

CONTENTS

PART 1: SYSTEM RPL

1. Introduction
2. Object Types
3. Binary Integers
 - 3.1 Built-in Binary Integers
 - 3.2 Binary Integer Conversion
 - 3.3 Binary Integer Tests
 - 3.4 Binary Integer Arithmetic
4. Characters
5. Character Strings
6. Hex Strings
7. Real & Complex Numbers
 - 7.1 Real & Complex Constants
 - 7.2 Real & Complex Conversion
 - 7.3 Real & Complex Testing
 - 7.4 Real & Complex Functions
8. Arrays
 - 8.1 General Operations
 - 8.2 Array Calculations
 - 8.3 Real & Complex Calculations for Arrays
 - 8.4 Statistics
9. Tagged Objects
10. Unit Objects
 - 10.1 General Operations
 - 10.2 Arithmetic Operations
11. Composite Objects
12. Meta Objects
 - 12.1 Meta Operations
 - 12.2 Meta & Object Operations
 - 12.3 Other Operations
13. Symbolics
 - 13.1 Symbolic Operations
 - 13.2 Meta Symbolics
14. Library & Backup objects
 - 14.1 Rom Pointers
 - 14.2 Libraries
 - 14.3 Backups
15. Stack operations
16. Checking Arguments
17. Error Generation & Trapping
 - 17.1 General Words
 - 17.2 Error Words
18. General Tests & Flags
 - 18.1 Flags
 - 18.2 General Tests
 - 18.3 Type Tests
19. Runstream Control
 - 19.1 Definite Loops
 - 19.2 Indefinite loops
 - 19.3 COLA & case words
20. Temporary Variables and Environments

- 21. Variables and Directories
 - 21.1 Temporary Memory
 - 21.2 Storing & Recalling Variables
 - 21.3 Directories
 - 21.4 Hidden Directory
- 22. Display Management
 - 22.1 Display Organization
 - 22.2 Preparing Display
 - 22.3 Clearing Display
 - 22.4 Displaying Text
 - 22.5 Window Coordinates
 - 22.6 Annunciator Control
 - 22.7 Display Refresh
- 23. Graphic Objects
 - 23.1 Dimensions
 - 23.2 Grob Handling
 - 23.3 Built-in Grobs
- 24. Menu Display
 - 24.1 General
 - 24.2 Labels
- 25. Keyboard Control
 - 25.1 General
 - 25.2 Inputline
 - 25.3 Parametrized Outer Loop
 - 25.4 System Catalog Browser
- 26. Time & Alarms
- 27. System Commands
 - 27.1 User & System Flags
 - 27.2 General
- 28. Inbuilt Programs
 - 28.1 System UI
 - 28.2 Interactive Stack
 - 28.3 Parser
 - 28.4 Decompiler
 - 28.5 Kermit
 - 28.6 Debugger
 - 28.7 Matrix Writer
 - 28.8 ParOuterLoop
 - 28.9 Editor

APPENDIX

- A. Memory organization
- B. Internal System Flags
- C. Default Keys
- D. Built-in Libraries
- E. Library Property Fields
- F. Internal System Flags
- G. RAM pointer area

1. Introduction

~~~~~

The reader is assumed to be a system rpl/ml guru. Makes it a lot easier to write this section. (Yes, I did have trouble writing it :)

If a symbol is listed with an address, the symbol is unsupported. The symbols that are at different locations of GX are listed with 2 addresses, first for rev E and second for rev M .

Warning: All of the GX addresses were obtained just by studying rev M rom dump. I have no GX to actually test any of the addresses.

\*\*\*\*\*

## 2. Object Types

~~~~~

	PRLG	Prolog signature	
	DOBINT	Binary number	#
	DOREAL	Real Number	%
02955	DOEREL	Extended Real Number	%%
	DOCMP	Complex Number	C%
0299D	DOECMP	Extended Complex Number	C%%
	DOCHAR	Character	chr
	DOARRAY	Array object	arry
02A0A	DOLNKARRY	Linked Array	lnkarry
	DOCSTR	Character String	\$
	DOHSTR, DOHXS	Hex string	hxs
	DOLIST	List	{}
02A96	DORRP	Directory object	rrp
	DOSYMB	Algebraic object	symb
	DOEXT	Unit object	unit
02AFC	DOTAG	Tagged object	tagged
	DOGROB	Graphics object	grob
	DOLIB	Library object	lib
02B62	DOBAK	Backup object	bak
	DOEXT0	Library Data	libdat
02BAA	DOEXT1	External type 1 (SX)	ext1
02BAA	DOACPTR	Access pointer (GX)	acptr
02BCC	D0EXT2	External type 2	ext2
02BEE	D0EXT3	External type 3	ext3
02C10	DOEXT4	External type 4	ext4
	DOCOL	Secondary object	::
	DOCODE	Code object	code
	DOIDNT	Identifier object	id
	DOLAM	Lambda variable	lam
	DOROMP	ROMPTR object	romptr
	SEMI	Composite end marker	
	comp	= symb :: {}	
	sym	= symb id lam	
	symf	= sym id lam % C%	
	pco	= primitive code object	
	ROMPTR	= ROMPTR object	
	romptr	= contents of ROMPTR (preceded by prop etc fields)	
	romp	= ROMPTR or romptr	
	F%	= % or C%	
	L%	= %% or C%%	
	B%	= %, C%, %% or C%%	

3. Binary Integers

~~~~~

#### 3.1 Built-in Binary Integers

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Type Numbers:

|       |          |              |       |          |              |
|-------|----------|--------------|-------|----------|--------------|
|       | TYPEREAL | ( -> #2933 ) | 03FB3 | TYPECOL  | ( -> #2D9D ) |
| 03FDB | TYPEEREL | ( -> #2955 ) | 03FBD | YPESYMB  | ( -> #2AB8 ) |
| 03F95 | TYPECMP  | ( -> #2977 ) | 03F9F | TYPELIST | ( -> #2A74 ) |
|       | TYPEIDNT | ( -> #2948 ) | 03FC7 | TYPERRP  | ( -> #2A96 ) |
| 03FD1 | TYPELAM  | ( -> #2E6D ) | 03FE5 | TYPEEXT  | ( -> #2ADA ) |

HP:                    Help:    Key:                    (unsupported)

|               |      |               |                 |            |
|---------------|------|---------------|-----------------|------------|
| ZERO          | ##0  |               | (any)           |            |
| ONE           | ##1  | kpNoShift     | real            | kcMenuKey1 |
| TWO           | ##2  | kpLeftShift   | (cmp)           | kcMenuKey2 |
| THREE         | ##3  | kpRightShift  | str             | kcMenuKey3 |
| FOUR          | ##4  | kpANoShift    | (arry)          | kcMenuKey4 |
| FIVE          | ##5  | kpALeftShift  | list            | kcMenuKey5 |
| SIX           | ##6  | kpARightShift | id, idnt        | kcMenuKey6 |
| SEVEN         | ##7  | kcMathMenu    | (lam)           |            |
| EIGHT         | ##8  | kcPrgmMenu    | seco            |            |
| NINE          | ##9  | kcCustomMenu  | symb            |            |
| TEN           | ##A  | kcVarsMenu    | sym             |            |
| ELEVEN        | ##B  | kcUpArrow     | (hxs)           |            |
| TWELVE        | ##C  | kcNextRow     | (grob)          |            |
| THIRTEEN      | ##D  | kcTick        | (tagged)        |            |
| FOURTEEN      | ##E  | kcSto         | (unitob)        |            |
| FIFTEEN       | ##F  | kcEval        | (romp)          |            |
| SIXTEEN       | ##10 | kcLeftArrow   | REALOB          |            |
| SEVENTEEN     | ##11 | kcDownArrow   | 2REAL, REALREAL |            |
| EIGHTEEN      | ##12 | kcRightArrow  |                 |            |
| NINETEEN      | ##13 | kcSin         |                 |            |
| TWENTY        | ##14 | kcCos         |                 |            |
| TWENTYONE     | ##15 | kcTan         |                 |            |
| TWENTYTWO     | ##16 | kcSqrt        |                 |            |
| TWENTYTHREE   | ##17 | kcPower       |                 |            |
| TWENTYFOUR    | ##18 | kcInverse     |                 |            |
| TWENTYFIVE    | ##19 | kcEnter       |                 |            |
| TWENTYSIX     | ##1A | kcNegate      |                 |            |
| TWENTYSEVEN   | ##1B | kcEnterExp    |                 |            |
| TWENTYEIGHT   | ##1C | kcDelete      |                 |            |
| TWENTYNINE    | ##1D | kcBackSpace   | REALEXT         |            |
| THIRTY        | ##1E | kcAlpha       |                 |            |
| THIRTYONE     | ##1F | kc7           | bin             |            |
| THIRTYTWO     | ##20 | kc8           |                 |            |
| THIRTYTHREE   | ##21 | kc9           |                 |            |
| THIRTYFOUR    | ##22 | kcDivide      |                 |            |
| THIRTYFIVE    | ##23 | kcLeftShift   |                 |            |
| THIRTYSIX     | ##24 | kc4           |                 |            |
| THIRTYSEVEN   | ##25 | kc5           |                 |            |
| THIRTYEIGHT   | ##26 | kc6           |                 |            |
| THIRTYNINE    | ##27 | kcTimes       |                 |            |
| FORTY, FOURTY | ##28 | kcRightShift  |                 |            |
| FORTYONE      | ##29 | kc1           |                 |            |

|            |      |          |         |
|------------|------|----------|---------|
| FORTYTWO   | ##2A | kc2      |         |
| FORTYTHREE | ##2B | kc3      |         |
| FORTYFOUR  | ##2C | kcMinus  |         |
| FORTYFIVE  | ##2D | kcOn     |         |
| FORTYSIX   | ##2E | kc0      |         |
| FORTYSEVEN | ##2F | kcPeriod | (rrp)   |
| FORTYEIGHT | ##30 | kcSpace  |         |
| FORTYNINE  | ##31 | kcPlus   |         |
| FIFTY      | ##32 |          |         |
| FIFTYONE   | ##33 |          |         |
| FIFTYTWO   | ##34 |          |         |
| FIFTYTHREE | ##35 |          | STRLIST |
| FIFTYFOUR  | ##36 |          |         |
| FIFTYFIVE  | ##37 |          |         |
| FIFTYSIX   | ##38 |          |         |
| FIFTYSEVEN | ##39 |          |         |
| FIFTYEIGHT | ##3A |          |         |
| FIFTYNINE  | ##3B |          |         |
| SIXTY      | ##3C |          |         |
| SIXTYONE   | ##3D |          |         |
| SIXTYTWO   | ##3E |          |         |
| SIXTYTHREE | ##3F |          | (ereal) |
| SIXTYFOUR  | ##40 | BINT40h  |         |

( Some more are supported but not listed )

Multiple Binary Integers + Stack Handling:

|              |             |            |
|--------------|-------------|------------|
| ZEROZERO     | ONEONE      | aka ONEDUP |
| #ZERO#ONE    | #ONE#27     |            |
| ZEROZEROZERO | #TWO#ONE    |            |
| ZEROZEROONE  | #TWO#TWO    |            |
| ZEROZEROTWO  | #TWO#FOUR   |            |
| #ZERO#SEVEN  | #THREE#FOUR |            |
|              | #FIVE#FOUR  |            |

|       |           |   |             |     |            |   |
|-------|-----------|---|-------------|-----|------------|---|
| 64449 | DROPZERO  | ( | ob          | --> | #0         | ) |
|       | 3DROPZERO | ( | ob1 ob2 ob3 | --> | #0         | ) |
|       | DROPONE   | ( | ob          | --> | #1         | ) |
|       | 2DROP00   | ( | ob1 ob2     | --> | #0 #0      | ) |
|       | DUPZERO   | ( | ob          | --> | ob ob #0   | ) |
|       | DUPONE    | ( | ob          | --> | ob ob #1   | ) |
|       | DUPTWO    | ( | ob          | --> | ob ob #2   | ) |
|       | SWAPONE   | ( | ob1 ob2     | --> | ob2 ob1 #1 | ) |
|       | ZEROSWAP  | ( | ob          | --> | #0 ob      | ) |
|       | ZEROOVER  | ( | ob          | --> | ob #0 ob   | ) |
|       | ZEROFALSE | ( |             | --> | #0 FALSE   | ) |
|       | ONESWAP   | ( | ob          | --> | #1 ob      | ) |
|       | ONEFALSE  | ( |             | --> | #1 FALSE   | ) |

### 3.2 Binary Integer Conversion

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

|       |   |     |     |     |   |       |   |     |     |     |   |
|-------|---|-----|-----|-----|---|-------|---|-----|-----|-----|---|
| #>HXS | ( | #   | --> | hxs | ) | CHR># | ( | chr | --> | #   | ) |
| HXS># | ( | hxs | --> | #   | ) | #>CHR | ( | #   | --> | chr | ) |
|       |   |     |     |     |   | #:>\$ | ( | #   | --> | \$  | ) |
|       |   |     |     |     |   | #>\$  | ( | #   | --> | \$  | ) |

```

COERCE      ( % --> # )
COERCEDUP   ( % --> # # )
COERCESWAP  ( ob % --> # ob )
COERCE2     ( % % --> # # )
%ABSCOERCE  ( % --> # )
%IP>#       ( % --> # )          (Actually takes ABS too)
UNCOERCE    ( # --> % )
UNCOERCE2   ( # # --> % % )
UNCOERCE%%  ( # --> %% )
19529 UNCOERCE{}2 ( { # ( # ) } --> { % ( % ) } )

```

### 3.3 Binary Integer Tests

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

#=      #<>      #<      #>      2DUP#=      2DUP#<  2DUP#>
#0=      #0<>
#1=      #1<>      #>1      DUP#0=  DUP#0<>
#2=      #2<>
#3=      #<3
#5=      DUP#<7

```

```

ONE_EQ      (EQ check, not EQUAL check)
OVER#0=
OVER#=
OVER#<
OVER#>

```

```

2#0=OR      ( #m #n --> flag )      Either one zero?

```

### 3.3 Binary Integer Arithmetic

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

OVF means 0 <= result <= FFFFF

```

#+      #-      #*      #/      (-> #rem #quo)
#+OVF (25B0B)  #-OVF (6D28B)  #*OVF
#1+      #1-      #1+ '
#2+      #2-      #2*      #2/      (Rounded down)
#3+      #3-
#4+      #4-
#5+      #5-
#6+      #6-      #6*
#7+
#8+      #8*
#9+
#10+      #10*
#12+

#-+1      #1--      #+-1
#-#2/

#+OVER      #+SWAP      #+DUP
#-OVER      #-SWAP      #-DUP
OVER#+      ROT#+      3PICK#+      4PICK#+
OVER#-      SWAP#-      ROT#-

#1+DUP      #1-DUP      #1+SWAP      #1+ROT
DUP#1+      DUP#1-      SWAP#1+,SWP1+  SWAP#1-      ROT#1+

```

```
#1-1SWAP
#1-ROT
ROT#1+UNROT
#1-UNROT
2DUP#+
4PICK#+SWAP, 4PICK+SWAP
DUP#2+
DUP3PICK#+
DROP#1-
SWAP#1+SWAP
SWAP#1-SWAP
SWAPOVER#-
ROT#+SWAP, ROT+SWAP

#MIN
#MAX
#AND      ( #2 #1 -> # ) Bitwise AND
```

\*\*\*\*\*

#### 4. Characters

~~~~~

Conversion functions:

```
CHR>#    ( chr --> # )
#>CHR    ( # --> chr )
CHR>$    ( chr --> $ )
```

Constants:

```
CHR_# CHR_* CHR_+ CHR_, CHR_- CHR_. CHR_/
CHR_0 CHR_1 CHR_2 CHR_3 CHR_4 CHR_5 CHR_6 CHR_7 CHR_8 CHR_9
CHR_: CHR_; CHR_< CHR_= CHR_> CHR_-> CHR_<< CHR_>>
CHR_[ CHR_] CHR_{ CHR_} CHR_<= CHR_>= CHR_<>
CHR_A ... CHR_Z CHR_a ... CHR_z
CHR_Angle CHR_Deriv CHR_Integral CHR_LeftPar CHR_Newline
CHR_Pi CHR_RightPar CHR_Sigma CHR_Space CHR_UndScore CHR_DblQuote
CHR_00 CHR_...
```

5. Character Strings

~~~~~

String constants are not listed in this file, please see rplman.doc or other entry lists.

Conversion:

```
ID>$      ( id/lam --> $      )      DOCHR    ( % --> $ )
$>ID      (      $ --> id      )      DONUM    ( $ --> % )
DUP$>ID    (      $ --> $ id    )

1CA26     LEN$      ( $      --> #      )      Returns lenght in bytes
1CA26     LEN$>%     ( $      --> %      )
1CA26     DUPLLEN$   ( $      --> $ #      )
1782E     OVERLEN$   ( $ ob --> $ ob #    )
127CA     2LEN$#+    ( $ $ --> $ $ #      )      Adds lengths of 2 strings.
127CA     DROPDUPLLEN$1+ ( $ ob --> $ #len+1 )

CAR$      ( $ --> chr / $ )      NULL$ for NULL$
CDR$      ( $ --> $'      )      NULL$ for NULL$

SUB$      (      $ #start #end --> $      )
#1-SUB$   (      $ #start #end --> $      )
1_#1-SUB$ (      $      #end --> $      )
LAST$     (      $ #start      --> $      )
#1+LAST$  (      $ #start      --> $      )
SUB$SWAP  ( ob $ #start #end --> $ ob )
SUB$1#    (      $ #pos --> #char ) Returns nth char as #
1C8BB     XEQSUB$    (      $ %start %end --> $      )
1C8BB     COERCE$22  (      $ --> $'      )
1C8BB     If len>21 then sub until 21 and append ... char
SEP$NL    ( $ --> $2 $1 )
Split $ at first newline
```



|       |               |                                     |          |                                                       |
|-------|---------------|-------------------------------------|----------|-------------------------------------------------------|
|       | NULL\$        | ( --> \$ )                          |          |                                                       |
|       | NULL\$SWAP    | ( ob --> \$ ob )                    |          |                                                       |
|       | NULL\$TEMP    | ( --> \$ )                          |          | Creates NULL\$ in TEMPOB                              |
|       | DROPNULL\$    | ( ob --> NULL\$ )                   |          |                                                       |
| 04D57 | 2DROPNULL\$   | ( ob1 ob2 --> NULL\$ )              |          |                                                       |
|       | NULL\$?       | ( \$ --> flag )                     |          |                                                       |
|       | DUPNULL\$?    | ( \$ --> \$ flag )                  |          |                                                       |
|       |               |                                     |          |                                                       |
|       | >H\$          | ( \$ chr --> \$' )                  |          | Prepends char to \$                                   |
|       | >T\$          | ( \$ chr --> \$' )                  |          | Appends char to \$                                    |
|       | &\$           | ( \$1 \$2 --> \$1+\$2 )             |          |                                                       |
|       | SWAP&\$       | ( \$1 \$2 --> \$2+\$1 )             |          |                                                       |
|       | &\$SWAP       | ( ob \$1 \$2 --> \$1+\$2 ob )       |          |                                                       |
|       | !append\$     | ( \$1 \$2 --> \$1+\$2 )             |          |                                                       |
|       |               |                                     |          | Tries &\$, if not enough memory then !!append\$?      |
|       | !append\$SWAP | ( ob \$1 \$2 --> \$1+\$2 ob )       |          |                                                       |
|       | !insert\$     | ( \$1 \$2 --> \$2+\$1 )             |          |                                                       |
|       |               |                                     |          | Effectively SWAP !append\$                            |
|       | !!append\$?   | ( \$1 \$2 --> \$1+\$2 )             |          |                                                       |
|       |               |                                     |          | Attempts appending "in place" if target is in tempob. |
|       | !!append\$    | ( \$1 \$2 --> \$1+\$2 )             |          |                                                       |
|       |               |                                     |          | Tries appending in place.                             |
|       | !!insert\$    | ( \$1 \$2 --> \$2+\$1 )             |          |                                                       |
|       |               |                                     |          | Tries inserting in place.                             |
|       | NEWLINE\$&\$  | ( \$ --> \$' )                      |          | (aka NEWLINE&\$)                                      |
|       | APPEND_SPACE  | ( \$ --> \$' )                      |          |                                                       |
|       | Blank\$       | ( # --> \$ )                        |          | Creates a string of # spaces                          |
|       |               |                                     |          |                                                       |
|       | POS\$         | ( \$search \$find #start --> #pos ) |          |                                                       |
|       |               |                                     |          | Returns position of \$find in \$search.               |
|       |               |                                     |          | Search from start onwards. #pos=0 if not found.       |
|       | POSCHR        | ( \$search chr #start --> #pos )    |          |                                                       |
|       |               |                                     |          | Actually the same entry as above.                     |
| 1CAD7 | XEQPOS\$      | ( \$search \$find --> %pos )        |          |                                                       |
| 15EF6 | ONEPOS\$      | ( \$search \$find/chr --> #pos )    |          |                                                       |
|       |               |                                     |          | POS\$ with start location ONE                         |
|       | POS\$REV      | ( \$search \$find #limit --> #pos ) |          |                                                       |
|       |               |                                     |          | Starts search from end until limit.                   |
|       | POSCHRREV     | ( \$search chr #limit --> #pos )    |          |                                                       |
|       |               |                                     |          | Actually the same entry as above.                     |
|       |               |                                     |          |                                                       |
|       | AND\$         | ( \$1 \$2 --> \$1 AND \$2 )         | 188E6    | !AND\$                                                |
|       | OR\$          | ( \$1 \$2 --> \$1 OR \$2 )          | 188F5    | !OR\$                                                 |
|       | XOR\$         | ( \$1 \$2 --> \$1 XOR \$2 )         | 18904    | !XOR\$                                                |
| 188D2 | NOT\$         | ( \$ --> \$' )                      | 18961    | !NOT\$                                                |
|       |               |                                     |          |                                                       |
|       | Testing:      | ( \$1 \$2 --> %flag )               |          |                                                       |
|       |               |                                     |          |                                                       |
| 1429A | STR>STR       | 142BA                               | STR>=STR |                                                       |
| 142A6 | STR<STR       | 142E2                               | STR<=STR |                                                       |

\*\*\*\*\*

## 6. Hex Strings

~~~~~

Conversion:

```
4F3D1  #>HXS      ( # --> hxs )    Lenght will be 5
      HXS>#    ( hxs --> # )    Takes lower 5 nibbles
      2HXS>#   ( hxs hxs --> # # )
      2HXSLIST? ( { hxs1 hxs2 } --> #1 #2 )
      Converts list of two hxs into two bints
      Generates Bad Argument Value error for invalid input
      #>% aka HXS>% ( hxs --> % )
      %>#         ( % --> hxs )
      hxs>$       ( hxs --> $ )    Uses current disp. mode & wordsize
      HXS>$       ( hxs --> $ )    Does hxs>$, then appends base character
```

```
0EDE1  MAKEHXS      ( #nibbles --> HXS_000... )
3742D  !MAKEHXS     ( #nibbles --> HXS_random_data )
      EXPAND        ( hxs #length --> hxs' )

      &HXS          ( hxs1 hxs2 --> hxs' )    Appends hxs2 to hxs1
      NULLHXS       ( --> hxs )              Pushes NULLHXS
05566  NULLHXS?     ( hxs --> flag )          zero-lenght hxs?
      LENHXS        ( hxs --> # )            Returns lenght in nibbles
      SUBHXS        ( hxs #start #send --> hxs' )
```

Arithmetic:

All bit words assume 64-bit or shorter hex strings.
Result hxs lenght will be current wordsize.

```
53EC3  bitAND      bitOR        bitXOR        bitNOT
      bitSL       bitSLB       bitSR         bitSRB
      bitRR       bitRRB       bitRL         bitRLB
      bitASR
      bit+        bit-         bit*         bit/
      bitNEG

      bit%#/      ( % hxs --> hxs )
      bit%#/      ( hxs % --> hxs )
      bit%#*      ( % hxs --> hxs )
      bit%#*      ( hxs % --> hxs )
      bit%#-      ( % hxs --> hxs )
      bit%#-      ( hxs % --> hxs )
      bit%#+      ( % hxs --> hxs )
      bit%#+      ( hxs % --> hxs )
```

Testing:

```
544D9  HXS=        ( hxs hxs --> %flag )    == test
      HXS#HXS      ( hxs hxs --> %flag )    <> test
      HXS>HXS      ( hxs hxs --> %flag )    > test
      HXS<HXS      ( hxs hxs --> %flag )    > test
      HXS>=HXS     ( hxs hxs --> %flag )    >= test
      HXS<=HXS     ( hxs hxs --> %flag )    <= test
```

7. Real & Complex Numbers

~~~~~

### 7.1 Real & Complex Constants

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

|       |            |           |              |     |     |         |           |         |       |      |     |     |     |     |
|-------|------------|-----------|--------------|-----|-----|---------|-----------|---------|-------|------|-----|-----|-----|-----|
|       | %0         | %1        | %2           | %3  | %4  | %5      | %6        | %7      | %8    | %9   | %10 | %11 | %12 | %13 |
|       |            | %-1       | %-2          | %-3 | %-4 | %-5     | %-6       | %-7     | %-8   | %-9  |     |     |     |     |
|       | %%0        | %%1       | %%2          | %%3 | %%4 | %%5     | %%7       | %%10    | %%12  | %%60 |     |     |     |     |
|       | %14        | %15       | %16          | %17 | %18 | (1CDF2) | %19       | (1CE07) |       |      |     |     |     |     |
|       | %20        | %21       | %22          | %23 | %24 | %25     | %26       | %27     |       |      |     |     |     |     |
|       | % .1       | % .5      |              |     |     |         |           |         |       |      |     |     |     |     |
|       | %% .1      | %% .5     | %% .4        |     |     |         |           |         |       |      |     |     |     |     |
|       | %100       | %180      | %360         |     |     |         |           |         |       |      |     |     |     |     |
|       | %PI        |           |              |     |     |         | %MAXREAL  |         |       |      |     |     |     |     |
| 2A458 | %%PI       |           |              |     |     |         | %-MAXREAL |         |       |      |     |     |     |     |
|       | PI/180     |           |              |     |     |         | %MINREAL  |         |       |      |     |     |     |     |
| 514EB | %2PI       |           |              |     |     |         | %-MINREAL |         |       |      |     |     |     |     |
|       | %%2PI      |           |              |     |     |         |           |         |       |      |     |     |     |     |
|       | %e         |           |              |     |     |         |           |         |       |      |     |     |     |     |
| 0EFEE | %TICKSsec  |           | (8192)       |     |     |         |           | 22352   | %1200 |      |     |     |     |     |
| 0F003 | %TICKSmin  |           | (491520)     |     |     |         |           | 22367   | %2400 |      |     |     |     |     |
| 0F018 | %TICKShour |           | (29491200)   |     |     |         |           | 2237C   | %4800 |      |     |     |     |     |
| 0F02D | %TICKSday  |           | (707788800)  |     |     |         |           | 22391   | %9600 |      |     |     |     |     |
| 0F042 | %TICKSweek |           | (4954521600) |     |     |         |           |         |       |      |     |     |     |     |
|       | C%0        | (0,0)     |              |     |     |         |           |         |       |      |     |     |     |     |
|       | C%1        | (1,0)     |              |     |     |         |           |         |       |      |     |     |     |     |
|       | C%-1       | (-1,0)    |              |     |     |         |           |         |       |      |     |     |     |     |
|       | C%%1       | (%%1,%%0) |              |     |     |         |           |         |       |      |     |     |     |     |
| 5267F | C%i        | (0,1)     |              |     |     |         |           |         |       |      |     |     |     |     |
| 526AE | C%-i       | (0,-1)    |              |     |     |         |           |         |       |      |     |     |     |     |

### Constants + Stack Handling:

|       |              |       |            |
|-------|--------------|-------|------------|
| 5198F | DROP%0       | 54B1E | DROP%0ABND |
| 1F047 | 2DROP%0      |       |            |
| 56AFB | 4DROP%0      |       |            |
| 1CA0D | DROP%1       | 54A9C | DROP%1ABND |
| 50A3B | UNROT2DROP%0 |       |            |

## 7.2 Real & Complex Conversion

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

%%>%      ( %% --> % )
%>%%      ( %  --> %% )
%>%%SWAP   ( ob % --> %% ob )
2%>%%     ( % % --> %% % )
2%>%      ( %% % --> % % )

```

```

%>C%      ( %re %im  --> C% )
SWAP%>C%   ( %im %re  --> C% )
%%>C%      ( %%re %%im --> C% )
05C8A  %%>C% ( %%re %%im --> C% ) ( GX: 05C72 )
Re>C%      ( %re      --> C% )

```

```

51C6B  C%>%      ( C%  --> %re %im )
SWAP2C%>% ( C%2 C%1 --> %re1 %im1 %re2 %im2 )
C%>%%      ( C%  --> %%re %%im )
C%>%%SWAP   ( C%  --> %%im %%re )
C>Re%      ( C%  --> %re )
C>Im%      ( C%  --> %im )
C%>C%      ( C%  --> C% )
C%>%%      ( C%  --> %%re %%im )
51C84  SWAP2C%>% ( C%2 C%1 --> %%re1 %%im1 %%re2 %%im2 )

```

## 7.3 Real & Complex Testing

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

%=          %<>      %<      %>          %%<      %%>
                %<=      %>=          %%<=      %%>=
%0=          %0<>      %0<      %0>      %%0=      %%0<>      %0<      %0>
                %0>=          %%0<=      %%0>=

```

DUP%0=

C%0= C%0=

## 7.4 Real & Complex Functions

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

%+          51C16  C%+C  51BD0  C%+R  51BF8  C%R+C
%%+          51C3E  C%%+C  51C9D  C%%+R  51CB1  C%%R+C
%-          51CFC  C%-C  51CE8  C%-R  51CD4  C%R-C
%%-          51D10  C%%-C  51D38  C%%-R  51D24  C%%R-C
%*          51D88  C%C*C  51D4C  C%C*R  51D60  C%R*C
%%*          51DE2  C%%C*C  51DAB  C%%C*R  51DBF  C%%R*C
%/          51EC8  C%C/C  51E64  C%C/R  51E19  C%R/C
%%/          51F13  C%%C/C  51F7C  C%%C/R  51F3B  C%%R/C
%^          C%C^C  C%C^R  C%R^C
%%^          51A4A  C%*i  51A5F  C%/i

```

```

%ABS          %%ABS          C%ABS  52080  C%%ABS
%CHS          %%CHS          C%CHS          C%%CHS
%1/          %%1/          C%1/
%SQRT          %%SQRT          C%SQRT          1B3F5  CK%SQRT
1B47B  %SQ          1B48F  C%SQ
%SGN          C%SGN
C%CONJ          C%%CONJ
1B30D  %ARG          C%ARG
%EXP          %%EXP          C%EXP

```

|       |               |                                                |             |     |                 |
|-------|---------------|------------------------------------------------|-------------|-----|-----------------|
|       | %EXPM1        |                                                |             |     |                 |
|       | %LN           | %%LN                                           | C%LN        |     | 1B995 CK%LN     |
|       | %LNP1         | %%LNP1                                         |             |     |                 |
|       | %LOG          |                                                | C%LOG       |     | 1BA0C CK%LOG    |
|       | %ALOG         |                                                | C%ALOG      |     |                 |
|       | %SIN          | %%SIN                                          | C%SIN       |     |                 |
|       | %COS          | %%COS                                          | C%COS       |     |                 |
|       | %TAN          |                                                | C%TAN       |     |                 |
|       | %ASIN         |                                                | C%ASIN      |     | 1B6EA CK%ASIN   |
|       | %ACOS         |                                                | C%ACOS      |     | 1B775 CK%ACOS   |
|       | %ATAN         |                                                | C%ATAN      |     |                 |
|       | %SINH         | %%SINH                                         | C%SINH      |     |                 |
|       | %COSH         | %%COSH                                         | C%COSH      |     |                 |
|       | %TANH         |                                                | C%TANH      |     |                 |
|       | %ASINH        |                                                | C%ASINH     |     |                 |
|       | %ACOSH        |                                                | C%ACOSH     |     | 1B86C CK%ACOSH  |
|       | %ATANH        |                                                | C%ATANH     |     | 1B8DE CK%ATANH  |
| 2AC27 | %%SINRAD      | %%SINDEG                                       | %%ASINRAD   |     |                 |
|       | %%COSRAD      | %%COSDEG                                       | %%ACOSRAD   |     |                 |
|       | %%TANRAD      |                                                |             |     |                 |
|       | %MANTISSA     | %EXPONENT                                      | %FP         | %IP | %CEIL           |
|       | %FLOOR        | %%FLOOR                                        | %MOD        |     |                 |
| 2AFAC | %INT          | %%INT                                          |             |     |                 |
|       | %ANGLE        | %%ANGLE                                        | %>%%ANGLE   |     |                 |
|       | %%ANGLERAD    | %%ANGLEDEG                                     |             |     |                 |
|       | RNDXY         | ( %number %places --> %number' )               |             |     |                 |
|       | TRCXY         | ( %number %places --> %number' )               |             |     |                 |
|       | %COMB         | %PERM                                          |             |     |                 |
|       | %NFACT (Fact) | %FACT (!)                                      |             |     |                 |
|       | %NROOT        |                                                |             |     |                 |
|       | %MIN          | %MAX                                           | %%MAX       |     |                 |
|       | %MAXorder     | ( % % --> %max %min )                          |             |     |                 |
| 51AB7 | %MAXIMIZE     | ( % --> %0, %MAXR or -%MAXR)                   |             |     |                 |
|       | %RAN          | %RANDOMIZE                                     | DORANDOMIZE |     |                 |
|       | %OF           | %T                                             | %CH         |     |                 |
|       | %D>R          | ( % --> % )                                    |             |     |                 |
|       | %R>D          | ( % --> % )                                    |             |     |                 |
| 51A94 | %%SQR         | ( %%1 %%2 --> %% ) (stk1^2+stk2^2)             |             |     |                 |
| 520B2 | 2DUP%%R       | ( %%1 %%2 --> %%1 %%2 %% (sqrt(%%1^2+%%2^2)) ) |             |     |                 |
| 51A71 | 2%>%%SQR      | ( % % --> %% ) (stk1^2+stk2^2)                 |             |     |                 |
|       | %POL>%REC     | ( %x %y --> %radius %angle )                   |             |     |                 |
|       | %REC>%POL     | ( %radius %angle --> %x %y )                   |             |     |                 |
|       | %%R>P         | ( %%x %%y --> %%radius %%angle )               |             |     |                 |
|       | %%P>R         | ( %%radius %%angle --> %%x %%y )               |             |     |                 |
|       | %SPH>%REC     | ( %r %th %ph --> %x %y %z )                    |             |     |                 |
|       | %>%%-         | %>%%*                                          | %1+         | %1- | %>%%1/ %>%%SQRT |
|       | %10*          |                                                |             |     |                 |
|       | %+SWAP        |                                                |             |     |                 |
|       | %%*SWAP       | %%*ROT                                         | %%*UNROT    |     |                 |
|       | %%/>%         | SWAP%%/                                        |             |     |                 |

\*\*\*\*\*

## 8. Arrays

~~~~~

no ref check = Object is changed directly.
 Thus variables contents etc change too.
 Unless otherwise mentioned, ! words do no ref check
 {dims} = list of dimension sizes (#)

8.1 General Operations

^^^^^^

	ARSIZE	([] --> #elements)	
	OVERARSIZE	([] ob --> [] ob #elements)	
	DIMLIMITS	([] --> {dims}	
	MDIMS	([] --> #m FALSE / #m #n TRUE)	
	MDIMSDROP	([] --> #m / #m #n)	
193DA	CKARRYDIM	({ % (%) } --> { # (#) }	
		Errors if not valid location in array (0 indexes)	
	MAKEARRY	({dims} ob --> [ob])	
		Create unlinked array, all elements initialized to ob	
	XEQ>ARRAY	(F% ... F% {dims} --> [F%])	
1D02C	XEQ>VECTOR	(F% ... F% #n --> [F%]	
19294	>ARRY	(F% ... F% #n [F%] --> [F%]	
		Copies floats into array, starting in reverse order by putting stk3 to last location.	
1D0AB	DOARRY>	([] --> floats {dims})	
	SWAPROWS	([] #m #n --> []' #m #n)	
		Swap 2 rows. Doesn't NEWOB.	
37500	SWAPCOLUMNS	([] #m #n --> []' #m #n)	(GX: 37508)
		Swap 2 columns. Doesn't NEWOB.	
0358F	TYPEARRY@	([] --> #prolog)	
		Gets array element type as binary	
03685	ARRYEL?	({dims} [] --> #element TRUE)	
		({dims} [] --> FALSE)	
		Returns TRUE if array element exists	
0C4EC	GETEL	(#index [] --> ob TRUE / FALSE)	(GX: 0C501)
		(#index #addr_of_array --> ob TRUE / FALSE)	
		Works for most element types.	
	GETATELN	(# [] --> ob TRUE)	
		(# [] --> FALSE)	(no such element)
		Recall array element	
3558E	PULLEL	([F%] #n --> [F%] F%)	
	PULLREAL	([%] #n --> [%] %)	
	PULLCMPEL	([C%] #n --> [C%] C%)	
355D8	PULLLONGEL	([F%] #n --> [F%] L%)	(Converts element to long)
35602	PULLEREAL	([%] #n --> [%] %%)	(Converts real to long real)
	PUTEL	([F%] B% #n --> [F%]')	(Uses PACK!)
	PUTREAL	([%] % #n --> [%]')	
	PUTCMPEL	([C%] C% #n --> [C%]')	

8.2 Array Calculations

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

36115  MAT+
      ( [F%] [F%] --> [F%] )
36278  MAT-
      ( [F%] [F%] --> [F%] )
3643F  MAT*
      ( [F%]1 [F%]2 --> [F%]' )
36494  2DUPMAT*
      ( [F%]1 [F%]2 --> [F%]1 [F%]2 [F%]' )
36B60  MAT/
      ( [F%] [F%] --> [F%]' )

      MATCON ( [F%] F% --> [F%]' )
35D35  MATIDN ( [F%] --> [F%]' )
35F30  MATCONJ ( [F%] --> [F%]' )
35DEB  MATNEG ( [F%] --> [F%]' )
36B0B  MATINV ( [F%] --> [F%] )      ( GX: 36A99 )
      MATTRN ( [F%] --> [F%]' )
      MATREDIM ( [F%] { #dims } --> [F%]' )

35FA3  DUP%0CON      ( [F%] --> [F%] [%0] )
362DC  MATFLOAT*     ( [F%] or F%, [F%] or F% --> [F%]' )
3633C  MATFLOAT/     ( [F%] F% --> [F%] )      ( GX: 363DB )
36435  MATSQ         ( [F%] --> [F%]' )      ( GX: 36444 )
369CB  MATABS        ( [F%] --> % )          ( GX: 369E9 )
35F8F  MATRE         ( [F%] --> [F%]' )
35FEE  MATIM         ( [F%] --> [F%]' )
36782  MATCROSS      ( [F%] [F%] --> [F%] )   ( GX: 36791 )
366F6  MATDOT        ( [F%] [F%] --> [F%] )   ( GX: 36705 )
36A2A  MATDET        ( [%] --> % )           ( GX: 36A48 )
365AC  MATRSD        ( [%] [%] [%] --> [%] ) ( GX: 365BB )
368E5  MATRNRM       ( [%] --> % )           ( GX: 368F4 )
368F9  MATCNRM       ( [%] --> % )           ( GX: 3690D )

36039  MATR>C        ( [F%] [F%] --> [C%] )
360B6  MATC>R        ( [F%] --> [%re] [%im] )
35E2C  MATRND        ( [F%] % --> [F%]' )
35EA9  MATTRNC       ( [F%] % --> [F%]' )

35C2C  DOARRAYPRG1   Operate on F% elements ( seco [F%] --> [F%]' )
                        seco argument will be L%
35C63  DOARRAYPRG2   Operate on F% elements ( seco [orig] [new] --> [new]')

```

Examples:

```

      !MATNEG          = :: ' NEGF% SWAP DOARRAYPRG1 ;
      !MATCONJ         = :: ' C%%CONJ SWAP DOARRAYPRG1 ;

      MATIM            = :: DUP%0CON
                        OVER TYPERARRY? case SWAPDROP
                        ' :: DROP C%%>%% SWAPDROP ;
                        UNROT DOARRAYPRG2
                        ;

```

8.3 Real & Complex Calculations for Arrays

^^

Matrix calculations use special entries:

```

35B4F  SWITCHFLOATS      ( B% --> ? )      ( GX: 35B47 )
                                Operates on F% by dispatching on type.
                                Order is % C% %% C%%. All but last entry must be
                                addresses only. Example:
                                :: SWITCHFLOATS %CHS C%CHS %%CSH C%%CHS ;

35B90  SWITCH2FLOATS     ( B% B% --> ? ) ( GX: 35B88 )
                                Dispatch order for SX:
                                % %, C% %, % C%, C% C%, %% %, C%% %, %% C%, C%% C%
                                Dispatch order for GX: (L% only!)
                                %% %, C%% %, %% C%, C%% C%

35AF6  SWITCH2ARRY      ( [F%] [F%] --> ? ) ( GX: 35AEE )
                                Dispatch order: % %, C% %, % C%, C% C%

37DC4  F%>L%            ( % --> %, C% --> C% )      ( GX: 37D19 )
37DB5  B%ABS            ( B% --> B% )                ( GX: 37C75 )
37D7E  B%NEG            ( B% --> B% )                ( GX: 37CD3 )
37DA1  B%0=            ( B% --> flag )               ( GX: 37DF6 )

37BE9  DB%+            ( F% F% --> F% / L% L% --> L% ) (SX only)
37C7A  DB%-            ( F% F% --> F% / L% L% --> L% ) (SX only)

```

DB%+ and DB%- both use PACK!, thus no infinite result errors are generated. (Indicators are set though.)

