E8AC	TOLVar71!
2E861	TOLVar33@	2E887	TOLVar52@	2E8AD	TOLVar71@
2E862	TOLVar34!	2E888	TOLVar53!	2E8AE	TOLVar72!
2E863	TOLVar34@	2E889	TOLVar53@	2E8AF	TOLVar72@
2E864	TOLVar35!	2E88A	TOLVar54!	2E8B0	TOLVar73!
2E865	TOLVar35@	2E88B	TOLVar54@	2E8B1	TOLVar73@
2E866	TOLVar36!	2E88C	TOLVar55!	2E8B2	TOLVar74!
2E867	TOLVar36@	2E88D	TOLVar55@	2E8B3	TOLVar74@
2E868	TOLVar37!	2E88E	TOLVar56!	2E8B4	TOLVar75!
2E869	TOLVar37@	2E88F	TOLVar56@	2E8B5	TOLVar75@
2E86A	TOLVar38!	2E890	TOLVar57!	2E8B6	TOLVar76!
2E86B	TOLVar38@	2E891	TOLVar57@	2E8B7	TOLVar76@
2E86C	TOLVar39!	2E892	TOLVar58!	2E8B8	TOLVar77!
2E86D	TOLVar39@	2E893	TOLVar58@	2E8B9	TOLVar77@
2E86E	TOLVar40!	2E894	TOLVar59!	2E8BA	TOLVar78!
2E86F	TOLVar40@	2E895	TOLVar59@	2E8BB	TOLVar78@
2E870	TOLVar41!	2E896	TOLVar60!	2E8BC	TOLVar79!
2E871	TOLVar41@	2E897	TOLVar60@	2E8BD	TOLVar79@
2E872	TOLVar42!	2E898	TOLVar61!	2E8BE	TOLVar80!
2E873	TOLVar42@	2E899	TOLVar61@	2E8BF	TOLVar80@
2E874	TOLVar43!	2E89A	TOLVar62!	2E8C0	TOLVar81!
2E875	TOLVar43@	2E89B	TOLVar62@	2E8C1	TOLVar81@
2E876	TOLVar44!	2E89C	TOLVar63!	2E8C2	TOLVar82!

2E8C3	TOLVar82@	2E8E9	TOLVar101@	2E90F	TOLVar120@
2E8C4	TOLVar83!	2E8EA	TOLVar102!	2E910	TOLVar121!
2E8C5	TOLVar83@	2E8EB	TOLVar102@	2E911	TOLVar121@
2E8C6	TOLVar84!	2E8EC	TOLVar103!	2E912	TOLVar122!
2E8C7	TOLVar84@	2E8ED	TOLVar103@	2E913	TOLVar122@
2E8C8	TOLVar85!	2E8EE	TOLVar104!	2E914	TOLVar123!
2E8C9	TOLVar85@	2E8EF	TOLVar104@	2E915	TOLVar123@
2E8CA	TOLVar86!	2E8F0	TOLVar105!	2E916	TOLVar124!
2E8CB	TOLVar86@	2E8F1	TOLVar105@	2E917	TOLVar124@
2E8CC	TOLVar87!	2E8F2	TOLVar106!	2E918	TOLVar125!
2E8CD	TOLVar87@	2E8F3	TOLVar106@	2E919	TOLVar125@
2E8CE	TOLVar88!	2E8F4	TOLVar107!	2E91A	TOLVar126!
2E8CF	TOLVar88@	2E8F5	TOLVar107@	2E91B	TOLVar126@
2E8D0	TOLVar89!	2E8F6	TOLVar108!	2E91C	TOLVar127!
2E8D1	TOLVar89@	2E8F7	TOLVar108@	2E91D	TOLVar127@
2E8D2	TOLVar90!	2E8F8	TOLVar109!	2E91E	TOLVar128!
2E8D3	TOLVar90@	2E8F9	TOLVar109@	2E91F	TOLVar128@
2E8D4	TOLVar91!	2E8FA	TOLVar110!	2E920	TOLVar129!
2E8D5	TOLVar91@	2E8FB	TOLVar110@	2E921	TOLVar129@
2E8D6	TOLVar92!	2E8FC	TOLVar111!	2E922	TOLVar130!
2E8D7	TOLVar92@	2E8FD	TOLVar111@	2E923	TOLVar130@
2E8D8	TOLVar93!	2E8FE	TOLVar112!	2E924	TOLVar131!
2E8D9	TOLVar93@	2E8FF	TOLVar112@	2E925	TOLVar131@
2E8DA	TOLVar94!	2E900	TOLVar113!	2E926	TOLVar132!
2E8DB	TOLVar94@	2E901	TOLVar113@	2E927	TOLVar132@
2E8DC	TOLVar95!	2E902	TOLVar114!	2E928	TOLVar133!
2E8DD	TOLVar95@	2E903	TOLVar114@	2E929	TOLVar133@
2E8DE	TOLVar96!	2E904	TOLVar115!	2E92A	TOLVar134!
2E8DF	TOLVar96@	2E905	TOLVar115@	2E92B	TOLVar134@
2E8E0	TOLVar97!	2E906	TOLVar116!	2E92C	TOLVar135!
2E8E1	TOLVar97@	2E907	TOLVar116@	2E92D	TOLVar135@
2E8E2	TOLVar98!	2E908	TOLVar117!	2E92E	TOLVar136!
2E8E3	TOLVar98@	2E909	TOLVar117@	2E92F	TOLVar136@
2E8E4	TOLVar99!	2E90A	TOLVar118!	2E930	TOLVar137!
2E8E5	TOLVar99@	2E90B	TOLVar118@	2E931	TOLVar137@
2E8E6	TOLVar100!	2E90C	TOLVar119!	2E932	TOLVar138!
2E8E7	TOLVar100@	2E90D	TOLVar119@	2E933	TOLVar138@
2E8E8	TOLVar101!	2E90E	TOLVar120!	2E934	TOLVar139!

2E935	TOLVar139@	2E95B	TOLVar158@	2E981	TOLVar177@
2E936	TOLVar140!	2E95C	TOLVar159!	2E982	TOLVar178!
2E937	TOLVar140@	2E95D	TOLVar159@	2E983	TOLVar178@
2E938	TOLVar141!	2E95E	TOLVar160!	2E984	TOLVar179!
2E939	TOLVar141@	2E95F	TOLVar160@	2E985	TOLVar179@
2E93A	TOLVar142!	2E960	TOLVar161!	2E986	TOLVar180!
2E93B	TOLVar142@	2E961	TOLVar161@	2E987	TOLVar180@
2E93C	TOLVar143!	2E962	TOLVar162!	2E988	TOLVar181!
2E93D	TOLVar143@	2E963	TOLVar162@	2E989	TOLVar181@
2E93E	TOLVar144!	2E964	TOLVar163!	2E98A	TOLVar182!
2E93F	TOLVar144@	2E965	TOLVar163@	2E98B	TOLVar182@
2E940	TOLVar145!	2E966	TOLVar164!	2E98C	TOLVar183!
2E941	TOLVar145@	2E967	TOLVar164@	2E98D	TOLVar183@
2E942	TOLVar146!	2E968	TOLVar165!	2E98E	TOLVar184!
2E943	TOLVar146@	2E969	TOLVar165@	2E98F	TOLVar184@
2E944	TOLVar147!	2E96A	TOLVar166!	2E990	TOLVar185!
2E945	TOLVar147@	2E96B	TOLVar166@	2E991	TOLVar185@
2E946	TOLVar148!	2E96C	TOLVar167!	2E992	TOLVar186!
2E947	TOLVar148@	2E96D	TOLVar167@	2E993	TOLVar186@
2E948	TOLVar149!	2E96E	TOLVar168!	2E994	TOLVar187!
2E949	TOLVar149@	2E96F	TOLVar168@	2E995	TOLVar187@
2E94A	TOLVar150!	2E970	TOLVar169!	2E996	TOLVar188!
2E94B	TOLVar150@	2E971	TOLVar169@	2E997	TOLVar188@
2E94C	TOLVar151!	2E972	TOLVar170!	2E998	TOLVar189!
2E94D	TOLVar151@	2E973	TOLVar170@	2E999	TOLVar189@
2E94E	TOLVar152!	2E974	TOLVar171!	2E99A	TOLVar190!
2E94F	TOLVar152@	2E975	TOLVar171@	2E99B	TOLVar190@
2E950	TOLVar153!	2E976	TOLVar172!	2E99C	TOLVar191!
2E951	TOLVar153@	2E977	TOLVar172@	2E99D	TOLVar191@
2E952	TOLVar154!	2E978	TOLVar173!	2E99E	TOLVar192!
2E953	TOLVar154@	2E979	TOLVar173@	2E99F	TOLVar192@
2E954	TOLVar155!	2E97A	TOLVar174!	2E9A0	TOLVar193!
2E955	TOLVar155@	2E97B	TOLVar174@	2E9A1	TOLVar193@
2E956	TOLVar156!	2E97C	TOLVar175!	2E9A2	TOLVar194!
2E957	TOLVar156@	2E97D	TOLVar175@	2E9A3	TOLVar194@
2E958	TOLVar157!	2E97E	TOLVar176!	2E9A4	TOLVar195!
2E959	TOLVar157@	2E97F	TOLVar176@	2E9A5	TOLVar195@
2E95A	TOLVar158!	2E980	TOLVar177!	2E9A6	TOLVar196!

2E9A7	TOLVar196@	2E9CD	TOLVar215@	2EE64	SetDAsTemp
2E9A8	TOLVar197!	2E9CE	TOLVar216!	2EE65	SetDA12a3NoCh
2E9A9	TOLVar197@	2E9CF	TOLVar216@	2EE65	SetDA12a3NCh
2E9AA	TOLVar198!	2E9D4	TOLVarN!	2EE66	DA2aLess1OK?
2E9AB	TOLVar198@	2E9F8	TOLVarN@	2EE67	SetDA1Valid
2E9AC	TOLVar199!	2EA1C	ClrAllTVars	2EE68	SetDA2bValid
2E9AD	TOLVar199@	2EA52	ClrAllTOLVs	2EE69	SetDA1Temp
2E9AE	TOLVar200!	2EA6E	%0AllTopicVs	2EE6A	SetDA2bTemp
2E9AF	TOLVar200@	2EAA9	%0AllTOLVars	2EE6B	SetDA3Temp
2E9B0	TOLVar201!	2EAE4	TOLVarSet!	2EE6C	SetDA2aEcho
2E9B1	TOLVar201@	2EB11	SaveTOLVarSet	2EE6D	ClrDAsOK
2E9B2	TOLVar202!	2EB66	RestTOLVarSet	2EE6E	ClrDA3OK
2E9B3	TOLVar202@	2EBB1	%0TOLVarSet	2EE6F	SetDA12NoCh
2E9B4	TOLVar203!	2EC01	1getcxt!	2EE70	SetDA13NoCh
2E9B5	TOLVar203@	2EC15	DoInCxt	2EE71	SetDA12Temp
2E9B6	TOLVar204!	2EC6F	DoInCalcCxt	2EE72	SetDA1NoCh
2E9B7	TOLVar204@	2EC88	DoInAppCxt	2EE73	SetDA2aNoCh
2E9B8	TOLVar205!	2ECA1	DoInFuncCxt	2EE74	ClrDA1Bad
2E9B9	TOLVar205@	2ECBA	DoInPolarCxt	2EE75	ClrDA2aBad
2E9BA	TOLVar206!	2ECD3	DoInParamCxt	2EE76	SetDA2bNoCh
2E9BB	TOLVar206@	2ECEC	DoInSeqCxt	2EE77	SetDA3NoCh
2E9BC	TOLVar207!	2ED05	DoInStatCxt	2EE78	SetDA1Bad
2E9BD	TOLVar207@	2ED1E	DoInSolveCxt	2EE79	SetDA2aBad
2E9BE	TOLVar208!	2ED37	DoInOtherCxt	2EE7A	SetDA2bBad
2E9BF	TOLVar208@	2ED91	DoInOtherN	2EE7B	SetDA3Bad
2E9C0	TOLVar209!	2EDD7	DoInOtherU	2EE7C	SetDAsNoCh
2E9C1	TOLVar209@	2EE04	otherNG?	2EE7D	ClrDA1IsStat
2E9C2	TOLVar210!	2EE37	GET@tTYPER	2EE7E	DA2bIsEdL?
2E9C3	TOLVar210@	2EE5A	DispEditLine	2EE7F	SetDA2bIsEdL
2E9C4	TOLVar211!	2EE5B	DispTime?	2EE80	ClrDA2bIsEdL
2E9C5	TOLVar211@	2EE5C	BlankDA12	2EE81	ClrDA2bNoCh
2E9C6	TOLVar212!	2EE5D	?FlashAlert	2EE82	DA2aOK?
2E9C7	TOLVar212@	2EE5E	SysErrorTrap	2EE83	SetDA2aBadT
2E9C8	TOLVar213!	2EE5F	SysErrorTrapConf..	2EE84	DA2bOK?
2E9C9	TOLVar213@	2EE60	DoWarning	2EE85	SetDA2bBadT
2E9CA	TOLVar214!	2EE61	FlashWarning	2EE86	DA2OK?
2E9CB	TOLVar214@	2EE62	DA1OK?	2EE87	SetDA3BadT
2E9CC	TOLVar215!	2EE63	DA3OK?	2EE88	DAsOK?

2EE8A	SetDA2aTemp	2EEC1	DOBAUD	2EEE8	InitEdModes
2EE8B	MENoP&FixDA1	2EEC2	DOPARITY	2EEE9	EditString
2EE8D	ClrDA1OK	2EEC3	DOTRANSIO	2EEEA	CURSOR_END?
2EE8E	ClrDA2aOK	2EEC4	DOKERRM	2EEEB	EDITLINE$
2EE8F	ClrDA2bOK	2EEC5	DOBUFLEN	2EEEC	EDITPARTS
2EE90	ClrDA2OK	2EEC6	DOSBRK	2EEED	NoEditLine?
2EE91	SetDA2Valid	2EEC7	DOSRECV	2EEEE	APPprompt1!
2EE92	SetDAsValid	2EEC8	FLUSHRSBUF	2EEEF	AUTOSCALE
2EE93	SetDA2NoCh	2EEC9	CLOSEUART	2EEF0	PromptIdUtil
2EE94	SetDA23NoCh	2EECA	docr	2EEF1	PUTSCALE
2EE97	SetDA1ValidF	2EECB	DOCR	2EEF2	PUTINDEP
2EEA0	SetDA3ValidF	2EECC	DOPRLCD	2EEF3	PUTINDEPLIST
2EEA3	SetDA2aTempF	2EECD	DODELAY	2EEF4	PUTRES
2EEA5	SetDA2bTempF	2EECE	SetEcma94	2EEF5	GETPTYPE
2EEA6	DA2bTemp?	2EECF	TOD	2EEF6	PUTPTYPE
2EEA7	ClrDA2bTemp	2EED0	DATE	2EEF7	VSCALE
2EEA9	SetDA3TempF	2EED1	DDAYS	2EEF8	HSCALE
2EEAB	DA1IsStatus?	2EED2	DATE+DAYS	2EEF9	DOERASE
2EEAC	SetDA1IsStat	2EED3	TIMESTR	2EEFA	CROSS_HAIRS
2EEAD	NoRollDA2?	2EED4	Clr8	2EEFB	CROSS_OFF
2EEAE	SetNoRollDA2	2EED5	Clr8-15	2EEFC	MENUOFF
2EEAF	ClrNoRollDA2	2EED6	HBUFF_X_Y	2EEFD	MENUOFF?
2EEB0	DA1Bad?	2EED7	SysTime	2EEFE	CURRENTMARK?
2EEB1	DA2aBad?	2EED7	CLKTICKS	2EEFF	DispCoord1
2EEB2	DA2bBad?	2EED9	SYMBNUMSOLVE	2EF01	DOPX>C
2EEB3	ClrDA2bBad	2EEDA	STATCLST	2EF02	DOC>PX
2EEB4	DA3Bad?	2EEDC	STATN	2EF03	DOLCD>
2EEB5	ClrDA3Bad	2EEDD	STATSMAX	2EF04	DO>LCD
2EEB6	ClrDA3NoCh	2EEDE	STATMEAN	2EF05	DOCLLCD
2EEB7	DA2bNoCh?	2EEDF	STATSMIN	2EF06	CKPICT
2EEB9	DA2aNoCh?	2EEE0	STATSTDEV	2EF07	nmetasyms
2EEBA	DA1NoCh?	2EEE1	STATTOT	2EF26	SYMSHOW
2EEBB	SENDLIST	2EEE2	STATVAR	2EF53	SYMSQ
2EEBC	GETNAME	2EEE3	EchoChrKey	2EF59	MENP&FixDA12
2EEBD	DOFINISH	2EEE4	Echo$Key	2EF5A	apndvarlst
2EEBE	DOPKT	2EEE5	EditLevel1	2EF5E	BlankDA1
2EEBF	GetIOPAR	2EEE6	InitEd&Modes	2EF5F	InputLine
2EEC0	DOOPENIO	2EEE7	InitEdLine	2EF60	DOGRAPHIC

2EF61	WINDOW#	2EF8A	CMD_COPY	2EFB6	bitRL
2EF62	palparse	2EF8B	STO_CURS_POS	2EFB7	bitRLB
2EF66	SysMenuCheck	2EF8C	STO_CURS_POS2	2EFB8	bitASR
2EF67	SysDisplay	2EF8D	STO_CURS_POS3	2EFB9	bit+
2EF68	ClrDouseAlm	2EF8E	STO_CURS_POS4	2EFBA	bit-
2EF69	EvalParsed	2EF8F	STO_CURS_POS_VIS	2EFBC	bit*
2EF6A	Parse.1	2EF90	CAL_CURS_POS_VIS	2EFBD	bit/
2EF6B	Parse.2	2EF91	CAL_CURS_POS	2EFBE	WORDSIZE
2EF6C	AtUserStack	2EF92	XLINE_SIZE?	2EFBF	BASE
2EF6D	GetLastEdit	2EF93	VERIF_SELECTION	2EFC0	HXS>$
2EF6E	ParseFail	2EF94	PASTE.EXT	2EFC1	hxs>$
2EF6F	DispBadToken	2EF95	DEL_CMD	2EFC2	bit%#/
2EF70	ParseFail2	2EF96	NO_AFFCMD	2EFC3	bit#%/
2EF71	DispBadToken2	2EF97	InsertEcho	2EFC4	bit%#*
2EF72	CacheStack	2EF98	SetDA2aValid	2EFC5	bit#%*
2EF73	?Space/Go>	2EF99	SetDA3Valid	2EFC6	bit%#-
2EF74	CMD_PLUS	2EF9A	CommandLineHeight	2EFC7	bit#%-
2EF75	AddTrailingSpace	2EF9F	LINEON	2EFC8	bit%#+
2EF76	AddLeadingSpace	2EFA0	LINEOFF	2EFC9	bit#%+
2EF77	CMD_PAGEL	2EFA1	TOGLINE	2EFCA	HXS>%
2EF78	CMD_PAGER	2EFA2	LINEON3	2EFCB	%>#
2EF79	CMD_PAGEU	2EFA3	LINEOFF3	2EFCC	HXS==HXS
2EF7A	CMD_PAGED	2EFA4	TOGLINE3	2EFCD	HXS>HXS
2EF7B	CMD_BAK	2EFA5	DOHEX	2EFCE	HXS>=HXS
2EF7C	CMD_NXT	2EFA6	DOBIN	2EFCF	HXS<HXS
2EF7D	CMD_DEB_LINE	2EFA7	DOOCT	2EFDB	GROB+
2EF7E	CMD_END_LINE	2EFA8	DODEC	2EFEC	symbn
2EF7F	CMD_UP	2EFAA	dostws	2F002	Ticks>TOD
2EF80	CMD_DOWN	2EFAC	bitAND	2F003	Ticks>Date
2EF81	CMD_DROP	2EFAD	bitOR	2F004	Ticks>Rpt
2EF82	CMD_DEL	2EFAE	bitXOR	2F007	getxpos
2EF83	CMD_STO_DEBUT	2EFAF	bitNOT	2F008	getypos
2EF84	CMD_STO_FIN	2EFB0	bitSL	2F019	UNIT>$
2EF85	RCL_CMD_DEB	2EFB1	bitSLB	2F031	TURNMENUON
2EF86	RCL_CMD_FIN	2EFB2	bitSR	2F034	TURNMENUOFF
2EF87	RCL_CMD_POS	2EFB3	bitSRB	2F038	Save16
2EF88	CMD_CUT	2EFB4	bitRR	2F03B	>DATE
2EF89	CUT.EXT	2EFB5	bitRRB	2F05E	SaveLastEdit

