

A Few Basics	CKSM (Chk)	DEPND	FINDALARM	IP
ABS	CLKADJ	Derivative (∂)	FINISH	I→R
ACK/ACKALL	CLLCD	DET	FLAGS/list	ISOL
ADD	CLOSEIO	DETACH	FLASHEVAL	
ALARMS	CLVAR	DIAG→	FLOOR	KERRM
ALPHA mode	CLΣ	→DIAG	FONT series	KEY
ALOG	CMD	DIFFEQ	FOR-NEXT	Key Numbers
AMORT/AMOR	CNRM	DIR	FOURIER	KEYTIME
AND	COL-	DISP	FP	KGET
ANIMATE	COL+	DISPXY	FREE	KILL
ANS	→COL	DMS (degrees)	FREEZE	
APPLY	COL→	DOERR	FUNCTION	LABEL
APPROX mode	COLCT	DOLIST	F0λ	LABEL
ARC	COLLECT	DOSUBS		LANGUAGE
ARCHIVE	COLΣ	DOT	Garbage Collection	LASTARG/LAST
ARG	COMB	DO-UNTIL	GET/GETI	lbmol
ARRAY-LIST	COMPLEX mode	D→R	gmol	LCD series
Arrays/vectors	Complex numbers	DRAW	GOR	LIB
ARRY-/→ARRY	CON	DRAW3DMATRIX	GRAD	LIBEVAL
ASM	COND	DRAX	GRAPH	LIBRARY basics
ASN	CONIC	DTAG	Graphics menu	LIBS
ASR	CONJ		GRIDMAP	LINE
Assign(→)	CONLIB	e	→GROB	LINFIT
ATICK	Conn4x basics	E/EEX	GROBADD	LININ
ATTACH	CONST	EDIT/EDITB	GXOR	LIST series
AUTO	CONT	EGV		LNP1
AXES	CONVERT	EGVL	HALT	LOCAL
	COPY	ENDSUB	HEAD	LOGFIT
Backquotes	CORR	ENTRY	HEADER series	LQ
BAR	COV	EQ	Helptext (in pgms)	LR
BARPLOT	C→PX	EQ→	Hidden dir	LSQ
BASE	CR	EQLIB	HIST	LU
Batteries	C→R	=/== (equals)	HISTOGRAM	MANT
BAUD	CRDIR	EQW	HISTPLOT	MATCH
BEEP	CRLIB	ERASE	HMS series	Matrix Writer
BESTFIT	CROSS	ERR0		MAX
Binary Intgrs	CSWP	ERRM	i	MAXR
BINS	Cursor keys	ERRN	IDN	MAXΣ
BLANK	CUSTOM	EVAL	IF-THEN	MCALC
BOX	Custom Enter	EXACT mode	IFERR	MEAN
B→R	CUT	EXITED	IFFT	MEM
Brackets	CYLIN	EXP	IFT	MEMORY basics
BUFLEN		EXPAN/EXPAND	IFTE	MENU
BYTES		EXPFIT	IM	MENUXY
	DARCY	EXPM	INCR (Increment)	MERGE
CASCFG	DATE/DATE+	EYEPT	INDEP	MIN
CASDIR	→DATE		Infinity (∞)	MINEHUNT
CASE	dB	FACT	INFORM	MINIFONT
CEIL	DEBUG	factorial (!)	INPUT	MINIT
CENTR	DDAYS	FANNING	INT	MINR
%CH,%	DECR	FAST3D	Integers	MINΣ
CHARS	DEF/DEFINE	FFT	Integral (∫)	MITM
CHR	DELALARM	FILES/FILER	INV	MOD
	DELAY	FINANCE	IOPAR	MODE screens
	DELKEYS			