```

SX:                GX:
                   37BE9  L%+      ( L% L% --> L% )
                   37C0C  L%-      ( L% L% --> L% )
37CED  L%*          37C2F  L%*      ( L% L% --> L% )
37D24  L%/          37C52  L%/      ( L% L% --> L% )

```

In GX all matrix calculations are done in L% for accuracy. This change has removed DB%+ and added L%+ etc. See SWITCH2FLOATS.

8.4 Statistics

^^^^^^^^^^^^^^^^

```

2C270  STATCLST                (CLSigma)
2C1F3  STATRCL
2C1F3  STATSTO
                STATSADD%      (Sigma+ with %)
                STATN
                STATSMIN
                STATSMAX
                STATMEAN
                STATSDEV
                STATTOT
                STATVAR
2C684  STATCOL                ( GX: 2C675 )
2C6C5  STATXCOL               ( GX: 2C6B6 )
2C6DE  STATYCOL               ( GX: 2C6CF )
2C701  STATGETXCOL            ( GX: 2C6F2 )
2C715  STATGETYCOL            ( GX: 2C706 )
2C84B  STATCORR               ( GX: 2C83C )

```


2C8F5	STATCOV	(GX: 2C8E6)
2C94F	STATX	(GX: 2C940)
2C963	STATY	(GX: 2C959)
2C977	STATXX	(GX: 2C972)
2C99A	STATYY	(GX: 2C99A)
2C9BD	STATXY	(GX: 2C9C2)
2CA30	STATLR	(GX: 2CA0D)
2CB02	STATPREDY	(GX: 2CADF)
2CB75	STATPREDX	(GX: 2CB52)
2CCDF	ColumnMAX	(#column [[]] --> %) (GX: 2CCBC)
2CCF8	ColumnMIN	(#column [[]] --> %) (GX: 2CCD5)
2CD04	ColumnTOT	(GX: 2CCE1)
2CD13	ColumnMEAN	(GX: 2CCF0)
2CD22	ColumnVAR	(GX: 2CCFF)
2CD31	ColumnSDEV	(GX: 2CD0B)

9. Tagged Objects

~~~~~

```
>TAG      ( ob $ --> tagged )
USER$>TAG  ( ob $ --> tagged )      (Up to 255 chars valid)
%>TAG     ( ob % --> tagged )
ID>TAG     ( ob id/lam --> tagged )
05E9F  {}>TAG  ( { id ob } --> tagged )
TAGOBS     ( ob $ --> tagged )
           ( ob1 ... obn { $1 ... $n } --> tagged1 ... taggedn )

05EC7  TAG>      ( tagged --> ob $tag )  ( GX: 05EC9 )
        STRIPTAGS ( tagged --> ob )
        STRIPTAGS12 ( tagged ob' --> ob ob' )
```

\*\*\*\*\*

## 10. Unit Objects

~~~~~

SI Units:

```
um*      * marker
um/      / marker
um^      ^ marker
umP      Character prefix operator
umEND     Unit end operator
```

Hex Quantity String:

Contains: ?^nn mol^nn cd^nn K^nn s^nn A^nn m^nn kg^nn

Example: 1 Newton = 00000000FE000101h = s^-2 * m^1 * kg^1

```
puretemp?      ( hxs --> flag )
                Purely temperature unit?
```

Reals Related to Temperature:

```
cfF      ( %%5/9 for C <--> F conversion )
cfC      ( %%1 for K <--> C conversion )
```

```
10E9C    %%KZERO      ( %%273.15 )
10EB6    %%RZERO      ( %%459.67 )
```

10.1 General Operations

^^^^^^^^^^^^^^^^^^^^

```
UFACT      ( unit1 unit2 --> unit' )
UM>U        ( % unit --> unit' )
            Replaces number part.
UMU>        ( unit --> % unit' )
            Returns number and normalized unit parts of unit.
UMCONV      ( unit1 unit1 --> unit1' )
            Converts unit1 to units of unit2.
UMSI        ( unit --> unit' )
            Unit object to SI units.
U>NCQ       ( unit --> n%% cf%% qhxs )
            Returns number, conversion factor to base units
            and hex quantity string
```

```

U>nbr          ( unit --> % )
                Takes number part of unit.
Unbr>U         ( unit % --> unit' )
                Puts new number part for unit.
UNIT>$         ( unit --> $ )
                Decompiles a unit object with tics.

```

10.2 Arithmetic Operations

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

Tests:                                     ( unit1 unit2 --> %flag )

```

```

UM=?      UM#?      UM<?      UM>?      UM<=?      UM>=?

```

```

Arithmetic Operations with 2 args:         ( unit1 unit2 --> unit )

```

```

UM+      UM-      UM*      UM/
UMMAX    UMMIN    UM%      UM%CH    UM%T
UMXROOT

```

```

Arithmetic Operations with 1 arg:          ( unit --> unit' )

```

```

0FD68  UMABS    UMCHS    UMINV    UMSQ    UMSQRT
        UMSIGN    UMIP    UMFP    UMFLOOR    UMCEIL
        UMRND    UMTRC
        UMSIN    UMCOS    UMTAN

```

```

0FD4A  UMOOPER:
        Evaluates next ob on units number part, then builds unit again.

```

11. Composite Objects

~~~~~

|       |              |                                                                                                                                                                                                                      |
|-------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | &COMP        | ( comp1 comp2 --> comp' ) concatenate composites                                                                                                                                                                     |
|       | >TCOMP       | ( comp ob --> comp' ) add ob to tail of comp                                                                                                                                                                         |
|       | >HCOMP       | ( comp ob --> comp' ) add ob to head of comp                                                                                                                                                                         |
| 1AC93 | SWAP>HCOMP   | ( ob comp --> comp' ) add ob to head of comp                                                                                                                                                                         |
|       | CARCOMP      | ( comp --> ob / comp )<br>Returns first object in the core of the composite.<br>Returns a null comp if the argument is a null comp.                                                                                  |
|       | ?CARCOMP     | ( comp flag --> comp / ob )<br>If TRUE then CARCOMP                                                                                                                                                                  |
|       | CDRCOMP      | ( comp --> comp' )<br>Returns composite minus it's first object.<br>Returns null composite for null composite.                                                                                                       |
|       | LENCOMP      | ( comp --> #n )                                                                                                                                                                                                      |
|       | DUPLENCOMP   | ( comp --> comp #n )                                                                                                                                                                                                 |
| 1CA3A | LENCOMP>%    | ( comp %n )                                                                                                                                                                                                          |
|       | SUBCOMP      | ( comp #start #end --> comp' )<br>Makes all possible index checks too.                                                                                                                                               |
|       | NTHELCOMP    | ( comp #i --> ob TRUE )<br>( comp #i --> FALSE )<br>Returns FALSE if #i is out of range                                                                                                                              |
|       | NTHCOMPDROP  | ( comp #i --> ob / nothing )<br>Does NTHELCOMP DROP                                                                                                                                                                  |
|       | NTHCOMDDUP   | ( comp #i --> ob ob )<br>Does NTHCOMPDROP DUP                                                                                                                                                                        |
|       | NEXTCOMPOB   | ( comp #offset --> comp #next_offset ob TRUE )<br>( comp #offset --> comp FALSE )<br>Returns object at nibble offset from the start.<br>Returns FALSE if object is SEMI.<br>Use #5 as a start value (to skip prolog) |
|       | NULLCOMP?    | ( comp --> flag ) Returns TRUE if composite is empty                                                                                                                                                                 |
|       | DUPNULLCOMP? | ( comp --> comp flag )                                                                                                                                                                                               |
|       | matchob?     | ( ob comp --> ob TRUE / FALSE )<br>TRUE if ob is EQUAL to any element.                                                                                                                                               |
|       | POSCOMP      | ( comp ob pred --> #i / #0 )<br>If pred(any element,ob)=TRUE, return element location                                                                                                                                |
|       | EQUALPOSCOMP | ( comp ob -- #i / #0 )<br>POSCOMP with pred = EQUAL.<br>Returns position of ob in list (or #0).                                                                                                                      |
|       | NTHOF        | ( ob comp --> #i / #0 )<br>Same as SWAP EQUALPOSCOMP                                                                                                                                                                 |
|       | #=POSCOMP    | ( comp # --> #i / #0 )<br>POSCOMP with pred = #=                                                                                                                                                                     |
|       | Find1stTrue  | ( comp pred --> ob TRUE / FALSE )<br>If pred(any element)=TRUE, return element location                                                                                                                              |
|       | Lookup       | ( ob pred comp --> nextob TRUE / FALSE )<br>Does matching in groups of 2. If first matches<br>second is returned with TRUE.                                                                                          |
|       | EQLookup     | ( ob comp --> nextob TRUE / FALSE )<br>Lookup with pred = EQ                                                                                                                                                         |
|       | Embedded?    | ( ob1 ob2 --> flag )<br>Returns TRUE if ob2 is embedded in,<br>or is the same as, ob1. Else returns FALSE.                                                                                                           |

## Building/Exploding:

```

05331  >COMP          ( Meta #prolog --> comp )
        ::N          ( Meta --> seco )
        {}N          ( Meta --> list )
        SYMBN        ( Meta --> symb )
05481  UNITN         ( Meta --> unit )
5E0A3  XEQSYMBN      ( Meta --> symb )          With error protection
5E0DA  XEQ{}N        ( Meta --> {} )           With error protection
5E111  XEQ::N        ( Meta --> seco )          With error protection

```

The XEQ words first try the low level version, if it errors (insufficient memory) the composite is built 1 element at a time. This allows GC to happen while the composite is being built so that you can build larger composites. Use the XEQ words if you know the composite may be very large.

```

INNERCOMP      ( comp --> Meta )
DUPINCOMP      ( comp --> comp Meta )
SWAPINCOMP     ( comp ob --> ob Meta )
INCOMPDROP     ( comp --> obn ... obl )
INNERDUP       ( comp --> Meta #n )
1C973  INNERCOMP>% ( comp --> obn ... obl %n )
        INNER#1=    ( comp --> obn ... obl flag )
5E585  INNERtop&   ( Meta1 comp2 --> Meta )

```

## Lists:

```

NULL{}         ( --> {} )          Returns null list
DUPNULL{}?     ( {} --> {} flag )
ONE{}N         ( ob --> {} )
TWO{}N         ( obl ob2 --> {} )
THREE{}N       ( obl ob2 ob3 --> {} )
#1-{}N         ( obn ... obl #n+1 --> {} )
PUTLIST        ( ob #i {} --> {}' )
                Replaces object at nth position.
                Assumes valid #i.
0E461  INSERTN{} ( {} ob #n --> {}' )
                Insert ob to nth position
0E4DE  REMOVE{} ( {} #n --> {}' )
                Remove nth ob
49CD6  ROLL{}    ( {} --> {}' )
                Roll lists elements.
                (If NULL{} does "No Current Equation" error)
apndvarlst     ( {} ob --> {}' )
                Append ob to {} if not already there

```

## Secondaries:

```

NULL::         ( --> NULL:: )      Returns null secondary
Ob>Seco        ( ob --> seco )     Does ONE ::N
?Ob>Seco       ( ob --> seco )     If not seco then Ob>Seco
2Ob>Seco       ( obl ob2 --> seco ) Does TWO ::N

::NEVAL        ( obn .. obl #n --> ? ) Does ::N EVAL

```

\*\*\*\*\*

## 12. Meta Objects

~~~~~

Meta object is a 'stack object' consisting of n objects and their count. INNERCOMP produces a meta object from a composite.
ZERO is a null-Meta.

12.1 Meta Operations

^^^^^^

If the entry name is supported the address is marked with *
If I use another name than the supported one the address is listed
and the supported name is given with "aka name"

'Stack Operations':

dup	5E35C	(M --> M M)	
swap	5EB1C	(M1 M2 --> M2 M1)	aka psh
rot	5EB58	(M1 M2 M3 --> M2 M3 M1)	
unrot	5EBDB	(M1 M2 M3 --> M3 M1 M2)	
4roll	5EBC6	(M1 M2 M3 M4 --> M2 M3 M4 M1)	
4unroll	5EBEA	(M1 M2 M3 M4 --> M4 M1 M2 M3)	
5roll	5ED45				
5unroll	5ED5A				
N+1roll	5EBFC	(M1...Mn+1 #n --> M2...Mn+1 M1)	
N+1unroll	5ED6C	(M1...Mn+1 #n --> Mn+1 M1...Mn)	
swapdrop	63911	(M1 M2 --> M2)	aka SWAPUnDROP
rotswap	5E857	(M1 M2 M3 --> M2 M1 M3)	
rotdup	63F1A	(M1 M2 M3 --> M2 M3 M1 M1)	aka metaROTDUP
4rollunrot	5E870	(M1 M2 M3 M4 --> M2 M1 M3 M4)	

Meta combining:

top&	*5E415	(M1 M2 --> M1&M2)	
pshtop&	*5E4D1	(M1 M2 --> M2&M1)	
rotop&	63F2E	(M1 M2 M3 --> M2 M3&M1)	aka ROTUntop&
unrotop&	63F42	(M1 M2 M3 --> M3 M1&M2)	aka roll2top&
over&	5E3C0	(M1 M2 M3 --> M1&M2 M3)	
psh&	*5E3AC	(M1 M2 M3 --> M1&M3 M2)	
overrev&	5E843	(M1 M2 M3 --> M2&M1 M3)	
2top&	5E490	(M1 M2 M3 --> M1&M2&M3)	
top&pshtop&	5B861	(M1 M2 M3 --> M2&M3&M1)	

12.2 Meta & Object Operations

^^^^^^

pick1	5FC24	(Ob M --> Ob M Ob)	
get1	*61305	(Ob M --> M Ob)	
psh1	63105	(M Ob --> Ob M)	aka OVER#2+UNROL
pshm1	5E3E8	(M Ob --> Ob #1 M)	
addt	62904	(M Ob --> M&Ob)	aka SWAP#1+
psh1top&	*5E401	(M Ob --> Ob&M)	
pull	*5E4A9	(M&Ob --> M Ob)	
pulldrop	5EAF4	(M&Ob --> M)	
pullpshm1	5E6BB	(M&Ob --> Ob #1 M)	
pullrev	*5E4BD	(Ob&M --> M Ob)	
pickrev	6119E	(Ob&M --> Ob&M Ob)	aka DUP#1+PICK
pulldroppull	5FA45	(M&Ob1&Ob2 --> M Ob1)	
2pulldrop	5CC12	(M&Ob1&Ob2 --> M)	

repl	60F0E	(M&Ob1 Ob2 --> M&Ob2)	aka ROTDROPSWAP
revpulldrop	5FA63	(M&Ob1 Ob2 --> M Ob2)	
revpull&psh	548AA	(M&Ob1 Ob2 --> Ob1&Ob2 M)	
psh1&	*5E706	(M1 M2 Ob --> Ob&M1 M2)	
psh1&rev	*5E7A5	(M1 M2 Ob --> M1&Ob M2)	
addtpsh	57441	(M1 M2 Ob --> M2&Ob M1)	(GX: 57432)
unrot1	10AF9	(M1 M2 Ob --> Ob M1 M2)	
rot1	10ADB	(Ob M1 M2 --> M1 M2 Ob)	
pullpsh1&	*5E4EA	(M1 M2&Ob --> Ob&M1 M2)	
pullrev1&	5E503	(M1 M2&Ob --> M1&Ob M2)	
pshpullpsh1&	5D6F4	(M1&Ob M2 --> Ob&M2 M1)	
pshzer	*5E67A	(M --> #0 M)	
unzer	638FD	(#0 M --> M)	aka SWAPUnDROP
4psh	25322	(M1 Ob1 Ob2 Ob3 Ob4 --> M2 M1)	

Arguments from current stream:

addt:	5E51C	(M --> M&ob)
addt2:	5E530	(M --> M&ob1&ob2)
repl:	5E59E	(M&Ob --> M&ob)
psh1&rev:	5E549	(M1 M2 --> M1&ob M2)
psh1&rev2:	5E562	(M1 M2 --> M1&ob1&ob2 M2)
2psh1&rev:	5DD65	(M1 M2 --> M1&ob M2&ob)

Arguments from 1LAM or 3LAM:

addfunc	55288	(M --> M&LAM1)	aka 1GETLAMSWP1+
replfunc	55477	(M&Ob --> M&LAM1)	
addoper	560ED	(M1 M2 --> M1&M2&LAM1)	aka xssgeneral
addcon&oper	56101	(M --> LAM3&M&LAM1)	aka xnsgeneral
add&conoper	5611F	(M --> M&LAM3&LAM1)	aka xsngeneral
dropaddoper	562BE	(M1 M2 M3 --> M1&M2&LAM1)	
MetaUnCalc	56309	(M Ob --> LAM3 #1)	

12.3 Other Operations

^^^^^^^^^^^^^^^^^^^^

SubMetaOb	*64345	(Meta #start #end --> Meta')	
		Does range checks	
SubMetaOb1	*643BD	(Meta ob #start #n #start --> meta ob #n)	
		Drops obs from meta starting from #start+1	
metatail	*28296	(Meta #n --> #n-1-size(Meta) Meta')	
MEQU?	584B2	(M1 M2 --> M1 M2 TRUE/FALSE)	
		Same count, equal obs --> TRUE	
ObInMeta?	5768A	(M Ob --> M Ob TRUE/FALSE)	(GX: 57676)
		True if Ob is equal to some ob in Meta	
NoIdsInMeta?	58715	(M --> M TRUE/FALSE)	
		FALSE, if Meta has IDs/LAMs or secos	
		starting with CK0	
psh1&revUNTIL	5F370	(M1 M2 Ob --> M1&Ob M2)	
		If Meta2 is empty (#0) end loop	
pick1#0=case	5FB6E	(M1 M2 --> M1 M2)	
		If Meta1 is empty then COLA else SKIP	

DROPdrop	63FA6	(M	Ob	-->)	aka DROPNDROP
dropf	169A5	(M	-->	FALSE)	aka NDROPFALSE
dropDROPf	50F60	(Ob	M	-->	FALSE)
DROP2dropf	57428	(M1	M2	Ob	-->	FALSE) (GX: 57419)
2DROP2dropf	57414	(M1	M2	Ob1	Ob2	--> FALSE) (GX: 57405)
Repl0	5551C	(M	-->	%0	#1)
Repl1	55535	(M	-->	%1	#1)
Repl-1	5554E	(M	-->	%-1	#1)
2Repl0	56183	(M1	M2	-->	%0	#1)
2Repl-1	561D8	(M1	M2	-->	%-1	#1)
DropRepl0	5643A	(M	Ob	-->	%0	#1)
Repl0ABND	5499F	(M	-->	%0)

13. Symbolics

~~~~~

```

055F3 NULLSYMB      Null algebraic (do not eval)
5E067 SINNER        ( symb --> Meta / ob --> ob #1 )
                    If symbolic the INNERCOMP, else ONE.
5E30C 2SINNER       ( ob1 ob2 --> Meta1 Meta2 )
                    SINNER for 2 objects.
5E2F8 2SINNERtop&   ( ob1 ob2 --> Meta )
                    2SINNER, then top&
5E32A SINNERMETA    ( Meta --> Meta' )
                    Explodes each object in Meta with SINNER and merges the
                    result with top& commands.
5F2A3 EXPLODE       ( ob --> Meta )
                    Explodes ob with recursive SINNER calls.
5F2EE IMplode       ( Meta --> ob )
                    Builds symbolic obeying VUNS properties (UNSYM
                    element), checking fcnapply etc. Does not build
                    symbolic if result is a single object valid in
                    symbolics.
                    symcomp      ( ob --> ob' )
                    If ob is symbolic nothing, else ONESYMBN
5E661 ONESYMBN      ( ob --> symb )
5E085 CKSYMBN       ( Meta --> ob )
                    If size <> 1 then SYMBN, else DROPSYM
5F384 DROPSYM       ( ob1 ob2 --> ob )
                    Drop ob2, if ob1 is symf the nothing, else ONESYMBN.

1CF2E EQ>           ( symb --> arg1 arg2 )           Internal EQ->
1CFD0 EXPR>         ( symb --> arg1 ... argn %n ob ) Internal OBJ->

                    CRUNCH      ( ob --> % )
                    Internal ->NUM
22F68 SYMCRUNCH1    ( ob --> % )
                    Internal ->NUM, if id XEQRCCL is done before CRUCNH
22F86 SYMCRUNCH2    ( ob1 ob2 --> % ob2 )
                    Internal ->NUM for stack level 2 (Possible XEQRCCL)

uncrunch ( ob --> ? )
Clears num flag for next command only. (Flag 3)
Example: SYMCOLCT = :: uncrunch colct ;

cknumdsptchl ( sym --> symf )
Used by 1 argument functions to evaluate a symbolic or numeric
evaluation routine according to num flag.
Usage: :: cknnumdsptchl <Sym> <Num> ;
If numeric mode stkl is CRUNCHed and <Num> is COLAd.
In symbolic mode ckseval1: is called.
Example: SYMRE = :: cknnumdsptchl MetaRE xRE ;
54DBC ckseval1: ( symf --> symf' )
Binds next 2 in stream to LAMxSYMfcn and LAMxfcn. Explodes
symf, then evaluates next on Meta, then builds ob with CKSYMBN.
If symf is equation next is evaluated on both sides, then
equation is rebuilt. (ckevaleq1)
54E2A ckevaleq1 ( Meta&= --> symb )
Evaluates 2LAM on both sides of equation, the rebuilds symbolic
and abandons env.

```

```

sscknum2 ( sym sym --> symf )
Used by 2 argument functions to evaluate function according to
current (symbolic/numeric) mode.
Usage: :: sscknum2 <Sym> <Num> ;
In numeric mode stk1 and stk2 are CRUNCHED and <Num> is COLAd.
Else cksseval2: is called.
Example: SYM+ = :: sscknum2 Meta+ x+ ;

sncknum2 ( sym % --> symf )
Usage: :: sncknum2 <Sym> <Num> ;
In symbolic mode uses cksneval2:
Example: SYM+O = :: sncknum2 Meta+Con x+ ;

nscknum2 ( % sym --> symf )
Usage: :: nscknum2 <Sym> <Num> ;
In symbolic mode uses cknseval2:
Example: O+SYM = :: nscknum2 Con+Meta x+ ;

55657 ccknum2: ( symf symf --> symf )
Used by above 3 to determine (and possibly to CRUNCH) the
program to COLA.

557EC cksseval2: ( sym sym --> symf )
Binds next 2 to LAMxSYMfcn and LAMxfcn. Explodes stk1 and stk2,
evaluates next. If either is equation ckevaleq2 is called.
Rebuilds 1 symbolic.

5575B cknseval2: ( % sym --> symf )
SWAP, then cksneval2:

5576F cksneval2: ( sym % --> symf )
Binds % and next 2 to LAMscl, LAMxSYMfcn and LAMxfcn.
Explodes sym, evaluates LAMxSYMfcn, rebuilds symbolic.
If sym is equation ckevaleq1 is called.

58CE4 parameval ( sym param --> ?? )
Ensures sym is symbolic (with symcomp), then executes param on
each element in symbolic. param is bound to lLAM during the
loop. param should return TRUE or FALSE. If param returns TRUE
or stk1 is not operator loop continues, else possible COLCT
property is executed. (Better return TRUE always.)

58CEE eval ( sym --> ? )
parameval without binding of a new param. Use this for
recursive evaluation with the same parameter.
(See SHOWLS and showparam for examples)

5918A evalTRUE ( sym --> ? TRUE )
Used for recursive parameval.

```

### 13.1 Symbolic Operations

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

One argument Meta functions assume function in LAM1.  
Two argument Meta functions for constants also assume Con in LAM3.

|       |         |       |           |
|-------|---------|-------|-----------|
| 54AE0 | SYMRE   | 5542C | MetaRE    |
| 54EB9 | SYMIM   | 55495 | MetaIM    |
| 54ED2 | SYMNOT  |       | addfunc   |
| 54EEB | SYMNEG  | 553D2 | MetaNEG   |
| 54F04 | SYMABS  | 555B2 | MetaABS   |
| 54F1D | SYMCONJ | 55567 | MetaCONJ  |
| 54F36 | SYMINV  | 553EB | MetaINV   |
| 54F4F | SYMARG  |       | addfunc   |
| 54F68 | SYMSIGN |       | ?addrever |
| 54F81 | SYMSQRT |       | addfunc   |
| 54F9A | SYMSQ   | 555E9 | MetaSQ    |

|       |             |       |          |                     |
|-------|-------------|-------|----------|---------------------|
| 54FB3 | SYMSIN      | 5533C | MetaSIN  |                     |
| 54FCC | SYMCOS      | 55378 | MetaCOS  |                     |
| 54FE5 | SYMTAN      | 553A5 | MetaTAN  |                     |
| 54FFE | SYMSINH     | 5529C | MetaSINH | (?addinver: xASINH) |
| 55017 | SYMCOSH     | 552B0 | MetaCOSH | (?addinver: xACOSH) |
| 55030 | SYMTANH     | 552C4 | MetaTANH | (?addinver: xATANH) |
| 55049 | SYMASIN     |       | addfunc  |                     |
| 55062 | SYMACOS     |       | addfunc  |                     |
| 5507B | SYMATAN     |       | addfunc  |                     |
| 55094 | SYMASINH    |       | addfunc  |                     |
| 550AD | SYMACOSH    |       | addfunc  |                     |
| 550C6 | SYMATANH    |       | addfunc  |                     |
| 550DF | SYMEXP      | 552D8 | MetaEXP  | (?addinver: xLN)    |
| 5515C | SYMEXPM     | 55300 | MetaEXPM | (?addinver: xLNPl)  |
| 5512A | SYMALOG     | 552EC | MetaALOG | (?addinver: xLOG)   |
| 550F8 | SYMLN       |       | addfunc  |                     |
| 55111 | SYMLOG      |       | addfunc  |                     |
| 55143 | SYMLNP1     |       | addfunc  |                     |
| 55175 | SYMFACT     |       | addfunc  |                     |
| 5518E | SYMIP       |       | addfunc  |                     |
| 551A7 | SYMFP       |       | addfunc  |                     |
| 551C0 | SYMFLOOR    |       | addfunc  |                     |
| 551D9 | SYMCEIL     |       | addfunc  |                     |
| 551F2 | SYMEXPONENT |       | addfunc  |                     |
| 5520B | SYMMANT     |       | addfunc  |                     |
| 55224 | SYMD>R      |       | addfunc  |                     |
| 5523D | SYMR>D      |       | addfunc  |                     |
| 55256 | SYMUBASE    |       | addfunc  |                     |
| 5526F | SYMUVAL     |       | addfunc  |                     |

|       | addconoper |       | addcon&oper |       | ?addsimir |
|-------|------------|-------|-------------|-------|-----------|
| 55E4A | SYM%MAX    | 55E63 | %SYMMAX     | 55E7C | SYMMAX    |
| 55E95 | SYM%MIN    | 55EAE | %SYMMIN     | 55EC7 | SYMMIN    |
|       | addconoper |       | addcon&oper |       | addoper   |
| 5599A | SYM%AND    | 559B3 | %SYMAND     | 559CC | SYMAND    |
| 559E5 | SYM%OR     | 559FE | %SYMOR      | 55A17 | SYMOR     |
| 55A30 | SYM%XOR    | 55A49 | %SYM XOR    | 55A62 | SYM XOR   |
| 55A7B | SYMFLOAT== | 55A94 | FLOATSYM==  | 55AAD | SYM==     |
| 55AC6 | SYMFLOAT<> | 55ADF | FLOATSYM<>  | 55AF8 | SYM<>     |
| 55B11 | SYM%<      | 55B2A | %SYM<       | 55B43 | SYM<      |
| 55B5C | SYM%>      | 55B75 | %SYM>       | 55B8E | SYM>      |
| 55BA7 | SYM%<=     | 55BC0 | %SYM<=      | 55BD9 | SYM<=     |
| 55BF2 | SYM%>=     | 55C0B | %SYM>=      | 55C24 | SYM>=     |
| 55C3D | SYM%OF     | 55C56 | %SYM OF     | 55C6F | SYM OF    |
| 55C88 | SYM%CH     | 55CA1 | %SYM%CH     | 55CBA | SYM%CH    |
| 55CD3 | SYM%T      | 55CEC | %SYM%T      | 55D05 | SYM%T     |
| 55D1E | SYM%COMB   | 55D37 | %SYM COMB   | 55D50 | SYM COMB  |
| 55D69 | SYM%PERM   | 55D82 | %SYM PERM   | 55D9B | SYM PERM  |
| 55DB4 | SYM%RND    | 55DCD | RNDSYM      | 55DE6 | SYM RND   |
| 55DFF | SYM%TRNC   | 55E18 | TRNCSYM     | 55E31 | SYM TRNC  |
| 560A2 | SYM%XROOT  | 560BB | %SYM XROOT  | 560D4 | SYM XROOT |

|       |       |       |          |
|-------|-------|-------|----------|
| 55EE0 | SYM^O | 5645D | Meta^Con |
| 55EF9 | O^SYM | 562FA | Con^Meta |
| 55F12 | SYM^  |       | addoper  |
| 55F2B | SYM+O | 56543 | Meta+Con |

|       |         |       |            |
|-------|---------|-------|------------|
| 55F44 | O+SYM   | 56331 | Con+M      |
| 55F5D | SYM+    | 56160 | Meta+      |
| 55F76 | SYM-O   | 56566 | Meta-Con   |
| 55F8F | O-SYM   | 56359 | Con-Meta   |
| 55FA8 | SYM-    | 56174 | Meta-      |
| 55FC1 | SYM*O   | 56589 | Meta*Con   |
| 55FDA | O*SYM   | 56390 | Con*Meta   |
| 55FF3 | SYM*    | 561BA | Meta*      |
| 5600C | SYM/O   | 565CF | Meta/Con   |
| 56025 | O/SYM   | 563DB | Con/Meta   |
| 5603E | SYM/    | 56214 | Meta/      |
| 56057 | SYM%MOD | 5660B | MetamodCon |
| 56070 | %SYMMOD | 5642B | ConmodMeta |
| 56089 | SYMMOD  | 56250 | Metamod    |

COLCT  
\*\*\*\*\*

```
SYMCOLCT ( symf --> symf )
57DA4 colct ( symf --> symf )      Num flag is not checked!
```

Disassembly of colct:

```
:: EXPLODE
  pshzer colfac
  pshzer colrev
  ATTNFLG@ #0<> case :: CKSYMBN CK0NOLASTWD ?ATTNQUIT ;
  pshzer colunfac
  SYMBN COLA coleval
;
```

```
587AA colfac ( Metal Meta2 --> Meta' )
Appends objects in Meta2 tail to Metal tail replacing all
-,/,NEG,INV and SQ with +,*,^ and -1 as possible factor.
Example rules:
  'SQ(A)'      --> 'A^2'
  '-A'         --> '-1*A'
  'A-B'        --> 'A+-1*B'
  'A/B'        --> 'A*B^-1'
```

```
57E08 colrev ( Metal Meta2 --> Meta' )
Appends objects in Meta2 to tail of Metal collecting numeric
factors, ordering terms according to a comparison function,
collecting numeric terms to front. Only + and * factors are
checked.
58511 MetaLess? (Metal Meta2 --> M1 M2 flag )
58525 MetaMore? (Metal Meta2 --> M1 M2 flag )
585A7 BodyMore? (ob1 ob2 --> flag )
```

BodyMore? can be used for various sort programs, for example  
for strings (\$1 \$2 --> flag)

```
  if len($1)>len($2) then TRUE
  else if len($1)<len($2) then FALSE
  else if $1>$2 (alphabetically) then TRUE
```

EQUAL strings would have to be checked separately, otherwise  
the nibbles after the strings would determine the result.

```
58A20 colunfac ( Metal Meta2 --> Meta' )
Appends obs in Meta2 to head of Metal converting ^,+,* to
/,- when suitable.
```

```

58CDA coleval    ( ob --> ob' )
                  Passes FALSE as parameter to parameval. Thus eval uses
                  ?COLCT to check special evaluation.

EXPAN
^^^^^

57A0C SYMEXPAN  ( symf --> symf )
                  Expands symbolic or float.
57A48 expan     ( Meta1 Meta2 Meta3 --> Meta )
                  Expands Meta3. Successful part is added to the head of
                  Meta1, unsuccessful part is added to tail of Meta2.
                  Calls expan1 and larg until Meta3 becomes empty.
57AA2 expan1    ( Meta --> Meta1 Meta2 )
                  Expands Meta. Meta1 is the unsuccessful part, Meta2 the
                  succesful part (Could be just an operator).
                  Subexpanders:
57B63    ?expan^
                  If ^ then expands. (Returns if succesful)
57AB6    expansq
                  Expands SQ.
5BFD8    MetaD->
5C0B9    Meta<-D
5C2CE    MetaE^
5C348    MetaL*
57B4C    ?expanneginv
                  Prevents Meta->() from expanding [Expr INV NEG]
5C137    Meta->()
57B01    ?expanapp
                  If xFCNAPPLY then tries calling ?EXPAN

57C71    expansum^2
                  Expands (A+B)^2 or (A-B)^2. (Meta&<+/-> --> Meta')
57CF8    NXTPOT%?
                  Returns next number when expanding ^.
                  ( % --> flag %' TRUE / % FALSE )
                  flag indicates whether %>0. Do not use for %0.
                  Example: (-5 --> FALSE -4 TRUE )

INTEGRATE
^^^^^^^^^^

1F201    XEQINTEGID  ( ob ob ob id/lam --> symf )
1F27A    XEQINTEG    ( ob ob ob QN --> symf )
5AAC7    SYMINTEG    ( symf symf symf QN --> symf )
5662E    NUMINTEG    ( symf QN symf_lo symf_hi --> % )
52C36    CALCINTEG    ( seco %accuracy %low %hi --> %integral %error )
                  Low level numeric integration. If low=hi returns %0 %0.
                  Checks that 1E-12 <= %accuracy <= 1
                  seco gets % as input and should return one value.