2F062	StoIOPAR	2F09A	TempConv	2F153	CLKADJ*
2F063	StoPRTPAR	2F09B	Rcl&Edit	2F154	Ck&Input1
2F064	Sys@	2F09C	Rcl&View	2F155	Ck&Input2
2F066	STOAPPLDATA	2F09D	Roll&Edit	2F158	ChrAtCur
2F073	SWAPcompSWAP	2F09E	Roll&View	2F15A	CHOOSE_INIT
2F075	InitSysUI	2F0A1	RESETDEPTH	2F15B	CLEARMENU
2F076	puretemp?	2F0AC	PURGALARM%	2F15E	Clr16
2F07A	UM>U	2F0BC	PRINT	2F162	CHECKPICT
2F07B	U>nbr	2F0C5	PLOTPREP	2F163	CHECKPVARS
2F07C	UM#?	2F0D4	NotIDorLAM?	2F16D	Blank$
2F07D	UM%	2F0D5	NEWINDEP	2F177	AllowPrlcdCl
2F07E	UM%CH	2F0DB	MAKEPICT#	2F178	ALARMS@
2F07F	UM%T	2F0E6	KERMOPEN	2F179	AdjEdModes
2F080	UM*	2F0E7	InitIOEnv	2F17A	APPprompt2
2F081	UM+	2F0E8	INDEPVAR	2F17E	2HXSLIST?
2F082	UM-	2F0EC	ICMPDRPRTDRP	2F180	1REV
2F083	UM/	2F0EE	HXS#HXS	2F190	DcompWidth@
2F085	UM<=?	2F0EF	HXS<=HXS	2F191	!DcompWidth
2F086	UM<?	2F0FE	GETXMAX	2F192	DoNewEqw
2F087	UM=?	2F0FF	GETXMIN	2F193	UobROT
2F088	UM>=?	2F100	GETYMIN	2F194	CMD_PLUS2
2F089	UM>?	2F105	GDISPCENTER	2F195	CMD_PLUS3
2F08A	UMABS	2F106	GETINDEP	2F196	RCL_CMD
2F08B	UMCHS	2F107	GETPMIN&MAX	2F197	RCL_CMD2
2F08C	UMCONV	2F108	GETRHS	2F198	STO_CMD_MODE
2F08D	UMCOS	2F109	GETXPOS	2F199	RCL_CMD_MODE
2F08E	UMMAX	2F10A	GetRes	2F19A	ViewLevel1
2F08F	UMMIN	2F10D	GETRES	2F19B	OngoingText?
2F090	UMSI	2F10E	GETYMAX	2F19E	DispCommandLine
2F091	UMSIN	2F110	FINDVAR	2F19F	?DispCommandLine
2F092	UMSQ	2F113	FNDALARM{}	2F1A1	ErrorHandled?
2F093	UMSQRT	2F11C	Echo$NoChr00	2F1A3	SysErrorTrapAction
2F094	UMTAN	2F12E	DOSTIME	2F1A5	AskQuestion
2F095	UMU>	2F130	DOXMIT	2F1A7	CHARSEDIT
2F096	UMXROOT	2F13C	DoOldMatrix	2F1A8	EditFont
2F097	UM^	2F13D	DIFF_OR_ZERO	2F1A9	EDITF
2F098	Unbr>U	2F13F	DRAWLINE#3	2F1AB	Date>hxs13
2F099	U>NCQ	2F142	DoNewMatrix	2F1AC	StrEdit

2F1AD	CharEdit	2F265	UPDIR	2F2F1	DO>Del
2F1AE	ObEdit	2F266	USER$>TAG	2F2F2	FindStrInCmd
2F1AF	AlgObEdit	2F267	VARSIZE	2F2F3	GET.W->
2F1BF	Decomp%Short	2F268	Wait/GetKey	2F2F4	GET.W<-
2F1C6	DROP3PICK	2F292	XEQIOBACKUP	2F2F5	PUT_FONTE
2F1D5	MATR>C	2F296	XEQORDER	2F2F6	GET_CUR_FONT.EXT
2F1D6	MATC>R	2F297	XEQPURGEPICT	2F2F7	PUT_STYLE
2F205	laMGET0	2F2A3	XEQRCL	2F2F8	EXEC_CMD
2F215	CircleB	2F2A7	XEQSETLIB	2F2F9	DODEL.L
2F216	CircleG1	2F2A9	XEQSHOWLS	2F2FA	CMD_COPY.SBR
2F217	CircleG2	2F2C0	XEQSUB$	2F2FB	EVAL.SELECTION
2F218	CircleW	2F2C6	XEQXRCL	2F2FC	REPLACEALLNOSCREEN
2F219	CircleXor	2F2D4	dowait	2F2FF	OpenIO
2F21A	Dither	2F2D5	EVLNCKSTO	2F300	DispILPrompt
2F21B	ToGray	2F2D5	EVALNOCKSTO	2F312	OpenUartClr
2F21C	Lift	2F2DA	AlgCharEdit	2F313	OpenUart?Clr
2F21D	ViewObject	2F2DB	DOTEXTINFO	2F314	RCLALARM%
2F21E	ViewStrObject	2F2DC	ClearSelection	2F315	!#1+IF<dim-1
2F21F	ViewGrobObject	2F2DD	DoFarBS	2F316	!#1-IF>0
2F222	#1+ROT	2F2DE	DoFarDel	2F318	1GETLAMSWP1+
2F223	%>TAG	2F2DF	EditSelect	2F319	ACK_INIT
2F237	DOAPWL	2F2E0	ViewEditGrob	2F31A	APNDCRLF
2F23E	DOSTOSYSF	2F2E1	SELECT.LINE	2F31B	BlankDA2
2F242	EXPR>	2F2E2	SELECT.LINEEND	2F31C	BlankDA2a
2F244	Flag%isUser?	2F2E3	EVAL.LINE	2F31D	BOTROW
2F24E	LISTRCL	2F2E4	DO>BEG	2F31E	BUILDKPACKET
2F257	OCRC%	2F2E5	DO>END	2F31F	C%>#
2F258	PICTRCL	2F2E6	GOTOLABEL	2F320	CHECKHEIGHT
2F259	RCLSYSF	2F2E7	SELECT.FONT	2F321	CkChr00
2F25A	RCLSYSF2	2F2E8	DOFIND	2F324	CKGROBFITS
2F25B	RCLUSERF	2F2E9	DOREPL	2F325	ClrServMode
2F25C	RCLUSERF2	2F2EA	DONEXT	2F326	CMDSTO
2F25D	SETROMPART	2F2EB	DOREPLACE	2F327	convertbase
2F25E	SPLITEQ	2F2EC	DOREPLACE/NEXT	2F328	CROSSMARKON
2F25F	STOSYSF	2F2ED	REPLACEALL	2F329	Date>d$
2F260	STOSYSF2	2F2EE	DO<Skip	2F32A	DECODE
2F261	STOUSERF	2F2EF	DO>Skip	2F32B	DISPCOORD2
2F262	STOUSERF2	2F2F0	DO<Del	2F32C	DRAWBOX#

2F32D	drax	2F353	LINECHANGE	2F379	SetDA123NoCh
2F32E	DROPDEADTRUE	2F354	List	2F37A	SetDA2OKTemp
2F32F	DropSysErr$	2F355	MAKEPVARS	2F37B	SetIOPARErr
2F330	ENCODE	2F356	metatail	2F37C	SETLOOPENV
2F331	ENCODE1PKT	2F357	newBASE	2F37D	SetServMode
2F332	EQCURSOR?	2F358	NEWMARK	2F37E	SORTASLOW
2F333	EXCHINITPK	2F359	NEXTRRPOB	2F37F	STOALM
2F334	Extobcode	2F35A	NEXTSTEP	2F380	SysSTO
2F335	FcnUtilEnd	2F35B	NUMSOLVE	2F381	TOD>t$
2F336	FindNext	2F35C	OB>BAKcode	2F382	TOGGLELINE#3
2F337	FixRRP	2F35D	OpenIOPrt	2F383	TOP16
2F338	GetChkPRTPAR	2F35E	PLOTERR	2F384	TOP8
2F339	GetEqN	2F35F	PlotOneMore?	2F385	TOPROW
2F33A	GetKermPkt#	2F360	PREMARKON	2F386	TRPACKETFAIL
2F33B	GETKP	2F361	PrintGrob	2F387	UARTBUFLEN
2F33C	getmatchtok	2F362	PRINTxNLF	2F388	VerifyTOD
2F33D	GETPARAM	2F363	PtoR	2F389	VERSTRING
2F33E	GETSCALE	2F364	PUTSERIAL	2F38A	WINDOWBOT?
2F33F	GETSERIAL	2F365	PUTXMAX	2F38B	WINDOWLEFT?
2F340	GETYPOS	2F366	PUTXMIN	2F38C	WINDOWRIGHT?
2F341	GraphicExit	2F367	PUTYMAX	2F38D	WINDOWTOP?
2F342	GROB+#	2F368	PUTYMIN	2F38E	xnsgeneral
2F343	IncrLAMPKNO	2F369	RECORDX&YC%	2F38F	xsngeneral
2F344	InputLAttn	2F36A	REMAP	2F390	xssgeneral
2F345	InputLEnter	2F36B	RIGHTCOL	2F3A7	SEND_PACKET
2F346	IOCheckReal	2F36C	Rows8-15	2F3A8	RecvNextPkt
2F347	JUMPBOT	2F36D	SCROLLDOWN	2F3A9	STOALLFcont
2F348	JUMPLEFT	2F36E	SCROLLLEFT	2F3AA	STOALLFcont2
2F349	JUMPRIGHT	2F36F	SCROLLRIGHT	2F3B3	StoUserKeypatch
2F34A	JUMPTOP	2F370	SCROLLUP	2F3B6	Restore16
2F34B	KDispRow2	2F371	SENDACK	2F3BF	IsApple
2F34C	KDispStatus2	2F372	SENDEOT	2F3C0	IsMidApple
2F34D	KINVISLF	2F373	SENDERROR	2F3C1	IsBigApple
2F34E	KVIS	2F374	SENDNAK	2F3CF	Save16Patch
2F34F	KVISLF	2F375	SENDNULLACK	2F3D0	Rest16Patch
2F350	'LamKPSto	2F376	SENDPKT	2F458	SETIVLERR
2F351	LASTPT?	2F377	SendSetup	2F47D	PACKSB
2F352	LEFTCOL	2F378	SetCursor	2F4A2	PACK

2F62C	POP1%SPLITA	2FC19	%%10	2FF71	%.05
2F636	POP1%	2FC7D	%1200	2FF86	%.99
2F65E	POP2%	2FC92	%2400	2FF9B	%%>%
2F7E4	PUSH%	2FCA7	%4800	2FFAC	%>%%
2F899	PUSH%LOOP	2FCBC	%9600	2FFBD	SETDEG
2F937	%0	2FCD1	%15360	2FFDB	SETRAD
2F94C	%1	2FCD1	%15396	2FFEF	SETGRAD
2F961	%2	2FCE6	%11	3000D	%D>R
2F976	%3	2FCFB	%12	30017	PI/180
2F98B	%4	2FD10	%13	30040	%R>D
2F9A0	%5	2FD25	%14	3005E	%>HMS
2F9B5	%6	2FD3A	%15	30077	%HMS>
2F9CA	%7	2FD4F	%16	3008B	%HMS+
2F9DF	%8	2FD64	%17	300B3	%HMS-
2F9F4	%9	2FD79	%18	300C7	%%MAX
2FA09	%-1	2FD8E	%19	300E0	%MAX
2FA1E	%-2	2FDA3	%20	300F9	%MIN
2FA33	%-3	2FDB8	%21	30112	%%0<
2FA48	%-4	2FDCD	%22	30123	%0<
2FA5D	%-5	2FDE2	%23	30145	%%0=
2FA72	%-6	2FDF7	%24	30156	%0=
2FA87	%-7	2FE0C	%25	30173	%%0>
2FA9C	%-8	2FE21	%26	30184	%0>
2FAB1	%-9	2FE36	%27	301A6	%%0<>
2FAC6	%PI	2FE4B	%28	301BA	%0<>
2FADB	%%PI	2FE60	%29	301CE	%%0>=
2FAF5	%MAXREAL	2FE75	%30	301E2	%0>=
2FB0A	%-MAXREAL	2FE8A	%31	301F6	%%0<=
2FB1F	%MINREAL	2FE9F	%32	3020A	%%<
2FB34	%-MINREAL	2FEB4	%33	3025C	%<
2FB49	%%0	2FEC9	%34	3026A	%%>
2FB63	%%1	2FEDE	%35	30275	%>
2FB7D	%%2	2FEF3	%.461368	30280	%%>=
2FB97	%%3	2FF08	%50	3028B	%>=
2FBB1	%%4	2FF1D	%.2887	30296	%%<=
2FBCB	%%5	2FF32	%.522851	302A1	%<=
2FBE5	%%.1	2FF47	%.2776	302AC	%=
2FBFF	%%.5	2FF5C	%.2943	302B7	%<>

302C2	%SGN	305F1	%%SIN	309AD	%RAN
302DB	%%ABS	30602	%%SINDEG	30A2F	%RANDOMIZE
302EB	%ABS	30612	%%SINRAD	30A66	DORANDOMIZE
302FB	%%CHS	3062B	%COS	30AAF	%FACT
3030B	%CHS	30642	%%COS	30B24	%-260
3031B	%MANTISSA	30653	%%COSDEG	30BEA	%%7
3032E	%%+	30663	%%COSRAD	30CC7	%%12
3033A	%%-	3067C	%TAN	30CEB	%%60
30346	%>%%-	30693	%%TANRAD	30DC8	%%.4
3035F	%+	306AC	%ASIN	30E47	2%>%%
3036C	%-	306C3	%%ASINRAD	30E5B	2%%>%
30385	%%*	306DC	%ACOS	30E79	%REC>%POL
303A7	%*	306F3	%%ACOSRAD	30E83	%%R>P
303B4	%OF	3070C	%ATAN	30EA6	%POL>%REC
303D3	%%/	30723	%ANGLE	30EB0	%%P>R
303E9	%/	3073A	%%ANGLE	30EDD	%SPH>%REC
303F6	%T	30746	%>%%ANGLE	30F14	RNDXY
3041B	%CH	30757	%%ANGLEDEG	30F28	TRCXY
3044A	%%^	30767	%%ANGLERAD	31066	aMODF
3045B	%^	30780	%%SINH	31123	aH>HMS
3046C	%NROOT	30799	%SINH	31131	SPLITA
3047D	%%1/	307B2	%%COSH	31187	SPLTAC
30489	%>%%1/	307C5	%COSH	31193	SPLITC
3049A	%1/	307D8	%TANH	31219	Y<=X
304D5	%%SQRT	307EB	%ASINH	3125D	TST15
304E1	%>%%SQRT	307FE	%ACOSH	3133A	XYEX
304F4	%SQRT	30811	%ATANH	31348	STAB0
30507	%%EXP	30824	%EXPONENT	31356	STAB2
3051A	%EXP	30837	%NFACT	31364	STCD0
3052D	%EXPM1	3084D	%COMB	31372	STCD2
30546	%%LN	30860	%PERM	31380	EXAB0
30559	%LN	30912	%%H>HMS	3138E	EXAB2
3056C	%LOG	30938	%FP	3139C	RCAB0
3057F	%%LNP1	3094B	%IP	313A7	RCAB2
30592	%LNP1	3095E	%CEIL	313B2	RCCD0
305A5	%ALOG	30971	%FLOOR	313BD	RCCD2
305C7	%MOD	30984	%%FLOOR	313C8	CCSB1
305DA	%SIN	30984	%%INT	313D3	RNDC[B]

314CA	GETAB1	3314D	BINT7	331CF	BINT20
314E4	GETAB0	3314D	SEVEN	331CF	TWENTY
31518	GETCD0	33157	seco	331D9	TWENTYONE
31532	PUTAB0	33157	EIGHT	331D9	BINT21
31568	1/X15	33157	BINT8	331E3	BINT22
31586	RSUB1	33161	symb	331E3	TWENTYTWO
3158F	RADD1	33161	BINT9	331ED	BINT23
315A9	RADDF	33161	NINE	331ED	TWENTYTHREE
315BB	ADDF	3316B	sym	331F7	BINT24
316FD	MULTF	3316B	BINT10	331F7	TWENTYFOUR
31756	DIVF	3316B	TEN	33201	BINT25
317EE	SQRF	33175	hxs	33201	TWENTYFIVE
31994	DIV2	33175	BINT11	3320B	REALSYM
319C1	CLRFRC	33175	ELEVEN	3320B	BINT26
33107	any	3317F	BINT12	3320B	TWENTYSIX
33107	ZERO	3317F	grob	33215	TWENTYSEVEN
33107	BINT0	3317F	TWELVE	33215	BINT27
33111	real	33189	TAGGED	3321F	TWENTYEIGHT
33111	BINT1	33189	BINT13	3321F	BINT28
33111	MEMERR	33189	THIRTEEN	33229	BINT29
33111	ONE	33193	EXT	33229	TWENTYNINE
3311B	BINT2	33193	BINT14	33233	BINT30
3311B	TWO	33193	FOURTEEN	33233	REALEXT
3311B	cmp	33193	unitob	33233	THIRTY
33125	str	3319D	BINT15	3323D	THIRTYONE
33125	BINT3	3319D	rompointer	3323D	BINT31
33125	THREE	3319D	FIFTEEN	33247	BINT32
3312F	FOUR	331A7	REALOB	33247	THIRTYTWO
3312F	BINT4	331A7	BINT16	33251	BINT33
3312F	arry	331A7	SIXTEEN	33251	THIRTYTHREE
33139	FIVE	331B1	SEVENTEEN	3325B	THIRTYFOUR
33139	BINT5	331B1	2REAL	3325B	BINT34
33139	list	331B1	REALREAL	33265	BINT35
33143	id	331B1	BINT17	33265	THIRTYFIVE
33143	BINT6	331BB	EIGHTEEN	3326F	TTHIRTYSIX
33143	SIX	331BB	BINT18	3326F	BINT36
33143	idnt	331C5	NINETEEN	33279	THIRTYSEVEN
3314D	lam	331C5	BINT19	33279	BINT37

33283	THIRTYEIGHT	3332D	FIFTYFIVE	333F5	BINT75
33283	BINT38	33337	BINT56	333FF	BINT76
3328D	BINT39	33337	FIFTYSIX	33409	BINT77
3328D	THIRTYNINE	33341	FIFTYSEVEN	33413	BINT78
33297	FOURTY	33341	BINT57	3341D	SEVENTYNINE
33297	BINT40	3334B	FIFTYEIGHT	3341D	BINT79
33297	FORTY	3334B	BINT58	33427	EIGHTY
332A1	FORTYONE	33355	BINT59	33427	BINT80
332A1	BINT41	33355	FIFTYNINE	33431	LISTREAL
332AB	BINT42	3335F	SIXTY	33431	EIGHTYONE
332AB	FORTYTWO	3335F	BINT60	33431	BINT81
332B5	BINT43	33369	SIXTYONE	3343B	BINT82
332B5	FORTYTHREE	33369	BINT61	3343B	LISTCMP
332BF	BINT44	33373	BINT62	33445	BINT83
332BF	FORTYFOUR	33373	SIXTYTWO	33445	FIVETHREE
332C9	FORTYFIVE	3337D	BINT63	3344F	BINT84
332C9	BINT45	3337D	SIXTYTHREE	3344F	FIVEFOUR
332D3	BINT46	33387	SIXTYFOUR	33459	BINT85
332D3	FORTYSIX	33387	BINT64	33459	2LIST
332DD	BINT47	33387	BINT40h	33463	BINT86
332DD	FORTYSEVEN	33387	YHI	33463	FIVESIX
332E7	BINT48	33391	BINT65	3346D	BINT87
332E7	FORTYEIGHT	33391	ARRYREAL	3346D	LISTLAM
332F1	BINT49	3339B	FOURTWO	33477	BINT88
332F1	FORTYNINE	3339B	BINT66	33481	BINT89
332FB	BINT50	333A5	FOURTHREE	3348B	BINT90
332FB	FIFTY	333A5	BINT67	33495	BINT_91d
33305	FIFTYONE	333AF	SIXTYEIGHT	33495	BINT91
33305	BINT51	333AF	BINT68	3349F	BINT92
3330F	BINT52	333B9	BINT69	334A9	BINT93
3330F	FIFTYTWO	333B9	FOURFIVE	334B3	BINT94
33319	BINT53	333C3	SEVENTY	334BD	BINT95
33319	THREEFIVE	333C3	BINT70	334C7	BINT_96d
33319	STRLIST	333CD	BINT71	334C7	BINT96
33319	FIFTYTHREE	333D7	BINT72	334D1	BINT97
33323	FIFTYFOUR	333E1	BINT73	334D1	IDREAL
33323	BINT54	333EB	BINT74	334DB	BINT98
3332D	BINT55	333EB	SEVENTYFOUR	334E5	BINT99

334EF	ONEHUNDRED	3361B	BINT_130d	3376F	Err#Kill
334EF	BINT100	3361B	BINT130	33779	Err#NoLstStk
334F9	BINT101	33625	XHI	33783	#NoRoomForSt
33503	BINT102	33625	BINT131	3378D	#132
3350D	BINT103	33625	BINT131d	33797	REALSTRSTR
33517	BINT104	33625	BINT_131d	337A1	#134
33521	BINT105	3362F	#8F	337AB	#135
3352B	BINT106	33639	SYMBREAL	337B5	#136
33535	BINT107	33643	SYMBCMP	337BF	#137
3353F	BINT108	3364D	SYMBSYM	337C9	#138
33549	BINT109	33657	SYMBUNIT	337D3	#139
33553	BINT110	33661	backup	337DD	#13A
3355D	char	3366B	SYMOB	337E7	#13B
3355D	BINT111	33675	SYMREAL	337F1	#13D
33567	BINT112	3367F	SYMCMP	337FB	Err#Cont
33571	BINT113	33689	SYMLIST	33805	INTEGER337
3357B	BINT114	33693	SYMID	3380F	CMPOBOB
33585	BINT_115d	3369D	SYMLAM	33819	Err#NoLstArg
33585	BINT115	336A7	SYMSYMB	33823	STRREALREAL
3358F	BINT_116d	336B1	SYMSYM	3382D	ARRYREALREAL
3358F	BINT116	336BB	SYMEXT	33837	ARRYREALCMP
33599	BINT117	336C5	HXSREAL	33841	3ARRY
335A3	BINT118	336CF	2HXS	3384B	ARRYLISTREAL
335AD	BINT119	336D9	BINTC0h	33855	ARRYLISTCMP
335B7	BINT120	336E3	2GROB	3385F	LISTREALOB
335C1	BINT121	336ED	TAGGEDANY	33869	LISTREALREAL
335CB	BINT122	336F7	EXTREAL	33873	LISTLISTOB
335CB	BINT_122d	33701	EXTSYM	3387D	IDREALOB
335D5	BINT123	3370B	2EXT	33887	IDLISTOB
335DF	BINT124	33715	ROMPANY	33891	FSTMACROROM#
335E9	BINT125	3371F	BINT253	3389B	PROGIDREAL
335F3	BINT126	33729	BINT255d	338A5	PROGIDCMP
335FD	BINT127	33733	REALOBOB	338AF	PROGIDLIST
33607	BINT80h	3373D	#_102	338B9	PROGIDEXT
33607	BINT128	33747	#SyntaxErr	338C3	ATTNERR
33611	BINT129	33751	BINT_263d	338CD	SYMREALREAL
3361B	BINT130d	3375B	REALREALOB	338D7	SYMREALCMP
3361B	XHI-1	33765	3REAL	338E1	SYMREALSYM

338EB	SYMCMPREAL	33AD7	tok<<	33D1F	$_...
338F5	SYMCMPCMP	33AE3	tokexponent	33D2B	CHR_00
338FF	SYMCMPSYM	33AEF	tokanglesign	33D32	CHR_...
33909	SYMIDREAL	33AFB	tokSIGMA	33D39	CHR_DblQuote
33913	SYMIDCMP	33B07	tokWHERE	33D40	CHR_#
3391D	SYMIDLIST	33B13	14SPACES$	33D47	CHR_*
33927	SYMIDEXT	33B39	NEWLINE$	33D4E	CHR_+
33931	SYMSYMREAL	33B45	$DER	33D55	CHR_,
3393B	SYMSYMCMP	33B55	tok_	33D5C	CHR_-
33945	3SYM	33B55	SPACE$	33D63	CHR_.
3394F	XFERFAIL	33B61	tokUNKNOWN	33D6A	CHR_/
33959	PROTERR	33B79	tokquote	33D71	CHR_0
33963	InvalServCmd	33B85	tok'	33D78	CHR_1
3396D	Connecting	33B91	tok,	33D7F	CHR_2
33977	Retry	33B9D	tok.	33D86	CHR_3
33981	#CAlarmErr	33BA9	tok;	33D8D	CHR_4
3398B	EXTOBOB	33BB5	toklparen	33D94	CHR_5
33995	#EXITERR	33BC1	tokrparen	33D9B	CHR_6
3399F	MINUSONE	33BCD	tok^	33DA2	CHR_7
339A9	%e	33BD9	tok*	33DA9	CHR_8
339BE	%.5	33BE5	tok/	33DB0	CHR_9
339D3	%-.5	33BF1	tok+	33DB7	CHR_:
339E8	%10	33BFD	tok-	33DBE	CHR_;
339FD	%180	33C09	tok=	33DC5	CHR_<
33A12	%200	33C15	tokSQRT	33DCC	CHR_=
33A27	%360	33C21	tokDER	33DD3	CHR_>
33A3C	%400	33C2D	tokCTGROB	33DDA	CHR_A
33A51	tok]	33C3F	tokCTSTR	33DE1	CHR_B
33A5D	lbrac	33C4D	tok0	33DE8	CHR_C
33A6B	tok[	33C59	tok1	33DEF	CHR_D
33A77	tok{	33C65	tok2	33DF6	CHR_E
33A83	tok}	33C71	tok3	33DFD	CHR_F
33A8F	toksharp	33C7D	tok4	33E04	CHR_G
33A9B	tokuscore	33C89	tok5	33E0B	CHR_H
33AA7	tok$	33C95	tok6	33E12	CHR_I
33AB3	tok&	33CA1	tok7	33E19	CHR_J
33ABF	tokESC	33CAD	tok8	33E20	CHR_K
33ACB	tok>>	33CB9	tok9	33E27	CHR_L

33E2E	CHR_M
33E35	CHR_N
33E3C	CHR_O
33E43	CHR_P
33E4A	CHR_Q
33E51	CHR_R
33E58	CHR_S
33E5F	CHR_T
33E66	CHR_U
33E6D	CHR_V
33E74	CHR_W
33E7B	CHR_X
33E82	CHR_Y
33E89	CHR_Z
33E90	CHR_a
33E97	CHR_b
33E9E	CHR_c
33EA5	CHR_d
33EAC	CHR_e
33EB3	CHR_f
33EBA	CHR_g
33EC1	CHR_h
33EC8	CHR_i
33ECF	CHR_j
33ED6	CHR_k
33EDD	CHR_l
33EE4	CHR_m
33EEB	CHR_n
33EF2	CHR_o
33EF9	CHR_p
33F00	CHR_q
33F07	CHR_r
33F0E	CHR_s
33F15	CHR_t
33F1C	CHR_u
33F23	CHR_v
33F2A	CHR_w
33F31	CHR_x
33F38	CHR_y
33F3F	CHR_z
33F46	CHR_->
33F4D	CHR_<<
33F54	CHR_>>
33F5B	CHR_Angle
33F62	CHR_Deriv
33F69	CHR_Integral
33F70	CHR_LeftPar
33F77	CHR_Newline
33F7E	CHR_Pi
33F85	CHR_RightPar
33F8C	CHR_Sigma
33F93	CHR_Space
33F9A	CHR_UndScore
33FA1	CHR_[
33FA8	CHR_]
33FAF	CHR_{
33FB6	CHR_}
33FBD	CHR_<=
33FC4	CHR_>=
33FCB	CHR_<>
33FD2	$_R<<
33FE2	$_R<Z
33FF2	$_XYZ
34002	$_<<>>
34010	$_{}
3401E	$_[]
3402C	$_’’
3403A	$_::
34048	$_LRParens
34056	$_2DQ
34064	$_ECHO
34076	$_EXIT
34088	$_Undefined
340A4	$_RAD
340B4	$_GRAD
340CB	tokVersion
34133	tokCopyright
34144	RSWAP
3416E	XYZ>YXZ
3416E	ROTSWAP
34195	XYZ>ZY
34195	ROTDROPSWAP
341A8	XYZ>YZ
341A8	ROTDROP
341BA	XYZ>ZYX
341BA	UNROTSWAP
341BA	SWAPROT
341D2	3DROP
341D2	XYZ>
341D7	4DROP
341D7	XYZW>
341DC	5DROP
341E8	6DROP
341F4	7DROP
34202	4DropLoop
3421A	XY>Y
3421A	SWAPDROP
3422B	3UNROLL
3422B	UNROT
3422B	XYZ>ZXY
3423A	XYZW>YZWX
3423A	4ROLL
3423A	FOURROLL
34257	5ROLL
34257	FIVEROLL
34281	6ROLL
34281	SIXROLL
342BB	EIGHTROLL
342BB	8ROLL
342EA	SEVENROLL
342EA	7ROLL
34318	9ROLL
3432C	DUP4UNROLL
34331	FOURUNROLL

34331	XYZW>WXYZ	3457F	SWAPOVER	346CA	19GETLAM
34331	4UNROLL	34611	1PUTLAM	346CF	20PUTLAM
34357	5UNROLL	34616	1GETLAM	346D4	20GETLAM
34357	FIVEUNROLL	3461B	2PUTLAM	346D9	21PUTLAM
3438D	6UNROLL	34620	2GETLAM	346DE	21GETLAM
3438D	SIXUNROLL	34625	3PUTLAM	346E3	22PUTLAM
343BD	XYZ>Z	3462A	3GETLAM	346E8	22GETLAM
343BD	UNROT2DROP	3462F	4PUTLAM	346ED	23PUTLAM
343BD	ROTROT2DROP	34634	4GETLAM	346F2	23GETLAM
343CF	4UNROLL3DROP	34639	5PUTLAM	346F7	24PUTLAM
343CF	XYZW>W	3463E	5GETLAM	346FC	24GETLAM
343E1	2RDROP	34643	6PUTLAM	34701	25PUTLAM
343F3	3RDROP	34648	6GETLAM	34706	25GETLAM
34405	#-PICK	3464D	7PUTLAM	3470B	26PUTLAM
34417	#+PICK	34652	7GETLAM	34710	26GETLAM
34431	DUP#1+PICK	34657	8PUTLAM	34715	27PUTLAM
34436	#1+PICK	3465C	8GETLAM	3471A	27GETLAM
34451	#2+PICK	34661	9PUTLAM	3471F	DUP1PUTLAM
34465	#3+PICK	34666	9GETLAM	34724	1GETLAMSWAP
34474	#4+PICK	3466B	10PUTLAM	34729	DUP2PUTLAM
34485	3PICK	34670	10GETLAM	3472E	2GETLAMSWAP
3448A	4PICK	34675	11PUTLAM	34797	DUP4PUTLAM
3448F	5PICK	3467A	11GETLAM	347AB	DUPTEMPENV
34494	6PICK	3467F	12PUTLAM	3483E	GETLAMPAIR
34499	7PICK	34684	12GETLAM	348D2	#=case
3449E	8PICK	34689	13PUTLAM	348E2	OVER#=case
344A3	9PICK	3468E	13GETLAM	348F7	DUP#0=case
344A8	10PICK	34693	14PUTLAM	348FC	#0=case
344CB	#-ROLL	34698	14GETLAM	3490E	DUP#0=csedrp
344DD	#+ROLL	3469D	15PUTLAM	34920	EQcasedrop
344F2	#1+ROLL	346A2	15GETLAM	34939	#=casedrop
34504	get1	346A7	16PUTLAM	3494E	NOTcasedrop
34517	#2+ROLL	346AC	16GETLAM	3495D	casedrop
3452B	#-UNROLL	346B1	17PUTLAM	34976	NOTcase2drop
3453D	#+UNROLL	346B6	17GETLAM	34985	case2drop
34552	#1+UNROLL	346BB	18PUTLAM	34999	EQcase
34564	#2+UNROLL	346C0	18GETLAM	349B1	caseDROP
3457F	DUPUNROT	346C5	19PUTLAM	349C6	NOTcaseDROP

349D6	case2DROP	35064	DUPTYPELIB?	35159	TYPERRP?
349EA	NOTcase2DROP	35069	TYPELIB?	35163	DUPTYPESYMB?
349F9	case	35073	DTYPEMATRIX?	35168	TYPESYMB?
34A13	NOTcase	35073	DUPTYPEMATRIX?	35172	DUPTYPECOL?
34A22	IT	35078	TYPEMATRIX?	35172	DTYPECOL?
34A31	GOTO	35082	DUPTYPEFLASHPTR?	35177	TYPECOL?
34A46	?GOTO	35087	TYPEFLASHPTR?	35181	DUPTYPEGROB?
34A59	NOT?GOTO	35091	DUPTYPEZINT?	35186	TYPEGROB?
34A68	popflag	35096	TYPEZINT?	35190	DTYPELIST?
34A7E	#0=?SEMI	350A0	DUPTYPELNGREAL?	35190	DUPTYPELIST?
34A92	NOT?SEMI	350A5	TYPELNGREAL?	35195	TYPELIST?
34AA1	?SEMI	350AF	DUPTYPELNGCMP?	3519F	DUPTYPETAG?
34AAD	SEMILOOP	350B4	TYPELNGCMP?	351A4	TYPETAGGED?
34ABE	ITE_DROP	350BE	DUPTYPEFONT?	351AE	DUPTYPEEXT?
34AD3	COLA_EVAL	350C3	TYPEFONT?	351B3	TYPEEXT?
34AF4	COLARPITE	350CD	DUPTYPEAPLET?	351BD	DUPTYPEEXT0?
34B3E	ITE	350D2	TYPEAPLET?	351C2	TYPEEXT0?
34B4F	2'RCOLARPITE	350DC	DUPTYPELAM?	351F0	GPOverWrT/FL
34BAB	2@REVAL	350E1	TYPELAM?	351F3	GPOverWrTLp
34BBB	3@REVAL	350EB	DUPTYPEBINT?	351FA	OverWrF/TLp
34BD8	NOT?DROP	350F0	TYPEBINT?	351FD	OverWrTLoop
34BEF	ticR	350F5	#37258	35213	GPOverWrFLp
34C82	EXPAND	350FA	DUPTYPEHSTR?	3521A	OverWrT/FLp
34D00	CACHE	350FF	TYPEHSTR?	3521D	OverWrFLoop
34D51	SAVELAM	35109	DUPTYPECSTR?	35233	GPPushT/FLp
34D58	SAVESTACK	35109	DTYPECSTR?	35236	GPPushTLoop
34EBE	DUMP	3510E	TYPECSTR?	3523D	PushF/TLoop
34FA6	undo	35118	DUPTYPEREAL?	35240	PushTLoop
34FC0	DUPROM-WORD?	35118	DTYPEREAL?	3524F	GPPushFLoop
34FCD	ROM-WORD?	3511D	TYPEREAL?	35256	PushT/F
34FE6	Rom-Word?	35127	DUPTYPECMP?	35256	PushT/FLoop
35018	2SWAP	3512C	TYPECMP?	35259	PushFLoop
35037	DUPTYPECHAR?	35136	DTYPEARRY?	35268	OVER#=
3503C	TYPECHAR?	35136	DUPTYPEARRY?	35280	DROPTRUE
35046	DUPTYPEIDNT?	3513B	TYPEARRY?	35289	DROPFALSE
3504B	TYPEIDNT?	35145	DUPTYPEROMP?	35292	TYPERARRY?
35055	DUPTYPEBAK?	3514A	TYPEROMP?	352AD	TYPECARRY?
3505A	TYPEBAK?	35154	DUPTYPERRP?	352BD	DUP#0=

352E0	#3=	356B8	#6*	359E3	ORcase
352F1	#2=	356D5	5skipcola	359F7	REQcase
352FE	#1=	35703	3skipcola	35A10	REQcasedrop
3530D	#1<>	3570C	2skipcola	35A29	SAFESTO
3531C	DUP#1=	35715	skipcola	35A56	DUPSAFE@
3532B	DUP#0<>	3571E	DUP#2+	35A5B	SAFE@
3533C	!insert$	35733	DROPSWAP	35A88	?>ROMPTR
35346	SWAP&$	3574D	XYZ>Y	35AAB	?ROMPTR>
35369	!!append$?	3574D	DROPSWAPDROP	35AE2	MACRODCMP
353CD	!append$	3574D	ROT2DROP	35B32	2DROPFALSE
353EB	!!insert$	3576E	SWAPDUP	35B46	PALPTRDCMP
353F7	!!append$	3579C	ROTDUP	35B82	palrompdcmp
354CB	'RSAVEWORD	357BB	SWAP#-	35B96	#0=UNTIL
354CB	'RSaveRomWrd	357CE	DROPDUP	35BAF	INCOMPDROP
35511	#MIN	357E2	DUPLEN$	35BC3	NTHCOMPDROP
3551D	#MAX	357FC	#+DUP	35BD7	APPEND_SPACE
35552	#-#2/	35812	Push2#aLoop	35BEB	7UNROLL
3558C	DROPZERO	3581F	#-DUP	35BFF	RESOROMP
355A5	2DROP00	35830	#1+DUP	35C18	%10*
355C1	#9-	35841	#1-DUP	35C2C	DUP@
355C6	#8-	35857	SWAPDROPDUP	35C40	DUPROMPTR@
355CB	#7-	35872	SWAPDROPSWAP	35C54	#=ITE
355D0	#6-	35872	XYZ>ZX	35C68	INNERDUP
355D5	#5-	35872	UNROTDROP	35C7C	NOTAND
355DA	#4-	3588B	4ROLLDROP	35C90	TOTEMPSWAP
355DF	#3-	358A7	5ROLLDROP	35CA4	ROT2DUP
355FD	#3+	358C2	2DUP#<	35CB8	ROTAND
35602	#4+	358DC	2DUP#=	35CCC	ROTOVER
35607	#5+	358F8	2DUP#>	35CE0	DUPDUP
3560C	#6+	35912	DUP#1+	35CF4	OVERDUP
35611	#7+	3592B	SWP1+	35D08	COERCEDUP
35616	#8+	3592B	SWAP#1+	35D1C	UNROTDUP
3561B	#9+	35956	DUP#1-	35D30	2DUPSWAP
35620	#10+	3596D	DROPONE	35D30	DUP3PICK
35625	#11+	3597F	RDROPCOLA	35D44	4UNROLLDUP
3562A	#12+	35994	COLACOLA	35D58	NTHCOMDDUP
35675	#10*	359AD	COLAcase	35D6C	OVERUNROT
3569B	#8*	359C8	COLANOTcase	35D6C	OVERSWAP

35D80	ROLLSWAP
35D94	NULL$SWAP
35DA8	SUB$SWAP
35DBC	%MAXorder
35DDA	?SKIPSWAP
35DEE	1ABNDSWAP
35E07	ROT+SWAP
35E07	ROT#+SWAP
35E20	4PICK+SWAP
35E20	4PICK#+SWAP
35E39	#+SWAP
35E4D	#-SWAP
35E61	#1+SWAP
35E75	ZEROSWAP
35E89	#1-1SWAP
35EA2	ONESWAP
35EB6	COERCESWAP
35ECA	%>%%SWAP
35EDE	%%*SWAP
35EF2	XYZ>ZTRUE
35F06	4ROLLSWAP
35F1A	3PICKSWAP
35F2E	4PICKSWAP
35F42	1GETSWAP
35F56	?SWAP
35F6A	!append$SWAP
35F7E	NOT?SWAPDROP
35F97	?SWAPDROP
35FB0	#1+NDROP
35FB0	N+1DROP
35FC4	ROLLDROP
35FD8	MDIMSDROP
35FF3	DUPROT
36007	DROPROT
3601B	#1-ROT
3602F	%%*ROT
36043	FOURROLLROT
36043	4ROLLROT
36057	4UNROLLROT
3606B	DROPOVER
3607F	EQOVER
36093	#+OVER
360A7	#-OVER
360BB	ZEROOVER
360CF	UNROTOVER
360E3	4ROLLOVER
360F7	3PICKOVER
3610B	4PICKOVER
3611F	DUPPICK
36133	DUPROLL
36147	OVER#2+UNROL
3615B	8UNROLL
3616F	10UNROLL
36183	OVERARSIZE
3619E	'ERRJMP
361B2	caseERRJMP
361C6	?CARCOMP
361DA	NEWLINE$&$
361DA	NEWLINE&$
361EE	#1-{}N
36202	TWO{}N
36216	THREE{}N
3622A	DUPINCOMP
3623E	SWAPINCOMP
36252	DUPNULL$?
36266	DUPNULLCOMP?
3627A	DUPLENCOMP
3628E	#1-SUB$
362A2	1_#1-SUB
362A2	1_#1-SUB$
362B6	LAST$
362CA	#1+LAST$
362DE	DUP$>ID
362F2	SWAP%>C%
36306	'NOP
3631A	::NEVAL
3632E	2GETEVAL
36342	DROPRDROP
3635B	SWAPCOLA
3636F	XYZ>ZCOLA
36383	#0=?SKIP
3639C	#1=?SKIP
363B5	#=?SKIP
363CE	ONE_EQ
363E2	#>?SKIP
363FB	COLASKIP
3640F	NOT_UNTIL
36428	NOT_WHILE
36441	DUP#0<>WHILE
3645A	DUPINDEX@
3646E	SWAPINDEX@
36482	OVERINDEX@
36496	SWAPLOOP
364AF	DROPLOOP
364C8	DUP#0_DO
364E1	toLEN_DO
364FF	1GETABND
36513	DUP1LAMBIND
36518	1LAMBIND
3652C	caseTRUE
36540	TRUEFALSE
36540	TrueFalse
36554	FALSETRUE
36554	FalseTrue
36568	ZEROFALSE
3657C	ONEFALSE
36590	#=casedrpfls
365B3	casedrpfls
365CC	case2drpfls
365E5	caseFALSE
365F9	ORNOT
3660D	EQUALNOT
36621	2DUPEQ
36635	DUPEQ:

3663A	EQ:	368E7	GROB!ZERODRP	36C22	SWAP%%/
3664E	EQOR	368FB	casedrptru	36C36	caseDrpBadKy
36662	EQUALOR	36914	NOTcaseTRUE	36C4F	caseDEADKEY
36676	2#0=OR	3692D	?SEMIDROP	36C4F	caseDoBadKey
36694	OVER#0=	36946	SWAPUnDROP	36C68	GROBDIMw
366A8	OVER#<	3695A	SWAPUnNDROP	36C7C	%%*UNROT
366BC	#<3	3696E	DUP'	36C90	XYZW>YWZX
366D0	DUP#<7	36982	SWAP'	36C90	SWAP4ROLL
366E9	INNER#1=	36996	DROP'	36CA4	2DUP5ROLL
366FD	#5=	369AA	OVER'	36CB8	SWAP3PICK
36711	#2<>	369BE	STO'	36CCC	3PICK3PICK
36725	OVER#>	369D2	TRUE'	36CE0	SWAP4PICK
36739	ONE#>	369E6	ONEFALSE'	36CF4	OVER5PICK
36739	#>1	369FF	FALSE'	36D08	EQUALcasedrp
3674D	DUP3PICK#+	36A13	#1+'	36D21	DUP#0=csDROP
3674D	2DUP#+	36A27	'R'R	36D3A	jEQcase
36761	ROT#+	36A4A	'RRDROP	36D4E	ANDcase
36775	OVER#+	36A63	ONECOLA	36D62	EQUALcase
36789	3PICK#+	36A77	dvarlsBIND	36D76	#<case
3679D	4PICK#+	36A8B	'LAMLNAMESTO	36D8A	#1=case
367B1	ROT#-	36AA4	'xDEREQ	36D9E	#<>case
367C5	OVER#-	36ABD	DUPNULL{}?	36DB2	#>2case
367D9	INDEX@#-	36AD6	DUPZERO	36DCB	#>case
367ED	SWAPOVER#-	36AEA	DUPONE	36DDF	j%0=case
36801	ROT#1+	36AFE	SWAPONE	36DF3	REALcase
36815	#-+1	36B12	ONEDUP	36E07	dARRYcase
36815	#1--	36B12	ONEONE	36E2F	dZINTcase
36829	SWAP#1-	36B26	DUPTWO	36E43	dLISTcase
3683D	DROP#1-	36B3A	NOTcsdrpfls	36E57	EditExstCase
36851	#+-1	36B53	caseSIZEERR	36E6B	ANDNOTcase
36851	$1-+	36B67	NcaseSIZEERR	36E7F	EQUALNOTcase
36851	#1-+	36B7B	CKREAL	36E93	dIDNTNcase
36865	COLAITE	36BAA	NcaseTYPEERR	36EA7	dREALNcase
36883	ERROROUT	36BBE	'x*	36EBB	EQIT
36897	D0=DSKTOP	36BD2	'xDER	36ED4	DUP#0=IT
368A6	D1=DSKTOP	36BE6	%%/>%	36EED	ANDITE
368B5	SWAP2DUP	36BFA	UNCOERCE%%	36F01	EQITE
368C9	RSKIP	36C0E	DUP%0=	36F15	#0=ITE

36F29	#<ITE	3733A	#TWO#TWO	37B54	NEXTCOMPOB
36F3D	#>ITE	3734A	#TWO#FOUR	37C06	>LASTRAM-WORD
36F51	DUP#0=ITE	3735C	#THREE#FOUR	37F48	xIF
36F65	UserITE	3736E	#FIVE#FOUR	37F5C	tokIF-prompt
36F79	SysITE	37380	ZEROZEROZERO	37F7F	xTHEN
36F8D	top&Cr	37394	ZEROZEROONE	3805D	xELSE
36FA6	metaROTDUP	373A8	ZEROZEROTWO	3807D	xIFEND
36FBA	ROTUntop&	373D0	UNPICK	38093	xALG->
36FCE	roll2top&	37408	#1+UNPICK	380DB	xWHILE
36FCE	rolltwotop&	3741A	#+UNPICK	38105	xREPEAT
36FE2	plDRPpZparg	3742B	#1-UNPICK	3816B	xDO
36FF6	&$SWAP	37466	#<=	38195	xUNTIL
3700A	SWAPCKREF	3747D	#>=	381AB	xSTART
3701E	pZpargSWAPUn	374AA	SWAPFALSE	38252	xSTARTVAR
37032	DROPNDROP	374BE	SWAPDROPTRUE	38266	#FFFF
37046	2OVER	3760D	SubMetaOb	38275	#BB
3705A	?Ob>Seco	37685	SubMetaOb1	3831C	xNEXT
37073	Ob>Seco	376B7	matchob?	3851F	xSTEP
37087	2Ob>Seco	376C1	matchob?Lp	387AC	xIFERR
3709B	ExitAtLOOP	376EE	POSCOMP	3880D	xHALT
3709B	ZEROISTOPSTO	37702	nextpos	38837	xSILENT’
370AF	RclHiddenVar	37711	3DROPZERO	3885C	xRPN->
370C3	WithHidden	37752	#=POSCOMP	38999	x>>ABND
37104	StoHiddenVar	3776B	EQUALPOSCOMP	389B9	x<<
37118	PuHiddenVar	37784	NTHOF	389D4	x>>
3712C	SaveVarRes	37798	Find1stTrue	389EF	x’
3714A	SetHiddenRes	377C5	Lookup	38A14	xENDTIC
37186	RestVarRes	377DE	Lookup.1	38A2F	xWHILEEND
371B3	Embedded?	37829	EQLookup	38A54	xENDDO
371F9	UStackDepth	378FA	POS$	38ABA	xERRTHEN
3721C	Sig?ErrJmp	378FA	POSCHR	38B28	xCASE
37226	ListErrspecial	37906	POSCHRREV	38B43	xTHENCASE
37258	DupAndThen	37906	POS$REV	38BAE	xDIR
37287	ZEROZERO	37A78	CHR_A8	38BBF	xPROMPT
37294	#ZERO#ONE	37AA5	CHR>$	38C00	DoPrompt
37305	#ZERO#SEVEN	37ABE	STRIPTAGS	38C1B	xGROB
37315	#ONE#27	37AEB	STRIPTAGSl2	38C2C	xEVAL>
37328	#TWO#ONE	37B04	TAGOBS	38D2F	xNOEVAL>

38D72	xSTRUCT>	393CA	xCRDIR	39B3B	xi
38D83	x<STRUCT	393EA	xPATH	39B58	x+
38D94	xSTRUCT->	39405	xHOME	39C79	hxs70107
38DE1	xASR	39420	xUPDIR	39C8B	SWAP>HCOMP
38E01	xRL	3943B	xVARS	39C9F	$,Ob>$’
38E21	xRLB	39456	xTVARS	39CB3	Ob,$>$’
38E41	xRR	39480	xBYTES	39CD5	xNEGNEG
38E61	xRRB	394AA	xNEWOB	39CFC	x-
38E81	xSL	394C8	INHARDROM?	39DE8	x*
38EA1	xSLB	394F1	xKILL	39E6B	SYMARRY
38EC1	xSR	3950C	xOFF	39F2E	hxs80108
38EE1	xSRB	39527	xDOERR	39F49	x/
38F01	xR>B	3955B	xERR0	3A07D	ParseDataPdiv
38F21	xB>R	39576	xERRN	3A097	x^
38F41	xCONVERT	39591	xERRM	3A12D	#4FF
38F81	xUVAL	395AC	xEVAL	3A17F	ParseDataN^
38FB5	x>UNIT	395F3	xIFTE	3A18E	ParseDataP^
38FD7	xUBASE	39666	hxs0140626250	3A1C2	#304
3900B	xUFACT	396A4	xIFT	3A200	rpnXROOT
3900B	UMFACT	39705	xSYSEVAL	3A278	xXROOT
3905D	xTIME	39725	xDISP	3A2FA	SWAPUMXROOT
39078	xDATE	39745	xFREEZE	3A30E	SWAP%NROOT
39093	xTICKS	39765	xBEEP	3A32B	xINV
390AE	xWSLOG	39785	x>NUM	3A390	xARG
390C9	xACKALL	397E5	xLAST	3A3D1	%0%ANGLE
390E4	xACK	39819	xWAIT	3A3EE	xSIGN
39104	xSETDATE	39839	xCLLCD	3A442	xSQRT
39124	xSETTIME	39854	xKEY	3A4B0	PDataNSQRT
39144	xCLKADJ	3989C	xCONT	3A4BE	%2root
39164	xSTOALARM	398B9	x=	3A4EF	xSQ
3918E	xRCLALARM	39976	xNEG	3A54B	%SQ
391AE	xFINDALARM	399ED	CHSpdata	3A57C	xSIN
391D8	xDELALARM	39A07	xABS	3A5D0	xCOS
391F8	xTSTR	39A6C	xCONJ	3A624	xTAN
39218	xDDAYS	39AC7	xPI	3A678	xSINH
39238	xDATE+	39AE4	xMAXR	3A6C2	xCOSH
39277	#B437D	39B01	xMINR	3A70C	xTANH
39332	?GetMsg	39B1E	xCONSTANTe	3A756	xASIN

3A7DC	xACOS	3B2A6	SWAPUM%	3B928	#411
3A844	xATAN	3B2DC	x%T	3B93D	#415
3A88E	xASINH	3B362	x%CH	3B952	#451
3A8D8	xACOSH	3B3E6	xRAND	3B967	#855
3A94F	xATANH	3B401	xRDZ	3B976	#822
3A9B7	xEXP	3B423	xCOMB	3B9D2	xREPL
3AA01	xLN	3B477	xPERM	3B9FA	#313
3AA73	xLOG	3B4C9	xSF	3BA09	#515
3AAE5	xALOG	3B4E9	xCF	3BA18	#454
3AB2F	xLNP1	3B509	xFS?	3BA2D	#414
3AB6F	xEXPM	3B529	xFC?	3BAC1	xLIST>
3ABAF	xFACT	3B549	xDEG	3BADA	XEQLIST>
3ABD2	hxsB010	3B564	xRAD	3BAF5	xC>R
3ABFD	preFACT	3B57F	xGRAD	3BB1F	xSIZE
3AC3D	xIP	3B59A	xFIX	3BB94	xPOS
3AC87	xFP	3B5BA	xSCI	3BBBE	x>STR
3ACD1	xFLOOR	3B5DA	xENG	3BBD9	xSTR>
3AD1B	xCEIL	3B5FA	xSTD	3BBF9	xNUM
3AD65	xXPON	3B615	xFS?C	3BC19	xCHR
3ADA5	xMAX	3B635	xFC?C	3BC39	xTYPE
3AE2B	xMIN	3B655	xBIN	3BC43	XEQTYPE
3AEB1	xRND	3B670	xDEC	3BD4C	#AF
3AF3E	xTRNC	3B68B	xHEX	3BD65	#CF
3AFCB	xMOD	3B6A6	xOCT	3BDB2	xVTYPE
3B02E	xMANT	3B6C1	xSTWS	3BDE6	xEQ>
3B06E	xD>R	3B6FA	xRCWS	3BE38	xOBJ>
3B0AE	xR>D	3B715	xRCLF	3BE9B	x>ARRY
3B0EC	x>HMS	3B749	xSTOF	3BEC5	xARRY>
3B10C	xHMS>	3B76C	DOSTOALLF	3BEEC	xRDM
3B12C	xHMS+	3B7AD	#BBBB	3BF77	xCON
3B14C	xHMS-	3B7D2	x>LIST	3C02E	xIDN
3B16C	xRNRM	3B7ED	xR>C	3C084	xTRN
3B193	xCNRM	3B819	xRE	3C0BF	xPUT
3B1BA	xDET	3B87E	xIM	3C10F	ARRYLISTOB
3B1E1	xDOT	3B8D7	xSUB	3C11E	ARRYREALOB
3B208	xCROSS	3B8F5	#C55	3C139	xPUTI
3B22F	xRSD	3B904	#C22	3C16B	#750
3B251	x%	3B913	#455	3C17A	#710

3C1C7	xGET	3C83C	#82C	3D0BC	xOLDPRT
3C22D	xGETI	3C866	xLCD>	3D0D7	xPR1
3C2AC	xV>	3C881	x>LCD	3D0F2	xPRSTC
3C2D6	x>V2	3C8A1	x>GROB	3D10D	xPRST
3C30A	x>V3	3C8C6	xARC	3D128	xCR
3C33E	xINDEP	3C8D0	#2111	3D143	xPRVAR
3C372	xPMIN	3C8DF	#5B11	3D1C7	xDELAY
3C392	xPMAX	3C8FA	xTEXT	3D1E7	xPRLCD
3C3B2	xAXES	3C915	xXRNG	3D202	x∂
3C3DC	xCENTR	3C935	xYRNG	3D258	xDER
3C41A	xRES	3C955	xFUNCTION	3D28F	hxs0134250
3C444	x*H	3C967	xCONIC	3D2B4	CKSYMBTYPE
3C464	x*W	3C979	xPOLAR	3D393	xRCEQ
3C484	xDRAW	3C98B	xPARAMETRIC	3D3AE	xSTEQ
3C49F	xAUTO	3C99D	xTRUTH	3D3CE	xROOT
3C4BA	xDRAX	3C9AF	xSCATTER	3D434	x$\int$
3C4D5	xSCALE	3C9C1	xHISTOGRAM	3D47E	xINTEGRAL
3C4F5	xPDIM	3C9D3	xBAR	3D497	INTGPDATA
3C51F	xDEPND	3C9E5	xSAME	3D50D	SYMRRANY
3C553	xERASE	3CA07	xAND	3D51C	SYMSYMRANY
3C56E	xPX>C	3CA52	hxs50105	3D52B	SYMRSYMANY
3C58E	xC>PX	3CA61	XEQAND	3D549	SUMETCPDATA
3C5AE	xGRAPH	3CA8D	xOR	3D56B	x|
3C5C9	xLABEL	3CAD8	hxs40104	3D605	xWHERE
3C5E4	xPVIEW	3CAE7	XEQOR	3D619	hxs2214370B50
3C60E	xPIXON	3CB13	xNOT	3D6F6	xQUOTE
3C638	xPIXOFF	3CB4A	hxs0105	3D719	hxs014250
3C662	xPIX?	3CB5D	XEQNOT	3D7AC	xAPPLY
3C68C	xLINE	3CB7A	xXOR	3D7C0	hxs014360950
3C6B6	xTLINE	3CBCA	XEQXOR	3D81D	xFCNAPPLY
3C6E0	xBOX	3CBF6	x==	3DA3E	x->Q
3C70A	xBLANK	3CCA5	hxs60106	3DA63	x->QPI
3C72A	xPICT	3CCB4	SAME	3DAD0	xMATCHUP
3C74A	xGOR	3CD21	x#?	3DB04	xMATCHDN
3C7D8	xGXOR	3CE42	x<	3DB62	xFORMUNIT
3C7E2	#C5C	3CEE1	x>	3DB8F	hxsA0127
3C800	#C2C	3CF80	x<=?	3DBCA	xPREDIV
3C81E	#85C	3D01F	x>=?	3DBEA	xDUP

3DC05	xDUP2	3E03D	xXCOL	3E759	#8FD
3DC20	xSWAP	3E05D	xYCOL	3E7DA	#C8
3DC3B	xDROP	3E07D	xUTPC	3E7E9	#9F1
3DC56	xDROP2	3E09D	xUTPN	3E7FF	#8F1
3DC71	xROT	3E0BD	xUTPF	3E823	xSTO>
3DC8C	xOVER	3E0DD	xUTPT	3E85C	xDEFINE
3DCA7	xDEPTH	3E0FD	xSIGMACOL	3E87C	xPURGE
3DCC7	xDROPN	3E127	xSCLSIGMA	3E8C1	xMEM
3DCE2	xDUPN	3E156	xSIGMALINE	3E8F0	xORDER
3DCFD	xPICK	3E171	xBINS	3E91A	xCLUSR
3DD18	xROLL	3E17B	#111	3E97B	xTMENU
3DD33	xROLLD	3E196	xBARPLOT	3E9D4	xMENU
3DD4E	xCLEAR	3E1CA	xHISTPLOT	3EA01	ID_CST
3DD6E	xSTOSIGMA	3E1EF	xSCATRPLOT	3EA2E	xRCLMENU
3DD8E	xCLSIGMA	3E214	xLINFIT	3EA49	xPVARS
3DDA9	xRCLSIGMA	3E239	xLOGFIT	3EAA7	xPGDIR
3DDC4	xSIGMA+	3E25E	xEXPFIT	3EAC7	xARCHIVE
3DDEE	xSIGMA-	3E283	xPWRFIT	3EAE7	xRESTORE
3DE09	xNSIGMA	3E2C1	xBESTFIT	3EAFB	#9F
3DE24	xCORR	3E331	xSINV	3EB16	xMERGE
3DE3F	xCOV	3E35B	xSNEG	3EB2C	xFREE
3DE75	xSUMY	3E385	xSCONJ	3EB42	xLIBS
3DE90	xSUMX2	3E3AF	xSTO+	3EB64	xATTACH
3DEAB	xSUMY2	3E406	xSTO-	3EB84	xDETACH
3DEC6	xSUMXY	3E46C	xSTO/	3EB9D	dREALcase
3DEE1	xMAXSIGMA	3E4D2	xSTO*	3EC35	xXMIT
3DEFC	xMEAN	3E54C	xINCR	3EC55	xSRECV
3DF17	xMINSIGMA	3E576	xDECR	3EC75	xOPENIO
3DF32	xSDEV	3E5A0	xCOLCT	3EC95	xCLOSEIO
3DF4D	xTOT	3E5E9	xEXPAN	3ECB0	xSEND
3DF68	xVAR	3E632	xRULES	3ECE4	xKGET
3DF83	xLR	3E648	xISOL	3ED22	xRECN
3DF92	ListIntSlp	3E66F	xQUAD	3ED56	xRECV
3DF97	tokIntercept	3E696	xSHOW	3ED76	xFINISH
3DFB3	tokSlope	3E6CA	xTAYLR	3ED91	xSERVER
3DFDD	xPREDV	3E6F1	xRCL	3EDAC	xCKSM
3DFFD	xPREDY	3E739	xSTO	3EDCC	xBAUD
3E01D	xPREDX	3E743	#9FD	3EDEC	xPARITY