5ACC7    intg        (#0 #0 Meta --> Meta_ok Meta_fail )
                  Integrates Meta where variable of integration has been
                  changed to LAMdvar. Meta objects should be merged by
                  addition. Use colunfac to resume /,- etc from *,+.
```



|       |               |                                                                                               |                     |
|-------|---------------|-----------------------------------------------------------------------------------------------|---------------------|
| 591AD | SYMQUAD       | ( sym id --> symf )                                                                           |                     |
|       |               | Avoids the obvious in in solving a quadratic equation.                                        |                     |
| 595DD | SYMTAYLR      | ( sym id % --> symf )                                                                         |                     |
|       |               | Calculates taylor polynomial. Use % > 20 with a slightly complex function to impress friends. |                     |
| 572A2 | SYMISOL       | ( sym id --> symb )                                                                           | ( GX: 57293 )       |
| 1F113 | XEQSYMDERCON  | ( QN %/C%/Unit --> symf )                                                                     |                     |
| 1F0F5 | XEQSYMDERSTEP | ( QN sym --> symf )                                                                           |                     |
| 54977 | SYMDERSTEP    | ( QN sym --> symf )                                                                           | No CKSYMBTYPE check |
| 54954 | SYMDER        | ( sym sym --> symf )                                                                          |                     |
| 56949 | SYMSUM        | ( sym sym sym ob --> symf )                                                                   |                     |
| 56A06 | SYM%SUM       | ( sym sym % ob --> symf )                                                                     |                     |
| 56A4C | %SYMSUM       | ( sym % sym ob --> symf )                                                                     |                     |
| 56AC9 | %%SUM         | ( sym % % ob --> symf )                                                                       |                     |

## 13.2 Meta Symbolics

^^^^^^^^^^^^^^^^^^^^

Case words testing the 1st object:

|              |        |                                            |
|--------------|--------|--------------------------------------------|
| Mlst+case    | 5EF2E  | (Meta&+ ? case)                            |
| Mlst-case    | 5EF41  | (Meta&- ? case)                            |
| Mlst*case    | 5EF54  | (Meta&* ? case)                            |
| Mlst/case    | 5EF67  | (Meta&/ ? case)                            |
| MlstNEGcase  | 5EF7A  | (Meta&NEG ? case)                          |
| MlstINVcase  | 5EF8D  | (Meta&INV ? case)                          |
| Mlst^case    | 5EFA0  | (Meta&^ ? case)                            |
| MlstSQcase   | 5EFB3  | (Meta&SQ ? case)                           |
| MlstFcncase  | 5EFC6  | (Meta&FCNAPPLY ? case )                    |
| M-1potcase   | 58ADE  | (Meta&-1&^ ? case )                        |
| M-1stcasechs | *5EE10 | (Meta&NEG -> Meta COLA, Meta -> Meta SKIP) |
|              |        | (Meta&(%<0) -> Meta&ABS(%) COLA)           |
| MEQ1stcase   | *5EFD9 | (Meta&Ob1 Ob2 -> Ob1=Ob2 case) (Ob2 lost)  |
| AEQ1stcase   | *5EF15 | (Meta&Ob -> Ob=Nob case) (Next ob lost)    |
| MEQopscase   | *5EFF9 | (Meta1&Ob1 Meta2&Ob2 Ob3 --> ... )         |
| AEQopscase   | *5F048 | (Meta1&Ob1 Meta2&Ob2 --> ... )             |
| Mid1stcase   | *5F061 | (Meta&Ob -> TYPE(ob) =ID/LAM case)         |

Adding some common operators:

|         |       |                     |
|---------|-------|---------------------|
| addt-   | 5BC67 | (Meta -> Meta&-)    |
| addt+   | 5BC94 | (Meta -> Meta&+)    |
| addt*   | 5CD16 | (Meta -> Meta&*)    |
| addtNEG | 5CD2A | (Meta -> Meta&NEG)  |
| addtINV | 5CD3E | (Meta -> Meta&INV)  |
| repl/   | 5BCC1 | (Meta&Ob -> Meta&/) |
| repl*   | 5BCEE | (Meta&Ob -> Meta&*) |

## Changing operators:

|            |       |                                             |
|------------|-------|---------------------------------------------|
| Mlst+?Drp  | 5ACD6 | (Meta&+ -> Meta, Meta -> Meta)              |
| meta+      | 5BC5D | (Meta&NEG -> Meta&-) (Meta -> Meta&+)       |
| meta-      | 5BC8A | (Meta&NEG -> Meta&+) (Meta -> Meta&-)       |
| meta*      | 5BCB7 | (Meta&INV -> Meta&/) (Meta -> Meta&*)       |
| meta/      | 5BCE4 | (Meta&INV -> Meta&*) (Meta -> Meta&/)       |
| drpmeta+   | 5BD3E | (Meta&NEG&Ob -> Meta&-) (Meta&Ob -> Meta&+) |
| drpmeta-   | 5BD57 | (Meta&NEG&Ob -> Meta&+) (Meta&Ob -> Meta&-) |
| drpmeta*   | 5BD70 | (Meta&INV&Ob -> Meta&/) (Meta&Ob -> Meta&*) |
| drpmeta/   | 5BD89 | (Meta&NEG&Ob -> Meta&*) (Meta&Ob -> Meta&/) |
| metaneg    | 5BBE5 | (Meta&NEG --> Meta) (Meta --> Meta&NEG)     |
| metainv    | 5BC3F | (Meta&INV --> Meta) (Meta --> Meta&INV)     |
| metaneglft | 5BC03 | (Meta --> Meta') metaneg on left subexpr    |
| metainvlft | 5BC21 | (Meta --> Meta') metaneg on left subexpr    |

## Splitting algebraic metas:

|               |       |                                                                                                                                                                                                                                                                                      |
|---------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| parg          | 5EA9F | (Meta --> MetaLast MetaRest) aka pshzerpsharg<br>Pushes last subexpression in Meta. If Meta is a valid expression MetaRest will be empty.                                                                                                                                            |
| pargswap      | 63F92 | (Meta --> MetaRest MetaLast) aka pZpargSWAPUn<br>parg + swap                                                                                                                                                                                                                         |
| drpparg       | 63F56 | (Meta&ob --> MetaLast MetaRest ) aka plDRPpZparg<br>Drops ob, then parg                                                                                                                                                                                                              |
| pargop        | 5E68E | (Meta --> MetaLast&op MetaRest)<br>Pushes last subexpression ignoring first object in Meta. Thus is op is +,- etc MetaLast would be their second argument.                                                                                                                           |
| larg          | 5EAC2 | (Meta --> MetaRest MetaLast)<br>Splits last subexpression from Meta.                                                                                                                                                                                                                 |
| parg&         | 5E6F2 | (Meta1 Meta2 --> Meta1&MetaLast MetaRest)                                                                                                                                                                                                                                            |
| larg&         | 5CCEE | (Meta1 Meta2 --> Meta1&MetaRest MetaLast)                                                                                                                                                                                                                                            |
| drppargtop&   | 5CBF9 | (Meta&ob --> MetaLast&MetaRest)                                                                                                                                                                                                                                                      |
| swappargunrot | 57F4B | (Meta1 Meta2 --> MetaRest Meta2 MetaLast )                                                                                                                                                                                                                                           |
| drppargsym    | 1CF42 | (Meta&ob --> 'MetaRest' 'MetaLast')<br>Builds objects with CKSYMBN. Will give invalid expressions if ob is not a 2-argument function.                                                                                                                                                |
| tailpsh       | 5F996 | (Meta #n --> Meta1 Meta2)<br>Pushes n-1 last objects in Meta to Meta1.                                                                                                                                                                                                               |
| splitup       | 5F926 | (Meta #n #m --> Meta #level)<br>Calculates stack level of last object to be included when splitting last m subexpressions from Meta starting from stack level n. (2 1 would give level of first ob in the last subexpression)                                                        |
| splitdown     | 5F96E | (Meta #n #m --> Meta #lowlevel #args+1)<br>Seeks stack level n-1 downwards for extra operators for m expressions. Lowlevel is the stack level of the extra operator. args indicates how many expressions the lowlevel operator is still missing. Probably an inaccurate description. |



# Miscellaneous:

|             |       |                                                                                                                                                                                                       |
|-------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| argnum      | 5E889 | (Meta ob --> Meta ob #args )<br>Returns argument count of ob. Checks for xFCNAPPLY and infarg? before calling !argnum, so Meta must be present to be able to determine argument count in those cases. |
| !argnum     | 5E8DE | (seco --> seco #args)<br>Gives argument count for secondary. Checks first command in seco, if different from CK0 CK1&Dispatch ... CK4&Dispatch #5 is returned.                                        |
| infarg?     | 5E9A7 | (seco --> seco flag )<br>Is first command in seco CKINFARGS?                                                                                                                                          |
| ?spliteq    | 558BE | (Metal&Meta2&= -> Meta2 Metal, Meta -> Meta Meta)<br>(If Mlst is equal to =, split, else dup)                                                                                                         |
| ?addinver:  | 55314 | (Meta&Nob -> Meta, Meta -> Meta&LAM1)<br>(If next is equal to Mlst, drop Mlst, else addt 1LAM)                                                                                                        |
| ?addrever   | 5540E | (Meta&1LAM -> Meta&1LAM, Meta -> Meta&1LAM)<br>(If Mlst is equal to 1LAM, nada, else addr LAM1)                                                                                                       |
| ?addsimir   | 5613D | (Meta Meta -> Meta)<br>(Metal Meta2 -> Metal&Meta2&1LAM)                                                                                                                                              |
| count+      | 58C02 | (Meta -> Meta #0)<br>(Meta&+&+...&+ -> Meta #n)                                                                                                                                                       |
| count*      | 58C0E | (Above table for *)                                                                                                                                                                                   |
| MetaConCase | 549EC | (Meta --> Meta ) (COLA if no id/lam/symb/romptr)                                                                                                                                                      |
| dvars?      | 5AD08 | (Meta --> Meta TRUE/FALSE)                                                                                                                                                                            |
| >dvars      | 5670F | (Metal Meta2 -> Metal&Meta2')<br>(All IDs in Meta2 matching LAM 'dvar contents are changed to LAM_'dvar. (Metal can be #0))                                                                           |
| dvars>      | 5AC86 | (Meta -> Meta') (LAM 'dvar:s changed to 1LAM)                                                                                                                                                         |
| SYMBN:      | 54CEF | (Meta -> ALG&Nob)                                                                                                                                                                                     |
| replw%1     | 554B3 | (Meta&Ob -> Meta&%1)                                                                                                                                                                                  |
| replw%-1    | 55607 | (Meta&Ob -> Meta&%-1)                                                                                                                                                                                 |
| %lpshml     | 5FC38 | (Meta -> %1 #1 Metal)                                                                                                                                                                                 |
| pshDROP     | 638FD | (Metal Meta2 -> Meta2 metal_ob1 metal_ob2... metal_obn)                                                                                                                                               |
| COLAkeep1st | 5483C | Return & (Meta&Ob -> Ob)                                                                                                                                                                              |

Rules menu operations: ( Meta --> Meta' )

(For matching patterns the corresponding operation is COLAd, thus current stream is dropped in such cases.)

|       |               |                                              |
|-------|---------------|----------------------------------------------|
| 5BE56 | MetaMulInv    | Simplify combinations of INV and * (using /) |
|       | 58A61 colinv1 | [Expr1 INV Expr2 INV *] --> [Expr1 Expr2 /]  |
|       | 58A93 colinv2 | [Expr INV *] --> [Expr /]                    |
|       | 58AAC colinv3 | [Expr1 Expr2 INV Expr3 *] --> [E1 E3 E2 /]   |
| 5BE81 | Meta<-->      | Commute terms                                |
| 5BECE | Meta<-A       | Associate left term                          |
| 5BF53 | MetaA->       | Associate right term                         |
| 5BFD8 | MetaD->       | Delete right term (via expansion)            |
| 5C0B9 | Meta<-D       | Delete left term (via expansion)             |
| 5C102 | Meta<-D!      | Delete left term (Above - ^ expansion)       |
| 5C137 | Meta->()      | Remove prefix                                |
| 5C204 | Metal/()      | Double invert, then remove prefix            |

|       |            |                                               |
|-------|------------|-----------------------------------------------|
| 5C261 | Meta-()    | Double negate, then remove prefix             |
| 5C2CE | MetaE^     | Transform EXP(A*B) to EXP(A)^B                |
| 5C31B | MetaE()    | Transform EXP(A)^B to EXP(A*B)                |
| 5C348 | MetaL*     | Transform LN(A^B) to LN(A)*B                  |
| 5C375 | MetaL()    | Transform LN(A)*B to LN(A^B)                  |
| 5C3C2 | Meta<-M    | Merge common factor on left sides             |
| 5C4CF | MetaM->    | Merge common factor on right sides            |
| 5C53C | MetaAF     | Add fractions                                 |
| 5C589 | Meta(<-    | Include left term                             |
| 5C5D6 | Meta->)    | Include right term                            |
| 5C623 | Meta(())   | Put parentheses over nearest terms            |
| 5C670 | Meta->TRG  | Change EXP to trigonometric functions         |
| 5C68D | MetaT->    | Move nearest left term to right               |
| 5C6D9 | Meta<-T    | Move nearest right term to left               |
| 5C73D | Meta->()C% | Remove first RE,IM or CONJ                    |
| 5C845 | Meta->DEF  | Define function (SIN..SINH..ASIN..ASINH..)    |
| 5C91D | MetaTRG*   | Expand trigonometric function of a sum (or -) |

#### Repeated evaluation:

|       |             |       |               |
|-------|-------------|-------|---------------|
| 5CE15 | Meta<-Aall  | 5CE4C | MetaA->all    |
| 5CDF2 | Meta<-Dall  | 5CEF1 | MetaD->all    |
| 5CF5A | Meta<-Tall  | 5CF23 | MetaT->all    |
| 5CFF5 | Meta<-Mall  | 5D009 | MetaM->all    |
| 5CEBA | Meta(<-all  | 5CE83 | Meta->)all    |
| 5CF91 | Meta->()all | 5CFC3 | Meta->()C%all |

#### Repeaters:

|       |            |                                                                                                                                                                                                                    |
|-------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5CD52 | evalcase:  | ( Meta --> ? )<br>Evaluates next object. If it drops current stream then continue, else SKIP next.<br>Example:    :: evalcase: Meta<-D Meta<-Daga ;                                                                |
| 5CD7A | revalcase: | ( Meta --> ? )<br>Evaluates next ob for subexpressions until current stream is not dropped by ob.<br>Example:    Meta<-Daga<br>:: revalcase: Meta<-D COLA RDROP ;<br>(COLA RDROP is there to mark succesful oper.) |
| 5D01D | evalrept:  | ( Meta --> ? )<br>Repeats next until ??                                                                                                                                                                            |

Ones using MetaValidate drop remainder of stream if operation was succesful. (Metal Meta2 Meta --> Metal Meta2 Meta')

Some check is made on Metal before operations are allowed (?)

|       |            |               |
|-------|------------|---------------|
| 596D3 | MetaRCOLCT | Limited COLCT |
| 5971D | MetaDNEG   | Double negate |
| 5976B | MetaDINV   | Double invert |
| 597B5 | Metal*     | 1*            |
| 5983B | Meta^1     | 1^            |
| 59885 | Metal/     | 1/            |
| 5990F | Meta+1-1   | +1 -1         |

\*\*\*\*\*

## 14. Library & Backup objects

~~~~~

```

0AB51  PORTSTATUS      ( #port --> present? writeable? merged? #size #addr )
      PORTEND      ( #port --> #addr )      ( GX: 0AB22 )
      Get end address of port.
      NEXTLIBBAK    ( #addr --> backup/library #nextaddr TRUE / FALSE )
      Get next library/backup.
0B037  MERGE         ( #port --> )          ( GX: 0B409 )

```

14.1 Rom Pointers

^^^^^^^^^^^^^^^^

```

08CCC  #>ROMPTR      ( #libnum #cmdnum --> ROMPTR )
      ROMPTR>#      ( ROMPTR --> #libnum #cmdnum )
      ROMPTR@       ( ROMPTR --> ob TRUE / FALSE )
      DUPROMPTR@    ( ROMPTR --> ROMPTR ob TRUE / ROMPTR FALSE )
02FEF  DoRomptr     ( ROMPTR --> ? )
      Recall ROMPTR contents and EVAL
      Undefined XLIB error if not found
      ?>ROMPTR      ( ob --> ob' )
      If ROM_WORD? and TYPECOL? then RPL@
      ?ROMPTR>      ( ob --> ob' )
      If TYPEROMP? and contents exist INHARDROM?
      then return contents.
      RESOROMP      ( --> ob )
      Recalls contents of romptr in return stack
      ( :: 'R ROMPTR@ DROP ; )

07E76  PTR>ROMPTR   ( ob --> ROMPTR T / F )
0821F  ROMPTRDECOMP ( ROMPTR --> ID T / F )      ( GX: 081FB )
08207  PTR>ID       ( ob --> id T / F )      ( GX: 081E3 )
07BFD  ID>CMD       ( id --> id T / ROMPTR T/ F ) ( GX: 07C18 )
      Searches id from current path, if found returns TRUE.
      Else searches attached libraries.

      ROM-WORD?     ( ob --> flag )
      DUPROM-WORD?  ( ob --> ob flag )

```

14.2 Libraries

^^^^^^^^^^^^^^^^

```

0814F  NAME>ROMPART (      id --> #libnum T / F ) ( GX: 0814F )
081B9  ROMPARTNAME  ( #libnum --> id T / F ) ( GX: 08199 )
081EE  LIB>#        ( library --> #libnum T / F ) ( GX: 081DE )
0807F  ROMPARTADDR  ( #libnum --> #addr T / F ) ( GX: 08081 )
      Recall library address+10 (skipped prolog & length)
0809E  ROMPARTSIZE  ( #libnum --> #nibbles-10 T / F ) ( GX: 080BF )
080C9  NEXTROMPID   ( #libnum --> #nextlibnum T / F ) ( GX: 080DA )
      If #libnum exists, #libnum is returned with TRUE
08101  GETHASH      ( #libnum --> hxs_table T / F ) ( GX: 08112 )
0811C  GETMESG      ( #libnum --> [ ] T / F ) ( GX: 08130 )
08128  GETLINK      ( #libnum --> hxs_table T / F ) ( GX: 0813C )
08143  GETCONFIG     ( #libnum --> ob T / F ) ( GX: 08157 )
07F98  ROMPART      ( rrp --> { #lib1 ... #libn } T / F ) ( GX: 07F86 )
      ( ROMPTR -> #libnum )

```

```

TOSRRP      ( # --> )
              Attach library to home directory
076AE OFFSRRP ( # --> )
              Detach library from home directory
077C2 ONSRRP? ( # --> flag )          ( GX: 0778D )
              Is library attached to home directory?
07638 SETHASH ( hxs #libnum --> )    !! BUGGED !!
0764E SETMSG  ( [$] #libnum --> )

021E4 ROMPOLL ( --> )                  ( GX: 021DD )
              Configure internal & external libraries
02216 DOROMPOLL ( { #libnum1 #libnum2 ... } --> )    ( GX: 0210F )
              Configure libraries

```

14.3 Backups ^^^^^^^^^^^^^^

```

BAKNAME      ( backup --> id T / F )
BAK>OB       ( backup --> ob )
094A4 HOME>BAK ( --> bak )
              Home directory becomes unrooted.
09699 BAK>HOME ( bak --> )
              Restores HOME from backup.
0970A SETINITEN ( --> )

```

15. Stack operations

~~~~~

DEPTH ( obl ... obn --> #n )  
 reversym ( obl ... obn #n --> obn ... obl #n )

DROP 2DROP 3DROP 4DROP 5DROP 6DROP 7DROP  
 2DROP 2DUP 2SWAP 2OVER 2DUPSWAP 2DUP5ROLL  
 NDROP N+1DROP DROPNDROP NDUP NDUPN

|            |              |              |           |             |              |
|------------|--------------|--------------|-----------|-------------|--------------|
| DUP        | DROP         | SWAP         | OVER      | ROT         | UNROT        |
| DUPDUP     | DROPDUP      | SWAPDUP      | OVERDUP   | ROTDUP      | UNROTDUP     |
| -          | 2DROP        | SWAPDROP     | -         | ROTDROP     | UNROTDROP    |
| -          | DROPSWAP     | -            | OVERSWAP  | ROTSWAP     | UNROTSWAP    |
| -          | DROPOVER     | SWAPOVER     | -         | ROTOVER     | UNROTOVER    |
| DUPROT     | DROPROT      | SWAPROT      | -         | -           | -            |
| DUPUNROT   | -            | -            | OVERUNROT | -           | -            |
| -          | DROPSWAPDROP | -            | -         | -           | UNROTSWAPDRO |
| -          | -            | SWAPDROPDUP  | -         | -           | -            |
| -          | -            | SWAPDROPSWAP | -         | ROTDROPSWAP | -            |
| -          | -            | -            | -         | ROT2DROP    | UNROT2DROP   |
| -          | -            | SWAP2DUP     | -         | ROT2DUP     | -            |
| DUP3PICK   | -            | SWAP3PICK    | -         | -           | -            |
| -          | -            | SWAP4PICK    | -         | -           | -            |
| -          | -            | SWAP4ROLL    | -         | -           | -            |
| DUP4UNROLL | -            | -            | -         | -           | -            |
| -          | -            | -            | OVER5PICK | -           | -            |
| -          | -            | -            | -         | ROTROT2DROP | -            |

|            |            |              |
|------------|------------|--------------|
| 3PICKSWAP  | 3PICKOVER  | 3PICK3PICK   |
| 4PICKSWAP  | 4PICKOVER  |              |
| 4ROLLSWAP  | 4ROLLOVER  | 4ROLLDROP    |
| 4UNROLLDUP | 4UNROLLROT | 4UNROLL3DROP |

Picking:                      Rolling:                      Unrolling:

OVERSWAP, OVERUNROT

|                  |                  |                     |
|------------------|------------------|---------------------|
| OVER             | SWAP             | SWAP                |
| 3PICK            | ROT              | UNROT, 3UNROLL      |
| 4PICK            | 4ROLL, FOURROLL  | 4UNROLL, FOURUNROLL |
| 5PICK            | 5ROLL, FIVEROLL  | 5UNROLL, FIVEUNROLL |
| 6PICK            | 6ROLL, SIXROLL   | 6UNROLL, SIXUNROLL  |
| 7PICK            | 7ROLL, SEVENROLL | 7UNROLL             |
| 8PICK            | 8ROLL, EIGHTROLL | 8UNROLL             |
| -                | -                | 10UNROLL            |
| PICK             | ROLL             | UNROLL              |
| -                | ROLLDROP         | -                   |
| -                | ROLLSWAP         | -                   |
| DUPPICK          | DUPROLL          | -                   |
| #+PICK           | #+ROLL           | #+UNROLL            |
| #-PICK           | #-ROLL           | #-UNROLL            |
| #1+PICK          | #1+ROLL          | #1+UNROLL           |
| DUP#1+PICK       | -                | -                   |
| #2+PICK          | #2+ROLL          | #2+UNROLL           |
| 5FC24 DUP#2+PICK | DUP#2+ROLL       | -                   |
| -                | -                | OVER#2+UNROL        |

#3+PICK

-

-

#4+PICK

-

-

\*\*\*\*\*

## 16. Checking Arguments

~~~~~

	CK<n>	Saves current command to LASTCKCMD Verify atleast <n> objects on stack. Saves stack size to STACKMARK (@DSKBOT-D1 nibbles) If last arguments flag is set then Stores <n> to CKARGNUM (1 nibble) Store addresses of <n> args to CKARG1, CKARG2, CKARG3, CKARG4, CKARG5
	CK<n>NOLASTWD	Above minus saving current command
	CKNNOLASTWD	Does CK1NOLASTWD to check 1 argument (count) (See above for side effects) Does CKREAL Checks argument count
	CK&DISPATCH0	Dispatch on stack arguments
	CK&DISPATCH1	Dispatch on stack arguments, strip tags if necessary
18FA9	CK&DISPATCH2	Same as CK&DISPATCH1. Haven't found any hidden meanings yet. (Several words do check what the first command is in some composite. See EvalNoCK.)
	CK<n>&Dispatch	Combination of CK<n> and CK&DISPATCH1
5EA09	CKINFARGS	Used in aka words of functions that can take an unlimited number of arguments. Gets Meta as argument, checks lenght againts DEPTH and errors if Meta is too short. Collects the arguments to a list for marking the stack (CK1NOLASTWD), then explodes again (CKNEWMETA). Is usually followed by CKSYMBTYPE and nmetasyms checks to check argument types. (Internally used by yAPPLY and yWHERE)
	CK0	CK0NOLASTWD
	CK1	CK1NOLASTWD CK1&Dispatch
	CK2	CK2NOLASTWD CK2&Dispatch
	CK3	CK3NOLASTWD CK3&Dispatch
	CK4	CK4NOLASTWD CK4&Dispatch
	CK5	CK5NOLASTWD CK5&Dispatch
	CKN	CKNNOLASTWD
	0LASTOWDOB!	Clears command save done by the last CK<n> command aka 0LastRowWrd!
	AtUserStack	:: CK0NOLASTWD 0LASTOWDOB! ;
	CK1NoBlame	:: 0LASTOWDOB! CK1NOLASTWD ;
	'RSAVEWORD	Stores first object in prev to LASTCKCMD aka 'RSaveRomWrd
	EvalNoCK	(comp --> ?) Evaluate composite without saving as current command. If first command is CK<n>&Dispatch CK&DISPATCH1 is executed instead. If first command <> .. first command is skipped.
18F6A	EvalNoCK:	(next --> ?) EvalNoCK with next ob as argument

Type checking: (SETTYPEERR if not of correct type)

	CKREAL	(ob --> ob)	Checks for real
193C1	CKARRY	(ob --> ob)	Checks for array
194BB	CKRARRY	(ob --> ob)	Checks for real array
194D9	CKCARRY	(ob --> ob)	Checks for complex array
19443	CKLIST	(ob --> ob)	Checks for list
20BE0	CKNAMELIST	({} --> {})	Checks for non-empty list of names
1945C	CKLISTTYPE	(ob #prolog --> ob)	Checks for non empty list of certain type
	CKSYMBTYPE	(ob --> ob)	Checks for quoted name (Name as symbolic)
	nmetasyms	(Meta --> Meta)	Checks for Meta containing % C% Unit id lam symb
19207	CKNFLOATS	(obl ... obn any #n --> obl ... obn any #n zero)	Check for n floats (F%)
			zero = C%0 if atleast one float was complex, else %0

17. Error Generation & Trapping

~~~~~

### 17.1 General Words

^^^^^^^^^^^^^^^^

|       |               |                             |                                                                                                                                 |
|-------|---------------|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------|
|       | ERRBEEP       | ( --> )                     | *beep* (The irritating one)                                                                                                     |
|       | ERROR@        | ( --> # )                   | Returns current error number                                                                                                    |
|       | ERRORSTO      | ( # --> )                   | Stores new error number                                                                                                         |
|       | ERROROUT      | ( # --> )                   | Stores new error number and does ERRJMP                                                                                         |
|       | ERRORCLR      | ( --> )                     | Stores zero as new error number                                                                                                 |
|       | ERRJMP        | ( --> )                     | Invoke error handling subsystem                                                                                                 |
|       | 'ERRJMP       | ( --> ERRJMP )              |                                                                                                                                 |
|       | DO#EXIT       | ( #error --> )              | Stores new error number, AtUserStack, then ERRJMP                                                                               |
|       | DO%EXIT       | ( %error --> )              |                                                                                                                                 |
| 1501B | DOHXSEXIT     | ( hxs_error --> )           |                                                                                                                                 |
|       | DO\$EXIT      | ( \$ --> )                  | Stores \$ as EXITMSG, #70000 as new error, AtUserStack, then ERRJMP                                                             |
|       | ABORT         | ( --> )                     |                                                                                                                                 |
|       | ERRSET        | ( --> )                     | Does ERRORCLR and ERRJMP                                                                                                        |
|       |               | ( --> )                     | Set new error trap                                                                                                              |
|       | ERRTRAP       | ( --> )                     | Error trap marker, skips next object                                                                                            |
| 13FE5 | SAVEERRN      | ( --> )                     | Saves error number to last-error                                                                                                |
| 1400E | ERR0          | ( --> )                     | Clears last error                                                                                                               |
| 14039 | ERRN          | ( --> #error )              | Returns last error number                                                                                                       |
| 1404C | ERRN>HXS      | ( --> hxs_error )           | Returns last error number as hxs                                                                                                |
| 14065 | ERRM          | ( --> \$error )             | Returns last error message                                                                                                      |
|       | JstGETTHEMESG | ( # --> \$ )                | Fetches message from message table.<br>See binary number section for internal messages.<br>For libraries use libnum*#100+msgnum |
|       | GETTHEMESG    | ( # --> \$ )                | If #70000 then GETEXITMSG, else JstGETTHEMESG.                                                                                  |
| 04DD7 | SPLITMSG#     | ( #msg --> #error #libnum ) |                                                                                                                                 |
|       | EXITMSGSTO    | ( \$ --> )                  | Store \$ as EXITMSG (User defined error message)                                                                                |
| 04E07 | DRPGETEXITMSG | ( ob --> \$ )               |                                                                                                                                 |

## 17.2 Error Words

^^^^^^^^^^^^^^^^^^

|       |              |       |             |       |             |
|-------|--------------|-------|-------------|-------|-------------|
| 04FAA | SETPowerLost |       | SETMEMERR   | 04FC2 | SETDIRRECUR |
| 04FCE | SETUNDEFLAM  | 04FDA | SETINVCARD  | 04FE6 | SETOBINUSE  |
| 04FF2 | SETPORTNOTAV | 04FFE | SETNOROOM   | 0500A | SETOBNOTIN  |
| 05016 | SETROMPERR   | 0CBAE | NOALARMERR  | 10EEA | INVUNITERR  |
| 10EFA | CONSTUNITERR | 10F54 | NULLCHARERR | 10F64 | INVFUNCERR  |
| 10F74 | NOEQERR      |       | SYNTAXERR   | 10F96 | SETINVPPAR  |
| 10FA6 | SETNONREAL   | 10FB6 | SETISOLERR  | 10FC6 | SETHALTERR  |
| 10FE6 | LASTSTKERR   | 10FF6 | LASTCMDERR  | 11006 | ARGNUMERR   |
|       | SETCIRCERR   | 11026 | DIRARGERR   | 11036 | EMPTYDIRERR |
| 11046 | INVDEFERR    | 11056 | MISSLIBERR  | 11066 | IDCONFERR   |
| 18C92 | SETNONEXTERR |       | SETSIZEERR  |       | SETTYPEERR  |
| 18CC2 | FEWARGSERR   | 29DCC | POSFLOWERR  | 29DDC | NEGFLOWERR  |
| 29DEC | OVERFLOWERR  | 29DFC | UNDEFRESERR | 29E0C | INFRESERR   |

\*\*\*\*\*

## 18. General Tests & Flags

~~~~~

18.1 Flags

^^^^^^^^

```
COERCEFLAG      ( flag --> %1 / %0 )                (Also drops cur.stream)
( "UNCOERCEFLAG" = %0<> )

TRUE            ( --> TRUE )
FALSE           ( --> FALSE )
FALSETRUE       ( --> FALSE TRUE )      aka FalseTrue
TRUETRUE        ( --> TRUE TRUE )
TRUEFALSE       ( --> TRUE FALSE )      aka TrueFalse
TrueTrue        ( --> TRUE TRUE )
FalseFalse      ( --> FALSE FALSE )
failed          ( --> FALSE TRUE )

DROPTTRUE       (          ob --> TRUE )
2F542 2DROPTTRUE (          ob1 ob2 --> TRUE )
5F657 3DROPTTRUE (          ob1 ob2 ob3 --> TRUE )
10029 4DROPTTRUE (          ob1 ob2 ob3 ob4 --> TRUE )
DROPTFALSE      (          ob --> FALSE )
2DROPTFALSE     (          ob1 ob2 --> FALSE )
5F5E4 4DROPTFALSE (          ob1 ob2 ob3 ob4 --> FALSE )
5F6B1 5DROPTFALSE (          ob1 ... ob5 --> FALSE )
NDROPTFALSE     (          ob1 ... obn #n --> FALSE )
SWAPTRUE        (          ob1 ob2 --> ob2 ob1 TRUE )
SWAPDROPTTRUE   (          ob1 ob2 --> ob2 TRUE )
XYZ>ZTRUE       (          ob1 ob2 ob3 --> ob3 TRUE )

5DE41 COLATTRUE ( --> TRUE )
5DE55 COLAFALSE ( --> FALSE )      aka RDROPTFALSE

NOT              ( flag -> flag' )
Returns TRUE for FALSE and vice versa
AND             ( flag1 flag2 --> flag )
Returns FALSE if flags are different
OR              ( flag1 flag2 --> flag )
Returns TRUE if either flag is TRUE
XOR             ( flag1 flag2 --> flag )
Returns TRUE if flags are different
ORNOT           ( flag1 flag2 --> flag )
Returns FALSE if either flag is TRUE
NOTAND          ( flag1 flag2 --> flag )
Returns FALSE if flags are the same.
ROTAND          ( flag1 ob flag2 --> ob flag )
```

18.2 General Tests

^^^^^^^^^^^^^^^^^^^^

EQ (ob1 ob2 --> flag)
Returns TRUE if ob1 and ob2 are the same object, i.e.
occupy the same physical space in memory.
Tests only the addresses of the objects.

EQ: (ob --> flag)
EQ with next ob in this stream

DUPEQ: (ob --> ob flag)

2DUPEQ (ob1 ob2 --> ob1 ob2 flag)

EQOR (flag1 ob1 ob2 --> ob1 ob2 flag2)

EQOVER (ob3 ob1 ob2 --> ob3 flag ob3)

EQUAL (ob1 ob2 --> flag)
Returns TRUE if object prologs & contents are the same.
System equivalent of the user RPL command SAME.

EQUALNOT (ob1 ob2 --> flag)

EQUALOR (flag1 ob1 ob2 --> ob1 ob2 flag2)

18.3 Type Tests

^^^^^^^^^^^^^^^^^^^^

XEQTYPE (ob --> ob %type)
System equivalent of user RPL command TYPE.
Only saving the stack is left out.

TYPE (ob --> #prolog)
See binary number section for valid objects.

TYPEARRY?	DUPTYPEARRY?	DTYPEARRY?
TYPERARRY?	TYPECARRY?	
TYPEBINT?	DUPTYPEBINT?	
TYPECHAR?	DUPTYPECHAR?	
TYPECMP?	DUPTYPECMP?	
TYPECOL?	DUPTYPECOL?	DTYPECOL?
TYPECSTR?	DUPTYPECSTR?	DTYPECSTR?
TYPEEXT?	DUPTYPEEXT?	
TYPEGROB?	DUPTYPEGROB?	
TYPEHSTR?	DUPTYPEHSTR?	
TYPEIDNT?	DUPTYPEIDNT?	
TYPELAM?	DUPTYPELAM?	
TYPELIST?	DUPTYPELIST?	DTYPELIST?
TYPEREAL?	DUPTYPEREAL?	DTYPEREAL?
TYPEROMP?	DUPTYPEROMP?	
TYPERRP?	DUPTYPERRP?	
TYPESYMB?	DUPTYPESYMB?	
TYPE_TAGGED?	DUPTYPE_TAG?	

19. Runstream Control

~~~~~

|       |               |                         |                                                                                                                                                                                      |
|-------|---------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | NOP           | ( --> )                 | No operation.                                                                                                                                                                        |
|       | 'NOP          | ( --> NOP )             | Pushes NOP to stack.                                                                                                                                                                 |
|       | '             | ( --> ob )              | Pushes next object in stream to stack. (Skipping it)                                                                                                                                 |
|       | 'R            | ( --> ob )              | Pushes next object in return stack to stack. (Skipping it)<br>If top return stack body is empty (contains only SEMI) then<br>a null secondary is pushed and pointer is not advanced. |
|       | 'R'R          | ( --> ob1 ob2 )         | 'R twice                                                                                                                                                                             |
|       | ticR          | ( --> ob TRUE   FALSE ) | Pushes next object in return stack to stack + TRUE or<br>if top return stack body is empty drops it and pushes FALSE.                                                                |
|       | >R            | ( :: --> )              | Pushes :: to top of return stack (skips prolog)                                                                                                                                      |
|       | R>            | ( --> :: )              | Creates & pops a :: from top return stack body to stack                                                                                                                              |
|       | R@            | ( --> :: )              | Creates a :: from top return stack body to stack                                                                                                                                     |
| 14F2A | RROLL         | ( #n --> )              | Roll nth return stack level to top of return stack                                                                                                                                   |
|       | 'REVAL        | ( --> ? )               | Does 'R, then EVAL                                                                                                                                                                   |
|       | EVAL          | ( ob --> ? )            | Executes ob                                                                                                                                                                          |
|       | COMPEVAL      | ( comp --> ? )          | EVAL just pushes a list back, this one executes it.                                                                                                                                  |
|       | COLA_EVAL     | ( ob --> ? )            | Return & EVAL                                                                                                                                                                        |
|       | 2@REVAL       | ( --> ? )               | EVALs first object in prevprev stream                                                                                                                                                |
|       | 3@REVAL       | ( --> ? )               | EVALs first object in prevprevprev stream                                                                                                                                            |
|       | GOTO          | ( --> )                 | Jumps to address next in stream.                                                                                                                                                     |
|       | ?GOTO         | ( flag --> )            | If TRUE then jump to next address in stream, else skip 5 nibs                                                                                                                        |
|       | NOT?GOTO      | ( flag --> )            | If TRUE then skip 5 nibs, else jump to next address in stream                                                                                                                        |
|       | RDUP          |                         | Duplicates top return stack level                                                                                                                                                    |
|       | RDROP         |                         | Pops the return stack.                                                                                                                                                               |
|       | 2RDROP        |                         | Pops 2 return stack levels                                                                                                                                                           |
|       | 3RDROP        |                         | Pops 3 return stack levels                                                                                                                                                           |
|       | DROPRDROP     |                         | Pops stack and return stack ( ob --> )                                                                                                                                               |
|       | RDROPCOLA     |                         | RDROP & COLA                                                                                                                                                                         |
| 54C4F | RDROPCOLATRUE |                         | RDROP & COLATRUE                                                                                                                                                                     |
|       | RSWAP         |                         | Swap in return stack                                                                                                                                                                 |
|       | RSKIP         |                         | Skip 1st object in return stack                                                                                                                                                      |

## 19.1 Definite Loops

^^^^^^^^^^^^^^^^^^^^

```

DO                ( #stop #start --> )
*
ZERO_DO           ( #stop --> )
*
ONE_DO            ( #stop --> )
*
#1+_ONE_DO        ( #stop --> )

LOOP              ( --> )
+LOOP             ( # --> )      Increment INDEX by 1, do LOOP
07321 STOPLOOP    ( --> )      Destroy topmost DoLoop environment

INDEX@            ( --> # )      Recall topmost loop counter value
ISTOP@            ( --> # )      Recall topmost loop stop value
JINDEX@           ( --> # )      Recall 2nd topmost loop counter value
07264 JSTOP@       ( --> # )      Recall 2nd topmost loop stop value
INDEXSTO          ( # --> )      Stores # as new topmost loop counter value
ISTOPSTO          ( # --> )      Stores # as new topmost loop stop value
072AD JINDEXSTO    ( # --> )      Stores # as new 2nd topmost loop counter value
JSTOPSTO          ( # --> )      Stores # as new 2nd topmost loop stop value

ExitAtLOOP        ( --> )          aka ZEROISTOPSTO
3F78C DUPExitAtLOOP ( ob --> ob ob ) (Don't look at me, I just )
3F7EB ExitAtLOOPDUP ( ob --> ob ob ) (tell what's in there :)
4334F DRPExitAtLOOP ( ob --> )

*
DUP#0_DO          ( #stop --> #stop )
toLEN_DO          ( {} --> {} )      ( From ONE to #elements+1 )
37BCB ONE_DO_ARRAY ( --> )          ( From ONE to #elements+1 )

*
DROPLOOP          ( ob --> )
*
SWAPLOOP          ( ob1 ob2 --> ob2 ob1 )
54CB3 SWAPLOOPDROP
DUPINDEX@         ( ob --> ob ob # )
SWAPINDEX@        ( ob1 ob2 --> ob2 ob1 # )
OVERINDEX@        ( ob1 ob2 --> ob1 ob2 ob1 # )

INDEX@#-          ( # --> #' )
ISTOP-INDEX       ( --> # )

```

## 19.2 Indefinite loops

^^^^^^^^^^^^^^^^^^^^

```

0716B IDUP        ( --> )
          Push top body onto return stack
          (Listed in rplman.doc but not in entries.a)
BEGIN      ( --> )
          Push top body onto return stack
AGAIN      ( --> )
          Copy return stack to top
REPEAT     ( --> )
          Copy return stack to top
UNTIL      ( flag --> )
          If FALSE then AGAIN, else RDROP
WHILE      ( flag --> )
          If TRUE do nothing, else RDROP & 2SKIP

*
NOT_UNTIL
#0=UNTIL      (Actually it should be DUP#0=UNTIL)
NOT_WHILE
DUP#0<>WHILE

```

### 19.3 COLA & case words

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

|             |                                               |
|-------------|-----------------------------------------------|
| COLA        | Evaluate next & drop remainder of this stream |
| ONECOLA     | ONE & COLA                                    |
| SWAPCOLA    | SWAP & COLA                                   |
| XYZ>ZCOLA   | UNROT2DROP & COLA                             |
| COLACOLA    | SEMI & COLA                                   |
| COLAcase    | SEMI & case                                   |
| COLANOTcase | SEMI & NOTcase                                |
| REQcase     | ' & EQcase                                    |
| REQcasedrop | ' & EQcasedrop                                |

Initial stream:                    :: <Test> <Ob1> ... <ObN> <Rest> ;

In the 'effective' stream <Obn> <Rest> implies all in between.

m < n in comparisons.

obl and ob2 are different in comparisons.

f1 and f2 are different flags, T is TRUE and F is FALSE.

#set means that the corresponding flag is set.

Flag tests:

|       |              |   |           |                             |   |   |
|-------|--------------|---|-----------|-----------------------------|---|---|
|       | SKIP         | ( |           | --> :: <Ob2> <Rest>         | ; | ) |
| 0715C | 2SKIP        | ( |           | --> :: <Ob3> <Rest>         | ; | ) |
| 283D8 | 3SKIP        | ( |           | --> :: <Ob4> <Rest>         | ; | ) |
|       | ?SKIP        | ( | T         | --> :: <Ob2> <Rest>         | ; | ) |
|       | aka NOT_IT   | ( | F         | --> :: <Ob1> <Rest>         | ; | ) |
| 21C47 | MEMSKIP      | ( | ob        | --> #nextaddress            |   | ) |
|       | skipcola     | ( |           | --> :: <Ob2>                | ; | ) |
|       | 2skipcola    | ( |           | --> :: <Ob3>                | ; | ) |
|       | 3skipcola    | ( |           | --> :: <Ob4>                | ; | ) |
|       | 5skipcola    | ( |           | --> :: <Ob6>                | ; | ) |
|       | COLASKIP     |   |           | SEMI & SKIP                 |   |   |
| 283C4 | COLAskipcola |   |           | COLA & skipcola             |   |   |
|       | ?SEMI        | ( | T         | --> :: ;                    |   | ) |
|       |              | ( | F         | --> :: <Ob1> <Rest>         | ; | ) |
|       | NOT?SEMI     | ( | T         | --> :: <Ob1> <Rest>         | ; | ) |
|       |              | ( | F         | --> :: ;                    |   | ) |
|       | ?SEMIDROP    | ( | T         | --> :: ;                    |   | ) |
|       |              | ( | ob F      | --> :: <Ob1> <Rest>         | ; | ) |
|       | NOT?DROP     | ( | T         | --> :: <Ob1> <Rest>         | ; | ) |
|       |              | ( | ob F      | --> :: <Ob1> <Rest>         | ; | ) |
|       | ?SWAP        | ( | ob1 ob2 T | --> :: ob2 ob1 <Ob1> <Rest> |   | ) |
|       |              | ( | F         | --> :: <Ob1> <Rest>         | ; | ) |
|       | ?SKIPSWAP    | ( | T         | --> :: <Ob1> <Rest>         | ; | ) |
|       |              | ( | ob1 ob2 F | --> :: ob2 ob1 <Ob1> <Rest> | ; | ) |
|       | ?SWAPDROP    | ( | ob1 ob2 T | --> :: ob2 <Ob1> <Rest>     | ; | ) |
|       |              | ( | ob1 ob2 F | --> :: ob1 <Ob1> <Rest>     | ; | ) |
|       | NOT?SWAPDROP | ( | ob1 ob2 T | --> :: ob1 <Ob1> <Rest>     | ; | ) |
|       |              | ( | ob1 ob2 F | --> :: ob2 <Ob1> <Rest>     | ; | ) |
|       | RPIT         | ( | T ob      | --> :: ob <Ob1> <Rest>      | ; | ) |
|       |              | ( | F ob      | --> :: <Ob1> <Rest>         | ; | ) |
|       | RPITE        | ( | T ob1 ob2 | --> :: ob1 <Ob1> <Rest>     | ; | ) |
|       |              | ( | F ob1 ob2 | --> :: ob2 <Ob1> <Rest>     | ; | ) |
|       | COLARPITE    | ( | T ob1 ob2 | --> :: ob1 ;                |   | ) |
|       |              | ( | F ob1 ob2 | --> :: ob2 ;                |   | ) |

```

2'RCOLARPITE      Return, then ITE
IT                (      T --> :: <Ob1> <Rest> ;      )
                  (      F --> :: <Ob2> <Rest> ;      )
ITE               (      T --> :: <Ob1> <Ob3> <Rest> ;    )
                  (      F --> :: <Ob2> <Rest> ;      )
COLAITE           (      T --> :: <Ob1> ;                )
                  (      F --> :: <Ob2> ;                )
ITE_DROP          (      ob T --> :: <Ob2> <Rest>      ;    )
                  (      ob F --> :: ob <Ob1> <Rest> ;    )
ANDITE            (      f1 f1 --> :: <Ob1> <Ob3> <Rest> ;  )
                  (      f1 f2 --> :: <Ob2> <Rest> ;      )
case              (      T --> :: <Ob1> ;                )
                  (      F --> :: <Ob2> <Rest> ;          )
casedrop          (      ob T --> :: <Ob1> ;                )
                  (      ob F --> :: ob <Ob2> <Rest> ;    )
case2drop         ( ob1 ob2 T --> :: <Ob1> ;                )
                  ( ob1 ob2 F --> :: ob1 ob2 <Ob2> <Rest> ;  )
caseDROP          (      ob T --> :: ;                  )
                  (      ob F --> :: ob <Ob1> <Rest>      )
case2DROP         ( ob1 ob2 T --> :: ;                  )
                  ( ob1 ob2 F --> :: ob1 ob2 <Ob1> <Rest> )
NOTcase           (      T --> :: <Ob2> <Rest> ;          )
                  (      F --> :: <Ob1> ;                )
NOTcasedrop       (      ob T --> :: ob <Ob2> <Rest> ;    )
                  (      ob F --> :: <Ob1> ;                )
NOTcase2drop      ( ob1 ob2 T --> :: ob1 ob2 <Ob2> <Rest> ;  )
                  ( ob1 ob2 F --> :: <Ob1> ;                )
NOTcaseDROP       (      ob T --> :: ob <Ob1> <Rest>      )
                  (      ob F --> :: ;                  )
NOTcase2DROP      ( ob1 ob2 T --> :: ob1 ob2 <Ob1> <Rest> )
                  ( ob1 ob2 F --> :: ;                  )
NOTcsdrpfls      (      ob T --> :: ob <Ob1> <Rest> ;    )
                  (      ob F --> FALSE                  )
ANDcase           (      f1 f1 --> :: <Ob1> ;                )
                  (      f1 f2 --> :: <Ob2> <Rest> ;      )
ANDNOTcase        (      f1 f1 --> :: <Ob2> <Rest> ;      )
                  (      f1 f2 --> :: <Ob1> ;                )
ORcase            (      F F --> :: <Ob2> <Rest> ;          )
                  (      f1 f2 --> :: <Ob1> ;                )
caseTRUE          (      T --> T                        )
                  (      F --> :: <Ob1> <Rest> ;          )
casedrptru        (      ob T --> T                        )
                  (      F --> :: <Ob1> <Rest> ;          )
NOTcaseTRUE       (      T --> :: <Ob1> <Rest> ;          )
                  (      F --> T                        )
caseFALSE         (      T --> F                        )
                  (      F --> :: <Ob1> <Rest> ;          )
casedrpfls        (      ob T --> F                        )
                  (      F --> :: <Ob1> <Rest> ;          )
case2drpfls       ( ob1 ob2 T --> F                        )
                  (      F --> :: <Ob1> <Rest> ;          )

```



## Binary tests:

```

#0=?SEMI      (      #0 --> :: ; )
               (      # --> :: <Ob1> <Rest> ; )
#0=?SKIP      (      #0 --> :: <Ob2> <Rest> ; )
               (      # --> :: <Ob1> <Rest> ; )
#0=ITE        (      #0 --> :: <Ob1> <Ob3> <Rest> ; )
               (      # --> :: <Ob2> <Rest> ; )
DUP#0=IT      (      #0 --> :: #0 <Ob1> <Rest> ; )
               (      # --> :: # <Ob2> <Rest> ; )
DUP#0=ITE     (      #0 --> :: #0 <Ob1> <Ob3> <Rest> ; )
               (      # --> :: # <Ob2> <Rest> ; )

#1=?SKIP
#=?SKIP
#>?SKIP
#=ITE         (      #m #m --> :: <Ob1> <Ob3> <Rest> ; )
               (      #m #n --> :: <Ob2> <Rest> ; )
#<ITE        (      #m #n --> :: <Ob1> <Ob3> <Rest> ; )
               (      #n #m --> :: <Ob2> <Rest> ; )
#>ITE        (      #m #n --> :: <Ob2> <Rest> ; )
               (      #n #m --> :: <Ob1> <Ob3> <Rest> ; )
#=case        (      #m #m --> :: <Ob1> ; )
               (      #m #n --> :: <Ob2> <Rest> ; )
#=casedrop    (      #m #m --> :: <Ob1> ; )
               (      #m #n --> :: #m <Ob2> <Rest> ; )
#=casedrpf1s  (      #m #m --> FALSE )
               (      #m #n --> :: #m <Ob1> <Rest> ; )
OVER#=case    (      #m #m --> :: #m <Ob1> ; )
               (      #m #n --> :: #m <Ob2> <Rest> ; )
#<>case       (      #m #m --> :: <Ob2> <Rest> ; )
               (      #m #n --> :: <Ob1> ; )
#<case        (      #m #n --> :: <Ob1> ; )
               (      #n #m --> :: <Ob2> <Rest> ; )
#>case        (      #m #n --> :: <Ob2> <Rest> ; )
               (      #n #m --> :: <Ob1> ; )
#0=case       (      #0 --> :: <Ob1> ; )
               (      # --> :: <Ob2> <Rest> ; )
DUP#0=case    (      #0 --> :: #0 <Ob1> ; )
               (      # --> :: # <Ob2> <Rest> ; )
DUP#0=csdrp   (      #0 --> :: <Ob1> ; )
               (      # --> :: # <Ob2> <Rest> ; )
DUP#0=csDROP  (      #0 --> :: ; )
               (      # --> :: # <Ob1> <Rest> ; )
#1=case       (      #1 --> :: <Ob1> ; )
               (      # --> :: <Ob2> <Rest> ; )
#>2case       (      #0/#1/#2 --> :: <Ob2> <Rest> ; )
               (      # --> :: <Ob1> ; )

```

## Real tests:

```

num0=case      (      0 --> :: 0 <Ob1> ; )
               (      n --> :: n <Ob2> <Rest> ; )

```

0 means %0 or C%0. Similar syntax for following words:

|            |          |       |           |
|------------|----------|-------|-----------|
|            | %0=case  | 5F13B | C%0=case  |
| num1=case  | %1=case  | 5F19F | C%1=case  |
| num2=case  | %2=case  | 5F208 | C%2=case  |
| num-1=case | %-1=case | 5F285 | C%-1=case |

```

j%0=case      (      %0 --> :: <Ob1> ;                      )
               (      % --> :: <Ob2> <Rest> ;                )
5EEDB REALNEGcase (      %<0 --> :: % <Ob1> ;                  )
               (      ob --> :: % <Ob2> <Rest> ;              )

```

#### General Object tests:

```

nonopcase      (      seco --> :: seco <Ob1> ;              )
               (      ob --> :: seco <Ob2> <Rest> ;          )
idntcase       (      id --> :: id <Ob1> ;                  )
               (      ob --> :: ob <Ob2> <Rest> ;            )
dIDNTNcase     (      id --> :: id <Ob2> <Rest> ;            )
               (      ob --> :: ob <Ob1> ;                  )
idntlamcase    (      id/lam --> :: id <Ob1> ;               )
               (      ob --> :: ob <Ob2> <Rest> ;            )
REALcase       (      % --> :: <Ob1> ;                      )
               (      ob --> :: <Ob2> <Rest> ;                )
dREALNcase     (      % --> :: % <Ob2> <Rest> ;              )
               (      ob --> :: ob <Ob1> ;                  )
dARRYcase      (      [] --> :: [] <Ob1> ;                  )
               (      ob --> :: ob <Ob2> <Rest> ;            )
dLISTcase      (      {} --> :: {} <Ob1> ;                  )
               (      ob --> :: ob <Ob2> <Rest> ;            )
18E45 DNOTSYMB?SEMI (      symb --> :: symb <Ob1> <Rest> ; )
               (      ob --> :: ob ;                        )
NOTLISTcase
NOTSECOcase
NOTROMPcase
27224 NOTBAKcase
27234 NOTLIBcase
5EDFC DUPARF?case ( If % C% [] or lnkarry then COLA else SKIP )

```

```

EQIT           (      ob1 ob1 --> :: <Ob1> <Rest> ;          )
               (      ob1 ob2 --> :: <Ob2> <Rest> ;          )
EQITE          (      ob1 ob1 --> :: <Ob1> <Ob3> <Rest> ;    )
               (      ob1 ob2 --> :: <Ob2> <Rest> ;          )
EQcase         (      ob1 ob1 --> :: ob1 <Ob1> ;            )
               (      ob1 ob2 --> :: ob1 <Ob2> <Rest> ;      )
EQcasedrop     (      ob1 ob1 --> :: <Ob1> ;                )
               (      ob1 ob2 --> :: ob1 <Ob2> <Rest> ;      )
jEQcase        (      ob1 ob1 --> :: <Ob1> ;                )
               (      ob1 ob2 --> :: <Ob2> <Rest> ;          )
EQUALcase      (      ob1 ob1 --> :: <Ob1> ;                )
               (      ob1 ob2 --> :: <Ob2> <Rest> ;          )
EQUALcasedrp   (ob ob1 ob1 --> :: <Ob1> ;                    )
               (      ob1 ob2 --> :: <Ob2> <Rest> ;          )
EQUALNOTcase   (      ob1 ob1 --> :: <Ob2> <Rest> ;          )
               (      ob1 ob2 --> :: <Ob1> ;                )

```

#### Miscellaneous:

```

UserITE        (      #set --> :: <Ob1> <Ob3> <Rest> ;      )
               (      #clr --> :: <Ob2> <Rest> ;            )
SysITE         (      #set --> :: <Ob1> <Ob3> <Rest> ;      )
               (      #clr --> :: <Ob2> <Rest> ;            )

```

```

caseDoBadKey aka caseDEADKEY
    (          T --> :: DoBadKey ;          )
    (          F --> :: <Ob1> <Rest> ;      )
caseDrpBadKy  (      ob T --> :: DoBadKey ;      )
    (          F --> :: <Ob1> <Rest> ;      )
caseERRJMP
caseSIZEERR
NcaseSIZEERR
NcaseTYPEERR

```

\*\*\*\*\*

## 20. Temporary Variables and Environments

~~~~~

```

074E4  BIND          ( obn ... obl { lamn ... lam1 } --> )
      DOBIND       ( obn ... obl lamn ... lam1 #n --> )
      ABND         ( --> )
      CACHE        ( obn ... obl #n lam --> )
                  Binds all objects under the same name.
                  1LAM has count.
      DUMP         ( NULLLAM --> obl ... obn #n )
                  Inverse of CACHE. Always does garbage collection.
      SAVESTACK    Caches stack to SAVELAM
      undo         Dumps SAVELAM

      @LAM         ( lam --> ob TRUE / FALSE )
      STOLAM       ( ob lam --> )
02FD6  DoLam       ( lam --> ob )
                  Recall lam contents, Undefined Lam error if not found
0791E  @LAM1      ( lam --> ob TRUE / FALSE )      ( GX: 078E9 )
                  @LAM for 1st env only
0792A  @LAMN      ( lam #n --> ob TRUE / FALSE )   ( GX: 078F5 )
                  @LAM for nth env only

      GETLAM ( #n --> ob )          PUTLAM ( ob #n --> )
      1GETLAM ( --> ob )          1PUTLAM ( ob --> )
      2GETLAM          2PUTLAM
      3GETLAM          3PUTLAM
      ...             ...
      22GETLAM         22PUTLAM
                  DUP4PUTLAM

      1LAMBIND       ( ob --> )
      DUP1LAMBIND    ( ob --> ob )
      1ABNDSWAP      ( ob --> 1lamob ob )
      1GETABND       ( --> 1lamob )
      1GETSWAP       ( ob --> 1lamob ob )      ( No ABND )
      2GETEVAL       ( --> ? )                ( :: 2GETLAM EVAL ; )

      GETLAMPAIR     ( #n --> #n ob lam FALSE / #n TRUE )
                  Get lam contents & name ( 10 = 1lam, 20 = 2lam ... )
      DUPTEMPENV     ( --> )
                  Duplicates topmost tempenv (clears protection word)

      NULLLAM        ( --> NULLLAM )
      1NULLLAM{ }    ( --> { } )
36D5A  2NULLLAM{ }    ( --> { } )      ( GX: 37DB9 )
36D82  3NULLLAM{ }    ( --> { } )      ( GX: 37B17 )
      4NULLLAM{ }    ( --> { } )
3306C  7NULLLAM{ }    ( --> { } )
49199  8NULLLAM{ }    ( --> { } )      ( GX: 10E36 )

```

21. Variables and Directories

~~~~~

Conversion words:

```
05AED  ID>LAM      ( id --> lam )
05B01  LAM>ID      ( lam --> id )
      $>ID        ( $ --> id )
      ID>$        ( id/lam --> $ )
```

### 21.1 Temporary memory

\*\*\*\*\*

```
TOTEMPOB      ( ob --> ob' )
               Copies ob into TEMPOB and returns pointer to
               the new copy.

TOTEMPSWAP     ( ob1 ob2 --> ob2' ob1 )
CKREF          ( ob --> ob' )
               If ob is in TEMPOB, is not embedded in a composite,
               and is not referenced, the does nothing. Else copies
               ob to TEMPOB and returns the copy.
               ( :: INTERNOTREF? ?SEMI TOTEMPOB ; )

SWAPCKREF      ( ob1 ob2 --> ob2' ob1 )
INTERNOTREF?   ( ob --> ob flag )
               If the input object is in TEMPOB area,
               is not embedded in a composite object,
               and is not referenced, returns ob and
               TRUE, otherwise returns ob and FALSE.

06B3E  INTemp?  ( ob --> ob flag )
               Is ob in tempob area, not in composite?

065D9  PTRREF?  ( ob --> ob flag )
               Is object referenced?

065E5  REFERENCED? ( ob --> ob flag )
               Is object referenced or in composite?

06BC2  NOTREF?  ( ob --> ob flag )
               Object not referenced, not in composite?

06DDE  TOPTEMPOB ( ob --> ob' )

064BD  UNREFOB   ( ob --> ob' ob )
               Makes a standalone copy by moving references to
               a new copy.
               :: DUP TOTEMPSWAP UNREFINTemp ;

064D6  UNREFTEMP ( ob1 ob2 --> ob1 ob' )
               Move references from ob2 to ob1 (ob1 in tempob area)

064E2  UNREFINTemp ( ob1 ob2 --> ob1 ob' )
               Move references from ob2 to ob1 (ob1 in tempob area)
               References to ob2 body are moved too.
```

## 21.2 Storing and Recalling Variables

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```
@          ( id/lam --> ob TRUE / FALSE )
DUP@       ( id/lam --> id/lam ob TRUE / FALSE )
STO        ( ob id/lam --> )
For ids:
    Assumes ob is not PCO.
    If replacing some object that object is copied
    to TEMPOB first and pointers are updated.
For lams:
    Errors if lam is unbound.
SAFE@      ( id/lam --> ob TRUE / FALSE )
For lams @.
For ids does ?ROMPTR> to the ob found.
DUPSAFE@   ( id/lam --> id/lam ob TRUE / id/lam FALSE )
XEQRCL     ( id --> ob )
Same as SAFE@ but errors is variable is not found.
Works for lams too, but you get wrong error.
LISTRCL    ( { path id } --> ob )
SAFESTO     ( ob id/lam --> )
For lams STO.
For ids does ?>ROMPTR to ob before STO.

SAFE@_HERE ( id/lam --> ob TRUE / FALSE )
Same as SAFE@, but for ids searches only current rrp.
XEQRCLID1  ( id/lam --> id/lam ob )
SAFE@_HERE with UNDEFIDERR if no contents
XEQRCLID2  ( id/lam ob1 --> id/lam ob1 ob )
XEQRCLID3  ( id/lam ob2 ob1 --> id/lam ob2 ob1 ob )
XEQSTOID   ( ob id/lam --> )      aka ?STO_HERE
Sames as SAFESTO, but will only store to current
directory and will not overwrite a stored rrp.

5E5B7      @OBNOT      ( ob --> ob' FALSE / ob TRUE )
Recalls object contents for if,lam,ROMPTR. For other
types returns TRUE.
5E55E      SAFE@NOT    ( id --> ob FALSE / id TRUE )
5E602      @LAMNOT     ( lam --> ob FALSE / lam TRUE )
5E616      ROMPTR@NOT  ( ROMPTR --> ob FALSE / ROMPTR TRUE )
```

## 21.3 Directories

^^^^^^^^^^^^^^^^^^^^

```
CREATEDIR   ( id --> )
Creates an empty directory in the current directory.
Does ?PURGE_HERE first to delete the original
08E0F      !CREATEDIR ( id --> )
Doesn't check if the same name is already used
( :: #7FF CRDIR# SWAP CREATE ; )
07819      CRDIR#     ( #libnum --> rrp )      ( GX: 077E4 )
Creates an empty directory.
LASTRAM-WORD ( rrp --> ob TRUE / FALSE )
Recalls first object in rrp.
PREVRAM-WORD ( ob --> ob' TRUE / FALSE )
Recalls next object in rrp.
LastNonNull ( rrp --> ob TRUE / FALSE )
Recalls first object in rrp (not null named)
PrevNonNull ( ob --> ob' TRUE / FALSE )
Recalls next ob in rrp (not null named)
```

|       |              |                                                                                                                                                                                                                                                                            |
|-------|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | CkNonNull    | ( ob --> ob TRUE / FALSE )<br>Checks that variable (ob) has a name                                                                                                                                                                                                         |
|       | RAM-WORDNAME | ( ob --> id )<br>Recalls name of the object in current directory                                                                                                                                                                                                           |
|       | CREATE       | ( ob id --> )<br>Creates a RAM-WORD in the current directory.<br>Errors if ob is/contains current directory.<br>Assumes ob is not PCO.                                                                                                                                     |
|       | REPLACE      | ( newob oldob --> newob )<br>Replaces oldob with newob in memory.<br>Much faster than doing STO.                                                                                                                                                                           |
|       | PURGE        | ( id --> )<br>Purge variable. Does no type check, use XEQPGDIR for<br>directories instead.                                                                                                                                                                                 |
|       | XEQPGDIR     | ( id --> )<br>Purges a directory. Checks references etc first.                                                                                                                                                                                                             |
|       | XEQORDER     | ( { id1 id2.. } --> )<br>Orders context.                                                                                                                                                                                                                                   |
|       | DOVARS       | ( --> { id1 id2 .. } )<br>Returns list of variable names from context.                                                                                                                                                                                                     |
| 1867F | DODIRPRG     | ( :: TrueTrue DODIRPRG ; )<br>( any pred --> {} )<br>Executed pred on all directory variables.<br>At execution: any pred id_contents {} id<br>To be returned: any pred id_contents {} ob TRUE/FALSE<br>If TRUE is returned ob is added with >TCOMP, else it<br>is dropped. |
|       | PATHDIR      | ( --> { HOME dir dir .. } )<br>Returns current path                                                                                                                                                                                                                        |
|       | UPDIR        | ( --> )<br>Switches context to parent directory.                                                                                                                                                                                                                           |
| 08309 | GETUPDIR     | ( rrp --> rrp' TRUE / FALSE )<br>Get parent directory                                                                                                                                                                                                                      |
| 08DD4 | HOMEDIR?     | ( rrp --> flag )<br>Is rrp HOME directory?                                                                                                                                                                                                                                 |

#### Directory Pointers:

|       |                        |                                                                                                               |
|-------|------------------------|---------------------------------------------------------------------------------------------------------------|
|       | CONTEXT!               | ( rrp --> )<br>Sets new current directory.                                                                    |
|       | CONTEXT@               | ( --> rrp )<br>Recalls current directory.                                                                     |
| 08D4A | LCONTEXT!              | ( rrp --> )<br>Sets last directory.                                                                           |
| 08D82 | LCONTEXT@              | ( --> rrp )<br>Recalls last directory.                                                                        |
|       | HOMEDIR,<br>SYSCONTEXT | ( --> )<br>Sets HOME as current directory.                                                                    |
| 08DC4 | SYSLCONTEXT            | ( --> )<br>Sets HOME as last directory.                                                                       |
|       | SaveVarRes             | ( --> )<br>Binds CONTEXT and LCONTEXT to 2 NULLLAMs                                                           |
|       | RestVarRes             | ( --> )<br>Restores CONTEXT and LCONTEXT from 1&2LAMs (+ABND)<br>Sets home to both first (in case of errors?) |

## 21.4 Hidden Directory

^^^^^^^^^^^^^^^^^^^^

|              |                                                  |
|--------------|--------------------------------------------------|
| RclHiddenVar | ( id --> ob TRUE / FALSE )                       |
|              | Recalls hidden variable ( :: WithHidden @ ; )    |
| StoHiddenVar | ( ob id --> )                                    |
|              | Stores hidden variable ( :: WithHidden STO ; )   |
| PuHiddenVar  | ( id --> )                                       |
|              | Purges hidden variable ( :: WithHidden PURGE ; ) |
| WithHidden   | ( --> ? )                                        |
|              | Executes next command in hidden directory        |
| SetHiddenRes | ( --> )                                          |
|              | Sets HiddenDir to context and last context       |



\*\*\*\*\*

## 22. Display Management

~~~~~

22.1 Display Organization

^^

Pointer	Grob	Location
-----	-----	-----
HARDBUFF2 ->	+-----+ Menu labels	(Low Mem)
ABUFF ->	+-----+ text grob	
GBUFF ->	+-----+ graph grob	(Hi Mem)
	+-----+	

The text grob and graph grob may be enlarged, and may be scrolled.
VDISP contains pointer to current display grob.

	TOADISP	Choose text display	
	TOGDISP	Choose graphic display	
13167	GDISPON?	(--> flag)	
	ABUFF	(--> textgrob)	
	GBUFF	(--> graphgrob)	
	HARDBUFF	(--> HBgrob)	Current display grob
	HARDBUFF2	(--> menugrob)	
	HBUFF_X_Y	(--> HBgrob #x1 #y1)	
12B58	HBUFFDIMw	(--> #width)	
12B6C	HBUFFDIMh	(--> #height)	
	GBUFFGROBDIM	(--> #height #width)	
		Returns dimensions of graph grob	

Display Regions:

	+-----+	
DA1	directory time	Status line (16 lines)
	+-----+	
	4:	
DA2a	3:	Stack
	+-----+	Display
DA2b	2:	(40 lines total)
	1:	
	+-----+	
DA3	1 2 3 4 5 6	Menu labels (8 lines)
	+-----+	

Display area 2 is actually divided into areas 2a and 2b, a distinction most often used by the command line line. The boundary between 2a and 2b can move, but the overall sizes of areas 1, 2, and 3 are fixed.

22.2 Preparing the Display

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```
RECLAIMDISP      ( --> )
                  Chooses text grob, clears it, sets standard size
ClrDA1IsStat     ( --> )
                  Suspends ticking clock display
TURNMENUOFF      Turn menu grob off, enlarges ABUFF to fill screen.
TURNMENUON       Turn menu grob on
MENUOFF?         Returns TRUE if menu grob is off
130CA  RSZVDISP   Set standard size for VDISP
1297D  BROADENHbuff ( #cols --> )
12964  HEIGHTENHbuff ( #rows --> )
12BB7  BROADENGROB ( GDISP/ABUFF #cols --> )
        HEIGHTENGROB ( GDISP/ABUFF #rows --> ) (Unless grob is null)
13043  KILLPDISP
        KILLGDISP
```

Suggested usage in own applications:

```
::
    ClrDA1IsStat
    RECLAIMDISP TURNMENUOFF
    <application>
    ClrDAsOK                      ( Tell the 48 to redraw lcd )
    or
    SetDAsTemp                    ( Freeze all display areas )
;
```

See "Display Refresh" for more "DA" words.

22.3 Clearing the Display

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```
BLANKIT          ( #startrow #rows )
                  Clears #rows from HARDBUFF
CLEARVDISP       ( --> ) Clear HARDBUFF
BlankDA1         ( --> ) Clears status area from HARDBUFF
BlankDA2         ( --> ) Clears rows 16-56 from HARDBUFF
BlankDA12        ( --> ) Clears rows 0-56 from HARDBUFF
BlankDA2a        ( --> ) Clears area DA2a (depends on command line)
Clr16            ( --> ) Clear top 16 rows.
Clr8             ( --> ) Clear top 8 rows.
Clr8-15          ( --> ) Clear 2nd status line.
CLCD10           ( --> ) Clear status & stack area
CLEARLCD         ( --> ) Clear entire display
```

22.4 Displaying Text

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

When using text grob and display has not been scrolled.
Strings are fixed to 22 chars by adding spaces or cutting the end
and adding "..."

5x7 font:

5x9 font:

DISPROW1, DISP@01	(\$ -->)	BIGDISPROW1
DISPROW2, DISP@09	(\$ -->)	BIGDISPROW2
DISPROW3, DISP@17	(\$ -->)	BIGDISPROW3
DISPROW4, DISP@25	(\$ -->)	BIGDISPROW4
DISPROW5	(\$ -->)	
DISPROW6	(\$ -->)	

```

DISPROW7          ( $ --> )
DISPROW8          ( $ --> )
DISPN             ( $ #row --> )  BIGDISPN

Disp5x7           ( $ #startrow #maxrows --> )
                  Displays $ that spans several lines.
                  Too long lines are cut and ... is appended.

DISPROW1*         ( $ --> )      Displays relative to window corner.
DISPROW2*         ( $ --> )      Displays relative to window corner.
0E029 DISPROW1*!   ( $ --> )      = Clr8 + DISPROW1*

```

Displaying Messages:

Flash-words save the overwritten part and restore it after a delay loop.

```

DISPSTATUS2      ( $ --> )      Display in status area in 2 lines
FlashMsg         ( $ --> )      Flash message in status area
FlashWarning     ( $ --> )      Flash message & beep
38908 DispWarning ( $ --> )      Display message, beep & freeze

0E047 Save16      ( --> grob )    Save top 16 rows.
0E05B Restore16   ( grob --> )   Restore top 16 rows.

```

22.5 Window Coordinates ^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

TOP16             ( --> HBgrob #x1 #y1 #x1+131 #y1+16 )
TOP8              ( --> HBgrob #x1 #y1 #x1+131 #y1+8 )
Rows8-15          ( --> HBgrob #x1 #y1+8 #x1+131 #y1+16 )
WINDOWCORNER      ( --> #x #y )   Gets corner coordinates
TOPROW            ( --> #y )      Gets y-coordinate of top row
BOTROW            ( --> #y )      Gets y-coordinate of bottom row
LEFTCOL           ( --> #x )      Gets x-coordinate of left column
RIGHTCOL          ( --> #x )      Gets x-coordinate of right column

0E047 Save16      ( --> grob )    Save top 16 rows.
0E05B Restore16   ( grob --> )   Restore top 16 rows.

HBUFF_X_Y         ( --> HBgrob #x1 #y1 )
WINDOWXY          ( #x #y --> )   Sets corner coordinates
REDISPABUFF       ( --> )         Sets 0 0 window coordinates

```

With repeat:	Without repeat:	Testing:	Jumping to edge:
SCROLLDOWN	WINDOWDOWN	WINDOWBOT?	JUMPBOT
SCROLLLEFT	WINDOWLEFT	WINDOWLEFT?	JUMPLEFT
SCROLLRIGHT	WINDOWDOWN	WINDOWRIGHT?	JUMPRIGHT
SCROLLUP	WINDOWRIGHT	WINDOWTOP?	JUMPTOP

(Testing done against 7LAM for x- and 8LAM for y-directions)

22.6 Annunciator Control ^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

SetAlphaAnn       ClrAlphaAnn
SetRightAnn       ClrRightAnn
SetLeftAnn        ClrLeftAnn
11320 ClrPrgmAnn

```

	PrgmEntry?		AlgEntry?	11501	Lock?
	SetPrgmEntry		SetAlgEntry	11543	SetLock
1155C	ClrPrgmEntry	53984	ClrAlgEntry	1156C	ClrLock

ImmedEntry? (Not PrgmEntry? nor AlgEntry?)

LockAlpha (SetLock + SetKeysANS)

UnLockAlpha (ClrLock + SetKeysNS)

22.7 Display Refresh

^^^^^^^^^^^^^^^^^^^^

	DA1OK?	(DA1Bad? -> F, DA1ValidF? -> T DA1TempF? -> DA1BadT, DA1NoCh? -> T else F)
38DFC	DA2aOK?	(DA2aBad? -> F, DA2aValidF? -> T DA2aTempF? -> DA2aBadT, DA2aNoCh? -> T else F)
38E4C	DA2bOK?	(DA2bBad? -> F, DA2bValidF? -> T DA2bTempF? -> DA2bBadT, DA2bNoCh? -> T else F)
38E9C	DA2OK?	(DA2aOK? and DA2bOK?)
	DA3OK?	(DA3Bad? -> F, DA3ValidF? -> T DA3TempF? -> DA3BadT, DA3NoCh? -> T else F)
38F05	DA3OK?	(DA1OK? and DA2OK? and DA3OK?)
	MENoP&FixDA1	(SetDA1Bad)
	MENP&FixDA12	(SetDA1Bad SetDA2aBad SetDA2bBad)

38DE8	SetDA1BadT	(SetDA1Bad TRUE)
38E38	SetDA2aBadT	(SetDA2aBad TRUE)
38E88	SetDA2bBadT	(SetDA2bBad TRUE)
38EF1	SetDA3BadT	(SetDA3Bad TRUE)

DA1OK?NOTIT (SKIP if OK)
DA2aOK?NOTIT
DA2bOK?NOTIT
DA3OK?NOTIT

38F8C	InitDispModes	(SetDAsBad ClrDA2RollF ClrNoRollDA2)
38FB9	DA2RollReq?	(DA2RollF? AND NOT(Da2aBad?))
	SetDA1Valid	(SetDA1ValidF ClrDA1Bad ClrDA1IsStat)
	SetDA2aValid	(SetDA2aValidF Clr DA2aBad)
	SetDA2bValid	(SetDA2bValidF ClrDA2bBad Clr_70683:4)
	SetDA3Valid	(SetDA3ValidF ClrDA3Bad)
	SetDA2Valid	(SetDA2aValid SetDA2bValid)
39171	SetDAsValid	(SetDA1Valid SetDA2Valid SetDA3Valid)
	SetDA1Temp	(SetDA1TempF ClrDA1Bad ClrDA1IsStat)
39045	SetDA2aTemp	(SetDA2aTempF ClrDA2aBad)
39059	SetDA2bTemp	(SetDA2bTempF ClrDA2bBad Clr_70683:4)
39072	SetDA3Temp	(SetDA3TempF ClrDA3Bad)
	SetDA2OKTemp	(SetDA2aTemp SetDA2bTemp)
	SetDA12Temp	(SetDA1Temp SetDA2OKTemp)
	SetDAsTemp	(SetDA1Temp SetDA2OKTemp SetDA3Temp)

39086	?SetEditRoll	(EditExst?NOT ITE SetDA2RollF SetDA2aNoCh)
	MENoP&FixDA1	(SetDA1Bad)
	MENP&FixDA12	(SetDA1Bad SetDA2aBad SetDA2bBad)
	ClrDA1OK	(ClrDA1ValidF ClrDA1TempF ClrDA1NoCh)
390E5	ClrDA2aOK	(ClrDA2aValidF ClrDA2aTempF ClrDA2aNoCh)
390FE	ClrDA2bOK	(ClrDA2bValidF ClrDA2bTempF ClrDA2bNoCh)
	ClrDA2OK	(ClrDA2aOK ClrDA2bOK)
	ClrDA3OK	(ClrDA3ValidF ClrDA3TempF ClrDA3NoCh)
	ClrDAsOK	(ClrDA1OK ClrDA2OK ClrDA3OK)
393C5	DA1NoCh?	39275 DA1ValidF?
393D3	SetDA1NoCh	39283 SetDA1ValidF
393E1	ClrDA1NoCh	39291 ClrDA1ValidF
393EF	DA2aNoCh?	3929F DA2aValidF?
393FD	SetDA2aNoCh	392AD SetDA2aValidF
3940B	ClrDA2aNoCh	392BB ClrDA2aValidF
39419	DA2bNoCh?	392C9 DA2bValidF?
39427	SetDA2bNoCh	392D7 SetDA2bValidF
39435	ClrDA2bNoCh	392E5 ClrDA2bValidF
39443	DA3NoCh?	392F3 DA3ValidF?
	SetDA3NoCh	SetDA3ValidF
3945F	ClrDA3NoCh	3930F ClrDA3ValidF
	SetDA2NoCh	
	SetDA12NoCh	
391B2	SetDA23NoCh	
391C6	SetDA13NoCh	
391DA	SetDA12a3NoCh	
	SetDA123NoCh	
3946D	DA1Bad?	3931D DA1TempF?
	SetDA1Bad	3932B SetDA1TempF
39489	ClrDA1Bad	39339 ClrDA1TempF
	DA2aBad?	39347 DA2aTempF?
	SetDA2aBad	39355 SetDA2aTempF
	ClrDA2aBad	39363 ClrDA2aTempF
394C1	DA2bBad?	39371 DA2bTempF?
394CF	SetDA2bBad	3937F SetDA2bTempF
394DD	ClrDA2bBad	3938D ClrDA2bTempF
394EB	DA3Bad?	3939B DA3TempF?
394F9	SetDA3Bad	393A9 SetDA3TempF
	ClrDA3Bad	393B7 ClrDA3TempF
39248	DAsBad?	
39515	DA1IsStat?	
	SetDA1IsStat	
	ClrDA1IsStat	
3953F	DA2bEdit?	
3954D	SetDA2bEdit?	
3955B	ClrDA2bEdit?	
39569	NoRollDA2?	
	SetNoRollDA2	
3958B	ClrNoRollDA2	

23. Graphic Objects

~~~~~

### 23.1 Dimensions

^^^^^^

```
CKGROBFITS      ( grobl grob2 #n #m --> grobl grob2' #n #m )
                  Shrinks grob2 if it doesn't fit into grobl
GROBDIM          ( grob --> #height #width )
DUPGROBDIM      ( grob --> grob #height #width )
GROBDIMw        ( grob --> #width )
```

### 23.2 Grob Handling

^^^^^^

```
GROB!           ( grobl grob2 #col #row --> )
                  Stores grobl into grob2
GROB!ZERO       ( grob #x1 #y1 #x2 #y2 --> grob' )
                  Blanks a rectangular region of grob
GROB!ZERODRP    ( grob #x1 #y1 #x2 #y2 --> )
                  Blanks a rectangular region of grob
                  Assumes text or graph grob.
XYGROBDISP      ( #row #col grob --> )
                  Store grob into hardbuff. Expands if necessary.
GROB>GDISP      ( grob --> )
                  Stores new graph grob
DOLCD>          ( --> grob )
                  Current display as a 64x131 grob.
DO>LCD          ( grob --> )
                  Grob to display
DOCLLCD         ( --> )
                  Clears display

INVGROB         ( grob --> grob' )
SUBGROB         ( grob #x1 #y1 #x2 #y2 --> grob' )
MAKEGROB        ( #x #y --> grob )      Makes a blank grob

PIXON           ( #x #y --> )           Set pixel in text grob
PIXOFF          ( #x #y --> )           Clear text grob pixel
PIXON?          ( #x #y --> flag )      text grob pixel on?
PIXON3          ( #x #y --> )           Set pixel in graph grob
PIXOFF3         ( #x #y --> )           Clear graph grob pixel
PIXON?3         ( #x #y --> flag )      graph grob pixel on?

LINEON          ( #x1 #y1 #x2 #y2 --> )
                  Draws a line to text grob.
                  Note: For all line drawing commands #x2 must be > #x1
                  (Use ORDERXY#)
LINEOFF         ( #x1 #y1 #x2 #y2 --> )
                  Clears a line in text grob.
LINEON3         ( #x1 #y1 #x2 #y2 --> )
                  Draws a line in graph grob.
LINEOFF3        ( #x1 #y1 #x2 #y2 --> )
                  Clears a line in graph grob.
TOGLINE         ( #x1 #y1 #x2 #y2 --> )
                  Toggles a line of pixels in text grob.
TOGLINE3        ( #x1 #y1 #x2 #y2 --> )
                  Toggles a line of pixels in graph grob.
```

```

ORDERXY#      ( #x1 #y1 #x2 #y2 --> #x1' #y1' #x2' #y2' )
               Orders two points for line drawing.

$>BIGGROB     ( $ --> grob ) ( 5x9 font )
$>GROB        ( $ --> grob ) ( 5x7 font )
$>grob        ( $ --> grob ) ( 3x5 font )

RIGHT$3x6     ( $ #n --> flag grob )
               Make grob of n rightmost rows
CENTER$3x5    ( grob #x #y $ #?? --> grob )

Symb>HBuff    ( symb --> )
               Displays symb in HARDBUFF in Equation Writer from.
               Enlarges HARDBUFF if necessary.

12996  SCREENupN      ( #n --> )
               Moves stack display up one line. ( 1 < n < 4 )
12A4A  SCREENdnN      ( #n --> )
12A0D  Screenup       ( --> )
               Moves stack display up one line ( 5x7 )
12AF6  Screendn       ( --> )

```

### 23.3 Built-in Grobs

^^^^^^^^^^^^^^^^^^^^

```

CURSOR1      5x9 Insert Cursor      (arrow)
CURSOR2      5x9 Replace Cursor     (solid box)
SmallCursor  3x5 Cursor             (box outlines)
MediumCursor 5x7 Cursor             (box outlines)
BigCursor    5x9 Cursor             (box outlines)
CROSSGROB    5x5 Cross Cursor       ("+" symbol)
MARKGROB     5x5 Mark Cursor        ("X" symbol)
505B2  NULLGROB 0x0 Cursor
39B2D  LineGrob  131x2 Line          (Status area divider)
3A337  StdLabelGrob 21x8 Box        (Normal menu key)
3A399  BoxLabelGrob 21x8 Box        ("Flagged" menu key)
3A3FB  DirLabelGrob 21x8 Box        (Directory menu key)
3A45D  InvLabelGrob 21x8 Box        (Solver menu key)

```

\*\*\*\*\*

## 24. Menu Display

~~~~~

24.1 General

^^^^^^^^

```
SysMenuCheck      Checks menu validity
                  If DA3NoCh? then nothing
                  If Track? then ?DoTrackAct@
                  If Rebuild? then SetThisRow

?DispMenu         :: KEYINBUFFER? NOTcase DispMenu SetDA3Bad ;
DispMenu          :: DispMenu.1 SetDA3Valid ;
DispMenu.1        Turns menu grob on, displays menu
```

Better use this so that all flags are checked:
:: DA3OK?NOTIT ?DispMenu ;

Internal menus: (SX Only!!!)

~~~~~

|        |      |               |                                 |
|--------|------|---------------|---------------------------------|
| 3B234  | {}   | Menus         | All internal menus in a list    |
| Addr   | Type | Label         | Special settings                |
| 3B239  | ::   | CstMenu       | ???????????????                 |
| 3F6D8  | ::   | VarMenu       | ???????????????                 |
| 3B284  | ::   | MathMenu      | SetDirMenu SetDirKeys           |
| 3B298  | {}   | PartsMenu     |                                 |
| 3B2BB  | {}   | ProbMenu      |                                 |
| 3B2DC  | {}   | HypMenu       |                                 |
| 3B2FB  | {}   | MatrixMenu    |                                 |
| 3B31E  | {}   | VectorMenu    |                                 |
| 3B341  | {}   | BaseMenu      |                                 |
| 3B542  | ::   | PrgmMenu      | SetDirMenu SetDirKeys           |
| 3B556  | {}   | StkMenu       |                                 |
| 3B575  | {}   | ObjMenu       |                                 |
| 3B594  | {}   | DsplMenu      |                                 |
| 3B5B5  | {}   | CtrlMenu      |                                 |
| 3B5D6  | {}   | BrchMenu      |                                 |
| 3B5F7  | {}   | TestMenu      |                                 |
| 3B972  | {}   | PrintMenu     |                                 |
| 3B9A4  | {}   | IOMenu        |                                 |
| 3E71A  | {}   | SetupMenu     | Sets ReviewIO, SetFixDA1Exit    |
| 3BB46  | {}   | ModeMenu      |                                 |
| 3BC8D  | {}   | CustomMenu    |                                 |
| 3BCE7  | {}   | MemoryMenu    |                                 |
| 3BD46  | {}   | AritMenu      |                                 |
| 3F376  | ::   | LibraryMenu   | ???                             |
| 3BD82  | ::   | Port0Menu     | SetRebuild ' Port0Build         |
| 3BDAA  | ::   | Port1Menu     | SetRebuild ' Port1Build         |
| 3BDD2  | ::   | Port2Menu     | SetRebuild ' Port2Build         |
| 3BDFA* | {}   | EditMenu      |                                 |
| 3BE22  | ::   | SolveMenu     | Sets SolveReview, SetExitRedraw |
| 3BE2C  | {}   | SolvrMenu     | [SolvrMenuInit]                 |
| 15200  | ::   | SolvrMenuInit | ??????????                      |
| 3BEB8  | ::   | PlotMenu      | Sets PtypeKeysNS, SetFixDA1Exit |
| 3BEC2  | {}   | PlotrMenu     | [PlotrMenuInit]                 |
| 3BF21  | {}   | PtypeMenu     | [PtypeMenuInit]                 |



|       |    |               |                                               |
|-------|----|---------------|-----------------------------------------------|
| 3C039 | :: | PtypeMenuInit | Sets PtypeKeysNS and MENoP&FixDA1             |
| 3C0AF | :: | PlotrMenuInit | Sets PlotrReview, SetFixDA12Exit              |
| 3C483 | {} | AlgebraMenu   |                                               |
| 3C4C9 | :: | TimeMenu      | Sets TimeReview                               |
| 3C4D3 | {} | TimeSetMenu   | [SetMenuInit] SetTimeMenu                     |
| 3C51A | {} | AdjstMenu     | [AdjstMenuInit] SetTimeMenu                   |
| 3C565 | {} | AlrmMenu      | [AlrmMenuInit] PromptAlarm SetTimeMenu12      |
| 3C671 | {} | AdjstMenuInit |                                               |
| 3C79C | :: | AlrmMenuInit  | Stes ReviewAlrm                               |
| 3C842 | {} | RptMenu       | PromptRpt                                     |
| 3CAA7 | :: | StatMenu      | Sets StatRowAct and StatReview, SetFixDA1Exit |
| 3CD96 | :: | FitMenu       | Sets PtypeKeysNS and MENoP&FixDA1             |
| 3CE65 | :: | UnitsMenu     | SetDirMenu, SetDirKeys                        |
| 3EBF6 | {} | ModlMenu      |                                               |
| 3CE79 | {} | LenghtMenu    | SetUnitKeys                                   |
| 3CE9A | {} | AreaMenu      | SetUnitKeys                                   |
| 3CEBB | {} | VolumeMenu    | SetUnitKeys                                   |
| 3CEDA | {} | UTimeMenu     | SetUnitKeys                                   |
| 3CEFB | {} | SpeedMenu     | SetUnitKeys                                   |
| 3CF1E | {} | MassMenu      | SetUnitKeys                                   |
| 3CF3F | {} | ForceMenu     | SetUnitKeys                                   |
| 3CF62 | {} | EnergyMenu    | SetUnitKeys                                   |
| 3CF83 | {} | PowerMenu     | SetUnitKeys                                   |
| 3CFA4 | {} | PressMenu     | SetUnitKeys                                   |
| 3CFC7 | {} | TempMenu      | SetUnitKeys                                   |
| 3CFE8 | {} | ElecMenu      | SetUnitKeys                                   |
| 3D009 | {} | AngleMenu     | SetUnitKeys                                   |
| 3D02A | {} | LightMenu     | SetUnitKeys                                   |
| 3D04D | {} | RadMenu       | SetUnitKeys                                   |
| 3D061 | {} | ViscMenu      | SetUnitKeys                                   |
| 3DAF2 | {} | UnitsCmdMenu  |                                               |
| 3DE93 | :: | GraphMenu     | NoExitAction                                  |
| 3DEBB | {} | FcnMenu       | NoExitAction                                  |
| 4EE1A | :: | ZoomMenu      |                                               |

LabelDef and MenuKey for each menukey is stored to TOUCHTAB.

Also the executor of menukey presses is stored there.

|          |                     |                          |
|----------|---------------------|--------------------------|
| GETDF    | ( #menukey --> ob ) | ( #1 <= #menukey <= #6 ) |
| STODF    | ( ob #menukey --> ) | ( #1 <= #menukey <= #6 ) |
| GETPROC  | ( #menukey --> ob ) | ( #1 <= #menukey <= #6 ) |
|          | ( #7 --> executor ) |                          |
| STOPPROC | ( ob #menukey --> ) | ( #1 <= #menukey <= #6 ) |
|          | ( executor #7 --> ) |                          |

Menu properties:

~~~~~

MenuDef	Actual menu
MenuKeys	Menu keys in a list
MenuRow	Menu page
LabelDef	Label builder for menu
MenuRowAct	Action taken when menu row changes (Or if LastMenu)
ExitAction	Action taken when menu changes
TrackAct	Action taken if Track? (SysMenuCheck)
ReviewKey	Action taken when REVIEW is pressed
MenuKeysNS	Action taken when menu key is pressed
MenuKeysLS	Action taken when menu key is pressed with left shift
MenuKeysRS	Action taken when menu key is pressed with right shift
BadMenu?	Menu must be redrawn?
Rebuild?	Menurow has changed?

Track? If context has changed is there a prg to execute?

Track? examples:

SolverMenu has DoSolveMenu as TrackAct, because there might be another EQ variable to use.

CstMenu just restarts itself because CST may have changed.

(CstTrack = :: NoExitAction MenuDef@ InitMenu ;)

417D1	Track?	41804	Rebuild?	41837	BadMenu?
417C0	SetTrack		SetRebuild	41826	SetBadMenu
417AF	ClrTrack	417E2	ClrRebuild	41815	ClrBadMenu
	MenuRow!		MenuRow@		
	LastMenuRow!		LastMenuRow@		
41894	MenuDef!	418A4	MenuDef@		
418B4	MenuData!			418C4	DoMenuData
	MenuRowAct!			418E4	DoMenuRowAct@
	LabelDef!				DoLabel
	MenuKeyNS!	41924	MenuKeyNS@	41934	DoMenuKeyNS@
	MenuKeyLS!			41954	DoMenuKeyLS@
	MenuKeyRS!			41974	DoMenuKeyRS@
	ReviewKey!			41994	DoReviewKey@
419A4	TrackAct!			419B4	DoTrackAct@
	ExitAction!			419D4	DoExitAction@
	LastMenuDef!		LastMenuDef@		
41A04	SavedContext!	41A14	SavedContext@		
	InitTrack:	(-->)			
		Stores next as TrackAct, saves context, SetTrack			
41787	?DoTrackAct@	(-->)			
		If CONTEXT@ = SavedContext then nothing, else			
		SaveContext & DoTrackAct@			

Default menu properties set by InitMenu:

~~~~~

SetBadMenu, ClrRebuild, ClrTrack, ClrShowClock

|              |                    |
|--------------|--------------------|
| MenuDef      | = ob               |
| MenuRowAct   | = SetDA12noCh      |
| LabelDef     | = >Menu            |
| MenuKeysNS   | = StdMenuKeysNS    |
| MenuKeysLS   | = StdMenuKeysLS    |
| MenuKeysRS   | = StdMenuKeysRS    |
| ReviewKey    | = Review           |
| TrackAct     | = NOP              |
| ExitAction   | = SaveLastMenu     |
| SavedContext | = ZERO             |
| MenuKeys     | = [ob EVAL] result |

Building menus:

~~~~~

21196	xMENU	(% --> [InitMenu%])
		(Menu --> [Store to CST, ' CstMenu InitMenu])
2115D	xTMENU	(% --> [InitMenu%])
		(Ob --> [@LIST InitMenu])
	InitMenu%	(%0 --> [LastMenu])
		(%menu.page --> [Get Menu, coerce page, InitMenu#])
	InitMenu	(Menu --> [ONE InitMenu])
41373	InitSysMenu	(-->)
		Starts MathKeys with NoExitAction.

```

40F9A  InitMenu#      ( Menu #page --> )
                          StartMenu, if error then recover settings
StartMenu      ( Menu #page --> )
                          Does ExitAction (Previous menu!)
                          Sets default menu properties + page
                          Evaluates ob, stores result to MenuKeys
                          Finally SetThisRow

Building help:
~~~~~
SetThisRow      ( --> )
                  Builds a new TOUCHTAB, SetBadMenu
410E9  GetThisRow  ( Meta --> Meta' )
                  Takes submeta of MenuKeys according to MenuRow.
LoadTouchTbl    ( MenuKey1 ... MenuKeyN #n --> ) (max(n)=6)
                  Builds new TOUCHTAB from menukeys.
411D4  Key>Touch  ( MenuKey --> Def Proc )
                  Extracts data for STOPPROC and STODF from menukey

```

Menu format:

```

~~~~~
{ MenuKeys } or :: <Settings> { MenuKey1 MenuKey2 ... MenuKeyN } ;
<Settings> override default settings made by InitMenu.

```

MenuKey format:

```

~~~~~
NullMenuKey
{ LabelObj KeyProcNS }
{ LabelObj { KeyProcNS KeyProcLS } }
{ LabelObj { KeyProcNS KeyProcLS KeyProcRS } }

```

LabelObj takes no arguments and should return the argument(s) for LabelDef. TakeOver will override normal LabelDef, in this case grob should be returned.

KeyProc is the action taken upon key press. Executed by the executor. TakeOver will override executor.

```

If MenuKey = List,
    Proc = 2nd element (if non-existent then Proc = DoBadKey)
    Def = 1st element (if non-existent then Def = NULL$ )
    If Def = :: and TakeOver? Def = ::, else Def = "::"
If MenuKey = ID,  Def = ID,  Proc = ID
If MenuKey = $,   Def = $,   Proc = $
If MenuKey = #,   Def = #,   Proc = #
If MenuKey = CHR, Def = CHR, Proc = CHR
If MenuKey = Grob, Def = "Graphic...", Proc = grob
If MenuKey = Unit, Def = "UnitPart", Proc = Unit
If MenuKey = Tagged, Proc = Tagged
    If tag = 0,1,2,& Def = tagob
    Else Def = Tagged
Else Def = "MenuKey", Proc = MenuKey

```

Although # is supported in SX menu definitions, it is not supported by built-in label builders. >Menu will error when getting # as argument. Since DispMenu is not errortrapped by SysOuterLoop, HP will start VLM which is the out-of-memory error handler. This will again see that there is enough memory and restart SysOuterLoop *ignoring

the first command InitSysUI*, thus DispMenu is called again with no changes in the menu. The effect is that the screen keeps flashing. Change menu to stop the effect (DispMenu does KEYINBUFFER? test before displaying so this is possible).

ExitActions:

~~~~~

ExitAction is executed by InitMenu before setting the new menu. Thus ExitAction is the action taken when menu is exited.

NoExitAction      Sets NOP as ExitAction

Label building:

~~~~~

Label builders place \$ to grob with CENTER\$3x5.

```
MakeStdLabel      ( $ --> grob ) (StdLabelGrob + INVGROB)
MakeBoxLabel      ( $ --> grob ) (BoxLabelGrob + INVGROB)
MakeDirLabel      ( $ --> grob ) (DirLabelGrob + INVGROB)
MakeInvLabel      ( $ --> grob ) (InvLabelGrob)
Box/StdLabel      ( $ flag --> grob )
                  If TRUE then MakeBoxLabel else MakeStdLabel
Std/BoxLabel      ( $ flag --> grob )
                  If TRUE then MakeStdLabel else MakeBoxLabel
```

Label building & displaying:

~~~~~

Normal #col numbers: (#0 #16 #2C #42 #58 #6E)

```
3A260  >Menu      ( #col Grob/$/Seco/ID --> )
      Grob>Menu  ( #col Grob --> )
                  Displays Grob
      Str>Menu   ( #col $ --> )
                  Displays result of MakeStdLabel
      Seco>Menu  ( #col :: --> )
                  [EVAL DoLabel]
      Id>Menu    ( #col ID --> )
                  If @ finds object then MakeDirLabel/MakeStdLabel,
                  else ID>$ MakeStdLabel. Displays result.
      DoLabel    ( #col ob --> )
                  Executes LabelDef.
```

DispMenu will take care of the #col numbers when calling DoLabel. During the execution of DoLabel INDEX@ will give the menukey number.

#### Special displayers that use TakeOver:

~~~~~

3ED6B	DirLabel:	(--> grob) Makes dir label of next string
	Box/StdLbl:	(:: Box/StdLbl: \$label <tester> ;) Puts next to stack, evals next after it and executes Box/StdLbl.
3ECD0	FStd/BoxLbl:	(:: Box/StdLbl: \$label #flag ;) TestSysFlag for #flag, then Std/BoxLabel
3ECE6	FBox/StdLbl:	(:: Box/StdLbl: \$label #flag ;) TestSysFlag for #flag, then Box/StdLabel
3ED25	BBox/StdLbl:	(:: BBox/StdLbl: \$label #base ;) If BASENUM is EQ to #base then...
3ED48	MBox/StdLbl:	(:: MBox/StdLbl: \$label #fixmode ;) If @FixMode is EQ to #fixmode then...

Special menu types & related key settings:

~~~~~

|       |             |                                                         |
|-------|-------------|---------------------------------------------------------|
| 3F036 | SetDirMenu  | LabelDef = DirLabelDef, NoExitAction                    |
| 3EFE6 | SetDirKeys  | MenuKeysNS = DoDirKey, LS & RS = DoBadKey               |
| 3EF14 | SetUnitKeys | MenuKeyNS = UnitMenuKeyNS,...,MenuKeyRS = UnitMenuKeyRS |
| 3F00E | SetKeysNS   | (ob --> ) Sets ob as MenuKeysNS, DoBadKey to LS & RS    |

\*\*\*\*\*

## 25. Keyboard Control

~~~~~

25.1 General

^^^^^^^^

41F13 DELUSERKEYS (Deletes keys + sets CRC)

41E32 STouserKEYS (Also recalculates CRC)

41F2C StoUserKeys (Stores to UserKeys)

GetUserKeys (Pushes @UserKeys)

41F52 ClrUserKeys (Clears UserKeys)

Allowed internal forms of UserKeys:

Array of Array of System binary (49x6)

or List of Lists (49x6) / Array of System Binary (6x1)

is always passed to @PTR.

Key Codes:

#1 <= #Plane <= #6 (See binary number section)

ONE <= #KeyCode <= FORTYNINE (See binary number section)

Ck&DecKeyLoc (%rc.p --> #KeyCode #Plane)

CodePl>%rc.p (#KeyCode #Plane --> %rc.p)

Waiting for a Key:

FLUSH, FLUSHKEYS (-->)

Flush the key buffer

CHECKKEY (--> #KeyCode TRUE / FALSE)

Return next key in the buffer (No pop, does pollkeys)

GETTOUCH (--> #KeyCode TRUE / FALSE)

Pops next key from key buffer (Does pollkeys)

REPKEY? (#KeyCode --> flag)

Key pressed?

KEYINBUFFER? (--> flag)

Returns TRUE if keybuffer is not empty.

WaitForKey (--> #KeyCode #Plane)

Returns next full key press (Shifts too)

Wait/GetKey (% --> ?)

Internal WAIT command

ATTN? (--> flag)

Returns TRUE if ATTN has been pressed.

(ATTNFLAG <> 0)

ATTNFLAG@ (--> #)

Recall [ATTN] key counter

ATTNFLAGCLR (-->)

Clear [ATTN] key counter (buffer remains the same)

25.2 Inputline

^^^^^^^^^^^^^^^^

InputLine

Displays prompt in display area 2a
Sets keyboard entry modes
Initializes the edit line
Accept user input until [ENTER]
Returns TRUE if exited by [ENTER], FALSE if by [ATTN]
Parses, evaluates or just returns the user-input

Stack arguments:

\$Prompt	The prompt to be displayed during user input
\$EditLine	The initial edit line
CursorPos	The initial edit line cursor position, specified as a binary integer character number or a two-element list of binary integer row and column numbers. For all numbers, #0 indicates the end of the edit line, row, or column.
#Ins/Rep	The initial insert/replace mode: #0 current insert/replace mode #1 insert mode #2 replace mode
#Entry	The initial entry mode: #0 current entry mode plus program entry #1 program/immediate entry #2 program/algebraic entry
#AlphaLock	The initial alpha-lock mode: #0 current alpha lock mode #1 alpha lock enabled #2 alpha lock disabled
ILMenu	The initial InputLine menu in the format specified by "ParOuterLoop"
#ILMenuRow	The initial InputLine menu row number in the format specified by "ParOuterLoop"
AttnAbort?	A flag specifying whether pressing Attn while a non-null edit line exists should abort "InputLine" (TRUE) or just clear the edit line (FALSE)
#Parse	How to process the resulting edit line: #0 Return the edit line as a string #1 Return the edit line as a string AND as a parsed object #2 Parse and evaluate the edit line

Return values:

#Parse	Stack	Description
#0	\$EditLine TRUE	Edit line
#1	\$EditLine ob TRUE	Edit line and parsed edit line
#2	Ob1 ... Obn TRUE	Resulting object or objects
	FALSE	Attn pressed to abort edit

```

4A770  InputName      ( $messageline $editline --> id TRUE / FALSE )
                        SX only

                        ( This is almost the example in rplman-doc )
                        SETTYPEERR if no id was gotten.
                        Returns with FALSE if [ATTN] was pressed.
                        Arguments given for InputLine:
                        ONE      * Cursor position
                        ONE      * Insert mode
                        ONE      * Program/Immediate mode
                        ONE      * Alpha lock enabled
                        NULL{}   * No menu
                        ONE      * Menu row number
                        TRUE     * Attn aborts
                        ONE      * Returns as string and as parsed object

```

25.3 Parametrized Outer Loop

^^

Stack Arguments for ParOuterLoop:

AppDisplay	The display update object to be evaluated before each key evaluation. "AppDisplay" should handle display updating not handled by the keys themselves, and should also perform special handling of errors.
AppKeys	The hard key assignments, a secondary object in the format described below.
NonAppKeyOK?	A flag specifying whether the hard keys not assigned by the application should perform their default actions or be canceled.
DoStdKeys?	A flag used in conjunction with "NonAppKeyOK?" specifying whether standard key definitions are to be used for non-application keys instead of default key processing.
AppMenu	The menu key specification, a secondary or list in the format specified in the menu key assignments document, or FALSE. Note: FALSE means that current menu is active. Use NULL{} to avoid unnecessary updating.
#AppMenuRow	The initial application menu row number. For most applications, this should be binary integer one.
SuspendOK?	A flag specifying whether or not any user command that would create a suspended environment and restart the system outer loop should instead generate an error.
ExitCond	An object that evaluates to TRUE when the outer loop is to be exited, or FALSE otherwise. "ExitCond" is evaluated before each application display update and key evaluation.

AppError The error-handling object to be evaluated if an error occurs during the key evaluation part of the parameterized outer loop.
Note: 'ERRJMP saves 5 nibbles :)

How to use lams in ParOuterLoop:

ParOuterLoop executes POLSaveUI, which collects the current UI programs to a list and BINDs it to LAMSavedUI. Thus you get one extra lambda environment on top of the lams created before ParOuterLoop. Thus to fetch nullnamed lams you have to add 2 for every GETLAM and PUTLAM index. (See notes on LamEnvs at the end)

To create lams inside the ParOuterLoop is dangerous because POLRestoreUI makes no distinction what's in the topmost LamEnv. To get around this problem split ParOuterLoop into parts:

```
ParOuterLoop2
::
  POLSaveUI
  ERRSET
  ::
    (bind you own lambda environment)
    POLSetUI POLKeyUI
    ABND (Removing your own lamda environment)
  ;
  ERRTRAP POLResUI&Err
  POLRestoreUI
;
```

AppKeys format:

```
Stack: ( #KeyCode #Plane --> ob TRUE / FALSE )
```

```
If AppKeys does not define the key (FALSE) then
  if NonAppKeyOK? is TRUE then
    if DoStdKeys? is TRUE
      then Do standard key
      else Do standard/user key (USER or USR1 mode)
    end
  else *beep*
  end
else DoKeyOb
end
```

'Suggested' AppKeys format:

```
::
  kpNoShift       #=casedrop :: (process noshift plane) ;
  kpLeftShift    #=casedrop :: (process l-shift plane) ;
  ...
  2DROP FALSE
;
```

And for each plane:

```
::
  kcKeyCode1 ?CaseKeyDef :: (process key) ;
  kcKeyCode2 ?CaseKeyDef :: (process key) ;
  ...
  DROP FALSE
;
```

Note: TakeOver as a first command in :: (process key) ; marks a valid object to execute in edit mode.

Note: If :: (process key) ; is replaced with a ROMPTR use ?CaseRomptr@ if TakeOver test is to be done.

AppMenu format:

Note: if menu key presses are to be processed insert
DUP#7< casedrpfls into AppKeys plane handlers

```
{
  MenuKey1_Definition
  ...
  MenuKeyN_Definition
}
```

and each definition is one of

```
NullMenuKey

{ LabelObj Action }

{ LabelObj {
  No_Shift_Action
  Left_Shift_Action
} }

{ LabelObj {
  No_Shift_Action
  Left_Shift_Action
  Right_Shift_Action
} }
```

LabelObj should return a \$, a grob or some other valid menu label definition. See "Menu Key Labels" section for examples.

Choosing only some standard keys to work in ParOuterLoop:

In this example TRUE and TRUE are given as NAppKeysOK? and DoStdKeys?.

AppKeys example:

```

::
kpNoShift #=casedrop
::
    DUP#7< casedropfls          (Menu keys handled by DoKeyOb)
    kcOwnKey  ?CaseKeyDef :: (Handle own key) ;
    (more own keys)
    (Then the keys that are supposed to work normally, for example these)
    kcNextRow  #=casedrpfls (Will be processed by DoKeyOb)
    kcLeftShift  #=casedrpfls
    kcRightShift #=casedrpfls
    DROPDEADTRUE          (Drop key number, return DoBadKey as key def)
;

(Other planes similarly)

2DROP 'DoBadKeyT          (Drop plane & key, return DoBadKey as key def)
;

```

Thus in above example menu keys + [NEXT] and shifts will work, but any other key not 'defined' by user will cause an error beep.

Related Commands:

	POLSaveUI	Saves current UI to LAMSavedUI			
	POLSetUI	Sets new UI, same arguments as to ParOuterLoop			
	POLKeyUI	Displays, reads and evaluates keys according to set UI.			
38B09	1POLKeyUI	Executes UI once, doesn't check exit condition			
38B45	POLKeyErr	Handles error caused by a key press			
38994	POLSet&KeyUI	:: POLSetUI POLKeyUI ;			
	POLRestoreUI	Restores saved UI from LAMSavedUI.			
	POLResUI&Err	Restores saved UI & executes ERRJMP			
389CB	RclUI	Recalls current ParOuterLoop UI			
38A11	RclHPUI	Recalls system UI.			
38BD6	NormAppFlags	Sets default application flags.			
38A3E	SavedUILS	{ LAMSavedUI }			
38A3E	LAMSavedUI				
	AppDisplay!	38C18	AppDisplay@	38C28	NoAppDisplay!
	AppKeys!	38C48	AppKeys@	38C58	NoAppKeys!
	AppExitCond!	38C78	AppExitCond@	38C88	NoAppExitCnd!
	AppError!	38CAB	AppError@	38CBE	NoAppError!
38CD1	AppError?	38CDF	SetAppError	38CED	ClrAppError
	AppMode?		SetAppMode	38D17	ClrAppMode
38D25	NAppKeyOK?		SetNAppKeyOK	38D41	ClrNAppKeyOK
38D4F	DoStdKeys?		SetDoStdKeys	38D6B	ClrDoStdKeys
38D79	AppSuspOK?	38D8A	SetAppSuspOK	38D9B	ClrAppSuspOK

25.4 System Catalog Browser

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

47C9A DoCat (Menu { Variable Names } -->)
SX only.
Cat is a POL application, Menu is just the application menu.
The list should be DOVARS output or similar.

Cat can be hooked by defining a special Menu for it. The current status of the browser is cached to LAMs 2-8 with the following start values:

2LAM : FALSE	; ??
3LAM : NULL{}	; Cat Status line message, NULL{} = none
4LAM : FALSE	; Exit browser?
5LAM : NOP	; Review program done when exiting
6LAM : ONE	; Current position
7LAM : ONE	; Page top position
8LAM : { Names }	; The variable names

27. System Commands

~~~~~

### 27.1 User & System Flags

^^^^^^^^^^^^^^^^^^^^

|       |              |                |                                        |
|-------|--------------|----------------|----------------------------------------|
| 1C64E | RCLUSERF     | ( --> hxs )    | Recall user flags                      |
| 1C6F7 | USERSTOF     | ( hxs --> )    | Stores user flags                      |
|       | RCLSYSF      | ( --> hxs )    | Recall system flags                    |
| 1C731 | SYSSTOF      | ( hxs --> )    | Stores system flags                    |
| 1C6E3 | XEQSYSSTOF   | ( hxs --> )    | Above + checks changes in LASTARG flag |
| 53C96 | STWS         | ( % --> )      |                                        |
|       | dostws       | ( # --> )      |                                        |
| 53CF0 | RCWS         | ( --> % )      |                                        |
|       | WORDSIZE     | ( --> # )      |                                        |
| 54050 | BASEHEX      | ( --> # )      | (#10h #10d #10o or #10b)               |
| 5407A | BASECHAR     | ( --> char )   | (b,o,d or h)                           |
|       | ClrUserFlag  | ( # --> )      |                                        |
|       | SetUserFlag  | ( # --> )      |                                        |
|       | ClrSysFlag   | ( # --> )      |                                        |
|       | SetSysFlag   | ( # --> )      |                                        |
| 3EDA2 | TogSysFlag   | ( # --> )      | Toggles system flag                    |
|       | TestUserFlag | ( # --> flag ) |                                        |
|       | TestSysFlag  | ( # --> flag ) |                                        |
| 1C4CE | TestUserClr  | ( # --> flag ) | Clears flag after testing              |
| 1C4EC | TestSysClr   | ( # --> flag ) | Clears flag after testing              |
|       | DOENG        | ( # --> )      |                                        |
|       | DOFIX        | ( # --> )      |                                        |
|       | DOSCI        | ( # --> )      |                                        |
|       | DOSTD        | ( --> )        |                                        |
|       | DOHEX        | ( --> )        |                                        |
|       | DOBIN        | ( --> )        |                                        |
|       | DOOCT        | ( --> )        |                                        |
|       | DODEC        | ( --> )        |                                        |
|       | SETRAD       | ( --> )        | RAD? ( --> flag )                      |
|       | SETDEG       | ( --> )        | 53BC9 DEG? ( --> flag )                |
|       | SETGRAD      | ( --> )        |                                        |
| 53B88 | CLRNUM       | ( --> )        |                                        |
| 53B9C | SETNUM       | ( --> )        |                                        |
| 53BB0 | NOTNUM?      | ( --> flag )   |                                        |
| 53C0A | NOTCONST?    | ( --> flag )   |                                        |
| 53C23 | PRSOL?       | ( --> flag )   |                                        |

DPRADIX? ( --> flag ) Returns TRUE if current radix is .

### 27.2 General

^^^^^^^^^^^^

DOBEEP ( %freq %duration --> )  
setbeep ( #mHz #msec --> )

TurnOff ( --> ) Internal OFF  
041D4 !TurnOff ( --> ) Internal OFF, does no alarm etc checks

```

DEEPSLEEP      ( --> flag ) Returns TRUE if Invalid Card Data
ShowInvRomp    ( --> ) Flashes Invalid Card Data message

?FlashAlert    ( --> )
                Displays possible system warnings
04546 GETWARN#  ( --> # )      ( GX: 04544 )
                Recalls last system warning
                #0 = OK          #1 = Alarm
                #2 = LowBat(S)   #4 = LowBat(P1) #8 = LowBat(P2)
                ( #F means all )
04577 #>WARN$   ( # --> $ )    ( GX: 04575 )
                Recall system warning message

0D2A3 WSLOG     ( --> $4 $3 $2 $1 ) Recall warmstart log messages
0D18A WSLOGN    ( #n --> $n )   Recall nth (1-4) wslog message
0 - log cleared by <ON> <SPC> then <ON>
1 - low batteries, auto deep sleep mode invoked to save battery
2 - hardware failed during IR (time out)
3 - run through address 0
4 - system time corrupt
5 - deep sleep mode wake up (alarm?)
6 - unused
7 - CMOS test word in ram was corrupted
8 - abnormality was detected about device config
9 - corrupt alarm list
A - unused
B - card module pulled
C - hardware reset
D - system RPL error handler not found in run stream
E - corrupt config table
F - system RAM card pulled

21B4E WarmStart ( --> )
                Forces a warm start
04912 LiteSlp   ( --> )
                Enters light sleep
GARBAGE        ( --> )
                Forces garbage collection.
MEM            ( --> # )
                Returns the amount of free memory.
                Doesn't force garbage collection.
                ( User level MEM does ).
OSIZE          ( ob --> #bytes )
                Forces garbage collection.
OCRC           ( ob --> #nibbles hxs_checksum )
OCRC%          ( ob --> hxs %bytes )
VARSIZE        ( id --> hxs %bytes )

getnibs        ( hxsDATA hxsADDR --> hxsDAT' )
                Peek.
                hxsDATA is overwritten for it's length (max 16)
                with nibbles starting from address hxsADDR.
putnibs        ( hxsDATA hxsADDR --> )
                Poke. See above.
0C612 @PTR      ( ob -> ob' )      SX only
                If ob is #addr then pushes ob' at addr to stack
                If addr is in hidden ROM ob is copied to tempob area
2709B @PTRTRUE  ( ob --> ob' TRUE ) SX only

```

|       |               |                                         |         |
|-------|---------------|-----------------------------------------|---------|
| 5DE69 | @PTREVAL      | ( ob --> ? )                            | SX only |
|       | CHANGETYPE    | ( ob #prolog -> ob' )                   |         |
|       |               | Changes prolog of object, does TOTEMPOB |         |
| 05ACC | !CHANGETYPE   | ( ob #prolog -> ob' )                   |         |
|       |               | Changes prolog of object                |         |
|       | INHARDROM?    | ( ob --> ob flag )                      |         |
|       |               | Is objects address < #70000 ?           |         |
| 19350 | NOTINHARDROM? | ( ob --> ob flag )                      |         |
|       |               | Is objects address >= #70000 ?          |         |



\*\*\*\*\*

## 28. Inbuilt Programs

~~~~~

28.1 System UI

^^^^^^^^^^^^^^

Execution order after WarmStart checks:

```
021E4  ROMPOLL          ( GX: 021DD )
      Configures internal and external libraries
021CC  Boot            ( GX: 021C5 )
      Executes program stored to FIRSTPROC
```

Normally FIRSTPROC points to StartupProc

	StartupProc	Default system UI
385A7	SysOuterLoop	Default system UI without VLM errortrap
	VLM	System errortrap for memory out

=StartupProc

::

```
    SysOuterLoop
    ERRTRAP VLM
```

;

=SysOuterLoop

::

```
    InitSysUI
    BEGIN
        AtUserStack
        SysMenuCheck
        SysDisplay
        GetKeyOb
        ERRSET DoKeyOb
        ERRTRAP SysErrorTrap
    AGAIN
```

;

InitSysUI	Clears LCD
	Inits display modes
	Inits keys
	Inits first menu (MATH)
	Inits editing modes
	Sets normal application flags
	Clears MEMOUTHXS
	Clears last arguments
	Checks system warnings
	Print memory clear if needed
	If UNDO_ON? does SAVESTACK
	Does AtUserStack
SysMenuCheck	Does defined actions if context or menurow changes.
SysDisplay	Updates display
SysErrorTrap	Handles errors in key presses

DoKeyOb		Executes key press (Following for SX only)
char:		EchoChrKey
str:	404A9	EchoStrKey
unit:	404BD	DoKeyUnit
seco:	404E5	DoKeySeco
romp:	4051C	DoKeyRomp
symb:	405BC	DoKeySymb
other:	405F3	DoKeyNormal

28.2 Interactive Stack

^^^^^^^^^^^^^^^^^^^^

4365D DoUStackKey

ParOuterLoop parameters:

43770 UStackDisp
 43842 UStackKeys
 43D06 UStackMenu

28.3 Parser

^^^^^^^^^^^^

palparse (\$ --> ob TRUE / \$ #error_pos \$_error_token FALSE)
 Parse a string into an object.

!*trior (FALSE --> <SKIP>)
 (TRUE TRUE --> <COLA>)
 (FALSE TRUE --> FALSE TRUE <SEMI>)

!*triand (TRUE TRUE -->)
 (FALSE TRUE --> FALSE TRUE <SEMI>)
 (FALSE --> FALSE TRUE <SEMI>)

tok8cktrior (\$1 \$1 --> :: \$1 <Ob1> ;)
 (\$1 \$2 --> :: \$1 <Ob2> <Rest> ;)

tok8trior (GNT data \$1 \$1 --> :: GNT data GetNextToken ;)
 (GNT data \$1 \$2 --> :: \$1 <Ob1> <Rest> ;)

nultrior (NULL\$ --> :: ;)
 (\$ --> :: \$ <Ob1> <Rest> ;)

GetNextToken (hxs-mask \$ #start --> hxs-mask \$ #next \$token)

getmachtok (hxs-mask \$ #loc \$_tok --> hxs-mask \$ #next \$match)

25D3A MainMask general
 25E44 HexMask Hex strings
 25F4E SolverMask Algebraics
 26058 UnitMask Units

28.4 Decompiler

^^^^^^^^^^^^^^^^

```

DO>STR          ( ob --> $ )
                  Internal version of ->STR
                  Stores 0 as DcompWidth, then if not $ Decompile
PromptIdUtil    ( id ob --> $ )
                  Returns string of the form "id: ob"
EDITDECOMP$     ( ob --> $ )
                  Decompiles object for editing.
                  Sets 19 as DcompWidth, calls Decompile
DECOMP$         ( ob --> $ )
                  rplman.doc: 'Decompiles object for stack display.'
                  Internally 'stkdecomp..' words decompile for stack
                  display, DECOMP$ output is the same as from EDITDECOMP$
                  except for ids (Surrounding quotes). It does do
                  SetDcompLine though. Calls Dicompile
15955 1stkdecomp$l ( #line ob --> $ )
                  Decompiles 1 line for stack display, adds line

1596E !1stkdecomp$l ( #line ob #width --> $ )
                  Decompiles width for stack display, adds line
15978 1stkdecomp$w ( #line ob --> $ )
                  Decompiles for stack display, adds line (obeys DW)
159D2 stkdecomp$r  ( ob --> $ )
                  SetDcompLine stkdecomp$w

```

Dispatchers for objects:

```

          stkdecomp$w ( ob --> $ )   ST=1 3 ST=0 4 Stack display
15B31  Decompile    ( ob --> $ )   ST=0 3 ST=1 4 Editing
15B3D  Dicompile    ( ob --> $ )   ST=0 3 ST=0 4 'Stack display'

```

If stk1 is RomWord (Named ROMPTR) then dispatching is not done and a special decompiler is called:

16153 DcRomWord

	stkdecomp\$w:		D?compile:	
	~~~~~		~~~~~	
%	162AC DiReal aka a%>\$,	162B8	DeReal aka a%>\$	
C%	1746E DcCmp	1746E	DcCmp	
#	15D06 DcSystBin	15D06	DcSystBin	
id	160D1 DiIdnt	160E5	DeIdnt	
lam	160D1 DiIdnt	160E5	DeIdnt	
\$	15F3D DiString	15ED8	DeString	(Stack\$?)
hxs	HXS>\$	15F83	DeBinary	(Stack\$?)
romptr	1616C DcRomptr	1616C	DcRomptr	
lib	15D38 DcLibrary	15D28	DcLibrary	
bak	15D97 DcBackUp	15D97	DcBackUp	
grob	15DF1 DiGrob	15E1F	DeGrob	(Stack\$?)
symb	16103 DiSymb	161D0	DeComp	
ext	16103 DiSymb	161D0	DeComp	
arry	1622A DiComp	161D0	DeComp	
list	1622A DiComp	161D0	DeComp	
seco	1622A DiComp	161D0	DeComp	
tagged	1622A DiComp	161D0	DeComp	
rrp	1622A DiComp	161D0	DeComp	

(Stack\$?) means that if Stack\$? is set stkdecomp\$w dispatchees are used.

	!DcompWidth	( # --> )	Stores new decompile width
17980	@DcompWidth	( --> # )	Recalls decompile width
159FA	SetDcompLine	( --> )	Sets 19 as DcompWidth
11472	@Indent	( --> # )	Recalls indent level
11495	!Indent	( # --> )	Stores indent level
178E2	IncIndent	( --> )	Adds 2 to indent level
178FB	DecIndent	( --> )	Subtracts 2 from indent level
1792D	INDENT\$	( \$ --> \$' )	!insert\$ @Indent spaces
17914	?INDENT\$	( \$ --> \$' )	If Stack\$? then nothing, else INDENT\$

## 28.5 Kermit

^^^^^^^^^^^^

MAXRETRY           Maximum number of retries = 10

Related lams:

	LAMPKNO	Contains packet number
	LAMPACKET	
	LAMRETRY	Contains #retry
2D3D9	LAMERRMSG	
	LAMKP	
	LAMLNAME	
	LAMOBJ	
	LAMOPOS	
2D42E	LAMEXCHP	
	LAMKLIST	
	LAMKMODE	
2D480	LAMKPTRN	
	LAMKRM	
2D4A2	LAMMaxR	
	LAMKML	
	'LamKPSto	
2D816	DOREC�	Internal RECN ( \$/id/lam)
2D9F5	DOSERVER	Internal SERVER
	SENDLIST	Internal SEND ( {} --> )
2E5AB	SENDNAME	Internal SEND ( id/lam --> )
	GETNAME	Internal KGET ( \$/id/lam --> )
	DOFINISH	Internal FINISH
	DOPKT	Internal PKT ( \$ \$ --> )
	DOBAUD	Internal BAUD ( % --> )
	DOPARITY	Internal PARITY ( % --> )
	DOTRANSIO	Internal TRANSIO ( % --> )
2ED4C	DOCKSM	Internal CKSM
	DOKERRM	Internal KERRM
	DOBUFLCN	Internal BUFLCN
2EDF5	DOSTIME	Internal STIME
	DOSBRK	Internal SBRK ( --> )
2EE6F	DOXMIT	Internal XMIT ( \$ --> )
	DOSRECV	Internal SRECV ( % --> )
	CLOSEUART	Internal CLOSEIO
	DOCR	Internal CR

DOPR1	Internal PR1	
DOPRSTC	Internal PRSTC	
DOPRST	Internal PRST	
DODELAY	Internal DELAY	
KDispRow2		
KDispStatus2		
EXCHINITPK		
SENDEOT		
SEND_PACKER		
SENDNAK		
SENDERROR		
SENDPKT		
BUILDPACKET		
InitIOEnv		
KERMOPEN		
DOOPENIO		
OpenIO		
OpenIOPrt		
APNDCRLF		
docr		
IOCheckReal		
StdIOPAR	Default IOPAR	( --> { 9600 0 0 0 3 1 } )
GetIOPAR		
StoIOPAR		
SetIOPARErr		
StdPRTPAR		
StoPRTPAR		
GetChkPRTPAR		
OUTUART		
POPUART		
OpenUartClr		
PRINT		
PRINTxNLF		
REMAP		
SetEcma94		
AllowPrlcdCl		
PrintGrob		
SetServMode		
ClrServMode		
SendSetup		
TRPACKETFAIL		
IncrLAMPKNO	Increases packet number	
GetKermPkt#		
RecvNextPkt		
KVISLF		
KVIS		
KINVISLF		
GETKP		
SENDACK		
SENDNULLACK		
ACK_INIT		
CHOOSE_INIT		
ENCODE1PKT		
ENCODE		
DECODE		
UARTBUFLen		
FLUSHRSBUF		
PUTSERIAL		
GETSERIAL		

```

VERSTRING      ( --> $ )      Returns version string
3037A  VERSTRING?      ( $ --> $ flag ) Checks if $ starts with VERSTRING

```

---

28.6 Debugger  
 ^^^^^^^^^^^^^^^

---

28.7 Matrix Writer  
 ^^^^^^^^^^^^^^^

---

28.8 ParOuterLoop  
 ^^^^^^^^^^^^^^^

ParOuterLoop

```

::
  POLSaveUI
  ERRSET
  ::
    POLSetUI
    POLKeyUI
  ;
  ERRTRAP POLResUI&Err
  POLRestoreUI
;

```

POLSaveUI

```

::
  AppMode? ITE
    RclUI
    RclHPUI
  {}N SaveUILS BIND
;
  * Recall current ParOuterLoop defs
  * Recall system UI definitions
  * !! 1 lam created !!
  * Thus add 2 for every GET/PUTLAM
  * to use lams created before ParOuterLoop

```

RclUI

```

::
  LastMenuDef@
  LastMenuRow@
  AppDisplay@
  AppKeys@
  NAppKeyOK?
  DoStdKeys?
  MenuDef@
  MenuRow@
  AppSuspOK?
  AppExitCond@
  AppError@
  ELEVEN
;

```

RclHPUI

```

::
  LastMenuDef@
  LastMenuRow@
  MenuDef@
  MenuRow@
  FOUR
;

```

```
POLSetUI
::
  AppError!
  NoAppDisplay!
  NoAppKeys!
  ClnAppKeyOK
  SetAppSuspOK
  NoAppExitCnd!
  NoAppError!
;
```

---

28.9 Editor  
~~~~~

\*\*\*\*\*

APPENDIX

A. Memory Organization for SX:

Approximate program areas:

- 00100 - IO Registers
- 00140 - System ML routines
- 028FC - Provided objects
- 0314C - General purpose routines
- 0CAD7 - Time/Alarm routines
- 0F075 - Unit handling
- 1158F - Grob handling / Display
- 13FE5 - Debugger
- 15955 - Decompiler programs
- 17B86 - Low level system errortrap
- 1957B - Internal library 2 main programs
- 22DFE - Internal library 1792 main programs
- 23879 - Parser
- 26886 - Symbolics parser
- 29DDC - Real number functions
- 2C1F3 - Statistics
- 2D359 - Kermit / Printer
- 32F77 - Root finder
- 34D2B - Symbolics decompiler
- 3551D - Array handling
- 3858E - System UI
- 3A5CD - Menus & Keys
- 424DA - Editor
- 4365D - Interactive stack
- 44948 - Matrix Writer
- 4737D - Plotter & other graphics
- 5193B - Complex number handling
- 52C36 - Numeric intergration
- 53C37 - Hex string handling
- 54565 - Symbolics
- 60EBD - General purpose routines
- 64B12 - Constants area (binary & real numbers, strings, characters)
- 659DE - Equation Writer

Hidden ROM:

- 70000 - System test routines (Memory scanner etc)
- 72000 - Message, unit, hash tables
- 75F9E - Machine language for real functions
- 7A929 - Character data
- 7AA5D - Default key definitions
- 7B5F2 - Symbolics for inversion, integration, derivation etc
- 7F6D1 - Handling port objects

Normal RAM structure:

70000 - 707D9 RAM pointer & status area

| Pointer | Name | Description | Sample values |
|----------|------------|------------------------------------|---------------|
| 70551 | VDISP2 | Menu display grob | 708A4 |
| 70556 | ADISP | Text display grob | 709C8 |
| 7055B | VDISP | Current display grob | 709C8 |
| 70560* | PDISP | Pict grob??? | 709C8 |
| 70565 | GDISP | Graphic display grob | 709C8 |
| 7056A* | TEMPBOT | Bottom of tempob area (?) | 71AF0 |
| 7056F | TEMPTOP | Top of tempob area | 72527 |
| 70574* | RETTOP | Top of rpl return stack | 7254F |
| FREE RAM | | | |
| 70579 | DSKTOP | Top of stack | 7F197 |
| 7057E* | DSKBOT | Bottom of stack | 7F1BA |
| | EDITLINE\$ | Backward string, terminates in 00. | |
| 70583 | TEMPENV | Temporary environments | 7F1EA |
| 70588 | LOOPENV | Loop environments | 7F20D |
| 7058D | TOUCHTAB | Menu key names & executor | 7F212 |
| 70592* | HOMETOP | HOME directory | 7F260 |
| 70597* | PORT0TOP | Start of PORT0 | 7FFFB |

@TEMPTOP: List of linked objects (to lower memory)

```

@TEMPBOT CON(5) 0          End of objects
          CON(1) ?
          ObjectN
          CON(5) (*)-(LinkN)
          ...
Link1     CON(5) (*)-(Link2)
          CON(1) ?
          Object1
          CON(5) (*)-(Link1)
@TEMPTOP CON(5) 0

```

@RETTOP: List of 5 nibble addresses to RPL streams.

```

@TEMPTOP CON(5) 0          End of return stack
          CON(5) =rstkn
          ...
          CON(5) =rstk1
@RETTOP  CON(5) #free

```

@DSKTOP: List of 5 nibble addresses to objects. 00000 = end

```

          CON(5) =stk1
          ...
          CON(5) =stkn
          CON(5) 0
@DSKBOT  CON(5) 0          Some marking system??? (UpStack???)