3EE0C	xTRANSIO	8000A	HOMEMASK	8053B	PORT1EOS
3EE2C	xKERRM	8000F	HRAMEND	80540	PORT2EOS
3EE47	xBUFLEN	80010	FAILSTK1	805DB	INTRAM
3EE62	xSTIME	80022	FAILSTK2	805EB	SAVE_MODES
3EE82	xSBRK	80034	FAILSTK3	805F0	SAVE_C[A]
3EE9D	xPKT	80046	FAILSTK4	805F5	SAVE_A
3EEBD	xINPUT	80058	NEXTIRQ	80605	SAVE_ST
3EEE7	xASN	80065	TIMECRC	80608	SAVE_B
3EF07	xSTOKEYS	80069	TIMExmit	80618	SAVE_D
3EF3B	xDELKEYS	80069	TIMEOUT	80628	SAVE_R0
3EF79	xRCLKEYS	80076	TIMEOUTCLK	80638	SAVE_PC
3EF97	ID_S	80077	LoBatTime	8063D	SAVE_D0
3EFB1	x->TAG	80078	StartTime	80642	SAVE_OR
3EFEF	xDTAG	80085	FailTime	80642	ORghost
3F007	xINT	80092	TESTMSG	80645	DRSTART
3F033	xANS	800BE	SW_Image	8064A	DREND
3F053	x;	800D4	SW_ETime	8064F	IREG
3F070	xR>I	800E1	PortStat	80652	SEMAPH
3F0B7	xI>R	800E2	Port1CRC	80654	IOSAVE
3F0FC	xNOVAL	800E6	AccessInit	80655	CSPEED
3F11C	xCMDAPPLY	800E8	COVERstate	8065A	INITEN
3F218	xRPL>	800EB	COVERsave	8065B	DISABLE_KBD
3F22E	xUNROT	800F5	IRAMBUFF	8065B	HANDSHK
3F249	xUNPICK	80127	IRAMBUFF2	8065C	KEYSTATE
3F264	xNIP	8030E	GraphPrtHook	80669	KEYBUFFER
3F27F	xPICK3	8030E	IRAMBEND	80669	INPUTSTREAM
3F29A	xDUPDUP	80319	uart_buffer	8068B	POPPEDKEY
3F2B5	xNDUPN	80519	uart_buf_end	8068D	DISP1CTLg
3F2DF	xFAST3D	8051B	uart_error	80692	LINENIBSg
3F2EA	DUPXEQRCL	8051C	uart_buf_st	80695	DISP2CTLg
3F33F	CKARRY	8051E	uart_handshk	8069A	LINECOUNTg
3F3C1	CKLIST	8051F	uart_modes	8069C	GreyOn?
3F481	COERCE2	80520	uart_parity	8069C	Stk0save
3F495	UNCOERCE2	80521	uart_timeout	8069D	GreyScr1
80000	RAMSTART	80523	IOCNIB	806A1	Stk1save
80000	CMOS	80524	CONFRAM	806A2	GreySoft1
80000	HARDROMEND	8052B	CONFTAB	806A6	Stk2save
80005	IRAMMASK	80536	PORT0EOS	806A7	GreyScr2

806AB	Stk3save	8073E	NOTESCXT	807ED	MenuDef
806AC	GreySoft2	80743	apletPTR	807F2	LastMenuDef
806B0	Stk4save	80748	funcPTR	807F7	MenuData
806B1	GreyScr3	8074D	polarPTR	807FC	MenuRowAct
806B5	Stk5save	80752	paramPTR	80801	LabelDef
806B6	GreySoft4	80757	seqPTR	80806	MenuKeyNS
806BA	R2[A]save	8075C	statPTR	8080B	MenuKeyLS
806BF	R2[S]save	80761	solvePTR	80810	MenuKeyRS
806C0	R1[A]save	80766	otherPTR	80815	ReviewKey
806C5	SAVE_BO	8076B	INTRPPTR	8081A	LastContext
806C6	SAVE_LC	8076B	OBUPSTART	8081F	TrackAct
806C8	SAVE_LN	80770	OSAVE	80824	MenuExitAct
806CB	SAVE_OFFSET	80775	LASTARG	80829	LASTROMWDOB
806D0	VDISP2	80775	LASTARG1	8082E	KeyOb
806D5	ADISP	8077A	LASTARG2	80833	FlagMBox
806DA	SYSUPSTART	8077F	LASTARG3	80838	ViewMBox
806DA	VDISP	80784	LASTARG4	8083D	ProgMBox
806DA	VDISP1	80789	LASTARG5	80842	Title
806DF	VDISP3	8078E	leeway	80847	HiLitePtr
806E4	GDISP	80793	ITEM1STATE	8084C	WindowPtr
806E9	TEMPOB	80798	HISTORY1	80851	HStackPtr
806EE	TEMPTOP	8079D	HISTORY2	80856	HStackTop
806F3	RSKTOP	807A2	HISTORY3	8085B	GraphContext
806F8	DSKTOP	807A7	HISTORY4	8086A	TopicVar1
806FD	EDITLINE	807AC	PDCHXS	8086F	TopicVar2
80702	TEMPENV	807B1	PDCSYMB	80874	TopicVar3
80707	DOLPENV	807B1	KERMERRM	80879	TopicVar4
8070C	TOUCHTAB	807B6	PAINTTREE	8087E	TopicVar5
80711	USEROB	807BB	EXITMSG	80883	TopicVar6
80716	ROMPARTS	807C0	AppDisplay	80888	TopicVar7
8071B	CONTEXT	807C5	AppKeys	8088D	TopicVar8
80720	STOPSIGN	807CA	AppExitCond	80892	TopicVar9
80725	UserKeys	807CF	AppError	80897	TopicVar10
8072A	ALARMS	807D4	AppSuspend	8089C	TopicVar11
8072F	FSTVGERPTR	807D9	AppResume	808A1	TopicVar12
8072F	VSTACK	807DE	AppCursor	808A6	TopicVar13
80734	CALCCXT	807E3	AppDoKeyOb	808AB	TopicVar14
80739	PGMCXT	807E8	CtlAlarm	808B0	TopicVar15

808B5	TopicVar16	80973	TopicVar54	80A31	TOLVar1
808BA	TopicVar17	80978	TopicVar55	80A36	TOLVar2
808BF	TopicVar18	8097D	TopicVar56	80A3B	TOLVar3
808C4	TopicVar19	80982	TopicVar57	80A40	TOLVar4
808C9	TopicVar20	80987	TopicVar58	80A45	TOLVar5
808CE	TopicVar21	8098C	TopicVar59	80A4A	TOLVar6
808D3	TopicVar22	80991	TopicVar60	80A4F	TOLVar7
808D8	TopicVar23	80996	TopicVar61	80A54	TOLVar8
808DD	TopicVar24	8099B	TopicVar62	80A59	TOLVar9
808E2	TopicVar25	809A0	TopicVar63	80A5E	TOLVar10
808E7	TopicVar26	809A5	TopicVar64	80A63	TOLVar11
808EC	TopicVar27	809AA	TopicVar65	80A68	TOLVar12
808F1	TopicVar28	809AF	TopicVar66	80A6D	TOLVar13
808F6	TopicVar29	809B4	TopicVar67	80A72	TOLVar14
808FB	TopicVar30	809B9	TopicVar68	80A77	TOLVar15
80900	TopicVar31	809BE	TopicVar69	80A7C	TOLVar16
80905	TopicVar32	809C3	TopicVar70	80A81	TOLVar17
8090A	TopicVar33	809C8	TopicVar71	80A86	TOLVar18
8090F	TopicVar34	809CD	TopicVar72	80A8B	TOLVar19
80914	TopicVar35	809D2	TopicVar73	80A90	TOLVar20
80919	TopicVar36	809D7	TopicVar74	80A95	TOLVar21
8091E	TopicVar37	809DC	TopicVar75	80A9A	TOLVar22
80923	TopicVar38	809E1	TopicVar76	80A9F	TOLVar23
80928	TopicVar39	809E6	TopicVar77	80AA4	TOLVar24
8092D	TopicVar40	809EB	TopicVar78	80AA9	TOLVar25
80932	TopicVar41	809F0	TopicVar79	80AAE	TOLVar26
80937	TopicVar42	809F5	TopicVar80	80AB3	TOLVar27
8093C	TopicVar43	809FA	TopicVar81	80AB8	TOLVar28
80941	TopicVar44	809FF	TopicVar82	80ABD	TOLVar29
80946	TopicVar45	80A04	TopicVar83	80AC2	TOLVar30
8094B	TopicVar46	80A09	TopicVar84	80AC7	TOLVar31
80950	TopicVar47	80A0E	TopicVar85	80ACC	TOLVar32
80955	TopicVar48	80A13	TopicVar86	80AD1	TOLVar33
8095A	TopicVar49	80A18	TopicVar87	80AD6	TOLVar34
8095F	TopicVar50	80A1D	TopicVar88	80ADB	TOLVar35
80964	TopicVar51	80A22	TopicVar89	80AE0	TOLVar36
80969	TopicVar52	80A27	TopicVar90	80AE5	TOLVar37
8096E	TopicVar53	80A2C	TopicVar91	80AEA	TOLVar38

80AEF	TOLVar39	80BAD	TOLVar77	80C6B	TOLVar115
80AF4	TOLVar40	80BB2	TOLVar78	80C70	TOLVar116
80AF9	TOLVar41	80BB7	TOLVar79	80C75	TOLVar117
80AFE	TOLVar42	80BBC	TOLVar80	80C7A	TOLVar118
80B03	TOLVar43	80BC1	TOLVar81	80C7F	TOLVar119
80B08	TOLVar44	80BC6	TOLVar82	80C84	TOLVar120
80B0D	TOLVar45	80BCB	TOLVar83	80C89	TOLVar121
80B12	TOLVar46	80BD0	TOLVar84	80C8E	TOLVar122
80B17	TOLVar47	80BD5	TOLVar85	80C93	TOLVar123
80B1C	TOLVar48	80BDA	TOLVar86	80C98	TOLVar124
80B21	TOLVar49	80BDF	TOLVar87	80C9D	TOLVar125
80B26	TOLVar50	80BE4	TOLVar88	80CA2	TOLVar126
80B2B	TOLVar51	80BE9	TOLVar89	80CA7	TOLVar127
80B30	TOLVar52	80BEE	TOLVar90	80CAC	TOLVar128
80B35	TOLVar53	80BF3	TOLVar91	80CB1	TOLVar129
80B3A	TOLVar54	80BF8	TOLVar92	80CB6	TOLVar130
80B3F	TOLVar55	80BFD	TOLVar93	80CBB	TOLVar131
80B44	TOLVar56	80C02	TOLVar94	80CC0	TOLVar132
80B49	TOLVar57	80C07	TOLVar95	80CC5	TOLVar133
80B4E	TOLVar58	80C0C	TOLVar96	80CCA	TOLVar134
80B53	TOLVar59	80C11	TOLVar97	80CCF	TOLVar135
80B58	TOLVar60	80C16	TOLVar98	80CD4	TOLVar136
80B5D	TOLVar61	80C1B	TOLVar99	80CD9	TOLVar137
80B62	TOLVar62	80C20	TOLVar100	80CDE	TOLVar138
80B67	TOLVar63	80C25	TOLVar101	80CE3	TOLVar139
80B6C	TOLVar64	80C2A	TOLVar102	80CE8	TOLVar140
80B71	TOLVar65	80C2F	TOLVar103	80CED	TOLVar141
80B76	TOLVar66	80C34	TOLVar104	80CF2	TOLVar142
80B7B	TOLVar67	80C39	TOLVar105	80CF7	TOLVar143
80B80	TOLVar68	80C3E	TOLVar106	80CFC	TOLVar144
80B85	TOLVar69	80C43	TOLVar107	80D01	TOLVar145
80B8A	TOLVar70	80C48	TOLVar108	80D06	TOLVar146
80B8F	TOLVar71	80C4D	TOLVar109	80D0B	TOLVar147
80B94	TOLVar72	80C52	TOLVar110	80D10	TOLVar148
80B99	TOLVar73	80C57	TOLVar111	80D15	TOLVar149
80B9E	TOLVar74	80C5C	TOLVar112	80D1A	TOLVar150
80BA3	TOLVar75	80C61	TOLVar113	80D1F	TOLVar151
80BA8	TOLVar76	80C66	TOLVar114	80D24	TOLVar152

80D29	TOLVar153	80DE7	TOLVar191	80E9B	AVMEM
80D2E	TOLVar154	80DEC	TOLVar192	80EA0	LANGUAGE
80D33	TOLVar155	80DF1	TOLVar193	80EA5	ERROR
80D38	TOLVar156	80DF6	TOLVar194	80EAB	ATTNFLG
80D3D	TOLVar157	80DFB	TOLVar195	80EB0	FIRSTPROC
80D42	TOLVar158	80E00	TOLVar196	80EC0	SysNib1
80D47	TOLVar159	80E05	TOLVar197	80EC1	SysNib2
80D4C	TOLVar160	80E0A	TOLVar198	80EC2	SysNib3
80D51	TOLVar161	80E0F	TOLVar199	80EC3	SysNib4
80D56	TOLVar162	80E14	TOLVar200	80EC4	SysNib5
80D5B	TOLVar163	80E19	TOLVar201	80EC5	SysNib6
80D60	TOLVar164	80E1E	TOLVar202	80EC6	SysNib7
80D65	TOLVar165	80E23	TOLVar203	80EC7	SysNib8
80D6A	TOLVar166	80E28	TOLVar204	80EC8	SysNib9
80D6F	TOLVar167	80E2D	TOLVar205	80EC9	EDITFLAG
80D74	TOLVar168	80E32	TOLVar206	80EC9	SysNib10
80D79	TOLVar169	80E37	TOLVar207	80EC9	EDITLFLAG
80D7E	TOLVar170	80E3C	TOLVar208	80ECA	ParenModFLAG
80D83	TOLVar171	80E41	TOLVar209	80ECA	SysNib11
80D88	TOLVar172	80E46	TOLVar210	80ECB	SysNib12
80D8D	TOLVar173	80E4B	TOLVar211	80ECC	SysNib13
80D92	TOLVar174	80E50	TOLVar212	80ECD	SizeMLDisp
80D97	TOLVar175	80E55	TOLVar213	80ECD	SysNib14
80D9C	TOLVar176	80E5A	TOLVar214	80ECE	SysNib15
80DA1	TOLVar177	80E5F	TOLVar215	80ECF	SysNib16
80DA6	TOLVar178	80E64	TOLVar216	80ED0	SysNib17
80DAB	TOLVar179	80E69	CatalogCache	80ED1	SysNib18
80DB0	TOLVar180	80E6E	Clipboard	80ED2	SysNib19
80DB5	TOLVar181	80E73	FindPattern	80ED3	SysNib20
80DBA	TOLVar182	80E78	ReplacePatte	80ED4	AppCount
80DBF	TOLVar183	80E7D	ObjectU1	80ED6	ITEM1LINES
80DC4	TOLVar184	80E82	ObjectU2	80ED7	VIEWLEVEL
80DC9	TOLVar185	80E87	ObjectU3	80EDC	DEPTHSAVE
80DCE	TOLVar186	80E8C	ObjectU4	80EE1	RNSEED
80DD3	TOLVar187	80E91	OBUPEND	80EF0	SAVECLK
80DD8	TOLVar188	80E91	ObjectU5	80EF1	ALARMSDUE
80DDD	TOLVar189	80E96	SYSNOUPSTART	80EF2	PASTDUE
80DE2	TOLVar190	80E96	RAMEND	80EF3	DOUSEALARM

80EF4	NOALARMSRV	80FCD	DcompWidth	8107D	LastMenuRow
80EFF	LPD_HIST	80FCF	FONTHEIGHT	81082	FlashPtrBkp
80F00	ANNUNCIATORS	80FD0	FONTWIDTH	8108E	HeaderHeight
80F02	SystemFlags	80FD1	FONTCOUNT	8108E	T_HEADER
80F12	FLAG_SYSTEM2	80FD4	NODECOUNT	81093	StackHeight
80F22	UserFlags	80FD7	OBTREELEN	81093	NB_LIGNE
80F32	FLAG_USER2	80FDA	LASTOP	81098	FontHeight
80F42	ELEMENT	80FDB	LEFTTREE	81098	H_FONTE
80F44	FIRSTCHAR	80FDE	RIGHTTREE	8109D	TYPE_HEADER
80F49	CR_COUNT	80FE1	PARENTTREE	810A2	BEGIN_REL
80F4E	STACKNUM	80FE4	PRECSTACK	810A7	END_REL
80F53	TOPLINE	80FEB	KEYLIST	810AC	BEGX
80F59	HISTORYLEVEL	80FF0	KEYLOCK	810B1	ENDX
80F5A	LASTARGCOUNT	80FF1	ACCUM	810B6	BEG
80F5B	LASTARGf	80FF3	COLWIDTH	810BB	END
80F5C	LASTERROR	80FF5	ENTRWISE	810C0	T_ECRAN
80F61	CURSOREPOSN	80FF6	PARENCOUNT	810C0	SizeCLScreen
80F61	CURSOR	80FF8	STRETCHCOUNT	810E8	HashCLE
80F66	CURSORPART	80FFA	ClkOnNib	810E8	TAB_CMD
80F66	CURSORROW	80FFB	XmitSrcvTOut	8125A	T_BLOC
80F6B	CURSORPOSN	80FFD	DelayCt	8125F	CHECK_VAL
80F6B	CURSOROFFSET	80FFF	GCOLCOUNT	81264	CHECK_VAL2
80F6D	CURSORSTATE	81001	COLCOUNT	81269	SavTEMPENV
80F6E	CURSORCHR	81003	PrtStatus	81269	SAUV_80702
80F70	CURSORGROB	81006	IOCsave	8126E	SavFIRSTCHAR
80F98	CURSORX	81007	LineByteCt	8126E	SAUV_80865
80F9D	CURSORY	81009	FifoByteCt	81273	CHECK_TEXTE
80FA2	CURRENTMENU	8100B	LastPrntTime	81273	CheckCLE
80FA4	MENULEVEL	81016	PFIFO	81278	SAUV_MATRIX
80FA9	OLDMENU	81026	MenuRow	81278	SavMatrix
80FAB	T1COUNT	8102B	EqPtr	812A0	SizeLine
80FAC	PADCOUNT	81030	KeyRomPtr0	812A0	T_LIGNE
80FAD	GARBSCRATCH1	8103B	KeyRomPtr1	812A5	WidthScreen
80FB2	GARBSCRATCH2	81046	KeyRomPtr2	812A5	T_LARGEUR
80FB7	SAVECROSS	81051	KeyRomPtr3	812AA	NbFont
80FC1	PADJSAVE1	8105C	KeyRomPtr4	812AA	NB_FONTE
80FC2	PADJSAVE2	81067	KeyRomPtr5	812AF	VERIF_CARD
80FCC	KERMMODE	81072	KeyRomPtr6	812B4	SWITCH

812C3	MINI_FONT.OBJ
812C3	MiniFontObj
812CF	MINI_FONT
812CF	MiniFont
818CF	SavChars
818CF	SAUV_CHARS
818EE	FreeRoom
818F3	SAUV_REGA
818F3	SavRegA
818F8	SavRegB
818F8	SAUV_REGB
818FD	SavRegC
818FD	SAUV_REGC
81902	SavRegD
81902	SAUV_REGD
81907	SAUV_REGD1
81907	SavRegD1
8190C	SavRegisters
8190C	SAUV_REGISTR
81971	@FONTE
81971	ArryFont
8201D	TAB_FONTE
8201D	HashArryFont
8221D	SavMisc
8221D	SAUV_DIVERS
8229E	GROBSCR1
822B2	SCREEN1
822B2	ECRAN
82B32	GROBSCR2
82B46	SCREEN2
833C6	GROBSCR3
833DA	SCREEN3
83C5A	GROBSCR4
83C6E	SCREEN4
844EE	GROBSCR5
84502	SCREEN5
84D82	FONTE_SYSTEM
84D82	SystemFont
85F94	RealX
85FA9	RealY
85FBE	CplxX
85FE3	CplxY
86008	DIGITS
8600D	UserInt1
86012	UserInt2
86017	UserInt1g
8601C	UserInt2g
86021	nb_line_f_s
86026	has_font_f_s
86028	misc1_f_s
8602D	misc2_f_s
86032	misc3_f_s
86037	KSTATEVGER
86047	LastKey
86049	LastKeyTime
86051	BounceTiming
86059	CatalogEntry
8605E	FROMPTAB0_15
860AE	FROMPTABPTR
860B3	CurROMBank1
860B8	CurROMBank2
860BD	CurRAMBank1
860C2	CurRAMBank2
860C7	CurRAMBank3
860CC	FlashROMTAB2
8611C	RESRAMEND0
8611D	RESRAMEND
8611D	ROMPTAB
8611D	FlashROMPTAB
90000	HARDRAMEND
004002	^RunChooseSimple
005002	^sysCHOOSE
007002	^Ck&DoMsgBox
014002	^LIBS
015002	^GETLIBS
02E002	^DoAlert
02F002	^MkTitle
06E002	^Choose2
06F002	^Choose2Save
070002	^Choose2Index
072002	^Choose3
073002	^Choose3Save
074002	^Choose3Index
075002	^ChooseDefHandler
076002	^Choose3CANCL
077002	^Choose3OK
088002	^SaveHARDBUFF
089002	^RestoreHARDBUFF
09D002	^DoCKeyOK
09E002	^DoCKeyCancel
09F002	^DoCKeyCheck
0A0002	^DoCKeyChAll
0AE002	^DoMKeyOK
0AF002	^DoKeyCancel
0B0002	^DoCKeyUnChAll
0B1002	^LEDispItem
0B2002	^LEDispList
0B3002	^LEDispPrompt
0B4002	^DoKeyOK
0B5002	^DoKeyEdit
0BB002	^GetFieldVals
0BC002	^IFEDispField
0BD002	^DOTVARS{}
0BE002	^ChangeFocus
0C4002	^SERIAL
0C8002	^DISPROW1_plus
0C9002	^DISPROW2_plus
06C003	^laDELROW
06D003	^laINSROW
06E003	^laGPROW
09A003	^StrCutNchr
09B003	^StrCutNchr2
0A4003	^BRdone
0A5003	^BRDispItems

0A6003	^BRinverse	023004	^IfSetGrob	04A004	^IfInitDepth
0A7003	^BRViewItem	024004	^IfSetFieldValue	04B004	^IfTet
0AB003	^BRGetItem	025004	^IfSetCurrentFie..	04C004	^IfGetPrlgFromTy..
0AC003	^SWAPROWS	026004	^IfGetFieldValue	04D004	^IsUncompressDat..
001004	^FSTR1	027004	^IfGetCurrentFie..	04E004	^KeyLookup
002004	^FSTR2	028004	^IfGetFieldMessa..	067004	^Filer
003004	^FSTR3	029004	^IfGetFieldType	068004	^Arbo
004004	^FSTR4	02A004	^IfGetFieldObjec..	069004	^RENAME
005004	^FSTR5	02B004	^IfGetFieldDecom..	06D004	^FILER_MANAGER
006004	^FSTR6	02C004	^IfGetFieldChoos..	06E004	^FILER_MANAGERTYPE
007004	^FSTR7	02D004	^IfGetFieldChoos..	06F004	^FontBrowser
008004	^FSTR8	02E004	^IfGetFieldReset..	070004	^BrowseMem.1
009004	^FSTR9	02F004	^IfSetFieldReset..	08E006	^BerlekampP
00A004	^FSTR10	030004	^IfGetFieldInter..	08F006	^Berlekamp
00B004	^FSTR11	031004	^IfDisplayFromData	090006	^ErrInfRes
00C004	^FSTR12	032004	^IfGetNbFields	091006	^ErrUndefRes
00D004	^FSTR13	033004	^IfCheckSetValue	092006	^ErrBadDim
00E004	^algparse	034004	^IfCheckFieldtype	093006	^ALG48MSOLV
00F004	^algunwrap	035004	^IfReset	094006	^GMSOLV
010004	^EQW3	036004	^IfSetField	095006	^GBASIS
011004	^EQW3Edit	037004	^IfKeyChoose	096006	^GSOLVE
012004	^EQW3StartEdit	038004	^IfKeyEdit	097006	^GFACTOR
013004	^EQW3ViewMargin	039004	^IfKeyTypes	098006	^GREDUCE
014004	^EQW3ViewLeftX	03A004	^IfKeyCalc	099006	^REDUCE
015004	^EQW3ViewRightX	03B004	^IfKeyInvertCheck	09A006	^FASTREDUCE
016004	^EQW3ViewLeft	03C004	^IfONKeyPress	09B006	^ONE{}POLY
017004	^EQW3ViewRight	03D004	^IfEnterKeyPress	09C006	^TWO{}POLY
018004	^EQW3ViewRightRPL	03F004	^IfSetHelpString	09D006	^THREE{}POLY
019004	^EQW3GROB	040004	^IfSetTitle	09E006	^TWO::POLY
01A004	^EQW3GROBStk	041004	^IfSetTitle2	09F006	^::POLY
01B004	^EQW3CursorOn	042004	^IfMain2	0A0006	^{}POLY
01C004	^EQW3CursorOff	043004	^IfPutFieldsOnSt..	0A1006	^>TPOLY
01D004	^EQW3Code	044004	^IfSetFieldPos	0A2006	^>HPOLY
01E004	^EQW3GROBsys	045004	^IfGetFieldPos	0A3006	^>TPOLYN
01F004	^EQW3GROBmini	046004	^IfDisplayFromDa..	0A4006	^>HPOLYN
020004	^IfMain	047004	^IfSetAllLabelsM..	0A5006	^MKPOLY
021004	^IfSetFieldVisible	048004	^IfSetAllHelpStr..	0A6006	^ONE>POLY
022004	^IfSetSelected	049004	^IfCreateTitleGrob	0A7006	^>POLY

0A8006	^ALG48FCTR?	0CE006	^SPollard	0F4006	^Z>ZH
0A9006	^MFactTriv	0CF006	^BFactor	0F5006	^R>Z
0AA006	^CheckPNoExt	0D0006	^BrentPow	0F6006	^Z>R
0AB006	^PPP	0D1006	^ZPrime?	0F7006	^DupQIsZero?
0AC006	^PFactor	0D2006	^ZIsPrime?	0F8006	^QIsZero?
0AD006	^PSqff	0D3006	^SIsPrime?	0F9006	^DupZIsOne?
0AE006	^PHFctr	0D4006	^BIsPrime?	0FA006	^ZIsOne?
0AF006	^PHFctr1	0D5006	^BRabin	0FB006	^DupZIsNeg?
0B0006	^PHFctr0	0D6006	^ZTrialDiv2	0FC006	^ZIsNeg?
0B1006	^DeCntMulti	0D7006	^ZTrialPrime?	0FD006	^ListPos
0B2006	^DoLS	0D8006	^ZTrialDiv	0FE006	^AppendList
0B3006	^PNFctr	0D9006	^QMod	0FF006	^Contains?
0B4006	^PSQFF	0DA006	^QMODSYMext	100006	^SortList
0B5006	^LiftZAdic	0DB006	^ModPow	101006	^ZTrim
0B6006	^LFCProd	0DC006	^ZQUOText	102006	^ZAbs
0B7006	^UFactor	0DD006	^ZMod	103006	^PNMax
0B8006	^UFactor1	0DE006	^ZDIVext	104006	^LISTMAXext
0B9006	^MonicLf	0DF006	^QRoot	105006	^ZNMax
0BA006	^DemonicLf	0E0006	^ZSQRT	106006	^ZNMin
0BB006	^LiftLinear	0E1006	^PEvalMod	107006	^ZNLT?
0BC006	^LiftGeneral	0E2006	^QAddMod	108006	^DISTDIVext
0BD006	^UFactorDeg2	0E3006	^QSubMod	109006	^DupZIsTwo?
0BE006	^CombineFac	0E4006	^QMulMod	10A006	^DupZIsEven?
0BF006	^CombProd	0E5006	^QDivMod	10B006	^Univar?
0C0006	^CombInit	0E6006	^QInvMod	10C006	^SUnivar?
0C1006	^CombNext	0E7006	^QGcdMod	10D006	^ZBits
0C2006	^RmCombNext	0E8006	^QGcdExMod	10E006	^ZBit?
0C3006	^PFactTriv	0E9006	^IsV>V?	10F006	^LOPMext
0C4006	^VarFactor	0EA006	^PEvalFast?	110006	^SWAPRMULT
0C5006	^PFactPowCnt	0EB006	^PZadic	111006	^QMul
0C6006	^PDivLk	0EC006	^GCDHEUext	112006	^RMULText
0C7006	^Prime+	0ED006	^H>Z	113006	^RASOP
0C8006	^Prime-	0EE006	^#>Z	114006	^SWAPRSUB
0C9006	^ZFactor	0EF006	^Z2BIN	115006	^QSub
0CA006	^NFactor	0F0006	^COERCE2Z	116006	^RSUBext
0CB006	^NFactorSpc	0F1006	^Z>S	117006	^SWAPRADD
0CC006	^DupTypeS?	0F2006	^S>Z	118006	^QAdd
0CD006	^SFactor	0F3006	^S>Z?	119006	^RADDext

11A006	^SWAPRDIV	140006	^xssSYM%	166006	^NDXQext
11B006	^RDIVext	141006	^addt%CH	167006	^TYPEIRRQ?
11C006	^QDiv	142006	^xssSYM%CH	168006	^DTYPEIRRQ?
11D006	^R15SIMP	143006	^addt%T	169006	^BESTMATRIXTYPE
11E006	^PPow#	144006	^xssSYM%T	16A006	^{}TO[]
11F006	^RP#	145006	^addtMOD	16B006	^[]TO{}
120006	^MPext	146006	^xssSYMMOD	16C006	^DUPNULL[]?
121006	^MP0	147006	^addtTRNC	16D006	^MDIMS
122006	^MPEXEC	148006	^xssSYMTRCXY	16E006	^DIMLIMITS
123006	^RPext	149006	^addtRND	16F006	^CKSAMESIZE
124006	^PREPARext	14A006	^xssSYMRNDXY	170006	^DTYPENDO?
125006	^x+ext	14B006	^addtCOMB	171006	^DTYPFMAT?
126006	^x-ext	14C006	^xssSYMCOMB	172006	^CKNUMARRY
127006	^x*ext	14D006	^addtPERM	173006	^2DMATRIX?
128006	^x=ext	14E006	^xssSYMPERM	174006	^MATRIXDIM
129006	^x/ext	14F006	^addtOR	175006	^SAMEMATRIX
12A006	^2SYMBINCOMP	150006	^xssSYMOR	176006	^SAMEMATSCTYPE
12B006	^x^ext	151006	^addtAND	177006	^CKMATRIXELEM
12C006	^EXPAND^	152006	^xssSYMAND	178006	^MATRIX2ARRAY
12D006	^addtXROOT	153006	^addtXOR	179006	^MATRIX2LIST
12E006	^xssSYMXROOT	154006	^xssSYMXOR	17A006	^LIST2MATRIX
12F006	^addtMIN	155006	^2LAMBIND	17B006	^LENMATRIX
130006	^xssSYMMIN	156006	^3LAMBIND	17C006	^XEQARRY>
131006	^addtMAX	157006	^SYMBINCOMP	17D006	^MATEXPLODE
132006	^xssSYMMAX	158006	^CKINNERCOMP	17E006	^ARRAY2MATRIX
133006	^addt<	159006	^DUPCKLEN{}	17F006	^XEQ>ARRY
134006	^xssSYM<?	15A006	^CKCARCOMP	180006	^XEQ>ARRAY1
135006	^addt<=	15B006	^CARCOMPext	181006	^CKALG
136006	^xssSYM<=?	15C006	^RISCH13	182006	^TYPEZ?
137006	^addt>	15D006	^CXRIext	183006	^DUPTYPEZ?
138006	^xssSYM>?	15E006	^RIXCext	184006	^CK1Z
139006	^addt>=	15F006	^IRXCext	185006	^CK2Z
13A006	^xssSYM>=?	160006	^IRXC2	186006	^CK3Z
13B006	^addt==	161006	^SWAPNDXF	187006	^CK1Cext
13C006	^xssSYM=?	162006	^NDXFext	188006	^C2C%%
13D006	^addt!=	163006	^SWAPFXND	189006	^ZZ2C%%ext
13E006	^xssSYM#?	164006	^FXNDext	18A006	^Z2%%
13F006	^addt%	165006	^QXNDext	18B006	^C%>C%%

18C006	^E%%>C%%	1B2006	^METADERIFTE	1D8006	^FLAGFACTOR
18D006	^R2Zext	1B3006	^DERARG	1D9006	^FLAGLISTEXEC
18E006	^Z2Sext	1B4006	^METADEREXP	1DA006	^FLAGSYMBEXEC
18F006	^CKFPOLYext	1B5006	^METADERLN	1DB006	^FLAGIDNTEXEC
190006	^CK2FPOLY	1B6006	^METADERLNP1	1DC006	^FLAGINTVX
191006	^IDNTLAM?	1B7006	^METADERLOG	1DD006	^DERVX
192006	^FLOAT?	1B8006	^METADERALOG	1DE006	^SOLVEXFLOAT
193006	^CKSYMREALCMP	1B9006	^METADERABS	1DF006	^SYMLIMIT
194006	^TYPEIDNTLAM?	1BA006	^METADERINV	1E0006	^FLAGMATRIXLIMIT
195006	^REAL?	1BB006	^METADERNEG	1E1006	^TAYLOR0
196006	^TYPEREALZINT?	1BC006	^METADERSQRT	1E2006	^FLAGSERIES
197006	^OBJ2REAL	1BD006	^METADER&NEG	1E3006	^PLOTSTK
198006	^METAINT?	1BE006	^METADERSQ	1E4006	^PLOTADD
199006	^METAPOSINT?	1BF006	^METADERSIN	1E5006	^FLAGIBP
19A006	^OBJINT?	1C0006	^METADERCOS	1E6006	^FLAGPREVAL
19B006	^OBJPOSINT?	1C1006	^METADERTAN	1E7006	^MATRIXRISCH
19C006	^CKINT>0	1C2006	^METADERSINH	1E8006	^FLAGRISCH
19D006	^Z>#	1C3006	^METADERCOSH	1E9006	^FLAGDERIV
19E006	^CLEANIDLAM	1C4006	^METADERTANH	1EA006	^FLAGLAP
19F006	^ssSYMDER	1C5006	^METADERASIN	1EB006	^FLAGILAP
1A0006	^SYMDER	1C6006	^METADERACOS	1EC006	^FLAGDESOLVE
1A1006	^DERIVext	1C7006	^METADERATAN	1ED006	^FLAGLDSSOLV
1A2006	^siSYMDER	1C8006	^METADERASH	1EE006	^FLAGLDECSOLV
1A3006	^DERIVIDNT	1C9006	^METADERACH	1EF006	^FLAGTEXPAND
1A4006	^DERIVIDNT1	1CA006	^METADERATH	1F0006	^FLAGLIN
1A5006	^DERIV	1CB006	^pshder*	1F1006	^FLAGTSIMP
1A6006	^METADERIV	1CC006	^SQRTINVpshd*	1F2006	^FLAGLNCOLLECT
1A7006	^DO>STRID	1CD006	^ckaddt*	1F3006	^FLAGEXPLN
1A8006	^METADEROP	1CE006	^ckaddt+	1F4006	^FLAGSINCOS
1A9006	^METADER+	1CF006	^ckaddt-	1F5006	^FLAGTLIN
1AA006	^METADER-	1D0006	^VERNUMext	1F6006	^FLAGTCOLLECT
1AB006	^METADER*	1D1006	^MENUXYext	1F7006	^FLAGTRIG
1AC006	^METADER/	1D2006	^SAVECASFLAGS	1F8006	^FLAGTRIGCOS
1AD006	^METADER^	1D3006	^SAFEPURGE	1F9006	^FLAGTRIGSIN
1AE006	^METADERFCN	1D4006	^RESTORECASFLAGS	1FA006	^FLAGTRIGTAN
1AF006	^METADERDER	1D5006	^CASFLAGEVAL	1FB006	^FLAGTAN2SC
1B0006	^METADERI4	1D6006	^FLAGEXPAND	1FC006	^FLAGHALFTAN
1B1006	^METADERI3	1D7006	^EXPANDBOTH	1FD006	^FLAGTAN2SC2

1FE006	^FLAGATAN2S	224006	^FLAGQXA	24A006	^GCD1MOD
1FF006	^FLAGASIN2T	225006	^FLAGAXQ	24B006	^INVMOD
200006	^FLAGASIN2C	226006	^FLAGGAUSS	24C006	^MINVMOD
201006	^FLAGACOS2S	227006	^FLAGSYLVESTER	24D006	^FLAGDIV2MOD
202006	^CK&CONVINT	228006	^PCAR	24E006	^FLAGPOWMOD
203006	^CK&CONV2INT	229006	^MADNOCK	24F006	^FLAGMPOWMOD
204006	^CONVBACK2INT	22A006	^SYSTEM	250006	^EXPAMOD
205006	^CONVBACKINT	22B006	^VANDERMONDE	251006	^FLAGEXPAMOD
206006	^STEPIDIV2	22C006	^HILBERTNOCK	252006	^FLAGFACTORMOD
207006	^FLAGDIV2	22D006	^FLAGJORDAN	253006	^MFACTORMOD
208006	^FLAGGCD	22E006	^CURL	254006	^RREFMOD
209006	^PEGCD	22F006	^DIVERGENCE	255006	^KEYEVAL
20A006	^IEGCD	230006	^LAPLACIAN	256006	^LIFCext
20B006	^ABCUV	231006	^HESSIAN	257006	^EvalNoCKx*
20C006	^IABCUV	232006	^HERMITE	258006	^EvalNoCKx+
20D006	^FLAGLGCD	233006	^TCHEBNOCK	259006	^EvalNoCKx-
20E006	^FLAGLCM	234006	^LEGENDRE	25A006	^EvalNoCKx/
20F006	^FLAGSIMP2	235006	^LAGRANGE	25B006	^EvalNoCKx^
210006	^FLAGPARTFRAC	236006	^FOURIER	25C006	^EvalNoCKxCHS
211006	^FLAGPROPFRAC	237006	^SIGNE	25D006	^EvalNoCKxINV
212006	^FLAGPTAYL	238006	^TABVAR	25E006	^EvalNoCKxMOD
213006	^FLAGHORNER	239006	^FLAGDIVPC	25F006	^EvalNoCKxPERM
214006	^EULER	23A006	^FLAGTRUNC	260006	^EvalNoCKxCOMB
215006	^PA2B2	23B006	^FLAGSEVAL	261006	^EvalNoCKxOR
216006	^FLAGCHINREM	23C006	^XNUM	262006	^EvalNoCKxAND
217006	^ICHINREM	23D006	^REORDER	263006	^EvalNoCKxXOR
218006	^ISPRIME	23E006	^USERLVAR	264006	^EvalNoCKxXROOT
219006	^SOLVE1EQ	23F006	^USERLIDNT	265006	^TABVALext
21A006	^SOLVEMANYEQ	240006	^EXLR	266006	^TOLISText
21B006	^ZEROS1EQ	241006	^ADDTMOD	267006	^FROMLISText
21C006	^ZEROSMANYEQ	242006	^MADDTMOD	268006	^PFEXECext
21D006	^FCOEF	243006	^SUBTMOD	269006	^LOP1ext
21E006	^FROOTS	244006	^MSUBTMOD	26A006	^LOPAext
21F006	^FACTORS	245006	^MULTMOD	26B006	^LISTSECOext
220006	^DIVIS	246006	^MAT*SCMOD	26C006	^rpnQOBJext
221006	^STUDMULT	247006	^SC*MATMOD	26D006	^CK1TONOext
222006	^STUDDIV	248006	^MAT*MATMOD	26E006	^COLCext
223006	^rref	249006	^DIVMOD	26F006	^SYMCOLCT

270006	^COLC1	296006	^FACTOOBJext	2BC006	^LASTCOMP
271006	^COLC2	297006	^SLVARext	2BD006	^SQFF2ext
272006	^MULMULText	298006	^SIMPLIFY	2BE006	^PPZ
273006	^METAMULMULT	299006	^SIMP1ext	2BF006	^PZHSTR
274006	^METAMM2	29A006	^SYMEXPAN	2C0006	^HORNER1ext
275006	^COMPLISText	29B006	^SIMPVAR	2C1006	^PEval
276006	^METACOMPRIM	29C006	^ID>DERext	2C2006	^RISCHext
277006	^METACOMP0	29D006	^SIMPIDNT	2C3006	^risch/
278006	^METACOMP1	29E006	^RCLALLIDNT	2C4006	^rischABS
279006	^ADDLISText	29F006	^RCL1IDNT	2C5006	^IBP
27A006	^DIVISext	2A0006	^SIMPSYMBS	2C6006	^SQRT_IN?
27B006	^FACT1ext	2A1006	^SYMINTEGRAL	2C7006	^IS_SQRT?
27C006	^FACTOext	2A2006	^SIMPUSERFCN	2C8006	^XROOT_IN?
27D006	^ZFACTO	2A3006	^EVALUSERFCN	2C9006	^IS_XROOT?
27E006	^SOLVext	2A4006	^SIMP|	2CA006	^STOPRIMIT
27F006	^FRND	2A5006	^DENOLCMext	2CB006	^CONTAINS_LN?
280006	^BICARREE?	2A6006	^METADENOLCM	2CC006	^ISNT_IDNT?
281006	^REALBICAR	2A7006	^SWPSIMPNDXF	2CD006	^RISCHPF
282006	^FEVIDENText	2A8006	^SIMPNDXFext	2CE006	^RISCHRAT
283006	^EVIDENText	2A9006	^SIMPext	2CF006	^rischlogpart
284006	^EVIDSOLV	2AA006	^SIMPEXTOK	2D0006	^PREVALext
285006	^DEG2ext	2AB006	^MAKEPROFOND	2D1006	^WARNSING
286006	^METADEG2	2AC006	^SLOWSIMP2L	2D2006	^INText
287006	^METADEG1	2AD006	^SIMPGCDext	2D3006	^INT3
288006	^DEG1	2AE006	^SIMP3ext	2D4006	^FOURIERext
289006	^FDEG2ext	2AF006	^SIMP3LISText	2D5006	^3DUP
28A006	^PIext	2B0006	^SIMP3LSTSLOW	2D6006	^#3+ROLL
28B006	^RACTOFACext	2B1006	^LPGCDext	2D7006	^2DROPTRUE
28C006	^FACTORACext	2B2006	^SLOWGCDext	2D8006	^IRRQ#ULTIMATE
28D006	^RFACText	2B3006	^QGcd	2D9006	^LESSCOMPLEX?
28E006	^RFACT2ext	2B4006	^GCDext	2DA006	^LISTIRRQ
28F006	^RFACTSTEP3	2B5006	^CGCDext	2DB006	^LIST1i-1-i
290006	^RFACTSTEP5	2B6006	^CMODext	2DC006	^LIST10-10
291006	^METASOLV	2B7006	^ZGCDext	2DD006	^TABLECOSext
292006	^METASOLVOUT	2B8006	^ZGcd	2DE006	^TABLETANext
293006	^METASOLV2	2B9006	^TSIMP2ext	2DF006	^DROPZ1
294006	^METASOLV4	2BA006	^TSIMPext	2E0006	^DROPZ0
295006	^ADDMULTIPL	2BB006	^TSIMP3ext	2E1006	^TESTINFINI