```

@TEMPENV: List of temporary environments.

| | | |
|---------|---------------------|------------------------------------|
| | REL(5) LamEnv2 | Link to next temporary environment |
| | CON(5) #protection1 | Protection counter |
| | CON(5) =lam1name | Pointer to lam name |
| | CON(5) =lam1ob | Pointer to object |
| | CON(5) =lam2name | |
| | CON(5) =lam2ob | |
| | ... | |
| LamEnv2 | REL(5) LamEnv3 | |
| | ... | |
| LamEnvN | CON(5) 0 | End of temporary environments |

@LOOPENV: Loop environment counter & list of fixed size (#F) environments

| |
|---------------------|
| CON(5) #LoopEnvs |
| CON(5) #protection1 |
| CON(5) #index1 |
| CON(5) #stopvalue1 |
| CON(5) #protection2 |
| ... |
| CON(5) #stopvalueN |

@TOUCHTAB: List of menu key names, associated objects and a main executor

| | |
|-------------------|-----------------------------------|
| CON(3) #??? | |
| CON(5) =MenuName1 | |
| ... | |
| CON(5) =MenuName6 | |
| CON(5) =NULL\$ | Name of key definition program??? |
| CON(5) =MenuOb1 | |
| ... | |
| CON(5) =MenuOb6 | |
| CON(5) =Executor | Changed by shift keys |

\*\*\*\*\*

C. Default Keys for SX

(In GX the keys are stored to libraries A1-A6.)

3FB1F r\_KDefs Reference to key definition array (#)
 7B4E4 KDefs Key definition array. Contains pointers to 49 arrays
 which in turn contain pointers to objects executed in
 normal, LS, RS, A, ALS, RLS modes.
 (Characters marked with lowe case chr are in hidden
 ROM. Their addresses are not marked)

| No | Addr1 | Label1 | Addr2 | Label2 | Addr3 | Label3 |
|----|-------|--------------|-------|---------------|-------|--------------|
| 11 | 3FFDB | DoMenuKey1NS | 3FFF4 | DoMenuKey1LS | 4000D | DoMenuKey1RS |
| | | CHR_A | | CHR_a | | chr_alpha |
| 12 | 4003F | DoMenuKey2NS | 40026 | DoMenuKey2LS | 40058 | DoMenuKey2RS |
| | | CHR_B | | CHR_b | | chr_beta |
| 13 | 40071 | DoMenuKey3NS | 4008A | DoMenuKey3LS | 400A3 | DoMenuKey3RS |
| | | CHR_C | | CHR_c | | chr_Delta |
| 14 | 400BC | DoMenuKey4NS | 400D5 | DoMenuKey4LS | 400EE | DoMenuKey4RS |
| | | CHR_D | | CHR_d | | chr_delta |
| 15 | 40107 | DoMenuKey5NS | 40120 | DoMenuKey5LS | 40139 | DoMenuKey5RS |
| | | CHR_E | | CHR_e | | chr_epsilon |
| 16 | 40152 | DoMenuKey6NS | 4016B | DoMenuKey6LS | 40184 | DoMenuKey6RS |
| | | CHR_F | | CHR_f | | chr_theta |
| 21 | 3AD57 | DoMathMenu | 3AE33 | DoPrintMenu | | xPR1 |
| | | CHR_G | | CHR_g | | chr_gamma |
| 22 | 3AE1A | DoPrgmMenu | 3A6E5 | DoIOMenu | | xSERVER |
| | | CHR_H | | CHR_h | | chr_eta |
| 23 | 3AB72 | DoCstMenu | 3ADA2 | DoModeMenu | 3ADBB | DoCustomMenu |
| | | CHR_I | | CHR_i | | chr_inf |
| 24 | 3AF37 | DoVarMenu | 3AD70 | DoMemoryMenu | 3AD89 | DoAritMenu |
| | | CHR_J | | CHR_j | | chr_ |
| 25 | 3A93D | DoUpKey | 3AE6F | DoLibraryMenu | 3B00E | DoUpKeyRS |
| | | CHR_K | | CHR_k | | chr_up |
| 26 | | DoNextRow | | DoPrevRow | | DoFirstRow |
| | | CHR_L | | CHR_l | | chr_lamda |
| 31 | 3A69A | Do 'Key | | xUPDIR | | xHOME |
| | | CHR_M | | CHR_m | | chr_' |
| 32 | 3A992 | DoStoKey | | xDEFINE | | xRCL |
| | | CHR_N | | CHR_n | | chr_mu |
| 33 | | xEVAL | | x>Q | | x>NUM |
| | | CHR_O | | CHR_o | | chr_Omega |
| 34 | 3A834 | DoLeftKey | | xGRAPH | 3AFE6 | DoLeftKeyRS |
| | | CHR_P | | CHR_p | | chr_<- |
| 35 | 3A645 | DoDownKey | 3AE4C | DoReviewKey | 3AF69 | DoDownKeyRS |
| | | CHR_Q | | CHR_q | | chr_down |
| 36 | 3A8DE | DoRightKey | | xSWAP | 3A80C | DoRightKeyRS |
| | | CHR_R | | CHR_r | | chr_rho |
| 41 | | xSIN | | xSIN | | xDER |
| | | CHR_S | | CHR_s | | chr_sigma |
| 42 | | xCOS | | xACOS | | xINTEGRAL |
| | | CHR_T | | CHR_t | | chr_tau |
| 43 | | xTAN | | xATAN | | xSUM |
| | | CHR_U | | CHR_u | | chr_% |
| 44 | | xSQRT | | xSQ | | xXROOT |
| | | CHR_V | | CHR_v | | chr_~ |

| | | | | | | |
|----|-------|--------------|-------|---------------|-------|---------------|
| 45 | | x^ | | xALOG | | xLOG |
| | | CHR_W | | CHR_w | | chr_omega |
| 46 | | xINV | | xEXP | | xLN |
| | | CHR_X | | CHR_x | | chr_xbar |
| 51 | 3A7F3 | DoEnterKey | 3AF50 | DoEnterKeyLS | 3B068 | DoEnterKeyRS |
| | 3A7F3 | DoEnterKey | | chr_& | | chr_@ |
| 52 | 3AA82 | DoNegKey | 3A7C6 | DoEditKey | 3B12B | DoVisitKey |
| | | CHR_Y | | CHR_y | | chr_+/- |
| 53 | 3AC3A | DoEexKey | 3B15D | Do2DKey | 3B18F | Do3DKey |
| | | CHR_Z | | CHR_z | | chr_Pi |
| 54 | 3AB09 | DoDelKey | 3B1DF | DoPurgeKey | 3AAEB | DoClvarKey |
| | 3AB09 | DoDelKey | | chr_! | | chr_inv_! |
| 55 | 3A5F0 | DoBackspcKey | | xDROP | | xCLEAR |
| | 3A5F0 | DoBackspcKey | | chr_? | | chr_inv_? |
| 61 | | 1A/LockA | 3B0DB | TogUser | 3B081 | TogEntry |
| | 3B036 | LockA/0A | 3E5EE | TogLowerCase | 3E5AF | TogIns |
| 62 | | CHR_7 | 3AE88 | DoSolveMenu | | DoSolvrMenu |
| | | CHR_7 | 3F167 | DoInv7Key | 3F17B | DoInv7KeyRS |
| 63 | | CHR_8 | | DoPlotMenu | 48FD6 | DoPlotrMenu |
| | | CHR_8 | 3F18F | DoInv8Key | 3F1A3 | DoInv8KeyRS |
| 64 | | CHR_9 | 3AB59 | DoAlgebraMenu | 47B5A | DoPCatKey |
| | | CHR_9 | 3F1B7 | DoInv9Key | 3F1CB | DoInv9KeyRS |
| 65 | | x/ | 3A6CC | Do()Key | 3AC08 | Do#Key |
| | | CHR_/ | 3A6CC | Do()Key | | CHR_# |
| 71 | 3A893 | DoLSKey | 3A8AC | DoNSKey | 3A893 | DoLSKey |
| | 3AA37 | DoLSKeyA | 3AA50 | DoNSKeyA | 3AA37 | DoLSKeyA |
| 72 | | CHR_4 | 3A7A3 | DoTimeMenu | 47AE7 | DoTimeCat |
| | | CHR_4 | | chr_\$ | | chr_cent |
| 73 | | CHR_5 | 3AEB5 | DoStatMenu | 3AEE2 | DoStatMenuRS |
| | | CHR_5 | | chr_pound | | chr_yen |
| 74 | | CHR_6 | 3AF05 | DoUnitsMenu | 3AF1E | DoUnitCmdMenu |
| | | CHR_6 | | chr_currency | | chr_degree |
| 75 | | x* | 3ABA4 | Do[]Key | 3A6B3 | Do_Key |
| | | CHR_* | 3ABA4 | Do[]Key | | CHR_UndScore |
| 81 | 3A8C5 | DoRSKey | 3A8AC | DoRSKey | 3AA69 | DoNSKey |
| | 3AA69 | DoRSKeyA | 3AA69 | DoRSKeyA | 3AA50 | DoNSKeyA |
| 82 | | CHR_1 | 3B1CB | DoRadKey | 3B1B7 | DoPolarKey |
| | | CHR_1 | | \$_== | | chr_<> |
| 83 | | CHR_2 | 3ACF8 | DoLastStk | | xLASTARG |
| | | CHR_2 | | CHR_< | | CHR_> |
| 84 | | CHR_3 | 3ACBC | DoLastCmd | 3A6FE | DoLastMenu |
| | | CHR_3 | | chr_<= | | chr_>= |
| 85 | | x- | 3ABD6 | Do<<>>Key | 3AC21 | Do" "Key |
| | | CHR_- | 3ABBD | Do<<>>KeyLS | | CHR_DblQuote |
| 91 | 3A5CD | TurnOnKey | | xCONT | | TurnOffKey |
| | 3A5CD | TurnOnKey | | xCONT | | TurnOffKey |
| 92 | | CHR_0 | | x= | | xALG-> |
| | | CHR_0 | | CHR_= | | CHR_-> |
| 93 | 3A753 | Do.Key | 3A77B | Do,Key | 3ADD4 | DoNewlineKey |
| | 3A753 | Do.Key | 3A77B | Do,Key | 3ADD4 | DoNewlineKey |
| 94 | | CHR_Space | | xPI | | CHR_Angle |
| | | CHR_Space | | CHR_Pi | | CHR_Angle |
| 95 | | x+ | 3AB8B | Do{ }Key | 3ABEF | Do::Key |
| | | CHR_+ | 3AB8B | Do{ }Key | | CHR_: |

\*\*\*\*\*

D. Built-in Libraries in SX

| Library | Addr | Link | Hash | #Hash | Mesg | #Mesg | Config |
|---------|-------|-------|-------|-------|-------|-------|----------------------|
| 000 | 028E3 | - | - | - | 04CDC | 72000 | 0BB40 (Hidden dir) |
| 001 | 10F14 | - | - | - | 653CE | 72704 | 11086 (Hidden keys) |
| 002 | 189E8 | 22651 | 22647 | 7448A | 653D8 | 72DCF | 188A1 (Attach) |
| 003 | 29DB3 | - | - | - | 653E2 | 72F1E | - |
| 005 | 35504 | - | - | - | 653EC | 736F9 | - |
| 006 | 2C086 | - | - | - | - | 72FE6 | - |
| 00A | 32F5E | - | - | - | 6540A | 72281 | - |
| 00B | 0F05C | - | - | - | 653C4 | 726A5 | - |
| 00C | 2D340 | - | - | - | 653F6 | 7232C | 2D359 (IO prepare) |
| 00D | 0CABE | - | - | - | 65400 | 7260A | 0CAD7 (Hidden alarm) |
| 019 | 0BE14 | - | - | - | - | - | - |
| 0F0 | 38336 | 3834F | - | - | - | - | - |
| 700 | 22DE5 | 22E08 | 22DFE | 7427C | - | - | 22EA3 (Attach) |

Library 002

Names are those defined in USRLIB.EXE.

| | | |
|----------------------|---------------------|---------------------|
| 000 1957B xASR | 001 1959B xRL | 002 195BB xRLB |
| 003 195DB xRR | 004 195FB xRRB | 005 1961B xSL |
| 006 1963B xSLB | 007 1965B xSR | 008 1967B xSRB |
| 009 1969B xR>B | 00A 196BB xB>R | 00B 196DB xCONVERT |
| 00C 1971B xUVAL | 00D 1974F x>UNIT | 00E 19771 xUBASE |
| 00F 197A5 xUFACT | 010 197F7 xTIME | 011 19812 xDATE |
| 012 1982D xTICKS | 013 19848 xWSLOG | 014 19863 xACKALL |
| 015 1987E xACK | 016 1989E xSETDATE | 017 198BE xSETTIME |
| 018 198DE xCLKADJ | 019 198FE xSTOALARM | 01A 19928 xRCLALARM |
| 01B 19948 xFINDALARM | 01C 19972 xDELALARM | 01D 19992 xTSTR |
| 01E 199B2 xDDAYS | 01F 199D2 xDATE+ | 020 1A105 xCRDIR |
| 021 1A125 xPATH | 022 1A140 xHOME | 023 1A15B xUPDIR |
| 024 1A194 xVARS | 025 1A1AF xTVARS | 026 1A1D9 xBYTES |
| 027 1A2BC xNEWOB | 028 1A303 xKILL | 029 1A31E xOFF |
| 02A 1A339 xDOERR | 02B 1A36D xERR0 | 02C 1A388 xERRN |
| 02D 1A3A3 xERRM | 02E 1A3BE xEVAL | 02F 1A3FE xIFTE |
| 030 1A4CD xIFT | 031 1A52E xSYSEVAL | 032 1A584 xDISP |
| 033 1A5A4 xFREEZE | 034 1A5C4 xBEEP | 035 1A5E4 x>NUM |
| 036 1A604 xLAST | 037 1A71F xWAIT | 038 1A858 xCLLCD |
| 039 1A873 xKEY | 03A 1A8BB xCONT | 03B 1A8D8 x= |
| 03C 1A995 xNEG | 03D 1AA1F xABS | 03E 1AA6E xCONJ |
| 03F 1AABD xPI | 040 1AADF xMAXR | 041 1AB01 xMINR |
| 042 1AB23 xCONSTANTE | 043 1AB45 xi | 044 1AB67 x+ |
| 045 1ACDD xNEGNEG | 046 1AD09 x- | 047 1ADEE x* |
| 048 1AF05 x/ | 049 1B02D x^ | 04A 1B185 rpnxROOT |
| 04B 1B1CA xXROOT | 04C 1B278 xINV | 04D 1B2DB xARG |
| 04E 1B32A xSIGN | 04F 1B374 xSQRT | 050 1B426 xSQ |
| 051 1B4AC xSIN | 052 1B505 xCOS | 053 1B55E xTAN |
| 054 1B5B7 xSINH | 055 1B606 xCOSH | 056 1B655 xTANH |
| 057 1B6A4 xASIN | 058 1B72F xACOS | 059 1B79C xATAN |
| 05A 1B7EB xASINH | 05B 1B830 xACOSH | 05C 1B8A2 xATANH |
| 05D 1B905 xEXP | 05E 1B94F xLN | 05F 1B9C6 xLOG |
| 060 1BA3D xALOG | 061 1BA8C xLNP1 | 062 1BAC2 xEXPM |
| 063 1BB02 xFACT | 064 1BB41 preFACT | 065 1BB6D xIP |

| | | |
|------------------------|-----------------------|---------------------|
| 066 1BBA3 xFP | 067 1BBD9 xFLOOR | 068 1BC0F xCEIL |
| 069 1BC45 xXPON | 06A 1BC71 xMAX | 06B 1BCE3 xMIN |
| 06C 1BD55 xRND | 06D 1BDD1 xTRNC | 06E 1BE4D xMOD |
| 06F 1BE9C xMANT | 070 1BEC8 xD>R | 071 1BEF4 xR>D |
| 072 1BF1E x>HXS | 073 1BF3E xHMS> | 074 1BF5E xHMS+ |
| 075 1BF7E xHMS- | 076 1BF9E xRNRM | 077 1BFBE xCNRM |
| 078 1BFDE xDET | 079 1BFFE xDOT | 07A 1C01E xCROSS |
| 07B 1C03E xRSD | 07C 1C060 x% | 07D 1C0D7 x%T |
| 07E 1C149 x%CH | 07F 1C1B9 xRAND | 080 1C1D4 xRDZ |
| 081 1C1F6 xCOMB | 082 1C236 xPERM | 083 1C274 xSF |
| 084 1C2D5 xCF | 085 1C313 xFS? | 086 1C360 xFC? |
| 087 1C399 xDEG | 088 1C3B4 xRAD | 089 1C3CF xGRAD |
| 08A 1C3EA xFIX | 08B 1C41E xSCI | 08C 1C452 xENG |
| 08D 1C486 xSTD | 08E 1C4A1 xFS?C | 08F 1C520 xFC?C |
| 090 1C559 xBIN | 091 1C574 xDEC | 092 1C58F xHEX |
| 093 1C5AA xOCT | 094 1C5C5 xSTWS | 095 1C5FE xRCWS |
| 096 1C619 xRCLF | 097 1C67F xSTOF | 098 1C783 x>LIST |
| 099 1C79E xR>C | 09A 1C7CA xRE | 09B 1C819 xIM |
| 09C 1C85C xSUB | 09D 1C8EA xREPL | 09E 1C95A xLIST> |
| 09F 1C98E xC>R | 0A0 1C9B8 xSIZE | 0A1 1CAB4 xPOS |
| 0A2 1CB0B x>STR | 0A3 1CB26 xSTR> | 0A4 1CB46 xNUM |
| 0A5 1CB66 xCHR | 0A6 1CB86 xTYPE | 0A7 1CE28 xVTYPE |
| 0A8 1CE33 xEQ> | 0A9 1CF7B xOBJ> | 0AA 1D009 x>ARRY |
| 0AB 1D092 xARRY> | 0AC 1D0DF xRDM | 0AD 1D186 xCON |
| 0AE 1D2DC xIDN | 0AF 1D392 xTRN | 0B0 1D407 xPUT |
| 0B1 1D5DF xPUTI | 0B2 1D7C6 xGET | 0B3 1D8C7 xGETI |
| 0B4 1DD06 xV> | 0B5 1DE66 x>V2 | 0B6 1DEC2 x>V3 |
| 0B7 1E04A xINDEP | 0B8 1E07E xPMIN | 0B9 1E09E xPMAX |
| 0BA 1E0BE xAXES | 0BB 1E0E8 xCENTR | 0BC 1E126 xRES |
| 0BD 1E150 x*H | 0BE 1E170 x*W | 0BF 1E190 xDRAW |
| 0C0 1E1AB xAUTO | 0C1 1E1C6 xDRAX | 0C2 1E1E1 xSCALE |
| 0C3 1E201 xPDIM | 0C4 1E22B xDEPND | 0C5 1E25F xERASE |
| 0C6 1E27A xPX>C | 0C7 1E29A xC>PX | 0C8 1E2BA xGRAPH |
| 0C9 1E2D5 xLABEL | 0CA 1E2F0 xPVIEW | 0CB 1E31A xPIXON |
| 0CC 1E344 xPIXOFF | 0CD 1E36E xPIX? | 0CE 1E398 xLINE |
| 0CF 1E3C2 xTLINE | 0D0 1E3EC xBOX | 0D1 1E416 xBLANK |
| 0D2 1E436 xPICT | 0D3 1E456 xGOR | 0D4 1E4E4 xGXOR |
| 0D5 1E572 xLCD> | 0D6 1E58D x>LCD | 0D7 1E5AD x>GROB |
| 0D8 1E5D2 xARC | 0D9 1E606 xTEXT | 0DA 1E621 xXRNG |
| 0DB 1E641 xYRNG | 0DC 1E661 xFUNCTION | 0DD 1E681 xCONIC |
| 0DE 1E6A1 xPOLAR | 0DF 1E6C1 xPARAMETRIC | 0E0 1E6E1 xTRUTH |
| 0E1 1E701 xSCATTER | 0E2 1E721 xHISTOGRAM | 0E3 1E741 xBAR |
| 0E4 1E761 xSAME | 0E5 1E783 xAND | 0E6 1E809 xOR |
| 0E7 1E88F xNOT | 0E8 1E8F6 xXOR | 0E9 1E972 x== |
| 0EA 1EA9D x#? | 0EB 1EBBE x<? | 0EC 1EC5D x>? |
| 0ED 1ECFC x<=? | 0EE 1ED9B x>=? | 0EF 1EE38 xOLDPRT |
| 0F0 1EE53 xPR1 | 0F1 1EE6E xPRSTC | 0F2 1EE89 xPRST |
| 0F3 1EEA4 xCR | 0F4 1EEBF xPRVAR | 0F5 1EF43 xDELAY |
| 0F6 1EF63 xPRLCD | 0F7 1EF7E rpnDER | 0F8 1EFD2 xDER |
| 0F9 1F133 xRCEQ | 0FA 1F14E xSTEQ | 0FB 1F16E xROOT |
| 0FC 1F1D4 rpnINTG | 0FD 1D223 yINTEGRAL | 0FE 1F2C9 xSUM |
| 0FF 1F354 rpnWHERE | 100 1F3F3 xWHERE | 101 1F500 xQUOTE |
| 102 1F55D rpnAPPLY | 103 1F5C5 xAPPLY | 104 1F640 xFCNAPPLY |
| 105 1F996 COMPLEXDUMMY | 106 1F9AE POLARDUMMY | 107 1F9C4 x->Q |
| 108 1F9E9 x->QPI | 109 1FA59 xMATCHUP | 10A 1FA8D xMATCHDN |
| 10B 1FAEB xFORMUNIT | 10C 1FB5D xPREDIV | 10D 1FB87 xDUP |
| 10E 1FBA2 xDUP2 | 10F 1FBBD xSWAP | 110 1FBD8 xDROP |
| 111 1FBF3 xDROP2 | 112 1FC0E xROT | 113 1FC29 xOVER |
| 114 1FC44 xDEPTH | 115 1FC64 xDROPN | 116 1FC7F xDUPN |
| 117 1FC9A xPICK | 118 1FCB5 xROLL | 119 1FCD0 xROLLD |

| | | |
|---------------------|---------------------|----------------------|
| 11A 1FCEB xCLEAR | 11B 1FD0B xSTOSIGMA | 11C 1FD2B xCLSIGMA |
| 11D 1FD46 xRCLSIGMA | 11E 1FD61 xSIGMA+ | 11F 1FD8B xSIGMA- |
| 120 1FDA6 xNSIGMA | 121 1FDC1 xCORR | 122 1FDDC xCOV |
| 123 1FDF7 xSUMX | 124 1FE12 xSUMY | 125 1FE2D xSUMX2 |
| 126 1FE48 xSUMY2 | 127 1FE63 xSUMXY | 128 1FE7E xMAXSIGMA |
| 129 1FE99 xMEAN | 12A 1FEB4 xMINSIGMA | 12B 1FECF xSDEV |
| 12C 1FEEA xTOT | 12D 1FF05 xVAR | 12E 1FF20 xLR |
| 12F 1FF7A xPREDV | 130 1FF9A xPREDY | 131 1FFBA xPREDX |
| 132 1FFDA xXCOL | 133 1FFFA xYCOL | 134 2001A xUTPC |
| 135 2003A xUTPN | 136 2005A xUTPF | 137 2007A xUTPT |
| 138 2009A xSIGMACOL | 139 200C4 xSCLSIGMA | 13A 200F3 xSIGMALINE |
| 13B 2010E xBINS | 13C 20133 xBARPLOT | 13D 20167 xHISTPLOT |
| 13E 2018C xSCATRLOT | 13F 201B1 xLINFIT | 140 201D6 xLOGFIT |
| 141 201FB xEXPFIT | 142 20220 xPWRFIT | 143 2025E xBESTFIT |
| 144 202CE xSINV | 145 2034D xSNEG | 146 203CC xSCONJ |
| 147 2044B xSTO+ | 148 20538 xSTO- | 149 2060C xSTO/ |
| 14A 20753 xSTO* | 14B 208F4 xINCR | 14C 209AA xDECR |
| 14D 20A15 xCOLCT | 14E 20A49 xEXPAN | 14F 20A7D xRULES |
| 150 20A93 xISOL | 151 20AB3 xQUAD | 152 20AD3 xSHOW |
| 153 20B20 xTAYLR | 154 20B40 xRCL | 155 20CCD xSTO |
| 156 20D65 xDEFINE | 157 20EFE xPURGE | 158 20FAA xMEM |
| 159 20FD9 xORDER | 15A 210FC xCLUSR | 15B 2115D xTMENU |
| 15C 21196 xMENU | 15D 211E1 xRCLMENU | 15E 211FC xPVARs |
| 15F 2123A xPGDIR | 160 2125A xARCHIVE | 161 2133C xRESTORE |
| 162 2137F xMERGE | 163 213D1 xFREE | 164 2142D xLIBS |
| 165 21448 xATTACH | 166 2147C xDETACH | 167 21E75 xXMIT |
| 168 21E95 xSRECV | 169 21EB5 xOPENIO | 16A 21ED5 xCLOSEIO |
| 16B 21EF0 xSEND | 16C 21F24 xKGET | 16D 21F62 xRECN |
| 16E 21F96 xRECV | 16F 21FB6 xFINISH | 170 21FD1 xSERVER |
| 171 21FEC xCKSM | 172 2200C xBAUD | 173 2202C xPARITY |
| 174 2204C xTRANSIO | 175 2206C xKERRM | 176 22087 xBUFLEN |
| 177 220A2 xSTIME | 178 220C2 xSBRK | 179 220DD xPKT |
| 17A 224CA xINPUT | 17B 224F4 xASN | 17C 22514 xSTOKEYS |
| 17D 22548 xDELKEYS | 17E 22586 xRCLKEYS | 17F 225BE x->TAG |
| 180 22633 xDTAG | | |

Library 002 Special Functions, addresses for SX

| Cmd | Label | PRP | EQWR | AKA | DISP | DER | RINV | COLCT | RULES | INTG | WHERE | VUNS |
|-----|-----------|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 02F | xIFTE | 1A3 | | | 1A471 | 56D26 | | 1A48A | | | 1A494 | 5A270 |
| 03B | x= | 5C8 | 6DE50 | | 26E47 | 56C13 | 5745F | | 3F13F | | | |
| 03C | xNEG | 5C8 | 6B355 | | 1A9F9 | 56C90 | 574A5 | | 3F135 | | | |
| 03D | xABS | 080 | | | | 56C1D | | | | | | |
| 03E | xCONJ | 0C8 | | | | 56CAE | 57603 | | 3F149 | | | |
| 03F | xPI | 080 | | | | 56D12 | | | | | | |
| 040 | xMAXR | 080 | | | | 56D12 | | | | | | |
| 041 | xMINR | 080 | | | | 56D12 | | | | | | |
| 042 | xCONST..e | 080 | | | | 56D12 | | | | | | |
| 043 | xi | 080 | | | | 56D12 | | | | | | |
| 044 | x+ | 5C8 | 6BA97 | | 1AC75 | 56B91 | 57455 | | 3F103 | | | |
| 045 | xNEGNEG | 500 | 6B3A5 | | 1A9F9 | | | | | | | |
| 046 | x- | 5C8 | 6BA97 | | 1AC75 | 56B9B | 5745F | | 3F103 | | | |
| 047 | x* | 5C8 | 6BA97 | | 1AEDE | 56B87 | 57469 | | 3F10D | | | |
| 048 | x/ | 5C8 | 6C02D | | 1AFF5 | 56BA5 | 57473 | | 3F10D | | | |
| 049 | x^ | 5CC | 6C4D3 | | 1B0E6 | 56D6C | 5747D | | 3F117 | 5B906 | | |
| 04A | rpnXROOT | A | | 1B1CA | | | | | | | | |
| 04B | xXROOT | 401 | 6B8A8 | | | | | | | | | 5A2C5 |
| 04C | xINV | 0CC | | | | 56CEA | 574B9 | | 3F135 | 1B2C8 | | |

| | | | | | | | | | | |
|-----|-----------|-----|-------|-------|-------|-------|-------|-------|-------|-------------|
| 04D | xARG | 080 | | | 56C5E | | | | | |
| 04E | xSIGN | 004 | | | | | | | 1B361 | |
| 04F | xSQRT | 5C4 | 6B86C | | 1B3C9 | 56D4E | 575BD | | 1B3BF | |
| 050 | xSQ | 0C4 | | | | 56D44 | 5759F | | 1B471 | |
| 051 | xSIN | 0CC | | | | 56D30 | 574EB | 3F153 | 1B4F2 | |
| 052 | xCOS | 0CC | | | | 56CCC | 574F5 | 3F153 | 1B54B | |
| 053 | xTAN | 0CC | | | | 56D58 | 574FF | 3F153 | 1B5A4 | |
| 054 | xSINH | 0CC | | | | 56D3A | 57509 | 3F153 | 1B5F3 | |
| 055 | xCOSH | 0CC | | | | 56CD6 | 57513 | 3F153 | 1B642 | |
| 056 | xTANH | 0CC | | | | 56D62 | 5751D | 3F153 | 1B691 | |
| 057 | xASIN | 0CC | | | | 56C68 | 57527 | 3F15D | 1B6E0 | |
| 058 | xACOS | 0CC | | | | 56C40 | 5753B | 3F15D | 1B67B | |
| 059 | xATAN | 0CC | | | | 56C7C | 5754F | 3F15D | 1B7D8 | |
| 05A | xASINH | 0C8 | | | | 56C72 | 57563 | 3F15D | | |
| 05B | xACOSH | 0C8 | | | | 56C4A | 57577 | 3F15D | | |
| 05C | xATANH | 0C8 | | | | 56C86 | 5758B | 3F15D | | |
| 05D | xEXP | 0CC | | | | 56CE0 | 574CD | 3F121 | 5BB6D | |
| 05E | xLN | 0CC | | | | 56CF4 | 574D7 | 3F12B | 1B98B | |
| 05F | xLOG | 0CC | | | | 56D08 | 575D1 | 3F12B | 1BA02 | |
| 060 | xALOG | 0CC | | | | 56C54 | 575F9 | 3F121 | 1BA79 | |
| 061 | xLNP1 | 0C0 | | | | 56CFE | 575E5 | | | |
| 062 | xEXPM | 0C4 | | | | 56CE0 | 57617 | | 1BAEF | |
| 063 | xFACT | 500 | 6BCE5 | | 1BB2A | | | | | |
| 06E | xMOD | 100 | | | 1BE84 | | | | | |
| 09A | xRE | 008 | | | | | | 3F149 | | |
| 09B | xIM | 008 | | | | | | 3F149 | | |
| 0E5 | xAND | 100 | | | 1E7CE | | | | | |
| 0E6 | xOR | 100 | | | 1E854 | | | | | |
| 0E7 | xNOT | 500 | 6B3F5 | | 1E8CB | | | | | |
| 0E8 | xxOR | 100 | | | 1E854 | | | | | |
| 0E9 | x== | 100 | | | 1EA21 | | | | | |
| 0EA | x#? | 100 | | | 1EA21 | | | | | |
| 0EB | x<? | 100 | | | 1EA21 | | | | | |
| 0EC | x>? | 100 | | | 1EA21 | | | | | |
| 0ED | x<=? | 100 | | | 1EA21 | | | | | |
| 0EE | x>=? | 100 | | | 1EA21 | | | | | |
| 0F8 | rpnDER | E | 6CB90 | 1EFD2 | | | | | | |
| 0F9 | xDER | 5A7 | 6CB90 | | 1F00E | 56F36 | 1F024 | | 1F02E | 1F038 5A211 |
| 0FC | rpnINTG | E | 6C596 | 1F223 | | | | | | |
| 0FD | xINTEGRAL | 5A7 | 6C596 | | 1F241 | 57284 | 1F25C | | 5B875 | 1F26B 5A22A |
| 0FE | xSUM | 5A3 | 6C8B6 | | 1F319 | 5728E | 1F334 | | | 1F33E 5A243 |
| 0FF | rpnWHERE | E | 6BD7B | 1F3F3 | | | | | | |
| 100 | xWHERE | 5A3 | 6BD7B | | 1F40C | 5727A | 1F3F3 | | | 1F42A 5A257 |
| 101 | xQUOTE | 121 | | | 1F523 | | 1F533 | | | 5A284 |
| 102 | rpnAPPLY | A | | 1F5C5 | | | | | | |
| 103 | xAPPLY | 181 | | | 1F5D9 | 57298 | | | | 5A298 |
| 104 | xFCNAPPLY | 4A2 | 6B2F6 | | | 56F40 | 1F6CC | | 1F6D6 | |
| 10B | xFORMUNIT | 501 | 6BA97 | | 1FB1D | | | | | 5A2B1 |
| 10C | xPREDIV | 500 | 6C069 | | 26DF3 | | | | | |

(The missing column is for EXPND property that is not used internally)

Library 1792

| No | Addr | Label | NAME | tNAME | P | PARSE | SST/PROMPT |
|----|-------|-----------|--------|---------------|----|-------------------|-------------------|
| 00 | 22EC3 | xIF | IF | | 5 | 23BD2 PARSEIF | 22ED7 PROMPTIF |
| 01 | 22EFA | xTHEN | THEN | | A | | 14C30 SSTTHEN |
| 02 | 22F5B | xELSE | ELSE | | A | | 14C71 SSTELSE |
| 03 | 22FD5 | xIFEND | END | | 3 | | |
| 04 | 22FEB | xALG-> | -> | | 8 | 2419A PARSEALG-> | 14A55 SSTALG-> |
| 05 | 23033 | xWHILE | WHILE | | 9 | 23F79 PARSEWHILE | 148F2 SSTWHILE |
| 06 | 2305D | xREPEAT | REPEAT | | A | | 1494C SSTREPEAT |
| 07 | 230C3 | xDO | DO | | 9 | 23D0D PARSEDO | 148F2 SSTWHILE |
| 08 | 230ED | xUNTIL | UNTIL | | 2 | | |
| 09 | 23103 | xSTART | START | | 9 | 23E7F PARSESTART | 14AE6 SSTSTART |
| 0A | 231A0 | xSTARTVAR | FOR | | 9 | 23E9D PARSEFOR | 14B13 SSTFOR |
| 0B | 2324C | xNEXT | NEXT | | B | | 14B72 SSTSTEP |
| 0C | 23380 | xSTEP | STEP | | B | | 14B72 SSTSTEP |
| 0D | 233DF | xIFERR | IFEERR | | 9 | 24046 PARSEIFERR | 1483E SSTIFERR |
| 0E | 23472 | xHALT | HALT | | 8 | 24320 PARSEHALT | 14BE5 SSTHALT |
| 0F | 2349C | xSILENT' | | | 8 | TrueTrue | 14E2C SSTSILENT' |
| 10 | 234C1 | xRPN-> | | -> | 8 | 2419A PARGEALG-> | 149BF SSTRPN-> |
| 11 | 235FE | x>>ABND | >> | | B | | 14CDF SST>>ABND |
| 12 | 2361E | x<< | << | | 1 | 23DD0 PARSE<< | |
| 13 | 23639 | x>> | | >> | B | | 14D02 SST>> |
| 14 | 23654 | x' | ' | | 8 | 23A6F PARSE' | 14BB3 SST' |
| 15 | 23679 | xENDTIC | | ' | 0 | 23A6F PARSE' | |
| 16 | 23694 | xWHILEEND | | END | B | | 1499C SSTWHILEEND |
| 17 | 236B9 | xENDDO | | END | 1B | | 1490B SSTENDDO |
| 18 | 2371F | xERRTHEN | | THEN | 2 | | |
| 19 | 2378D | xCASE | CASE | | 1 | 23AE2 PARSECASE | |
| 1A | 237A8 | xTHENCASE | | THEN | 9 | failed | 14C99 SSTTHENCASE |
| 1B | 23813 | xDIR | DIR | C\$ GROB XLIB | 2 | | |
| 1C | 23824 | xPROMPT | PROMPT | | 8 | 24339 PARSEPROMPT | 14BFE SSTPROMPT |

\*\*\*\*\*

E. Library Property Fields

The visible romptrs in ROM are arranged in memory as follows:

```
CON(1)  #prop   or      CON(3)  #prop
CON(3)  #libnum
CON(3)  #cmdnum
<romprt body>
<ob/pointer/#>
<ob/pointer/#>
...
```

The prop field contains bit data which is used to indicate different properties of the romptr and may imply an associated element after the romptr body. In the following explanations 'flag nn has data' means that the flag set implies an element.

The element implied can be one of the following:

- A 5 nibble pointer to either of following
- A binary number, the body of which is the address of a valid object
This is used to reference objects in hidden ROM.
- A normal object

ROMPTRS for which libnum >= #700 (%1792)

~~~~~

SST	0x8	has SST data
PROMPT	0x4	has prompt message
	0x3	is end macro
	0x2	is middle macro
PARSE	0x1	has begin macro
	0x0	has stand-alone macro

Order of implied elements: PARSE SST PROMPT

Entries related to macro properties:

```
2699F  ?OKMACRO      ( ob --> flag )
Is ob romp with libnum >= #700?
26886  ?PARSE#      ( romp --> # )      ( GX: 268D6 )
0=stand-alone, 1=begin, 2=middle, 3=end
26906  ?PARSE      ( romp --> ob TRUE / FALSE )      ( GX: 26917 )
Recalls parser property
270AF  ?PROMPT      ( ob --> ob' TRUE / ob FALSE )      ( GX: 2708E )
Recalls prompt string
271B9  ?SST        ( ob --> ob' TRUE / ob FALSE )      ( GX: 281C5 )
Recalls single-step property
```

PARSE: Parser object. A begin macro parser must do the parsing until the end macro token. Required syntax:

```
( obl .. obn #n HXS $string #offset $token
--> obl'.. obn' #n' HXS' $string #offset' $token' flag TRUE
--> obl .. obn #n HXS $string #offset $token FALSE )
```

obl..obn are previously parsed objects, #n their count  
HXS is a token-type table  
\$string = the string being parsed  
\$token = the token to parse  
#offset = offset to next char after \$token (in \$string)

If FALSE is returned error was found  
If TRUE TRUE is returned parsing was succesful  
If FALSE TRUE is returned parsing was succesful but not  
completed (usually an error condition)

Example: PARSEHALT = :: 'Rapndit xHALT TrueTrue ;

SST: Single step program. Syntax:

( <Stack arguments> ROMPTR --> ? )

Debugged stream is in return stack. The usual exit is entry  
GsstFIN.

Example: SSTDO = :: sstDISP @ Displays "DO"  
RDUP @ Duplicates 'current' stream  
GsstFIN @ Exit SST  
;

PROMPT: Prompt string

Example: PROMPTIF = \$ "IF-prompt"

ROMPTRS for which libnum < #700 (%1792)

~~~~~

Commands

~~~~~

	0x8	command flag	- always set for commands
EQWR	0x4	has eqwr entry	
AKA	0x2	has algebraic alias	
DISP	0x1	has display data	- always clear?

#### Functions

~~~~~

| | | | |
|-------|-------|--------------------------|------------------------------|
| | 0x800 | command flag | - always clear for functions |
| EQWR | 0x400 | has eqwr entry | |
| AKA | 0x200 | has algebraic alias | - always clear for functions |
| DISP | 0x100 | has display data | |
| DER | 0x080 | has derivative data | |
| RINV | 0x040 | has inversion data | |
| COLCT | 0x020 | has colct data | |
| EXPND | 0x010 | has expand data | |
| RULES | 0x008 | has rules data | |
| INTG | 0x004 | has integral data | |
| WHERE | 0x002 | has where data | |
| VUNS | 0x001 | has very-unsyminner data | |

Order of implied elements is EQWR ... VUNS

Entries related to command/function properties:

```
?OKINALG      (ob --> ob flag )
Is object allowed in algebraics?
26B40 ?FUNCTION      (romptr --> romptr flag )      ( GX: 26B04 )
Is romptr command prop cleared?
26B73 ?AKA          (romptr --> ob T / romptr F )    ( GX: 26B2E )
Recall AKA property.
26BBD ?DISP         (ob --> ob ob_disp #args T / ob F ) (GX: 26BED)
Recall DISP property. (Other than romp gives FALSE)
26F0D ?DER          (ob --> ob_der   T / ob F )      ( GX: 26F18 )
26F1B ?RINV         (ob --> ob_rinv  T / ob F )      ( GX: 26F26 )
26F29 ?COLCT        (ob --> ob_colct T / ob F )      ( GX: 26F34 )
26F37 ?EXPAN        (ob --> ob_expan T / ob F )      ( GX: 26F42 )
26F45 ?RULES        (ob --> ob_rules T / ob F )      ( GX: 26F50 )
26F53 ?INTG         (ob --> ob_intg  T / ob F )      ( GX: 26F5E )
26F61 ?WHERE        (ob --> ob_where T / ob F )      ( GX: 26F6C )
26F6F ?VUNS         (ob --> ob_vuns  T / ob F )      ( GX: 26F7A )
270AF ?EQWR         (ob --> ob_eqwr  T / ob F )      ( GX: 2708E )
```

EQWR: Equation Writer routine to handle special features.
Structure / Arguments not interesting.