2E2006	^INFINIext	308006	^SINEXPA*1	32E006	^MATDOT
2E3006	^MINUSINFext	309006	^COSEXPA	32F006	^RNDARRY
2E4006	^PLUSINFext	30A006	^METACOSEXPA	330006	^TRCARRY
2E5006	^?ext	30B006	^COSEXPA+	331006	^Yext
2E6006	^POSINFext	30C006	^COSEXPA-	332006	^MAT/SCL
2E7006	^POSUNDEFext	30D006	^COSEXPA*	333006	^MAT/
2E8006	^pisur2	30E006	^COSEXPA*1	334006	^MATCHS
2E9006	^pisur-2	30F006	^EXPEXPA	335006	^MATSQUARE
2EA006	^pi	310006	^METAEXPEXPA	336006	^MATCONJ
2EB006	^metapi	311006	^EXPEXPA+	337006	^MATRE
2EC006	^'xPI	312006	^EXPEXPA-	338006	^MATIM
2ED006	^metai	313006	^EXPEXPA*	339006	^MATTRACE
2EE006	^'xi	314006	^EXPEXPANEG	33A006	^MATTRN
2EF006	^ipi	315006	^EXPEXPA*1	33B006	^MATTRAN
2F0006	^metaipi	316006	^LNEXPA	33C006	^mattran
2F1006	^meta-pi	317006	^METALNEXPA	33D006	^mattrn
2F2006	^metapi/2	318006	^LNEXPA*	33E006	^MATSUB
2F3006	^metapi/4	319006	^LNEXPA/	33F006	^submeta
2F4006	^meta-pi/2	31A006	^LNEXPA^	340006	^MATREPL
2F5006	^meta-pi/4	31B006	^LINEXPA	341006	^MATREDIM
2F6006	^pifois2	31C006	^MTRIG2SYMB	342006	^VRRDM
2F7006	^deuxipi	31D006	^LNCOLCext	343006	^VRRDMmeta
2F8006	^metapi*2	31E006	^METATANEXPA	344006	^MATRANM
2F9006	^base_ln	31F006	^TEXPAext	345006	^DIMRANM
2FA006	^meta_e	320006	^MAT+	346006	^MATDET
2FB006	^NEXTPext	321006	^MADD	347006	^MATRDET
2FC006	^INSERT{}N	322006	^MAT-	348006	^MATFNORM
2FD006	^COMPRIMext	323006	^MSUB	349006	^MATRNORM
2FE006	^TCOLLECT	324006	^VADD	34A006	^MATCNORM
2FF006	^SIGMAEXPext	325006	^VSUB	34B006	^MATRREF
300006	^LINEXPext	326006	^MAT*	34C006	^MATREF
301006	^SIGMAEXP2ext	327006	^MMMULT	34D006	^MATRANK
302006	^TCHEBext	328006	^MVMULT	34E006	^MATINV
303006	^SINEXPA	329006	^SCL*MAT	34F006	^MATREFRREF
304006	^METASINEXPA	32A006	^MAT*SCL	350006	^INXREDext
305006	^SINEXPA+	32B006	^VPMULT	351006	^METAMATRED
306006	^SINEXPA-	32C006	^MAT^	352006	^METAPIVOT
307006	^SINEXPA*	32D006	^MATCROSS	353006	^PIVOTNORM

354006	^PIVOTFLOAT
355006	^SYSText
356006	^STOSYSText
357006	^MAKESYSText
358006	^VARGENext
359006	^NULLVECTOR?
35A006	^FINDELN
35B006	^PULLEL[S]
35C006	^BANGARRY
35D006	^PUT[]
35E006	^ARSIZE
35F006	^MATRIX>DIAG
360006	^MATRIXDIAG>
361006	^la+ELEMsym
362006	^INSERTROW[]
363006	^insertrow[]
364006	^INSERTCOL[]
365006	^INSERT[]ROW[]
366006	^INSERT[]COL[]
367006	^MATRIXRCI
368006	^MATRIXRCIJ
369006	^MATRIXCSWAP
36A006	^MATRIXRSWAP
36B006	^MATRIX-ROW
36C006	^METAMAT-ROW
36D006	^MATRIX-COL
36E006	^METAMATCSWAP
36F006	^METAMATRSWAP
370006	^STOMAText
371006	^MATIDN
372006	^MATCON
373006	^MAKEARRY
374006	^OBJDIMS2MAT
375006	^LCPROG2M
376006	^MAKE2DMATRIX
377006	^make2dmatrix
378006	^ADDMATOBJext
379006	^VUNARYOP
37A006	^VBINARYOP
37B006	^PEVAL
37C006	^MATEGVL
37D006	^ROOTM2ROOT
37E006	^MADJ
37F006	^MATEGV
380006	^JORDAN
381006	^QXA
382006	^AXQ
383006	^GAUSS
384006	^SYLVESTER
385006	^metasplit
386006	^m-1&m+1
387006	^meta1/meta
388006	^1&meta
389006	^meta/2
38A006	^addt2
38B006	^addt/
38C006	^meta2*
38D006	^meta1-sq
38E006	^metasq+1
38F006	^metasq-1
390006	^meta-1
391006	^NDROPZERO
392006	^2DROPZ0
393006	^metaadd
394006	^metasub
395006	^metamult
396006	^metadiv
397006	^meta^
398006	^addt^
399006	^metapow
39A006	^metafraction?
39B006	^metaxroot
39C006	^top&addt*
39D006	^top&addt/
39E006	^addti
39F006	^metaEQUAL?
3A0006	^2metaundef#
3A1006	^1metaundef#
3A2006	^metaundef
3A3006	^2metainf#
3A4006	^1metainf#
3A5006	^metainftype
3A6006	^unsignedinf
3A7006	^plusinf
3A8006	^NDROPplusinf
3A9006	^minusinf
3AA006	^NDROPminusinf
3AB006	^MetaAdd
3AC006	^xssSYM+
3AD006	^MetaSub
3AE006	^xssSYM-
3AF006	^MetaMul
3B0006	^xssSYM*
3B1006	^MetaDiv
3B2006	^xssSYM/
3B3006	^NDROPZ0
3B4006	^NDROPZ1
3B5006	^MetaPow
3B6006	^xssSYM^
3B7006	^MetaNeg
3B8006	^xSYMCHS
3B9006	^metaneg
3BA006	^metackneg
3BB006	^metasimp
3BC006	^metapi?
3BD006	^metaCOMPARE
3BE006	^STRICTmetaCOMPARE
3BF006	^EQUALPOSMETA
3C0006	^EQUALPOS2META
3C1006	^vgerxssSYMSUM
3C2006	^DISTRIB/
3C3006	^metareal?
3C4006	^ModExpa
3C5006	^ModAdd

3C6006	^ModSub	3EC006	^QUOText	412006	^COS2TAN
3C7006	^ModMul	3ED006	^NEWDIVext	413006	^cos2tan
3C8006	^ModDiv	3EE006	^QDivRem	414006	^SIN2TAN
3C9006	^ModDiv2	3EF006	^DIV2LISText	415006	^sin2tan
3CA006	^ModInv	3F0006	^DIVOBJext	416006	^TRIGext
3CB006	^ModGcd	3F1006	^DIVMETAOBJ	417006	^HYP2EXPext
3CC006	^ModLGCD	3F2006	^LOPDext	418006	^EXPLNext
3CD006	^ModLOPD	3F3006	^QUOTOBJext	419006	^SERIESEXPLN
3CE006	^MODULOMODext	3F4006	^DIVISIBLE?	41A006	^LNP12LN
3CF006	^MODULOMAText	3F5006	^QDiv?	41B006	^LOG2LN
3D0006	^Mod	3F6006	^FastDiv?	41C006	^ALOG2EXP
3D1006	^ModFctr	3F7006	^POTENCEext	41D006	^EXPM2EXP
3D2006	^PARTFRAC	3F8006	^PDIV2ext	41E006	^SQRT2LNEXP
3D3006	^INPARTFRAC	3F9006	^PSetSign	41F006	^sqrt2lnexp
3D4006	^PARTFRACRAT	3FA006	^PLCZ	420006	^TAN2EXP
3D5006	^PFext	3FB006	^HSECO2RCext	421006	^tan2exp
3D6006	^IEGCDext	3FC006	^SECO2CMPext	422006	^ASIN2LN
3D7006	^REGCDext	3FD006	^SECO2CMPPOL	423006	^asin2ln
3D8006	^EGCDext	3FE006	^SECO2CMPCART	424006	^ACOS2LN
3D9006	^INEGCD	3FF006	^VALOBJext	425006	^acos2ln
3DA006	^EGCDSWAP	400006	^R2SYM	426006	^TAN2SCext
3DB006	^EGCDNEWG	401006	^VAL2ext	427006	^TAN2SC
3DC006	^PDer	402006	^INVAL2	428006	^sin/cos
3DD006	^INTEGRext	403006	^METAVAL2	429006	^SIN2TCext
3DE006	^LRDMext	404006	^VAL1	42A006	^SIN2TC
3DF006	^RRDMext	405006	^VAL1M	42B006	^cos*tan
3E0006	^DEGREext	406006	^addt0meta	42C006	^COS2ext
3E1006	^FHORNER	407006	^HALFTAN	42D006	^sqrt1-sin^2
3E2006	^HORNext	408006	^COS2TAN/2	42E006	^SIN2ext
3E3006	^HORN1	409006	^cos2tan/2	42F006	^sqrt1-cos^2
3E4006	^MHORNext	40A006	^1-x^2/1+x^2	430006	^ATAN2Sext
3E5006	^PTAYLext	40B006	^SIN2TAN/2	431006	^ATAN2ASIN
3E6006	^LAGRANGEext	40C006	^sin2tan/2	432006	^atan2asin
3E7006	^PSEUDOPREP	40D006	^2x/1+x^2	433006	^ASIN2Text
3E8006	^PSEUDODIV	40E006	^TAN2TAN/2	434006	^ASIN2ATAN
3E9006	^IDIV2	40F006	^tan2tan/2	435006	^asin2atan
3EA006	^BESTDIV2	410006	^addtTAN/2	436006	^ASIN2Cext
3EB006	^CDIV2ext	411006	^TRIGTAN	437006	^ASIN2ACOS

438006	^pi/2-acos	45E006	^SYMQFORM	484006	^LIMERR10!
439006	^pi/2-meta	45F006	^LISTEXEC	485006	^LIMNEG!
43A006	^ACOS2Sext	460006	^LISTEXEC1	486006	^LIMRAC!
43B006	^pi/2-asin	461006	^SECOEXEC	487006	^LIMINV!
43C006	^ACOS2ASIN	462006	^EQUATION?	488006	^LIM/!
43D006	^ATAN2LNext	463006	^USERFCN?	489006	^LIMPOW!
43E006	^atan2ln	464006	^SYMBEXEC	48A006	^LIMSQ!
43F006	^TAN2SC2ext	465006	^MEVALext	48B006	^LIM*!
440006	^TAN2SC2	466006	^CASNUMEVAL	48C006	^LIMDIVPC!
441006	^2*1-cos/sin	467006	^CASCOMPEVAL	48D006	^DIVPC!
442006	^TAN2CS2	468006	^REPLACE2BY1	48E006	^LIMPROFEND!
443006	^2*sin/1+cos	469006	^NR_REPLACE	48F006	^LIMPROF!
444006	^SIN2EXPext	46A006	^SYMBWHERE	490006	^LIM%#!
445006	^sin2exp	46B006	^CASCRUNCH	491006	^LIMPROF0!
446006	^COS2EXPext	46C006	^APPROXCOMPEVAL	492006	^LIMPROF1!
447006	^cos2exp	46D006	^LIMIText	493006	^LIMPROF2!
448006	^SINH2EXPext	46E006	^REWRITEIFINF	494006	^LIMINVLN!
449006	^sinh2exp	46F006	^SYMTAYLOR	495006	^LIMLN!
44A006	^COSH2EXPext	470006	^SYMPAPRX	496006	^LIMEXP!
44B006	^cosh2exp	471006	^TRUNCDL	497006	^LIMSINCOS!
44C006	^TANH2EXPext	472006	^LIMSERIES!	498006	^LIMATAN!
44D006	^tanh2exp	473006	^LIMITX!	499006	^LIMASIN!
44E006	^ASINH2LNext	474006	^LIMITNOVX!	49A006	^LIMSQRT!
44F006	^asinh2ln	475006	^LIMERR0!	49B006	^LIMFLOOR!
450006	^ACOSH2LNext	476006	^LIMERR1!	49C006	^LIMABS!
451006	^acosh2ln	477006	^LIMIT!	49D006	^LPROF!
452006	^ATANH2LNext	478006	^LIMSTEP1!	49E006	^LIM#VARX!
453006	^atanh2ln	479006	^LIMSTEP2!	49F006	^LIMBETA!
454006	^XROOT2ext	47A006	^LIMSTEP3!	4A0006	^LIMALPHA!
455006	^xroot2expln	47B006	^LIMSTEP4!	4A1006	^HORNEXP!
456006	^LN2ext	47C006	^LIMLIM!	4A2006	^HORNCOS!
457006	^SINCOSext	47D006	^n{}N	4A3006	^HORNSIN!
458006	^exp2sincos	47E006	^LIMLIM1!	4A4006	^LIMSC0!
459006	^metai*	47F006	^LIMCMPL!	4A5006	^LIMSC1!
45A006	^LN2ATAN	480006	^LIMEQUFR!	4A6006	^HORNATAN!
45B006	^VAR=LIST	481006	^LIMEQU!	4A7006	^LIMATAS!
45C006	^IDNTEXEC	482006	^LIMEQU0!	4A8006	^HORNASIN!
45D006	^SYMISOL	483006	^LIM+-!	4A9006	^HORNASIN1!

4AA006	^HORNLN!
4AB006	^LNOBJ!
4AC006	^NEWLIMHORN
4AD006	^LIMHORN!
4AE006	^LRDM!
4AF006	^LIMDL!
4B0006	^LIMDLINF!
4B1006	^LIMINFSIGN!
4B2006	^LIMMAX!
4B3006	^LIMCOMP!
4B4006	^VARCOMP2!
4B5006	^LIMSORT!
4B6006	^VARCOMP!
4B7006	^VARCOMPLN!
4B8006	^VARCOMP3!
4B9006	^VARCOMP31!
4BA006	^VARCOMP32!
4BB006	^VARCOMP33!
4BC006	^LIMERR6!
4BD006	^LIMVALOBJ!
4BE006	^LIMVAL!
4BF006	^EQUIV!
4C0006	^LVARXNX2!
4C1006	^SIMP1!
4C2006	^FindCurVar
4C3006	^LIMVAR!
4C4006	^VAR%
4C5006	^ISOL1
4C6006	^ISOLALL
4C7006	^ISOL2ext
4C8006	^BEZOUTMSOLV
4C9006	^ROOT{}N
4CA006	^MHORNER
4CB006	^MHORNER1
4CC006	^SQFFext
4CD006	^MSQFF
4CE006	^%1TWO
4CF006	^MZSQFF
4D0006	^MZSQFF1
4D1006	^MSECOSQFF
4D2006	^MLISTSQFF
4D3006	^METASQFFext
4D4006	^SECOSQFFext
4D5006	^CSQFFext
4D6006	^SUMSQRext
4D7006	^VXXLext
4D8006	^METALISTVXXL
4D9006	^VXXLFext
4DA006	^VXXL1ext
4DB006	^VXXL0
4DC006	^VXXL2NR
4DD006	^VXXL2
4DE006	^LIDNText
4DF006	^LVARXNXext
4E0006	^ISPOLYNOMIAL?
4E1006	^2POLYNOMIAL?
4E2006	^VXINDEP?
4E3006	^LVARXNX2ext
4E4006	^RLVARext
4E5006	^LLVARDext
4E6006	^VXLVARext
4E7006	^LVARext
4E8006	^VX>LVARext
4E9006	^VX>
4EA006	^VX!
4EB006	^prepvarlist
4EC006	^LIDNTLVAR
4ED006	^LISTOPRAC
4EE006	^LISTOPext
4EF006	^LISTOPSQRT
4F0006	^LVARDext
4F1006	^>VARLIST
4F2006	^DEPTHext
4F3006	^DEPTHOBJext
4F4006	^TRIMext
4F5006	^PTrim
4F6006	^TRIMOBJext
4F7006	^NEWTRIMext
4F8006	^>POLYTRIM
4F9006	^ELMGext
4FA006	^SWAPRNEG
4FB006	^QNeg
4FC006	^RNEGext
4FD006	^SWAPRRE
4FE006	^RREext
4FF006	^SWAPRIM
500006	^RIMext
501006	^xREext
502006	^xSYMRE
503006	^xIMext
504006	^xSYMIM
505006	^RCONJext
506006	^addtCONJ
507006	^xSYMCONJ
508006	^QCONJext
509006	^QABSext
50A006	^RABSext
50B006	^ZABS
50C006	^CZABS
50D006	^xABSext
50E006	^addtABS
50F006	^xSYMABS
510006	^addtABSEXACT
511006	^addtSIGN
512006	^xSYMSIGN
513006	^addtARG
514006	^xSYMARG
515006	^ARG2
516006	^INTERNALARG2
517006	^QUADRANT
518006	^CNORMext
519006	^CXIRext
51A006	^QNORMext
51B006	^SXSQRext

51C006	^XSQRext	542006	^addtSINH	568006	^addtMANT
51D006	^CK%%SQRT	543006	^xSYMSINH	569006	^xSYMMANT
51E006	^C%%SQRT	544006	^addtCOSH	56A006	^addtLNP1
51F006	^ZINTSQRT	545006	^xSYMCOSH	56B006	^xSYMLNP1
520006	^SHALT	546006	^addtTANH	56C006	^addtLOG
521006	^CKLN	547006	^xSYMTANH	56D006	^xSYMLOG
522006	^xLNext	548006	^xATANHext	56E006	^addtALOG
523006	^addtLN	549006	^addtATANH	56F006	^xSYMALOG
524006	^xSYMLN	54A006	^xSYMATANH	570006	^addtEXPM
525006	^EXPANDLN	54B006	^xASINHext	571006	^xSYMEXPM1
526006	^CMPLXLN	54C006	^addtASINH	572006	^factorial
527006	^LNATANext	54D006	^xSYMASINH	573006	^facts
528006	^REALLN	54E006	^xACOSHext	574006	^addtFACT
529006	^xEXPext	54F006	^addtACOSH	575006	^xSYMFACT
52A006	^xINVext	550006	^xSYMACOSH	576006	^factzint
52B006	^xvext	551006	^addtSQRT	577006	^addtNOT
52C006	^xCOSext	552006	^xSYMSQRT	578006	^xSYMNOT
52D006	^xSINext	553006	^xSQext	579006	^Verbose1
52E006	^xTANext	554006	^addtSQ	57A006	^Verbose2
52F006	^xCOSHext	555006	^xSYMSQ	57B006	^Verbose3
530006	^xSINHext	556006	^addtINV	57C006	^VerboseN
531006	^xTANHext	557006	^xSYMINV	57D006	^GETERABLEMSG
532006	^xASINext	558006	^addtEXP	57E006	^ERABLEERROR
533006	^xACOSext	559006	^xSYMEXP	57F006	^CANTFACTOR
534006	^xATANext	55A006	^addtD->R	580006	^TRANSCERROR
535006	^addtCOS	55B006	^xSYMD>R	581006	^NONUNARYERR
536006	^xSYMCOS	55C006	^addtR->D	582006	^INTERNALERR
537006	^addtSIN	55D006	^xSYMR>D	583006	^INVALIDOP
538006	^xSYMSIN	55E006	^addtFLOOR	584006	^ISOLERR
539006	^addtTAN	55F006	^xSYMFLOOR	585006	^NONINTERR
53A006	^xSYMTAN	560006	^addtCEIL	586006	^INTVARERR
53B006	^addtSINACOS	561006	^xSYMCEIL	587006	^Z>#ERR
53C006	^addtASIN	562006	^addtIP	588006	^Z<0ERR
53D006	^xSYMASIN	563006	^xSYMIP	589006	^VXINDEPERR
53E006	^addtACOS	564006	^addtFP	58A006	^NONPOLYSYST
53F006	^xSYMACOS	565006	^xSYMFP	58B006	^COMPLEXERR
540006	^addtATAN	566006	^addtXPON	58C006	^VALMUSTBE0
541006	^xSYMATAN	567006	^xSYMXPON	58D006	^SWITCHNOTALLOWED

58E006	^ERR$EVALext	089007	^ILAPRAText	0AF007	^RIGORMODE
58F006	^Sys1IT	08A007	^ILAPDELTA	0B0007	^SLOPPYMODE
590006	^ZSQ	08B007	^ILAPEXP	0B1007	^SLOPPY?
001007	^ListToArry	08C007	^ILAPEXPSC	0B2007	^MENUCHOOSE?
002007	^ArryToMatrix	08D007	^MENUext	0B3007	^MENUCHOOSE
003007	^ArryToList	08E007	^WRITEMENU	0B4007	^MENUGENE1
004007	^DIMS	08F007	^CFGDISPLAY	0B5007	^MENUBASE1
005007	^RunDoOldMatrix	090007	^NEWVX	0B6007	^MENUCMPLX1
006007	^RunDoNewMatrix	091007	^NEWMODULO	0B7007	^MENUTRIG1
007007	^DoNewMatrixReal	092007	^SWITCHON	0B8007	^MENUMAT1
008007	^DoNewMatrixCplx	093007	^SWITCHOFF	0B9007	^MENUARIT1
009007	^DoOldMatrixReal	094007	^FLAGNAME	0BA007	^MENUSOLVE1
00A007	^DoOldMatrixCplx	095007	^COMPLEXON	0BB007	^MENUEXPLN1
00B007	^DoNewMatrixReal..	096007	^COMPLEXOFF	0BC007	^MENUDIFF1
00C007	^DEB.MATRIX	097007	^EXACTON	0BD007	^PROMPTSTO1
00D007	^DEB.MATRIXTYPE	098007	^EXACTOFF	0BE007	^XGROBext
073007	^QpiZ	099007	^COMPLEXMODE	0BF007	^GROBADDext
074007	^QPI	09A007	^SETCOMPLEX	0C0007	^DISPLAYext
075007	^QpiSym	09B007	^COMPLEX?	0C1007	^SCROLLext
076007	^QpiArry	09C007	^REALMODE	0C2007	^RCLMODULO
077007	^QpiList	09D007	^CLRCOMPLEX	0C3007	^RCLPERIOD
078007	^Qpi	09E007	^EXACTMODE	0C4007	^RCLVX
079007	^Qpi%	09F007	^SETEXACT	0C5007	^STOVX
07A007	^GetRoot	0A0007	^NUMMODE	0C6007	^STOMODULO
07B007	^Approx	0A1007	^CLREXACT	0C7007	^RCLEPS
07C007	^#FACT	0A2007	^EXACT?	0C8007	^ISIDREAL?
07D007	^CHECKSING	0A3007	^STEPBYSTEP	0C9007	^ADDTOREAL
07E007	^DESOLVE	0A4007	^NOSTEPBYSTEP	0CA007	^RESETCASCFG
07F007	^ODE_INT	0A5007	^VERBOSEMODE	0CB007	^FRACPARITY
080007	^LINSOLV	0A6007	^SILENTMODE	0CC007	^POLYPARITY
081007	^LDECSOLV	0A7007	^RECURMODE	0CD007	^PARITYTEST
082007	^LDEGENE	0A8007	^NONRECMODE	0CE007	^COSTEST
083007	^LDEPART	0A9007	^PLUSAT0	0CF007	^SHRINKEVEN
084007	^LDSSOLVext	0AA007	^SETPLUSAT0	0D0007	^SINTEST
085007	^ODETYPESTO	0AB007	^PLUSATINFTY	0D1007	^SHRINK2SYM
086007	^ODE_SEPAR	0AC007	^CLRPLUSAT0	0D2007	^SHRINKSYM
087007	^LAPext	0AD007	^SPARSEDATA	0D3007	^SHRINK2ASYM
088007	^ILAPext	0AE007	^FULLDATA	0D4007	^SHRINKASYM

0D5007	^FR2ND%	0FB007	^SUMVX	0030AB	~xXXRNG
0D6007	^POLYSYM	0FC007	^FLAGSUMVX	0040AB	~xYYRNG
0D7007	^POLYASYM	0FD007	^RATSUM	0050AB	~xEYEPT
0D8007	^P2P#	0FE007	^FTAYL	0060AB	~xNUMX
0D9007	^NDEvalN/D	0FF007	^CSTFRACTION?	0070AB	~xNUMY
0DA007	^PEvalN/D	100007	^NONRATSUM	0080AB	~xWIREFRAME
0DB007	^POSITIFext	101007	^LINEARAPPLY	0090AB	~xPARSURFACE
0DC007	^SIGNE1ext	102007	^linearapply	00A0AB	~xGRIDMAP
0DD007	^SIGNEext	103007	^meta_cst?	00B0AB	~xYSLICE
0DE007	^SIGNUNDEF	104007	^HYPERGEO	00C0AB	~xSLOPEFIELD
0DF007	^SIGNPLUS	105007	^fk+1/fk	00D0AB	~xPCONTOUR
0E0007	^SIGNMOINS	106007	^A/B2PQR	00E0AB	~xDIFFEQ
0E1007	^SIGNELN	107007	^GOSPER?	00F0AB	~xVERSION
0E2007	^SIGNEEXP	108007	^ZEILBERGER	0110AB	~xRECT
0E3007	^SIGNESIN	109007	^SYMPSI	0120AB	~xCYLIN
0E4007	^SIGNECOS	10A007	^sympsi	0130AB	~xSPHERE
0E5007	^SIGNETAN	10B007	^SYMPSIN	0140AB	~xANIMATE
0E6007	^SIGNEATAN	10C007	^sympsin	0150AB	~xLININ
0E7007	^SIGNESQRT	10D007	^IBERNOULLI	0160AB	~xLIBEVAL
0E8007	^SUBSIGNE	10E007	^FLAGRESULTANT	0170AB	~xFLASHEVAL
0E9007	^SIGNERIGHT	10F007	^RESULTANT	0180AB	~xCONLIB
0EA007	^SIGNELEFT	110007	^RESULTANTLP	0190AB	~xCONST
0EB007	^>SIGNE	111007	^RESPSHIFTQ	01A0AB	~xFFT
0EC007	^SIGNE>	112007	^ADDONEVAR	01B0AB	~xIFFT
0ED007	^SIGNMULText	113007	^IROOTS	01C0AB	~xNDIST
0EE007	^ZSIGNECK	114007	^TYPEGAUSSINT?	01D0AB	~xPSDEV
0EF007	^SIGNEERROR	115007	^DTYPEGAUSSINT?	01E0AB	~xPVAR
0F0007	^ZSIGNE	116007	^DUPTYPEGAUSSINT?	01F0AB	~xPCOV
0F1007	^zsigne	117007	^PPZZ	0200AB	~xRKF
0F2007	^PASCAL_NEXTLINE	118007	^DISTRIB*	0210AB	~xRKFSTEP
0F3007	^DELTAPSOLVE	119007	^NONALGERR	0220AB	~xRKFERR
0F4007	^SOLVEMETASYST	11A007	^ALGCASCOMPEVAL	0230AB	~xRRK
0F5007	^REDUCEMETASYST	11C007	^%%PSI	0240AB	~xRRKSTEP
0F6007	^REDUCEMETAPSYST	1DC007	^PUSHFLAGS	0250AB	~xRSBERR
0F7007	^SOLVECRAMER	1DD007	^POPFLAGS	0260AB	~xCOND
0F8007	^QUOTExSIGMA	0000AB	~xXVOL	0270AB	~xTRACE
0F9007	^SUM	0010AB	~xYVOL	0280AB	~xSRAD
0FA007	^FLAGSUM	0020AB	~xZVOL	0290AB	~xSNRM

02A0AB	~xRANK	0510AB	~xHEAD	0770AB	~xMROOT
02B0AB	~xLSQ	0520AB	~xTAIL	0050B0	~IFMenuRow1
02C0AB	~xEGV	0530AB	~xSEQ	0060B0	~IFMenuRow2
02D0AB	~xEGVL	0540AB	~xDOSUBS	0860B0	~grobAlertIcon
02E0AB	~xSVD	0550AB	~x∆LIST	0870B0	~grobCheckKey
02F0AB	~xSVL	0560AB	~xNSUB	0C80B0	~gFldVal
0300AB	~xLU	0570AB	~xENDSUB	0D80B0	~sFldVal
0310AB	~xQR	0580AB	~xSTREAM	0DE0B0	~nNullBind
0320AB	~xLQ	0590AB	~xΣLIST	0000B1	~DoMsgBox
0330AB	~xSCHUR	05A0AB	~xΠLIST	0040B1	~MsgBoxMenu
0340AB	~xRREF	05B0AB	~xDOLIST	0000B3	~Choose
0350AB	~xRANM	05C0AB	~xADD	0050B3	~ChooseMenu0
0360AB	~x→ROW	05D0AB	~xREVLIST	0060B3	~ChooseMenu1
0370AB	~xROW→	05E0AB	~xSORT	0070B3	~ChooseMenu2
0380AB	~x→COL	05F0AB	~xZFACTOR	0150B3	~BBMoveTo
0390AB	~xCOL→	0600AB	~xFANNING	0190B3	~BBRecalOff&Disp
03A0AB	~x→DIAG	0610AB	~xDARCY	0220B3	~BBRunEntryProc
03B0AB	~xDIAG→	0620AB	~xF0λ	0230B3	~BBReReadPageSize
03C0AB	~xROW-	0630AB	~xSIDENS	0240B3	~BBReReadHeight
03D0AB	~xROW+	0640AB	~xTDELTA	0250B3	~BBReReadCoords
03E0AB	~xCOL-	0650AB	~xTINC	0260B3	~BBReReadWidth
03F0AB	~xCOL+	0660AB	~xgmol	0280B3	~BBRunENTERAction
0400AB	~xRSWP	0670AB	~xlbmol	0290B3	~BBRunCanclAction
0410AB	~xCSWP	0680AB	~xrpm	02F0B3	~BBReDrawBackgr
0420AB	~xRCI	0690AB	~xdB	0370B3	~BBGetNGrob
0430AB	~xRCIJ	06A0AB	~xPINIT	0380B3	~BBGetNStr
0440AB	~xPROOT	06B0AB	~xDRAW3DMATRIX	03B0B3	~BBRereadChkEnbl
0450AB	~xPCOEF	06C0AB	~x→KEYTIME	03C0B3	~BBRereadFullScr
0460AB	~xPEVAL	06D0AB	~xKEYTIME→	03D0B3	~BReReadMenus
0470AB	~xTVM	06E0AB	~xXSERV	03E0B3	~BBReReadNElems
0480AB	~xTVMBEG	06F0AB	~xROMUPLOAD	03F0B3	~BBGetN
0490AB	~xTVMEND	0700AB	~xXGET	04B0B3	~BBIsChecked?
04A0AB	~xTVMROOT	0710AB	~xXPUT	0520B3	~BBUpArrow
04B0AB	~xAMORT	0720AB	~xMSOLVR	0530B3	~BBDownArrow
04C0AB	~xINFORM	0730AB	~xMINIT	0540B3	~BBSpace
04E0AB	~xMSGBOX	0740AB	~xMITM	0590B3	~BBPgDown
04F0AB	~xXSEND	0750AB	~xMUSER	05A0B3	~BBPgUp
0500AB	~xXRECV	0760AB	~xMCALC	05B0B3	~BBEmpty?

05C0B3	~BBGetDefltHeight	0080DE	~xqr	02E0DE	~xTESTS
05F0B3	~$>grobOrGROB	0090DE	~xGRAMSCHMIDT	02F0DE	~xMATHS
0630B3	~ChooseSimple	00A0DE	~xSYST2MAT	0300DE	~xCOLLECT
0C70B5	~%TICKSweek	00B0DE	~xCHOLESKY	0310DE	~xUNASSIGN
0CB0B5	~%HrTicks	00C0DE	~xDIAGMAP	0320DE	~xHELP
0CD0B5	~%TICKSmin	00D0DE	~xISOM	0330DE	~xCASCMD
0CF0B5	~%TICKSsec	00E0DE	~xMKISOM	0340DE	~xPUSH
0000DD	~x→LANGUAGE	00F0DE	~xKER	0350DE	~xPOP
0010DD	~xLANGUAGE→	0100DE	~xIMAGE	0360DE	~xDEGREE
0020DD	~x→FONT	0110DE	~xBASIS	0370DE	~xDEDICACE
0030DD	~xFONT→	0120DE	~xIBASIS	0380DE	~xPOTENTIAL
0040DD	~x→HEADER	0130DE	~xAUGMENT	0390DE	~xVPOTENTIAL
0050DD	~xHEADER→	0140DE	~xPMINI	03F0DE	~xRCLVX
0060DD	~x→NDISP	0150DE	~xCYCLOTOMIC	0400DE	~xSTOVX
0070DD	~xEDIT	0160DE	~xSTURM	0030E0	~BRStoC1
0080DD	~xVISIT	0170DE	~xSTURMAB	0100E0	~BRbrowse
0090DD	~xEDITB	0180DE	~xFDISTRIB	0130E0	~BRoutput
00A0DD	~xVISITB	0190DE	~xDISTRIB	0180E0	~BRRclCurRow
00B0DD	~xEQW	01A0DE	~xEXP2POW	0190E0	~BRRclC1
00C0DD	~xFILER	01B0DE	~xPOWEXPAND	0120E4	~MESRclEqn
00D0DD	~xFONT8	01C0DE	~xTAN2CS2	0110E7	~UTVUNS1Arg
00E0DD	~xFONT7	01D0DE	~xCIRC	02E0E7	~PCunpack
00F0DD	~xFONT6	01E0DE	~xC2P	02F0E7	~UTTYPEEXT0?
0100DD	~xSREPL	01F0DE	~xP2C	0030E8	~dontuple#
0110DD	~x→MINIFONT	0200DE	~xMSLV	01E0E8	~INTEMPOB?
0120DD	~xMINIFONT→	0210DE	~xDOMAIN	000100	~x→H
0130DD	~xRENAME	0220DE	~xSIMPLIFY	001100	~xH→
0140DD	~xUFL1→MINIF	0230DE	~xDROITE	002100	~x→A
0150DD	~xDBUG	0240DE	~xSTORE	003100	~xA→
0160DD	~xDISPXY	0250DE	~xDEF	004100	~xA→H
0000DE	~xADDTOREAL	0260DE	~xASSUME	005100	~xH→A
0010DE	~xSIGMAVX	0270DE	~xUNASSUME	006100	~x→CD
0020DE	~xSIGMA	0280DE	~xREWRITE	007100	~xCD→
0030DE	~xPsi	0290DE	~xINTEGER	008100	~xS→H
0040DE	~xPSI	02A0DE	~xCONSTANTS	009100	~xH→S
0050DE	~xRESULTANT	02B0DE	~xHYPERBOLIC	00A100	~x→LST
0060DE	~xIBERNOULLI	02C0DE	~xMODULAR	00B100	~x→ALG
0070DE	~xGAMMA	02D0DE	~xPOLYNOMIAL	00C100	~x→PRG

00D100	~xCOMP→	00D314	~xRISCH	033314	~xSIMP2
00E100	~x→RAM	00E314	~xDERIV	034314	~xPARTFRAC
00F100	~xSREV	00F314	~xDESOLVE	035314	~xPROPFRAC
010100	~xPOKE	010314	~xLAP	036314	~xPTAYL
011100	~xPEEK	011314	~xILAP	037314	~xHORNER
012100	~xAPEEK	012314	~xLDEC	038314	~xEULER
013100	~xR~SB	013314	~xTEXPAND	039314	~xPA2B2
014100	~xSB~B	014314	~xLIN	03A314	~xCHINREM
015100	~xLR~R	015314	~xTSIMP	03B314	~xICHINREM
016100	~xS~N	016314	~xLNCOLLECT	03C314	~xISPRIME?
017100	~xLC~C	017314	~xEXPLN	03D314	~xNEXTPRIME
018100	~xASM→	018314	~xSINCOS	03E314	~xPREVPRIME
019100	~xBetaTesting	019314	~xTLIN	03F314	~xSOLVE
01A100	~xCRLIB	01A314	~xTCOLLECT	040314	~xZEROS
01B100	~xCRC	01B314	~xTRIG	041314	~xFCOEF
01C100	~xMAKESTR	01C314	~xTRIGCOS	042314	~xFROOTS
01D100	~xSERIAL	01D314	~xTRIGSIN	043314	~xFACTORS
01E100	~xASM	01E314	~xTRIGTAN	044314	~xDIVIS
01F100	~xER	01F314	~xTAN2SC	045314	~xTRAN
020100	~x→S2	020314	~xHALFTAN	046314	~xHADAMARD
021100	~xXLIB~	021314	~xTAN2SC2	047314	~xrref
001102	~xGETADR	022314	~xATAN2S	048314	~xREF
002102	~xGETNAME	023314	~xASIN2T	049314	~xAXM
003102	~xGETNAMES	024314	~xASIN2C	04A314	~xAXL
004102	~xGETNEAR	025314	~xACOS2S	04B314	~xQXA
000314	~xEXPAND	026314	~xDIV2	04C314	~xAXQ
001314	~xFACTOR	027314	~xIDIV2	04D314	~xGAUSS
002314	~xSUBST	028314	~xQUOT	04E314	~xSYLVESTER
003314	~xDERVX	029314	~xIQUOT	04F314	~xPCAR
004314	~xINTVX	02A314	~xREMAINDER	050314	~xJORDAN
005314	~xLIMIT	02B314	~xIREMAINDER	051314	~xMAD
006314	~xTAYLOR0	02C314	~xGCD	052314	~xLINSOLVE
007314	~xSERIES	02D314	~xLCM	053314	~xVANDERMONDE
008314	~xSOLVEVX	02E314	~xEGCD	054314	~xHILBERT
009314	~xPLOT	02F314	~xIEGCD	055314	~xLCXM
00A314	~xPLOTADD	030314	~xABCUV	056314	~xDIV
00B314	~xIBP	031314	~xIABCUV	057314	~xCURL
00C314	~xPREVAL	032314	~xLGCD	058314	~xLAPL

059314	~xHESS
05A314	~xLEGENDRE
05B314	~xTCHEBYCHEFF
05C314	~xHERMITE
05D314	~xLAGRANGE
05E314	~xFOURIER
05F314	~xSIGNTAB
060314	~xTABVAR
061314	~xTABVAL
062314	~xDIVPC
063314	~xTRUNC
064314	~xSEVAL
065314	~xTEVAL
066314	~xMAP
067314	~xXNUM
068314	~xXQ
069314	~xREORDER
06A314	~xLVAR
06B314	~xFXND
06C314	~xEXLR
06D314	~xLNAME
06E314	~xADDTMOD
06F314	~xSUBTMOD
070314	~xMULTMOD
071314	~xDIVMOD
072314	~xDIV2MOD
073314	~xPOWMOD
074314	~xINVMOD
075314	~xGCDMOD
076314	~xEXPANDMOD
077314	~xFACTORMOD
078314	~xRREFMOD
079314	~xMODSTO
07A314	~xMENUXY
07B314	~xKEYEVAL
07C314	~xGROBADD
07D314	~xSCROLL
07E314	~xCASCFG
07F314	~xMAIN
080314	xALGB
080314	~xBASE
081314	~xCMPLX
082314	~xTRIGO
083314	~xMATR
084314	~xDIFF
085314	~xARIT
086314	~xSOLVER
087314	~xEXP&LN
088314	~xEPSX0
089314	~x?
08A314	~x∞
08B314	~xPROMPTSTO
08C314	~xVER

Entry Index