AKA: Algebraic aka word for commands.
Used when function uses a special way to input arguments.
?AKA does ROMPTR@ so to use AKA property use this trick:
Put A ROMPTR as the aka property, the contents of which
should be the wanted aka property. Thus the ROMPTR will 'pass'
the correct property for DoKeyOb.

DISP: Display format as a hex number or returned by a hex number.
Syntax: (--> hxs)

First/Last nibbles act as priorities/terminators

```
0 = Any Argument
1 = Command Name
2 = Symbolic Argument
3 = Quoted Name (Symbolic containing only ID)
4 = (
5 = )
6 = ,
7 = =
8 = ???
9 = Repeat 60 = ANY
A = ???
B = Repeat 370 = QN=ANY
C = ???
D = ???
E = ???
F = ???
```

```
26C65 ?DISPARGS      (GX: 26C8B)
      (ob --> ob #args ob_disp )
Recall argument count & display number for program.
If no ?DISP gives false then UNDEFXLIBERR
```

```

26E56  DISPEXAM          (GX: 26E61)
      (hxs --> NameNot1st? #P1 #P2 Parens? )
      Examine DISP property, returns TRUE if function name is
      not first, priorities and TRUE if parentheses are
      second.
5E9E1  ?DISPINF
      ( ob --> flag )
      Execute ?DISP, if property exists and arg count is
      infinite (#FFFFFF) returns TRUE, else FALSE.

```

Internal DISP values for SX:

| | | | |
|-------|-------------|----------------------|----------------------|
| 26DD2 | DISPARGSINF | HXS 7 0140950 | F(A,B,C,D...) |
| 26DE3 | DISPARGS1 | HXS 6 014050 | F(A) |
| 26DF3 | DISPARGS2 | HXS 8 01406050 | F(A,B) |
| 26E05 | DISPARGS3 | HXS A 0140606050 | F(A,B,C) |
| 26E19 | DISPARGS4 | HXS C 014060606050 | F(A,B,C,D) |
| 26E2F | DISPARGS5 | HXS E 01406060606050 | F(A,B,C,D,E) |
| | | | |
| 1B0E6 | DISP^ | HXS 5 90109 | (HXS 7 9014050) |
| 1AFF5 | DISP/ | HXS 5 80108 | (HXS 7 8014050) |
| 1AEDE | DISP* | HXS 5 80108 | |
| 1BE84 | DISPMOD | HXS 5 80108 | |
| 1AC75 | DISP+ | HXS 5 70107 | |
| 1EA21 | DISP== | HXS 5 60106 | |
| 1E7CE | DISPAND | HXS 5 50105 | |
| 1E854 | DISPOR | HXS 5 40104 | |
| 26E47 | DISP= | HXS 5 00700 | |
| | | | |
| 1BB2A | DISP! | HXS 4 A010 | |
| 1B3C9 | DISPSQRT | HXS 4 010C | (DISPARGS1) |
| 1A9F9 | DISPNEG | HXS 4 0108 | |
| 1E8CB | DISPNOT | HXS 4 0105 | |
| 1FB1D | DISPUNIT | HXS 5 90127 | |
| | | | |
| 1F00E | DISPDER | HXS 7 0134250 | DER QN(SYMB) |
| 1F241 | DISPINTG | HXS C 014060626350 | I(ANY,ANY,SYMB,QN) |
| 1F319 | DISPSUM | HXS C 014370606250 | SUM(QN=ANY,ANY,SYMB) |
| 1A471 | DISPIFTE | HXS A 0140626250 | IFTE(ANY,SYMB,SYMB) |
| 1F523 | DISPQUOTE | HXS 6 014250 | QUOTE(SYMB) |
| 1F5D9 | DISPAPPLY | HXS 9 014360950 | APPLY(QN,ANY,...) |
| 1F40C | DISPWHERE | HXS A 2214370B50 | SYMB (QN=ANY,...) |

DER: Derivation program. Syntax:

```
( Meta ALG_id/lam --> Ob )
```

Meta contains the stack arguments to the function.
 ALG\_id/lam is the variable of derivation embedded into a
 single-object symbolic.

Ob returned should be the single-step derivative of the
 function, as a type suitable in algebraics.

NOTE!!

DERFCN is bugged. If you use it to take a derivative of your
 own ROMPTR which has normal arguments (not infinite) and der
 property the internal program drops the size of Meta instead of

the argument count for xFCNAPPLY. (ROLLDROP instead of 4ROLLDROP).

Only way to get around it is:

```
:: Bind                ( get rid of ALG_id/lam )
  TWO OVER splitup      ( find last args size )
  SWADROP #2-           ( Correct to Meta size )
  lGETABND              ( recover ALG_id/lam )
  <normal operation>
```

Now you have Meta ALG\_id/lam on the stack as you should have.
You can now use larg, parg etc to split Meta if it actually contains multiple args.

RINV: Inversion program. Used by ISOL. Syntax:

```
( Meta1 Meta2 id --> Meta1' Meta2' & continue ISOL)
(                               --> FALSE                               )
```

id = variable to isolate
Meta2 = Isolated so far
Meta1 = Argument of the function

```
Example: RINVLN      = :: ISOLSWAP: xEXP ;
          ISOLSWAP = :: addtprsh 'RRDROP SWAP#1+ ISOLCONT ;
```

COLCT: Colct execution of the function. Used for preventing evaluation of some of the arguments. Syntax is the same as for the function itself, including all error checking.

```
( obl...obn ob --> obl...obn ob FALSE / obl'..obN' TRUE)
FALSE should be returned for default operations, TRUE is
property does the evaluation.
```

EXPND: Expand program for the function.

RULES: Rules menu for the function. Internal RULES menus:
(# references to hidden ROM)

INTG: Integration program for the function.
(Meta #pos --> Meta FALSE / Meta' TRUE)

pos is the position of the interesting function (Not necessarily topmost function!) in Meta. 2 means topmost function in Meta, 3 next object etc.

WHERE: How to do WHERE (|) for the function.

```
( [QN1 ob1 ... QNN obN] [arg1...argM] --> expr)
```

VUNS: Imploding symbolics and finding command alias for functions.

NEEDQN? (Meta #n -->

UNSYM element builds symbolic from Meta:
(Builds arguments to symbolics if necessary)
([expr...expr' arg1..argN] --> [expr..expr' arg1'..argN'])
Expressions must not be touched.

QN element tests if a given argument should be quoted:

Fixed args: (#n --> flag)

Inf args: (#n #total --> flag)

UNAKA element splits algebraic for user level OBJ-> when rpn version of function takes different arguments than alg version.

([arg1...argN] alg\_ob --> [ob1..obN] rpl\_ob)

Internal VUNS values for SX:

| | | | | | |
|-------|-----------|--------------|------------|------------|---|
| 5A211 | VUNSDER | { UNSYMDER | QN1= | UNAKADER | } |
| 5A22A | VUNSINTG | { UNSYMINTG | QN4= | UNAKAINTG | } |
| 5A243 | VUNSSUM | { UNSYMSUM | QN1= | | } |
| 5A257 | VUNSWHERE | { UNSYMWHERE | QNEVEN | UNAKAWHERE | } |
| 5A270 | VUNSIFTE | { UNSYMIFTE | QNDRPFALSE | | } |
| 5A284 | VUNSQUOTE | { UNSYMQUOTE | QNDRPFALSE | | } |
| 5A298 | VUNSAPPLY | { UNSYMAPPLY | QNDRP1= | UNAKAAPPLY | } |
| 5A2B1 | VUNSUNIT | { UNSYMUNIT | QNDRPFALSE | | } |
| 5A2C5 | VUNSXROOT | { NOP | DROPFALSE | UNAKAXROOT | } |

\*\*\*\*\*

F. Internal System Flags

SX:

| | 8 | 4 | 2 | 1 |
|---------|------------|---------------------------|-------------|--------------|
| 70683 | IgnoreAlm? | DA2bEdit? | TypeStack? | NoRollDA2? |
| 70684 | BadMenu? | VarMenu? | ServMode | ShowClock? |
| 70685 | Print\$? | AlgEntry? | InsMode? | UNDO_ON? |
| 70686 | Rebuild? | DA3TempF? | DA2bTempF? | DA2aTempF? |
| 70687 | Stack\$? | LowCase? | BlinkOn? | LASTCMD_ON? |
| 70688 | DA1TempF? | AppError | AppSuspOK | USR1MODE? |
| 70689 | DA3ValidF? | DA2bValidF? | DA2aValidF? | DA1ValidF? |
| 7068A | DA3NoCh? | DA2bNoCh? | DA2aNoCh? | DA1NoCh? |
| 7068B | DA3Bad? | DA2bBad? | DA2aBad? | DA1Bad? |
| 7068C | Appmode | StdKeys? | NAppKeyOk | EditLExists? |
| 7068D | StsKeys | DA2RollF? | Symb\$? | EqwrParser? |
| 7068E | (unused?) | (unused?) | (unused?) | DA1IsStat? |
| 7068F \ | | | | |
| ... | - unused?? | (Used by <-RPL-> library) | | |
| 70696 / | | | | |

For GX start from 80801 onwards, some new flags may have been added.

\*\*\*\*\*

G. RAM pointer area

Pointers that used to be supported are marked with \*.
GX addresses follow the table.

| Addr | Len | Name | Description |
|-------|-------|--------------|---|
| 70000 | 5 | RAMID | ID for RAM ok (?) (#A5C3F set by warmstart) |
| 70005 | 5 | RAMSIZE | RAM size mask (default is #F0000) |
| 7000A | 1 | WSLOGDAT1 | Reason of last warmstart (4*(1+13+4) nibbles) |
| 7000B | 13+4 | WSLOGTICKS1 | Time of last warmstart (ticks) + crc |
| 70030 | x | WSLOGLAST | Used for data movement only |
| 70042 | x | WSLOGLASTWD | Used for data movement only |
| 70052 | 13 | ACCESSTIME | Timer accessed (??) |
| 7005F | 4 | ACCESSCRC | CRC for ACCESSTIME |
| 70063 | 13 | TIMEOUT | Time of timeout?? |
| 70070 | * 1 | TIMEOUTCLK | Timeout on? (F or 0) |
| 70071 | 1 | TIMEOUTCLK? | Timeout set??? |
| 70072 | 13 | TESTTIME | Time initialized by ON-D SIN |
| 7007F | 13 | TESTTIME2 | ? |
| 7008C | 44 | TESTMSG | System test message storage for displaying |
| 700DB | 1 | CARDST | Old CARDSTATUS |
| 700DC | 4 | CARD1CRC | Old card1 CRC |
| 700E0 | 4 | CARD2CRC | Old card2 CRC |
| 700E4 | 3 | RAMsave | Special mask for (old) RAM base and size |
| 700E7 | * 5 | COVERsave | Return address used by restoreiram |
| 700EC | 5 | COVERrstk | Return stack save |
| 700F1 | * 256 | IRAMBUFF | Data transfer area (From hidden ROM) |
| 701F1 | * 11 | GraphPrtHook | |
| 701FC | 512 | RECVBUF | RS-232 receive buffer |
| 703FC | 8 | RECVHEAD | Buffer header (Size, get and put counters) |
| 703FE | 1 | uart_open | Cleared by DOOPENIO |
| 703FF | 1 | | |
| 70400 | 1 | | |
| 70401 | * 1 | uart_handshk | Cleard by DOOPENIO |
| 70402 | 1 | | Cleared by DOOPENIO |
| 70403 | 1 | uart_parity | |
| 70404 | 2 | | |
| 70406 | 1 | uart_ioc | |
| 70407 | | | Port config ids??? |
| 7040D | | | |
| 70418 | | | |
| 70428 | * 4 | CONFTAB | CRC?? |
| 7042C | 1 | PORT1STATUS | [Present? Writeable? Merged? ?] |
| 7042D | 5 | PORT1SIZE | Size mask for port1 |
| 70432 | 5 | PORT1ADDR | Base address for port1 |
| 70437 | 1 | PORT2STATUS | [Present? Writeable? Merged? ?] |
| 70438 | 5 | PORT2SIZE | Size mask for port2 |
| 7043D | 5 | PORT2ADDR | Base address for port1 |
| 70442 | 11 | | |
| 7044D | 5 | PORT0END | End address of port0 |
| 70452 | 5 | PORT1END | End address of port1 |
| 70457 | 5 | PORT2END | End address of port2 |

| | | | |
|-------|-----|------------|--|
| 7045C | 103 | INTRSAVE | Temporary save for registers during interrupts |
| 7045C | 16 | INTRSAVEC | Temp save for C.W where C.A is D1 |
| 7046C | 5 | INTRSAVEp | Temp save for P, Carry, DEC/HEX, SB |
| 70471 | 5 | INTRSAVECA | Temp save for C.A |
| 70476 | 16 | INTRSAVEA | Temp save for A.W |
| 70486 | 3 | INTRSAVEST | Temp save for ST flags |
| 70489 | 16 | INTRSAVEB | Temp save for B.W |
| 70499 | 16 | INTRSAVED | Temp save for D.W |
| 704A9 | 16 | INTRSAVER0 | Temp save for R0 |
| 704B9 | 5 | INTRSAVERS | Temp save for RSTK level 1 |
| 704BE | 5 | INTRSAVED0 | Temp save for D0 |

Also used by UnCover4K & similar entries:

| | | | |
|-------|------|--------------|---|
| 7045C | 5 | TEMPA | Temp save for A.A |
| 70461 | 3 | TEMPST | Temp save for status flags (UnCover4K) |
| 70475 | 15 | GOTOSave | Temp save for C.A RSTK B.A |
| 70484 | 2 | GOTOMode | Temp save for DEC/HEX (xF=DEC) |
| 704C3 | * 3 | ORghost | Output register shadow |
| 704C6 | 5 | VDISPSTART | Address of 1st displayed grob line |
| 704CB | 5 | VDISPEND | End address of display grob |
| ??? | | | |
| 704D5 | 1 | ANN2SAVE | Saved ANNCTRL2 by #3C0 |
| 704D6 | 5 | CLKSPEED | 5 nibbles for clock speed (Hz??) |
| 704DB | 1 | INITEN | Disables some ON keys if HOME is being changed |
| 704DC | * 1 | DISABLE_KBD | 0 = keyboard enabled |
| 704DD | * 13 | KEYSTATE | Status of keypresses |
| 704EA | * 34 | KEYBUFFER | Index for start of key queue, end of queue and queue of keys (16) |
| 7050C | 2 | POPPEDKEY | Storage for key last popped from keybuffer |
| 7050E | 5 | DISPADDRg | Location of display (LCD) |
| 70513 | 3 | LINEOFFSg | Display line offset |
| 70516 | 5 | MENUADDRg | Menu display address |
| 7051B | 2 | LINECOUNTg | Display line counter |
| 7051D | 30 | RSTKSAVE | Temporary save for 6 rstk levels (ON_/) |
| 7053B | 11 | R1R2SAVE | Temporary save for R1.A and R2.A (ON_/) |
| 70546 | 1 | GLINEOFFFLOW | |
| 70547 | 2 | GLINECOUNTg | |
| 70549 | 3 | GLINESAVE | |
| 7054C | 5 | GLINEOFFSAVE | |
| 70551 | * 5 | VDISP2 | Pointer to menu grob |
| 70556 | * 5 | ADISP | Pointer to text grob |
| 7055B | * 5 | VDISP | Pointer to current display grob |
| 70560 | 5 | PDISP | Pointer to ?? grob |
| 70565 | 5 | GDISP | Pointer to graphic display grob |
| 7056A | 5 | TEMPBOT | Pointer to bottom of tempob area |
| 7056F | * 5 | TEMPTOP | Pointer to top of tempob area |
| 70574 | 5 | RETTOP | Pointer to top of rpl return stack |
| 70579 | * 5 | DSKTOP | Pointer to top of data stack |
| 7057E | 5 | DSKBOT | Pointer to bottom of data stack (??) |
| 70583 | * 5 | TEMPENV | Pointer to top of tempenv area |
| 70588 | 5 | LOOPENV | Pointer to top of loop data area |
| 7058D | * 5 | TOUCHTAB | Pointer to menu key info |
| 70592 | 5 | HOMETOP | Pointer to HOME directory |
| 70597 | 5 | PORT0TOP | Pointer to port0 |
| 7059C | 5 | CONTEXT | Pointer to current directory |
| 705A1 | 5 | LCONTEXT | Pointer to last directory |
| 705A6 | 5 | UserKeys | Pointer to userkeys list |

| | | | |
|-------|-----|--------------|--|
| 705AB | 5 | Alarms | Pointer to alarms list |
| 705B0 | 5 | RPLTOP | Saved D0 (Pointer to current RPL stream) |
| 705B5 | 5 | TEMPSAVEA | Temporary save for A.A |
| 705BA | 5 | CKARG1 | Argument1 save by CK<n>(NOLASTWD) |
| 705BF | 5 | CKARG2 | |
| 705C4 | 5 | CKARG4 | |
| 705C9 | 5 | CKARG4 | |
| 705CE | 5 | CKARG5 | |
| 705D3 | 5 | MEMOUTHXS | Pointer to hxs (memout handling memory) |
| 705D8 | 5 | | |
| 705DD | 5 | HISTORY1 | Address of last command |
| 705E2 | 5 | HISTORY2 | |
| 505E7 | 5 | HISTORY3 | |
| 705EC | 5 | HISTORY4 | |
| 705F1 | 5 | | ? |
| 705F6 | 5 | KERRM | Address of kermit error message |
| 705FB | 5 | | |
| 70600 | 5 | EXITMSG | Pointer to user error message |
| 70605 | 5 | AppDisplay | |
| 7060A | 5 | AppKeys | |
| 7060F | 5 | AppExitCond | |
| 70614 | 5 | AppError | |
| 70619 | 5 | CtlAlarm | |
| 7061E | 5 | MenuDef | |
| 70623 | 5 | LastMenuDef | |
| 70628 | 5 | MenuData | |
| 7062D | 5 | MenuRowAct | |
| 70632 | 5 | LabelDef | |
| 70637 | 5 | MenuKeysNS | |
| 7063C | 5 | MenuKeysLS | |
| 70641 | 5 | MenuKeysRS | |
| 70646 | * 5 | ReviewKey | |
| 7064B | 5 | SavedContext | |
| 70650 | 5 | TrackAct | |
| 70655 | 5 | ExitAction | |
| 7065A | 5 | LASTCKCMD | Last command saved by CK<n> |
| 7065F | 5 | KeyOb | |
| 70664 | 5 | PORT0rstk | |
| 70669 | 5 | PORT0save | |
| 7066E | 5 | FREETOP | Saved D.A (Free stack) |
| 70673 | 5 | ERRN | |
| ??? | | | |
| 70679 | * 1 | ATTNFLG | ATTN presses counter |
| ??? | | | |
| 7067E | * 5 | FIRSTPROC | Pointer to rpl interface |
| 70683 | 1 | SNIB1 | System flags |
| ... | | | |
| 7068E | 1 | | |
| 7068F | 16 | UNUSED | Unused nibbles??? |
| 7069F | 5 | STACKMARK | Stack size saved by CK<n>(NOLASTWD) |
| 706A4 | 15 | RANDOMSEED | |
| 706B3 | 1 | ClkOnsave | Saved ClkOnNib |
| 706B4 | 1 | AlarmSet? | |
| 706B5 | 1 | AlarmOff? | |
| 706B6 | 1 | Douse? | |
| 706B7 | 1 | | |
| 706B8 | 5 | LastAlarm | Number of last alarm |
| 706BD | 5 | PASTDUE# | Number of passed alarm |
| 706C2 | 1 | WarnData | [Lowbat(P2) LowBat(P1) LowBat(S) OK] |
| 706C3 | 2 | ANNUNCIATORS | [Busy = 80, alpha=40, right=20, left=10..] |

| | | | | |
|-------|---|--------|--------------|---|
| 706C5 | * | 16 | Systemflags | |
| 706D5 | * | 16 | UserFlags | |
| 706E5 | | 2 | | Unused?? |
| 706E7 | | 5 | FIRSTCHAR | Left offset to first char in a line (scrolling) |
| 706EC | | 5 | EditParts | Number of edit lines (EDITPARTS-1) |
| 706F1 | | 5 | | Not used? |
| 706F6 | | 5 | 1stEditLine | Number of 1st line displayed |
| 706FB | | 1 | | Not used? |
| 706FC | | 1 | HISTORYNUM | Number of commands saved |
| 706FD | | 1 | CKARGNUM | Number of arguments saved |
| 706FE | | ? | | Not used? |
| 706FF | | 5 | LASTERRN | Saved last error code |
| 70704 | | 5 | CURSORPOS | Position (total) of cursor |
| 70709 | | 5 | CURSORLINE | Position (line) of cursor |
| 7070E | | 2 | CURSOROFFSET | Position of cursor in screen |
| 70710 | | 1 | CURSORSTATE | Cursor mode (Cursor/Char drawn) |
| 70711 | | 2 | CurChar | Current character under cursor |
| 70713 | | 40 | CharGrob | Cursor grob for blinking. (Current character) |
| 7073B | | 5 | CURSORSX | Cursor x-location in display (bit columns) |
| 70740 | | 5 | CURSORY | Cursor y-location in display (bit rows) |
| 70745 | | ? | | not used? |
| 7074E | | 1 | Blink | Cursor blink counter |
| 7074F | | 1 | Indent | Indentation counter |
| 70750 | | 1 | RstkSave | Tempary RSTK save during GARBAGECOL |
| 70755 | | 5 | TempenvSave | |
| 7075A | | 5 | | Not used? |
| 7075F | | 5 | | not used? |
| 70764 | | 1 | StSave | Status save |
| 70765 | | 1 | Rstk1Save | |
| 7076A | | 5 | Rstk2Save | |
| 70770 | | 5 | DcompWidth | Decompile width |
| 70775 | | 2 | | not used? |
| 70777 | | ? | | |
| 70793 | | 1 | Lock | Delayed alpha lock indicator |
| 70794 | | 1 | PrgmEntry | PRG mode indicator |
| 70795 | | 1 | | not used? |
| 70796 | | 2 | MatWrWidth | Matrix writer width |
| 70798 | | 5 | MatWrGoDir | Matrix writer go direction |
| 7079D | * | 1 | ClkOnNib | 1 if clock is on? |
| 7079E | | 2 | | |
| 707A0 | | 2 | | |
| 707A2 | | 2 | | |
| 707A4 | | 2 | | |
| 707A6 | | 3 | IoStatus | |
| 707A9 | | 1 | | |
| 707AA | | 2 | | |
| 707AC | | 2 | | |
| 707AE | | ? | | |
| 707B9 | | ? | | |
| 707C7 | | 2 | | |
| 707C9 | | 5 | MenuRow | Menu page |
| 707D3 | | 5 | LastMenuRow | Last menu page |
| 707D4 | | ?????? | | |
| 707D7 | | 2 | | ???? |
| 707D9 | * | 5 | ROMPTAB | Count, list of libnums & addrs |

Followed by display grobs, tempob area, rpl return stack, etc

GX addresses:

| | | | |
|-------|--------------|-------|--------------|
| 80000 | RAMID | 80784 | AppDisplay |
| 80005 | RAMSIZE | 80789 | AppKeys |
| 8000A | RAMSIZE2 | 8078E | AppExitCond |
| 8000F | RAMEND | 80793 | AppError |
| 80010 | WSLOGDAT1 | 80798 | CtlAlarm |
| 80011 | WSLOGTICKS1 | 8079D | MenuDef |
| 80036 | WSLOGLAST | 807A2 | LastMenuDef |
| 80048 | WSLOGLASTTWD | 807A7 | MenuData |
| 80058 | ACCESSTIME | 807AC | MenuRowAct |
| 80065 | ACCESSCRC | 807B1 | LabelDef |
| 80069 | TIMEOUT | 807B6 | MenuKeysNS |
| 80076 | TIMEOUTCLK | 807BB | MenuKeysLS |
| 80077 | TIMEOUTCLK? | 807C0 | MenuKeysRS |
| 800E1 | CARDST | 807C5 | ReviewKey |
| 800E2 | CARD1CRC | 807CA | SavedContext |
| 800E8 | RAMsave | 807CF | TrackAct |
| 800F5 | IRAMBUFF | 807D4 | ExitAction |
| 80319 | RECVBUF | 807D9 | LASTCKCMD |
| 80519 | RECVHEAD | 807DE | KeyOb |
| 8051B | uart_open | 807E3 | PORT0rstk |
| 8051E | uart_handshk | 807E8 | PORT0save |
| 80520 | uart_parity | 807ED | FREETOP |
| 80523 | uart_ioc | 807F2 | ERRN |
| 8052F | PORTST | 807F7 | ATTNFLG |
| 805DB | INTRSAVE | 807FC | FIRSTPROC |
| 805E0 | STsave | 8080E | UNUSED |
| 805F4 | GOTOSave | 8081D | STACKMARK |
| 80603 | GOTOmode | 80822 | RANDOMSEED |
| 80605 | INTRSAVEST | 80831 | ClkOnsave |
| 80608 | INTRSAVEB | 80832 | AlarmSet? |
| 80642 | ORghost | 80833 | AlarmOff? |
| 80645 | VDISPSTART | 80834 | Douse? |
| 8064A | VDISPEND | 80835 | AlarmMode |
| 80654 | ANN2SAVE | 80836 | LastAlarm |
| 80655 | CLKSPEED | 8083B | PASTDUE# |
| 8065A | INITEN | 80840 | WarnData |
| 8065B | DISABLE_KBD | 80841 | ANNUNCIATORS |
| 8065C | KEYSTATE | 80843 | SystemFlags |
| 80669 | KEYBUFFER | 80853 | UserFlags |
| 8068B | POPPEDKEY | 80865 | FIRSTCHAR |
| 8068D | DISPADDRg | 8086A | EditParts |
| 80692 | LINEOFFSg | 80874 | 1stEditLine |
| 80695 | MENUADDRg | 8087A | HISTORYNUM |
| 8069A | LINECOUNTg | 8087B | CKARGNUM |
| 8069C | RSTKSAVE | 8087D | LASTERRN |
| 806BA | R1R2SAVE | 80882 | CURSORPOS |
| 806C5 | GLINEOFFFLOW | 80887 | CURSORLINE |
| 806C6 | GLINECOUNTg | 8088C | CURSOROFFSET |
| 806C8 | GLINESAVE | 8088E | CURSORSTATE |
| 806CB | GLINEOFFSAVE | 8088F | CurChar |
| 806D0 | VDISP2 | 80891 | CharGrob |
| 806D5 | ADISP | 808B9 | CURSORSX |
| 806DA | VDISP | 808BE | CURSORY |
| 806DF | PDISP | 808CC | Blink |
| 806E4 | GDISP | 808CD | Indent |
| 806E9 | TEMPBOT | 808CE | RstkSave |
| 806EE | TEMPTOP | 808D3 | TempenvSave |
| 806F3 | RETTOP | 808E2 | StSave |

| | | | |
|-------|-----------|-------|-------------|
| 806F8 | DSKTOP | 808E3 | Rstk1Save |
| 806FD | DSKBOT | 808E8 | Rstk2Save |
| 80702 | TEMPENV | 808EE | DcompWidth |
| 80707 | LOOPENV | 80911 | Lock |
| 8070C | TOUCHTAB | 80912 | PrgmEntry |
| 80711 | HOMETOP | 80914 | MatWrWidth |
| 80716 | PORT0TOP | 80916 | MatWrGoDir |
| 8071B | CONTEXT | 8091B | ClkOnNib |
| 80720 | LCONTEXT | 80924 | IoStatus |
| 80725 | UserKeys | 80947 | MenuRow |
| 8072A | Alarms | 80951 | KeyRomp |
| 8072F | RPLTOP | 8099E | LastMenuRow |
| 80734 | TEMPSAVEA | 809A3 | ROMPTAB |
| 80739 | CKARG1 | | |
| 8073E | CKARG2 | | |
| 80743 | CKARG3 | | |
| 80748 | CKARG4 | | |
| 8074D | CKARG5 | | |
| 80752 | MEMOUTHXS | | |
| 8075C | HISTORY1 | | |
| 80761 | HISTORY2 | | |
| 80766 | HISTORY3 | | |
| 8076B | HISTORY4 | | |
| 80775 | KERRM | | |
| 8077F | EXITMSG | | |

\*\*\*\*\*

\*\*\*\*\*

PART 2: MACHINE LANGUAGE

1. Introduction
2. General Purpose Routines
3. Memory Handling
4. Popping & Pushing addresses
5. Popping & Pushing flags
6. Binary Integers
7. Hex Strings
8. Character Strings
9. Real Numbers
10. Arrays
11. Library Objects
12. Key Handling
13. Display, Grobs, Annunciators
14. IR, Wire, Printing, Time
15. Miscellaneous

APPENDIX

- A. IO page
- B. Saturn Instruction Set

\*\*\*\*\*

1. Introduction

~~~~~

This sections really offers very little, only lists of subroutines in ROM. For those who have trouble grasping how ml works I can only recommend disassembling elementary ROM routines.

Necessary info:

D = amount of free memory (addresses) between return stack and stack  
D1 = pointer to top of stack, increasing order in memory  
    @D1.A is address of the object on stack level 1  
D0 = pointer to current stream  
    @D0.A is next pointer/prolog of object  
B = pointer to top of return stack, decreasing order in memory  
    @B.A is free, @(B.A-5) is address of top return stack stream  
    @@(B.A-5) is next pointer/prolog in return stack stream

*****

## 2. General Purpose Routines

~~~~~

| | |
|------------------|--|
| SAVPTR | D0 to RPLTOP
D1 to DSKTOP
B to RETTOP
D to FREETOP
Clear carry |
| GETPTR | Above in reverse. Clears Carry. |
| GETPTRLOOP | GETPTR, Loop |
| 3259D GPRplExit | GETPTR, execute rpl at C.A |
| GPMEMERR | GETPTR, Insufficient Memory error |
| D0=DSKTOP | Get new D0 from DSKTOP, uses A |
| D1=DSKTOP | Get new D1 from DSKTOP, uses C |
| 11A58 D1->DSKTOP | Gets old stk1 into D1, uses C |
| STAB0 | (A.W --> R0, B.W --> R1) |
| STAB2 | (A.W --> R2, B.W --> R3) |
| STCD0 | (C.W --> R0, B.W --> R1) |
| STCD2 | (C.W --> R2, B.W --> R3) |
| RCAB0 | (R0 --> A.W, R1 --> B.W) |
| RCAB2 | (R2 --> A.W, R3 --> B.W) |
| RCCD0 | (R0 --> C.W, R1 --> D.W) |
| RCCD2 | (R2 --> C.W, R3 --> D.W) |
| EXAB0 | (A.W <--> R0, B.W <--> R1) |
| EXAB2 | (A.W <--> R2, B.W <--> R3) |
| XYEX | (A:B <--> C:D) |
| ASRW5 | ASR W 5 times |
| ASLW5 | ASL W 5 times |
| CSRW5 | CSR W 5 times |
| CSLW5 | CSL W 5 times |

| | | |
|-------|------------|---|
| | SysPtr@ | Push @C.A, Loop (Pushes A.A by jumping to DOBINT !) |
| 08D14 | SysPtr! | Pop address from stk to @C.A, Loop |
| 08D2F | SysPtrClr | Store 00000 to @C.A, Loop |
| 08D35 | SysPtrClr? | Test if @C.A is 00000, push T/F, Loop |
| 08D75 | SysPtrDo | Continue rpl at @C.A |
| 114B3 | SysBin@ | Save C to A, SAVPTR, Push @C.A as #, Loop |
| 114B5 | SysBinA@ | SAVPTR, Push @A.A as #, Loop |
| 114C9 | SysBin! | Pop # to @C.A, Loop |
| 114E1 | SysBin+ | Increment @C.A, Loop (FFFFFF limit) |
| 114EC | SysBin- | Decrement @C.A, Loop (Rounds to FFFFF) |
| 114F8 | SysBinClr | Store 0 to @C.A, Loop |
| 11481 | SysNib@ | Push @A.A (1 nibble) as #, Loop |
| 114A8 | SysNibA! | Store A.1 to @C.A, Loop |
| 11578 | SysNibClr | Store 0 (1 nibble) to @C.A, Loop |
| 1154F | SysNibSet | Store 1 (1 nibble) to @C.A, Loop |
| 1151D | SysNibSet? | Test if @C.A (1 nibble) <> 0, Push T/F, Loop |
| 13E3D | SysNibS/C | If ST=0 then store 0, else store F to @C.A, Loop |
| 13E6F | SysNibS/C? | Test if @C.A (1 nibble) is <> 0, Push T/F, Loop |
| 13E95 | SysByteD0@ | Pushes @D0 as # (byte), Loop (Assumes SAVPTR done) |
| 13EAA | SysByte! | Pop #, store low 2 nibbles to @C.A, Loop |
| 13EEA | SysByteD0! | DAT0=A B, D0=C, Loop |
| 13EC8 | SysByteClr | Store 00 to @C.A, Loop |
| 13EDE | SysByteInc | Increments @D0 (byte), Loop (FF limit) |
| 13F53 | SysNum@ | Pushes @C.A as # |
| 13F75 | SysNumInc | FFFFFF upper limit |
| 13F1D | SysNumDec | No low limit |
| 13F8B | SysNumDec! | 00001 lower limit |
| 13F35 | SysNum! | Pops # to @C.A |

\*\*\*\*\*

3. Memory Handling

~~~~~

	ROOM	Calculates free memory between data and return stacks. --> C.A = @DSKTOP-@RETTOP Uses D0,A.A
	ADJMEM	Recalculates D in FREETOP after RAM movements ( @FREETOP=ROOM/5 )
	GARBAGECOL	Garbage collection
	DOGARBAGE	If ST=1 10 then GPMEMERR else GARBAGECOL and GETPTR
02D59	GARBAGERETC	Save registers (also A.A), GARBAGECOL Restore registers (also A.A), return to C.A ( C.A --> ?)
03416	GARBAGE1	GARBAGECOL with requirement of room for 1 address SAVPTR GARBAGECOL GETPTR, then if enough room for one address then decrement D and return carry clear, else SETMEMERR Uses R0.
0341D	GARBAGEC	GARBAGECOL with requirement of room for 1 address Uses same method as GARBAGE1 Example: To ensure room for 2 stack elements LC(5) 2 D=D-A A GONC MemOk GOSBVL GARBAGEC MemOk [Now for example D1=D1- 10 is ok]
05F0F	GARBAGE1!	Same as GARBAGE1 but does not use R0. Returns carry clear.
06863	MOVEREFS	Moves references to area A.A-B.A with C.A (Adds C.A)
0686E	MOVEREFS2	Also checks area R0-R1
0681E	MOVE1REF	Checks one reference
	MOVERSD	Moves A.A - return stack downwards C.A nibbles
	MOVERSU	Moves A.A -return stack upwards C.A nibbles
	MOVEDSD	Moves data stack - A.A downwards C.A nibbles
	MOVEDSU	Moves data stack - A.A upwards C.A nibbles
	MOVEUP	Copy downwards C.A nibbles from D0 to D1 D0 and D1 will point to the 'starts' of D0 and D1 Used: A.W C.A P
	MOVEDOWN	Copy upwards C.A nibbles from D0 to D1 D0 and D1 will point to the next locations Used: A.W C.A P
0675C	CLEARUP	Clears C.A nibbles from D1 D1 will point to next loc after the cleared area Used: A.W P
01C20	CLEARUPW	Clears C.A words from D0 D0 will point to the last word Used: A.W

CREATETEMP	Allocates C.A nibbles, carry set if not enough memory --> D0=bottom, D1=top of area --> B.A = C.A = @D1 = offset to previous tempob (= #nibbles+6)
GETTEMP	CREATETEMP with GARBAGECOL if necessary C.A = nibbles --> C.A = nibbles --> D0 = bottom, D1 = top GPMEMERR if not enough memory
MAKE\$	Creates character string to tempob area Does SETHEX, C=C+C A and then MAKE\$N
MAKE\$N	Creates character string to tempob area If not enough mem (even after GARBAGECOL) then memerr C.A = nibbles --> A = nibbles+5 (lenght field of string) --> B = nibbles+16 (lenght of tempob area for \$) --> C = address of stack (from DSKTOP) --> D0 = address of body of \$ (prolog and len are ok) --> D1 = address of stack (from DSKTOP) --> R0 = address of \$ Not used: R1-R4

*****

#### 4. Popping & Pushing addresses

~~~~~

| | | |
|-------|--------------|----------------------------------|
| | DropLoop | Pop stack, Loop |
| 62467 | GPDropLoop | GETPTR, Pop stack, loop |
| 60F88 | 3DropLoop | Pop 3, Loop |
| | 4DropLoop | Pop 4, Loop |
| | PopSavptr | Pop, SAVPTR |
| | PopASavPtr | Pop to A.A, SAVPTR |
| | PUSHA | Push A, Loop |
| | GPPushA | GETPTR, Push A, Clear Carry |
| 02D52 | GCPushALoop | Garbage collection, Push A, Loop |
| | GPOverWrR0Lp | GETPTR, OverWr R0, Loop |
| | GPOverWrALp | GETPTR, OverWr A, Loop |
| 03B65 | OverWrR0Lp | OverWr R0, Loop |
| 03678 | OverWrALp | OverWr A, Loop |
| 610ED | OverWrCLp | OverWr C, Loop |

\*\*\*\*\*

5. Popping & Pushing flags

~~~~~

	popflag	Pop to A.A, if TRUE then set carry
	OverWrTLoop	OverWr TRUE, Loop
	OverWrFLoop	OverWr FALSE, Loop
	GPOverWrTLp	GETPTR, OverWr TRUE, Loop
	GPOverWrFLp	GETPTR, OverWr FALSE, Loop
	OverWrT/FLp	OverWr TRUE/FALSE, Loop
	OverWrF/TLp	OverWr FALSE/TRUE, Loop
	GPOverWrT/FL	GETPTR, OverWr TRUE/FALSE, Loop
	PushTLoop	Push TRUE, Loop (DOTRUE)
	PushFLoop	Push FALSE, Loop (DOFALSE)
	GPPushTLoop	GETPTR, Push TRUE, Loop
	GPPushFLoop	GETPTR, Push FALSE, Loop
	PushT/FLoop	Push TRUE/FALSE, Loop (PushT/F)
	PushF/TLoop	Push FALSE/TRUE, Loop
	GPPushT/FLp	GETPTR, Push TRUE/FALSE, Loop
267D5	GPPushFTLp	GETPTR, Loop to FalseTrue
267DC	PushFTLp	Loop to FalseTrue
	GETPTRTRUE	GETPTR, Do TRUE
	GETPTRFALSE	GETPTR, Do FALSE

*****

## 6. Binary Integers

~~~~~

| | | |
|-------|-------------|--|
| 611B0 | GET# | Gets # to A.A, old D1 is in C.A |
| 62278 | COPY# | Gets # to A.A (Only C affected) |
| | POP# | Pop # to A.A |
| | POP2# | Pop # to A.A and # C.A |
| 6250B | POP2#-DIV2 | Pop # and #,SAVPTR, #-, #2/ (--> A.A) |
| | PUSH# | GETPTR, Push R0 as # |
| 03DC7 | Push#Loop | SAVPTR, R0=A, PUSH#, Loop |
| | PUSH2# | GETPTR, Push R0 & R1 as # |
| | PUSH#LOOP | GETPTR, Push R0 as #, Loop |
| | PUSH#ALoop | GETPTR, Push A as #, Loop |
| 03F0E | PUSH#ACLoop | R0=A, R1=C, then Push2#Loop |
| | Push2#Loop | GETPTR, Push R0 & R1 as #, Loop |
| | Push2#aLoop | GETPTR, Push R0 & A as #, Loop |
| 6226F | ?A=0Loop | Push TRUE if ?A=0 A, else FALSE. Loop |
| 622DD | ?A#0Loop | Push TRUE if ?A#0 A, else FALSE. Loop |
| | Push#TLoop | GETPTR, Push R0 as #, Do TRUE |
| | Push#FLoop | GETPTR, Push R0 as #, Do FALSE |
| | MUL# | (B.A = A.A*C.A) |
| | IntDiv | (A.A/C.A --> A.A C.A) |
| 06A8E | CDIV5 | (C.A /= 5) |
| 603BF | CMUL5 | (C.A *= 5) |
| 60236 | AMUL5 | (A.A *= 5) |

\*\*\*\*\*

7. Hex Strings

~~~~~

53F8D	POPHXS	Pop hxs to A, SAVPTR, Clear Carry
53F77	POP2HXS	Pop hxs to C and hxs to A, SAVPTR, Clear Carry
		P has current wordsize.
	PUSHhxsLoop	Push P+1 nibbles from A as hxs, Loop
5422C	PUSHhxs	Push P+1 nibbles from A as hxs
54021	getwordsize	Fetches current word size to A, Clear Carry
	MPY	Multiply A.W and C.W (--> A.W=C.W) Uses D.W, SB. Returns carry clear

*****

## 8. Character Strings

~~~~~

| | |
|--------------|---|
| GetStrLenStk | D1=addr(\$) --> C.A = length, D1 = body |
| GetStrLenC | C=addr(\$) --> C.A = length, D1 = body |
| GetStrLen | D1=\$ --> C.A = length, D1 = body |

\*\*\*\*\*

9. Real Numbers

~~~~~

%:	A.X = exponent	%%:	A.A = exponent (C.A)
	A.S = sign		A.S = sign (C.S)
	A.M = mantissa		B.W = mantissa (D.W)

unf = undefined value  
inf = infinite value  
uinf = either of above

unf =	000000000000F01	+--inf =	sxxxxxxxxxxxxxF00
	000rr00000000F00		xxxxxxxxxxxxxxxxxx

x = A.W & B.W	r0 = R0.W & R1.W
y = C.W & D.W	r2 = R2.W & R3.W

-----

STAB0	r0 = x	STAB2	r2 = x
RCAB0	x = r0	RCAB2	x = r2
STCD0	r0 = y	STCD2	r2 = y
RCCD0	y = r0	RCCD2	y = r2
EXAB0	x <--> r0		
EXAB2	x <--> r2		
XYEX	x <--> y		

	GETAB1	x = @D1+	2C017	GETCD1	y = @D1+
	GETAB0	x = @D0+		GETCD0	y = @D0+
	PUTAB0	@D0+ = x			
2C066	PUTAB1	@D1+ = x			

2A12A	CKINF	( --> ) If XM is clr, RTN. Else if P#3, undefined result error. Else if infinite not allowed then infinite result err. Else set infinite res. indicator, set x to max
2AEF6	GETANGMODE	( --> ) If RAD then ST=0 4, ST=1 9 If DEG then ST=0 4, ST=0 9 If GRAD then ST=1 4, ST=0 9

-----

29E21	PACKSB	( A,B --> 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
	PACK	( A,B --> A ) ... Converts %% to %. Uses roundup. Obeys and sets flow flags/indicators.
29F29	PACK!	( A,B --> A ) Converts %% to %. Uses roundup. Sets indicators but does not obey flags.
29F76	PACK!!	( A,B --> A ) Converts %% to %. Uses roundup. Does not obey flags nor set indicators.

	SPLITA	( A --> x )	Convert % to %%
2BCAC	SPLITC	( C --> y )	Convert % to %%
	SPLTAC	( A,C --> x, y )	Convert 2 reals to long reals

-----

	POP1%	( %pop --> A )	Pop %, SAVPTR
2A060	POP1%%	( %%pop --> x )	Pop %, SAVPTR
	POP1%SPLITA	( %pop --> x )	Pop %, convert to %, SAVPTR
	POP2%	( %pop1 %pop2 --> A,C )	Pop 2 reals, SAVPTR
2A0B1	POP2%%	( %%pop1 %%pop2 --> x,y )	
29FF8	POP2%SPLITAC	( %pop1 %pop2 --> x,y )	Pop 2 reals as %, SAVPTR

	PUSH%	( A --> %push )	Push A as %, GETPTR
2A24B	PUSH%%	( A:B --> %%push )	Push A:B as %, GETPTR
	PUSH%LOOP	( A --> %push )	Push A as %, GETPTR, Loop
2A235	PUSH%%LOOP	( A:B --> %%push )	Push A:B as %, GETPTR, Loop
52AB7	POPC%	( C%pop --> A:C )	Pop C% (SETDEC)
52B57	POPC%%	( C%%pop --> A:B C:D )	Pop C%% (SETDEC)
52ADB	PUSHC%	( A:C --> C%push )	Push C%
51B81	PUSHC%LOOP	( A:C --> C%push )	Push C%, Loop
52B95	PUSHC%%	( A:B:C:D --> C%%push )	Push C%%
51BA2	PUSHC%%LOOP	( A:B:C:D --> C%%push )	Push C%%, Loop

2AECC	CK%%LP	Checks uinf res, pushes x as %, Loop
2AED7	CKPACK%LP	Checks uinf res, pushes x as % (PACK), Loop
2AEDB	PACK%LP	Pushes x as % (PACK), Loop
2AEE5	CKPACKSB%LP	Checks inf res, pushes x as % (PACKSB), Loop
2AEE9	PACKSB%LP	Pushes x as % (PACKSB), Loop
52F13	PACK!%LP	Pushes x as % (PACK!), Loop

-----

Inputs for following:	x,y	XM=SB=0, SETDEC, P=any
Outputs for following:	x	P=unknown
All use:	A.W B.W C.W D.W P	

Addr:	Name:	Oper:	Uses:	Notes:
	RADDF (%%)	x=x+y	D0	
2B7DC	ADDF (%)	x=x+y	D0	0 result negative if ST10 or ST11
	MULTF	x=x*y		
	DIVF	x=x/y		x/0 -> inf XM P=3 : 0/0 -> unf.07 : XM P=4 SB=0 CC

Inputs for following: x,y P=any Both x and y can be uinf  
Outputs for following: x P=unknown  
All use: A.W B.W C.W D.W P

Addr:	Name:	Oper:	Uses:	Notes:
2B7BE	DORADDF	x=x+y	D0	XM,SB=0 P=3 -> AMDR
2B7C4	DORADDF!	x=x+y	D0	P=3 -> AMDR
2B7D0	DOADDF	x=x+y	D0	XM,SB=0 P=0 -> AMDR
2B7D6	DOADDF!	x=x+y	D0	P=0 -> AMDR
2B915	DOMULTF	x=x*y		XM,SB=0 P=1 -> AMDR
2B918	DOMULTF!	x=x*y		P=1 -> AMDR
2B96E	DODIVF	x=x/y		XM,SB=0 P=2 -> AMDR
2B971	DODIVF!	x=x/y		P=2 -> AMDR

AMDR is an operations dispatcher that first checks if either of the arguments is uinf, if not then executes RADDF, MULTF etc. Else result is formed according to function. For example inf-inf=unf etc.

Note that none of RADDF, ADDF, MULTF, DIVF can handle overflows!!!!  
DIVF can produce unf (0/0) or inf (x/0) but that's it.

Also note that P output is random but this is not such a bad problem since RADDF, ADDF, MULTF, DIVF, GETAB0, STAB0, RCAB0 etc all accept any P as input. PUSH%%LOOP etc all also init P to 0 so you really don't have to worry if you keep working with absolute fields. Main thing to remember is probably to init P before LC(x) commands.

2BD32	TST	tst(x)	CS if X% <= Y%
	TST15	tst(x)	CS if test is true
			P=1 <, P=2 =, P=4 >, P=13 <>
2BD64	POSX?		CS if X nonnegative
2BD6F	NEGX?		CS if X negative
2BD02	INFX?		CS if x is uinf
2BD0E	INFY?		CS if y is uinf
2B9A3	SETINF	x=inf	Preserves sign, rr=C.A, P=3, XM
2BB8C	SETUNF	x=unf	rr=C.B, P=4, XM
2BAA1	AMDR	dispatch	
2BB72	CHOOSEUNF	x=unf	Choose unf from x,y
2BBFC	FRCNUM	P=frcnum	Returns number of fractional digits
			If P=15 then integer/uinf
			If P=14 then none
2B6C2	CKXNORMAL	x=normal	Scrolls mantissa so that digit1<>0
2BC77	XNORMAL	x=normal	No uinf check
2BCC4	CKYNORMAL	y=normal	Scrolls mantissa so that digit1<>0
2BCD9	YNORMAL	y=normal	No uinf check
2BD1A	UNSIGNXY%		Adds 5 to A.XS and C.XS
2BE2D	UNSIGNX		Adds 5 to A.4
2BE40	UNSIGNY		Adds 5 to C.4



	1/X15	$x = \text{inv}(x)$	InP:any XM,SB=0 -> DODIVF!
	RSUB1	$x = x + 1$	InP:any SB=0 -> DORADDF!
	RADD1	$x = x - 1$	InP:any SB=0 -> DORADDF!
	DIV2	$x = x / 2$	InP:any No inf/unf protection
2BBD9	MUL2	$x = x * 2$	InP:any No inf/unf protection
2AA25	MUL100	$x = x * 100$	inP:any No inf/unf protection
	CCSB1		set SB using D.S
	SQRTF	$x = \text{sqrt}(x)$	$x = \text{unf}$ if $x < 0$ , also checks uinf
	RNDC[B]	$x = \text{round}(x)$	If ST1 then TRNC else RND
			Input: C.A = parter
2AAC0	POTF	$x = x^y$	
2AAD5	NROOTF	$x = \text{nroot}(x, y)$	
2B674	FPF	$x = \text{fp}(x)$	
	aMODF	$x = \text{mod}(x, y)$	
2B686	XPONF	$x = \text{xpon}(x)$	
2B68F	LN1+XF	$x = \text{lnpl}(x)$	
2B698	LNf	$x = \ln(x)$	
2B6A1	LOGF	$x = \log(x)$	
2B6AA	EXPF	$x = \exp(x)$	
2B6B3	EX-115F	$x = \text{expm}(x)$	
2B6BC	YX2-15	$x = x^y$	
2B6C5	YX060		
2B6CE	!NROOTF	$x = \text{nroot}(x, y)$	
2B6D7	ANGLEF	$x = \text{ang}(x, y)$	GETANGMODE needed
2B6E0	SINF	$x = \sin(x)$	GETANGMODE needed
2B6E9	COSF	$x = \cos(x)$	GETANGMODE needed
2B6F2	TANF	$x = \tan(x)$	GETANGMODE needed
2B6FB	ATANF	$x = \text{atan}(x)$	GETANGMODE needed
2B704	ASINF	$x = \text{asin}(x)$	GETANGMODE needed
2B70D	ACOSF	$x = \text{acos}(x)$	GETANGMODE needed
2B716	NFACTF	$x = \text{nfact}(x)$	
2B71F	PERMF	$x = \text{perm}(x, y)$	
2B728	COMBF	$x = \text{comb}(x, y)$	
2B731	HYPF	$x = \cosh(x), r = \sinh(x)$	
2B73A	TANHF	$x = \tanh(x)$	
2B743	ASINHF	$x = \text{asinh}(x)$	
2B74C	ACOSHF	$x = \text{acosh}(x)$	
2B755	ATANHF	$x = \text{atanh}(x)$	
2B75E	CEILF	$x = \text{ceil}(x)$	
2B767	FLOORF	$x = \text{floor}(x)$	
	aH>HMS	$x = h > \text{hms}(x)$	
2B779	aHMS>H	$x = \text{hms} > h(x)$	

*****

## 10. Arrays

~~~~~

3551D getrealel

35551 getcmpel

\*\*\*\*\*

11. Library Objects

~~~~~

Rom-Word?

Carry set if copy stk1 is a ROM-WORD

*****

## 12. Key Handling

~~~~~

| | | | |
|-------|--------------|---------------|---|
| | OnKeyDown? | (--> Carry) | |
| | OnKeyStable | (--> Carry) | |
| | Flush | (-->) | Flushes key buffer. |
| | FlushAttn | (-->) | Flushes attn counter. |
| | FlushState | (-->) | Flushes KEYSTATE |
| | 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) |
| 00D9F | PUSHKEY | (C.A -->) | Sets carry if buffer is full.
Else adds key to buffer. |
| 008E6 | ScanKeys | (--> A.W) | Scans keyboard for key presses
Uses: A.W C.W P=0 |
| 009A5 | ScanKeys2 | (--> A.W) | ScanKeys with stable presses only |
| 00768 | SrvcKbd | | Scans keys & SrvcKbdAB |
| | SrvcKbdAB | (A.W -->) | Sets KEYSTATE and KEYBUFFER |
| 00C8D | KeyCodeDown? | (A.A -->) | Is key down? (Code = #KeyCode) |
| 01E79 | ScanAnns | | Scans & sets shift anns |
| 00D14 | LSKeyDown? | | Is left shift down? |
| 00D1D | RSKeyDown? | | Is right shift down? |
| 00D26 | AKeyDown? | | Is alpha key down? |
| 00E0B | AnyKeyDown? | | Is any key down? |
| | KeyInBuff? | (-->) | Keybuffer not empty? |
| 04988 | Attn? | (-->) | Is ATTNFLG <> 0? |
| | ATTNchk | | ATTN exit check with restoreiram |
| 0CA88 | ATTNchk2 | | Exit (via ?ATTNQUIT) if ATTNFLG <> 0 |
| | GX:0CA81 | | Uses C.A, D.A (does SETHEX) |

| | Bit: | 1 | 2 | 4 | 8 |
|-----------|---------|--------|-------|--------|---|
| Nibble: 0 | ON | + | SPC | . | |
| 1 | 0 | ' | - | 3 | |
| 2 | 2 | 1 | A | RShift | |
| 3 | * | 6 | 5 | 4 | |
| 4 | MTH | LShift | / | 9 | |
| 5 | 8 | 7 | SIN | Alpha | |
| 6 | BackSpc | 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 | | | | |

\*\*\*\*\*

13. Display, Grobs, Annunciators

~~~~~

	DispOn	
	DispOff	
01BA5	DISPON	
01BD3	DISPOFF	
01BA9	AnnsOn	
01BD7	AnnsOff	
01BEB	SetBusy	
01BFF	ClearDisplay	
01C15	ClearMenu	
42359	BusyOff	
42333	BusyOn	
	D0->Row1	( --> D0 ) Gets addr of current display (1st row)
	D0->Sft1	( --> D0 ) Gets address of menu grob (1st row)
11642	h->H	( A.A = grob_ysize, C.A = y --> B.A = y_offset) Computes grob y nibble
	w->W	( A.A = x --> A.A = x_nibble ) Computes grob x nibble
11C58	hw->HW	( D1 = grob. x1,y1=R4,R2 --> A.A ) Computes grob nibble ( D1+= 20 )
	makegrob	( R0.A = x, R1.A = y --> D0 = body ) Makes a grob of size x,y Prolog is in D0-20
116B5	grob!	( R0,R1 = (row,col), D0 = grob1, D1 = grob2 --> ) Stores grob1 intop grob2
11BAA	PopArea#	( --> (x1,y1,dx,dy) = R4,R2,R0,R1 ) Pops x1,y1,x2,y2 If x2<=x1 or y2<=y1 carry set
11BED	D1->0Grob?	( --> ) Sets carry if grob in D1 is null (lenght <= 15)
11C06	GrobMarkNok?	( --> ) Clears carry if x1,y1 (R4,R2) exceeds grob (D1) bounds
11C29	FixGrobdxdy	( --> ) Fixes dx and dy so that grob bound are not exceeded
	\$5x7	( D.A B.A C.A D0 D1 --> ) Displays string body in D1 in grob in D0 C.A = chars B.A = xlocation D.A = row lenght in nibbles (usually #22) --> D1 = addr after \$ D0 = location of next char D.A = row length (R* registers unaffected?)
11D92	\$5x7/Big	Same as above, but XM determines font XM=0 --> medium Use RTNSXM for getting big

*****

#### 14. IR, Wire, Printing, Time

~~~~~

settimeout

clrtimeout

SavPtrTime\* (--> C.13) Gets TICKS to C.13

GetTime++ (--> C.13) Gets TICKS to C.13

CloseUart

PSubErr

ChkLowBat

ErrFixEIRU

FixEIRU

\*\*\*\*\*

15. Miscellaneous

~~~~~

### Stream Handling:

5F090	doskip	Exit to rpl SKIP
5F09D	docola	Exit to rpl COLA
	SEMILOOP	Jumps to SEMI+5
	SKIPOB	Skip object in D0, clears ST1, clears carry, P=0 --> D0 = addr past object Used: A.A C.A P ST1
03012	SKIPBODY	Skip body in D0, prolog in A.A, sets ST1 (Only start conditions differ from SKIPOB)
	DupAndThen	Do DUP, then execute next object in stream
6199C	Carrycase	If Carry is set then COLA else SKIP
5F6F9	ObDispatch	Dispatch on meta objects last elements Usage: LC(5) #data-address GOSBVL =ObDispatch <argument list> Where data is a list of pointers to xlibs. Example: #5F7E0: + - * ... And argument list contains argument numbers and pointers to XLIBs/XLIBs. Example: (Assuming 5F7E0) 10000 ROMPTR aaa bbb @ If ... + #n then .. 20000 PTR nnnnn @ If ... - #n then .. 10300 PTR nnnnn @ If ...* any + #n then .. F @ End of data (Examples: RULES menu routines at ~5C000)
5F6F2	ObDispatch1	ObDispatch with data at 5F7E0 +,-,*,/,^,NEG,INV,EXP,ALOG,LN,LOG,PI,%2

### Internal System Flags:

Internal system flag operations expect flag number in C.  
Example for SX:: C.A = #487 --> 70600+87:4

53AF2	testflag	Set carry according to ISysFlag
53B0A	clrflag	Clear ISysFlag
53B1D	toggleflag	Toggles ISysFlag
	setflag	Set ISysFlag
53B45	getflag	Gets ISysFlag addr to D0 (original D0 in A.A) C.5 = bit, P=5
539E6	ISysFlag?	Pushes T/F according to ISysFlag
539E1	ClrISysFlag	Clear ISysFlag, Loop
53A01	TogISysFlag	Toggle ISysFlag, Loop
	SetISysFlag	Set ISysFlag, Loop

	makebeep	( C = msec, D = Hz --> ) Beep. Checks BEEP flag.
	Warmstart	= ON-C
01FC6	Softstart	?
01FD3	Coldstart	= ON-A-F
01FDA	Hardstart	= ON-A-F with NO
0CBEF	GETPTRHEX	Sets P=0, SETHEX, GETPTR, Clear Carry
	AllowIntr	( --> ) Allow interrupts.
	DisableIntr	( --> ) Disable interrupts.
	UnCover4K	( --> ) Uncovers RAM so that low 4K remain visible. Uses mask #FE000 (--> #70000 - #71FFF remains )
0BF7D	UnCoverC	( C.A --> )        SX only Uncovers RAM according to mask in C.A
0BF7F	UnCoverB	( B.A --> )        SX only
	restoreiram	( --> ) Restores RAM to settings before UnCover call.
0C740	UnCov4K&Allow	( --> )        SX only
0C8DE	GOTO:	( --> )        SX only Takes next 5 nibbles from RSTK and jumps there. (Hidden ROM) Example: GOSBVL =GOTO: CON(5) #71111                    (Just an example address)
	AINRTN	( IN --> A ) A=IN
	CINRTN	( IN --> C ) C=IN (For hardware reasons (bug) C=IN must be at even addr)

*****

## APPENDIX

### A. IO Page

The entries marked with * are supported.

Addr:	Name:	
00100	DISPIO	Display bit offset for [DON OFF2 OFF1 OFF0] scrolling and DON (display on)
00101	CONTRAST	5 bit value for contrast. [CON3 CON2 CON1 CON0]
00102	DISPTEST	High bit of contrast + [VDIG LID TRIM CON4] display test bits. [LRT LRTD LRTC BIN]
00104*	CRC	4 nibbles for CRC. Every memory fetch updates CRC.
00108	POWERSTATUS	Low power registers [LB2 LB1 LB0 VLBI] read-only LowBat2 LowBat1 LowBat0 and VeryLowBatIndeed :)
00109	POWERCTRL	Low power detection [ELBI EVLBI GRST RST]
0010A	MODE	read-only
0010B*	ANNCTRL	Annunciator control [LA4 LA3 LA2 LA1] (alarm alpha -> <-)
0010C	ANNCTRL2	Annunciator control [AON XTRA LA6 LA5] (on extra io busy)
0010D	BAU	Serial baud rate [UCK BD2 BD1 BD0] UCK read-only 3 bits = {1200 1920 2400 3840 4800 7680 9600 15360}
0010E	CARDCTRL	Card control register [ECDT RCDT SMP SWINT] (EnableCardDetect RunCardDetect SetModulePulled SoftWareINTerrupt)
0010F	CARDSTATUS	Card status register [P2W P1W P2C P1C] read-only (Port2Writable .. Port1Cinserted) (My English sucks..)
00110*	IOC	IO Control [SON ETBE ERBF ERBZ] (Serial On Enableint&TransmitBuffEmpty Enableint&RecvBuffFull Enableint&RecBuffBuzy)
00111*	RCS	Receive Control/Status [RX RER RBZ RBF]
00112*	TCS	Transmit Control/Status [BRK LPB TBZ TBF]
00113*	CRER	nibble writing anything clears RER
00114*	RBR	Receive Buffer Reg. [byte received] reading clears RBF
00116*	TBR	Transmit Buffer Reg. [byte being transmitted] Writing sets TBF
00118	SRR	Service Request Register read-only
0011A*	IRC	IR Control [IRI EIRU EIRI IRE] IRI read-only
0011C	LCR	Led Control Reg. [LED ELBE LBZ LBF]
0011D	LBR	Led Buffer Reg. [0 0 0 LB0] bits 1-3 read zero
0011E	SCRATCHPAD	Scratch register used by interrupts
0011F	IRAM@	RAM base address usually 7 for SX, 8 for GX
00120	DISPADDR	Display start address write-only



00125	LINEOFFS	3 nibble offset for display	write-only
00128	LINECOUNT	Display line counter	[LC3 LC2 LC1 LC0]
		and miscellanoues	[DA19 M32 LC5 LC4]
0012E	TIMER1CTRL	TIMER1 control	[SRQ WKE INT XTRA]
0012F	TIMER2CTRL	TIMER2 control	[SRQ WKE INT TRUN]
00130	MENUADDR	Menu display start addr	write-only
00137*	TIMER1	1 nibble timer	decremented 16 times/s
00138*	TIMER2	8 nibble timer	decremented 8192 times/s

*****

## B. Saturn Instruction Set

### Field Selection

```

P  [P-P]  -----
WP [0-P]  15:14:13:12:11:10: 9: 8: 7: 6: 5: 4: 3: 2: 1: 0
XS [2-2]  -----
X  [0-2]  |S|                                     |XS|<-B->|
S  [15-15]|<----- A ----->|
M  [3-14]|<----- M ----->|<-- X ->|
B  [0-1]  |<----- W ----->|
W  [0-15]
A  [0-4]

```

### Modifier:

fs = field select character  
rfs = restricted field select character (S,P,WP and XS not allowed)  
d = single nibble field

### Code:

y = reference to a symbol (absolute or relative)  
a = field select in range 0-7  
b = field select in range 8-F  
c = single-nibble length field (load constant)  
f = field select including A field  
n = nibble whose value is d  
m = nibble whose value is d-1  
t = mnemon selector nibble

Note: Sometimes the high bit of a/b determines instruction, not field.

fs: P	WP	XS	X	S	M	B	W	A	fs: P	WP	XS	X	S	M	B	W
a: 0	1	2	3	4	5	6	7	F	b: 1	2	3	4	5	6	7	8
a: 8	9	A	B	C	D	E	-		b: 8	9	A	B	C	D	E	F

rfs: P	WP	XS	X	S	M	B	W	A
f:				3		5	6	7 F
f:				B		D	E	-

# Instruction Descriptions

MNEMON		CODE	CYC	FLG	DESCRIPTION
RTNSXM		00	9	HS0	Return an set HS0
RTN		01	9	---	Return from subroutine
RTNSC		02	9	CRY	Return and set CRY
RTNCC		03	9	CRY	Return and set CRY
SETHex		04	3	---	Set ALU to HEX mode
SETDEC		05	3	---	Set ALU to BCD mode
RSTK=C		06	8	---	Push C(A) onto stack
C=RSTK		07	8	---	Pop stack to C(A)
CLRST		08	5	ST	Clear ST[0:11]
C=ST		09	5	---	Load C(X) with ST[0:11]
ST=C		0A	5	ST	Load ST[0:11] with C(X)
P=P+1		0C	3	CRY	Hex increment of P.
P=P-1		0D	3	CRY	Hex decrement of P.
r=r&s	fs	0Eas	4+D	---	Logical AND
r=r!s	fs	0Eas	4+D	---	Logical OR
RTI		0F	9	---	Return from intr and enable intr handling.
ss=r		10t	19	---	Copy A/C to scratch
ss=r		11t	19	---	Copy scratch to A/C
rsSEX		12t	19	---	Exchange scratch with A/C
d=r		13t	8	---	Copy A/C to D0/D1
d=rS		13t	7	---	Copy nibbles 0-3 of A/C to D0/D1
rdEX		13t	8	---	Exchange A/C with D0/D1
rdXS		13t	7	---	Exchange nibbles 0-3 of A/C with D0/D1
DATn=s	A	14t	17	---	Write A/C to @D0/@D1
DATn=s	B	14t	14	---	Write A/C to @D0/@D1
s=DATn	A	14t	18	---	Read @D0/@D1 to A/C
s=DATn	B	14t	15	---	Read @D0/@D1 to A/C
DATn=r	fs	15ta	16+D	---	Write field of A/C to @D0/@D1
DATn=r	d	15tm	16+N	---	Write nib. d of A/C to @D0/@D1
r=DATn	fs	15ta	17+D	---	Read @D0/@D1 to field of A/C
r=DATn	d	15tm	17+N	---	Read @D0/@D1 to nib. d of A/C
D0=D0+	d	16m	7	CRY	Hex increment of D0 by d
D1=D1+	d	17m	7	CRY	Hex increment of D1 by d
D0=D0-	d	18m	7	CRY	Hex decrement of D0 by d
D0=(2)	nn	19nn	4	---	Load B field of D0 with nn
D0=(4)	nnnn	1Annnn	6	---	Load nib. 0-3 of D0 with nnnn
D0=(5)	nnnnn	1Bnnnnn	7	---	Load D0 with nnnnn
D1=D1-	d	1Cm	7	CRY	Hex decrement of D1 by d
D1=(2)	nn	1Dnn	4	---	Load B field of D1 with nn
D1=(4)	nnnn	1Ennnn	6	---	Load nib. 0-3 of D1 with nnnn
D1=(5)	nnnnn	1Fnnnnn	7	---	Load D1 with nnnnn
P=	d	2n	2	---	Load P with d
LCHEX	nn..n	3cnn..n	3+c	---	Load C with t+1 nibbles beginning at P
RTNC		400	10/3	---	Return if CRY is set
GOC		4aa	10/3	---	Branch if CRY is set
RTNNC		500	10/3	---	Return if CRY is clear
GONC		5aa	10/3	---	Branch if CRY is clear
GOTO		6aaa	11	---	Relative jump
GOSUB		7aaa	12	---	Relative gosub.
OUT=CS		800	4	---	Load nib 0 of OR with nib 0 of C
OUT=C		801	6	---	Load OR with C(X)
A=IN		802	7	---	Load nib 0-3 of A with INPUT
C=IN		803	7	---	Load nib 0-3 of C with INPUT
UNCNFG		804	12	---	Unconfigure a cfg:d device at C(A)

CONFIG		805	11	---	Configure an uncfg<:d device at C(A) that has DAISYIN=1
C=ID		806	11	---	Load C(A) with ID of an uncfg:d device that has DAISYIN=1
SHUTDN		807	6	---	Enters low power state
INTON		8080	5	---	Unmask maskable interrupts
RSI		80810	6	---	Resets interrupt detect circuitry
LAHEX	nn.n	8082cn	6+c	---	Load a with nibbles nn.n beginning at P
BUSCB		8083	7	---	Issues BUSCB bus command
ABIT=0	d	8084n	6	---	Clear bit of A register
ABIT=1	d	8085n	6	---	Set bit of A register
?ABIT=0	d	8086nyy	16+9	CRY	Set CRY if bit is clear
?ABIT=1	d	8087nyy	16+9	CRY	Set CRY is bit is set
CBIT=0	d	8088n	6	---	Clear nit of C register
CBIT=1	d	8089n	6	---	Set bit of C register
?CBIT=0	d	808Anyy	16+9	CRY	Set CRY if bit is clear
?CBIT=1	d	808Bnyy	16	CRY	Set CRY if bit is set
PC=(A)		808C	23	---	Indirect jump to @A.A
BUSCD		808D	7	---	Issues BUSCD bus command
PC=(C)		808E	23	---	Indirect jump ti @C.A
INTOFF		808F	5	---	Mask the maskable interrupts
C+P+1		809	8	CRY	Hex increment of C by P+1
RESET		80A	6	---	Issues RESET bus command.
BUSCC		80B	6	---	Issues BUSCC bus command
C=P	d	80Cn	6	---	Load C(d) with P
P=C	d	80Dn	6	---	Load P with C(d)
SREQ?		80E	7	HS2	Issue POLL bus command and latch system bus into C(0). HS2 is set if C(0)<>0
CPEX	d	80Fn	6	---	Exchange P with C(d)
rSLC		81s	21	---	Left circ nib shift.
rSRC		81s	21	HS1	Right circ nib shift. SB set if wrap<>0
rSRB		81s	20	HS1	Right bit shift. SB set if off<>0
r=r+CON	rfs,d	818fsn	5+D	CRY	Hex increment of register r by d
r=r-CON	rfs,d	818fsn	5+D	CRY	Hex decrement of register r by d
rSRB.F	fs	819as	6+D	HS1	Right bit shift. SB set if off<>0
ss=r.F	fs	81Aatr	6+D	---	Copy field of A/C to scratch
r=ss.F	fs	81Aatr	6+D	---	Copy field of scratch ro A/C
rssex.F	fs	81Aatr	6+D	---	Exchange field of A/C with scratch
PC=A		81B2	16	---	Absolute jump to A(A)
PC=C		81B3	16	---	Absolute jump to C(A)
A=PC		81B4	9	---	Move addr of next instr to A(A)
C=PC		81B5	9	---	Move addr of next instr to C(A)
APCEX		81B6	16	---	Exchange A(A) and PC (next instr)
CPCEX		81B7	16	---	Exchange C(A) and PC (next instr)
CLRHSN	n	82n	3	HSn	Clears HST bits specified by n.
XM=0		821	3	HS0	
SB=0		822	3	HS1	
SR=0		824	3	HS2	
MP=0		828	3	HS3	
CLRHST		82F	3	HS0-3	
?HS=0	n	83nyy	13+6	CRY	Set CRY if HST bits are zero.
?XM=0		831yy	13+6	CRY	Same
?SB=0		832yy	13+6	CRY	Same
?SR=0		834yy	13+6	CRY	Same
?MP=0		838yy	13+6	CRY	Same
ST=0	n	84n	4	STn	Clear bit n of ST
ST=1	n	85n	4	STn	Set bit n of ST
?ST=0	n	86nyy	14+7	CRY	Set CRY if bit n of ST is clear
?ST=1	n	87nyy	14+7	CRY	Set CRY if bit n of ST is set
?P#	n	88nyy	13+6	CRY	Set CRY if P<>n

?P=	n	89nyy	13+6	CRY	Set CRY if P=n
?XXX	A	8Atyy	18+11	CRY	Set CRY if equality test is true
?XXX	A	8Btyy	18+11	CRY	Set CRY if comparison test is true
GOLONG		8Caaaa	14	---	Long relative jump
GOVLNG		8Daaaaa	14	---	Absolute jump
GOSUBL		8Eaaaa	14	---	Long relative gosub
GOSBVL		8Faaaaa	15	---	Absolute gosub
?XXX	fs	9atyy	13+D+6+D	CRY	Set CRY if test on field is true
r=r+s	fs	Aat	3+D	CRY	Addition of registers
r=r+r	fs	Aat	3+D	CRY	Doubling of register
s=r+s	fs	Aat	3+D	CRY	Addition of registers
r=r-1	fs	Aat	3+D	CRY	Decrement register
r=0	fs	Aat	3+D	---	Clearing register
r=s	fs	Aat	3+D	---	Copying register
s=r	fs	Aat	3+D	---	Copying register
rsEX	fs	Aat	3+D	---	Exchanging register
r=r-s	fs	Bat	3+D	CRY	Subtraction of registers
r=r+1	fs	Bat	3+D	CRY	Incrementing register
s=s-r	fs	Bat	3+D	CRY	Subtraction of registers
r=s-r	fs	Bat	3+D	CRY	Subtraction of registers
rSL	fs	Bat	4+D	---	Left nib shift.
rSR	fs	Bat	4+D	HS1	Right nib shift. SB set if off<>0
r=-r	fs	Bat	3+D	CRY	2's or 10's complement
r=-r-1	fs	Bat	3+D	CRY	1's complement
r=r+s	A	Ct	3+D	CRY	Addition of registers
r=r+r	A	Ct	3+D	CRY	Doubling of register
s=r+s	A	Ct	3+D	CRY	Addition of registers
r=r-1	A	Ct	3+D	CRY	Decrement register
r=0	A	Dt	3+D	---	Clearing register
r=s	A	Dt	3+D	---	Copying register
s=r	A	Dt	3+D	---	Copying register
rsEX	A	Dt	3+D	---	Exchanging register
r=r-s	A	Et	3+D	CRY	Subtraction of registers
r=r+1	A	Et	3+D	CRY	Incrementing register
s=s-r	A	Et	3+D	CRY	Subtraction of registers
r=s-r	A	Et	3+D	CRY	Subtraction of registers
rSL	A	Ft	4+D	---	Left nib shift.
rSR	A	Ft	4+D	HS1	Right nib shift. SB set if off<>0
r=-r	A	Ft	3+D	CRY	2's or 10's complement
r=-r-1	A	Ft	3+D	CRY	1's complement

s	0Eas	81s	819as	818fsn	8Atyy	9atyy	8Btyy	9btyy
0	A=B&A	ASLC	ASRB.F fs	A=A+CON rfs,d	?A=B A	?A=B fs	?A>B A	?A>B fs
1	B=C&B	BSLC	BSRB.F fs	B=B+CON rfs,d	?B=C A	?B=C fs	?B>C A	?B>C fs
2	C=A&C	CSLC	CSRB.F fs	C=C+CON rfs,d	?C=A A	?C=A fs	?C>A A	?C>A fs
3	D=C&D	DSLC	DSRB.F fs	D=D+CON rfs,d	?D=C A	?D=C fs	?D>C A	?D>C fs
4	B=B&A	ASRC	ASRB.F fs	A=A+CON rfs,d	?A#B A	?A#B fs	?A<B A	?A<B fs
5	C=C&B	BSRC	BSRB.F fs	B=B+CON rfs,d	?B#C A	?B#C fs	?B<C A	?B<C fs
6	A=A&C	CSRC	CSRB.F fs	C=C+CON rfs,d	?C#A A	?C#A fs	?C<A A	?C<A fs
7	C=C&D	DSRC	DSRB.F fs	D=D+CON rfs,d	?D#C A	?D#C fs	?D<C A	?D<C fs
8	A=B!A		ASRB.F fs	A=A-CON rfs,d	?A=0 A	?A=0 fs	?A>=B A	?A>=B fs
9	B=C!B		BSRB.F fs	B=B-CON rfs,d	?B=0 A	?B=0 fs	?B>=C A	?B>=C fs
A	C=A!C		CSRB.F fs	C=C-CON rfs,d	?C=0 A	?C=0 fs	?C>=A A	?C>=A fs
B	D=C!D		DSRB.F fs	D=D-CON rfs,d	?D=0 A	?D=0 fs	?D>=C A	?D>=C fs
C	B=B!A	ASRB	ASRB.F fs	A=A-CON rfs,d	?A#0 A	?A#0 fs	?A<=B A	?A<=B fs
D	C=C!B	BSRB	BSRB.F fs	B=B-CON rfs,d	?B#0 A	?B#0 fs	?B<=C A	?B<=C fs
E	A=A!C	CSRB	CSRB.F fs	C=C-CON rfs,d	?C#0 A	?C#0 fs	?C<=A A	?C<=A fs
F	C=C!D	DSRB	DSRB.F fs	D=D-CON rfs,d	?D#0 A	?D#0 fs	?D<=C A	?D<=C fs

t	Aat		Abt		Bat		Bbt		Ct		Dt		Et		Ft	
0	A=A+B	fs	A=0	fs	A=A-B	fs	ASL	fs	A=A+B	A	A=0	A	A=A-B	A	ASL	A
1	B=B+C	fs	B=0	fs	B=B-C	fs	BSL	fs	B=B+C	A	B=0	A	B=B-C	A	BSL	A
2	C=C+A	fs	C=0	fs	C=C-A	fs	CSL	fs	C=C+A	A	C=0	A	C=C-A	A	CSL	A
3	D=D+C	fs	D=0	fs	D=D-C	fs	DSL	fs	D=D+C	A	D=0	A	D=D-C	A	DSL	A
4	A=A+A	fs	A=B	fs	A=A+1	fs	ASR	fs	A=A+A	A	A=B	A	A=A+1	A	ASR	A
5	B=B+B	fs	B=C	fs	B=B+1	fs	BSR	fs	B=B+B	A	B=C	A	B=B+1	A	BSR	A
6	C=C+C	fs	C=A	fs	C=C+1	fs	CSR	fs	C=C+C	A	C=A	A	C=C+1	A	CSR	A
7	D=D+D	fs	D=C	fs	D=D+1	fs	DSR	fs	D=D+D	A	D=C	A	D=D+1	A	DSR	A
8	B=B+A	fs	B=A	fs	B=B-A	fs	A=-A	fs	B=B+A	A	B=A	A	B=B-A	A	A=-A	A
9	C=C+B	fs	C=B	fs	C=C-B	fs	B=-B	fs	C=C+B	A	C=B	A	C=C-B	A	B=-B	A
A	A=A+C	fs	A=C	fs	A=A-C	fs	C=-C	fs	A=A+C	A	A=C	A	A=A-C	A	C=-C	A
B	C=C+D	fs	C=D	fs	C=C-D	fs	D=-D	fs	C=C+D	A	C=D	A	C=C-D	A	D=-D	A
C	A=A-1	fs	ABEX	fs	A=B-A	fs	A=-A-1	fs	A=A-1	A	ABEX	A	A=B-A	A	A=-A-1	A
D	B=B-1	fs	BCEX	fs	B=C-B	fs	B=-B-1	fs	B=B-1	A	BCEX	A	B=C-B	A	B=-B-1	A
E	C=C-1	fs	CAEX	fs	C=A-C	fs	C=-C-1	fs	C=C-1	A	CAEX	A	C=A-C	A	C=-C-1	A
F	D=D-1	fs	DCEX	fs	D=C-D	fs	D=-D-1	fs	D=D-1	A	DCEX	A	D=C-D	A	D=-D-1	A

t	10t	11t	12t		r	81Aa0r	81Aa1r	81Aa2r
0	R0=A	A=R0	AR0EX		0	R0=A.F	A=R0.F	AR0EX.F
1	R1=A	A=R1	AR1EX		1	R1=A.F	A=R1.F	AR1EX.F
2	R2=A	A=R2	AR2EX		2	R2=A.F	A=R2.F	AR2EX.F
3	R3=A	A=R3	AR3EX		3	R3=A.F	A=R3.F	AR3EX.F
4	R4=A	A=R4	AR4EX		4	R4=A.F	A=R4.F	AR4EX.F
5	R1=A	A=R1	AR1EX		5	R1=A.F	A=R1.F	AR1EX.F
6	R2=A	A=R2	AR2EX		6	R2=A.F	A=R2.F	AR2EX.F
7	R3=A	A=R3	AR3EX		7	R3=A.F	A=R3.F	AR3EX.F
8	R0=C	C=R0	CR0EX		8	R0=C.F	C=R0.F	CR0EX.F
9	R1=C	C=R1	CR1EX		9	R1=C.F	C=R1.F	CR1EX.F
A	R2=C	C=R2	CR2EX		A	R2=C.F	C=R2.F	CR2EX.F
B	R3=C	C=R3	CR3EX		B	R3=C.F	C=R3.F	CR3EX.F
C	R4=C	C=R4	CR4EX		C	R4=C.F	C=R4.F	CR4EX.F
D	R1=C	C=R1	CR1EX		D	R1=C.F	C=R1.F	CR1EX.F
E	R2=C	C=R2	CR2EX		E	R2=C.F	C=R2.F	CR2EX.F
F	R3=C	C=R3	CR3EX		F	R3=C.F	C=R3.F	CR3EX.F

t	13t	14t	15ta		t	13t	14t	15tm			
0	D0=A	DAT0=A	A	DAT0=A	fs	8	D0=AS	DAT0=A	B	DAT0=A	d
1	D1=A	DAT1=A	A	DAT1=A	fs	9	D1=AS	DAT1=A	B	DAT1=A	d
2	AD0EX	A=DAT0	A	A=DAT0	fs	A	AD0XS	A=DAT0	B	A=DAT0	d
3	AD1EX	A=DAT1	A	A=DAT1	fs	B	AD1XS	A=DAT1	B	A=DAT1	d
4	D0=C	DAT0=C	A	DAT0=C	fs	C	D0=CS	DAT0=C	B	DAT0=C	d
5	D1=C	DAT1=C	A	DAT1=C	fs	D	D1=CS	DAT1=C	B	DAT1=C	d
6	CD0EX	C=DAT0	A	C=DAT0	fs	E	CD0XS	C=DAT0	B	C=DAT0	d
7	CD1EX	C=DAT1	A	C=DAT1	fs	F	CD1XS	C=DAT1	B	C=DAT1	d

*****