Mode-Flag menu	PPAR	RKF series	STARTEQW	2D/3D
Mode-FMT menu	PREDV,X,Y	RL/RLB	STARTERR	256 MENU
Mode-Misc menu	PRINT series	RND	STARTOFF	TYPE
MROOT	Programming basics	RNRM	STARTUP	
MSGBOX	PROMPT	ROMUPLOAD	STAT	UBASE
MSLV	PROOT	ROOT	STEQ	UFACT
MSOLVR	PARSURFACE	→ROW	STO	UNBIND
MUSER	PRTPAR	ROW→	STOALARM	UNDO
	PSDEV	ROW+/-	STO arithmetic	→UNIT
n1, n2	PURGE	rpm	STOF	Unit basics
→NDISP	PUT/PUTI	RSBERR	STOKEYS	USER
NDIST	PVAR	RREF series	STOΣ	UTPC
NEG	PVARS	RSD	STOVX	UTPF
NEWLINE	PVIEW	RSWP	STREAM	UTPN
NEWOB	PWRFIT		STR series	UTPT
NEXT	PX→C		STWS	UVAL
NXT		s1,s2	SUB	
NOT	→Q/→Qπ	SAME	SVD/SVL	V→,→V2,→V3
NOVAL	QR/qr	SBRK	SYSEVAL	VAR
NSUB	QUAD	SCALE series		VARS
NUM	QUOTE	SCATTER/SCATR	%T	VER/VERSION
→NUM		SCHUR	TABLE	VISIT/VISITB
Numerical/Symbolic	RAND	SCLΣ	→TAG	VPAR
NUM.SLV	RANK	SCONJ	TAIL	VTYPER
NUMX/NUMY	RANM	SCROLL	TakeOver	
NΣ	RATIO	SD card basics	TAYLR	WAIT
	R→B	SDEV	TBLSET	WHERE ()
OBJ→	R→C	SEND	TDELTA	WHILE
OFF	RCEQ	SEQ	TETRIS	WIN
Old SYMB menu	RCI/RCIJ	SERVER	TEXT	WIREFRAME
ON+ Codes	RCL	Shortcut keys	Text Editors	WSLOG
OPENIO	RCLALARM	SHOW	Text to Grob	
OR	RCLF	SIDENS	TICK	XCOL
ORDER	RCLKEYS	Sigma (Σ	TICKS	XGET
	RCLMENU	SigmaDAT (DAT)	TIME/→TIME	XLAT
PARAMETRIC	RCLΣ	ΣLINE	TINC	XMIT
PARITY	RCWS	SigmaPAR (ΣPAR)	TLINE	XOR
PATH	R→D	Sigma stats	TMENU	XPON
Pauses	RDM	SIGN	TOFF	XPUT
PCOEF	RDZ	SINV	TOOL/EDIT menu	XRECV
PCONTOUR (Ps-)	RE	SIZE	TOT	XRNG
PCOV	REAL Mode	SL/SLB	TPAR	XROOT
PDIM	Real numbers	SLOPEFIELD	TRACE	XSEND
%	REC�	SNEG	TRAN	XSERV
%T	Recovery	SNRM	TRANSFERRING	XVOL
PERM	RECT	SOLVEQN	TRANSIO/XLAT	XXRNG
PEVAL	Rect/Polar	SOLVING basics	Translations	
PGDIR	RECV	SORT	TRIG keys	YCOL
PICT/PICTURE	REF	SPHERE	TRN	Y=
PINIT	Reflashing ROM	SR/SRB	TRNC	YRNG
PIX series	RENAME	SRAD	TRUTH	YSLICE
PKT	REPL	SRECV	TSTR	YVOL
Plots	RES	SREPL	TVARS	YYRNG
PLOT command	Reset	SREV	TVM	
PMAX/PMIN	RESTORE	Stack Commands	TVMBEG	ZFACT
POLAR	REVLIST	START	TVMEND	ZPAR
POS	R→I	STARTED	TVMROOT	ZVOL

Symbols:

{ } ∂ \leq \geq $=/=$ $>$ ∞ (infinity) \int \backslash $<$ \neq (not eq.) %CH,%
 π (pi) [] () { } $\langle \rangle$ $::$ +/- \rightarrow (Assign) Σ Σ DAT Σ LINE Σ PAR Σ stats
 _ | (WHERE) $x\sqrt{y}$ (XROOT) APPENDIX

NAVIGATING:

Click on any index or reference link. There's also a scrolling index at the left in Reader.

To **return to the Index**, click the Index link at the top of any page, or click the up arrow with the line over it on Reader's toolbar (next to the create-PDF icon).

To **return from a link** to where you were before, click the Back button (left arrow) on the upper left of the Reader toolbar (to the right of the page number).

A Few Basics

First, tips on HLP49, then some HP49/50 commands. HLP49 is for RPN mode on the HP49G, HP49G+ and HP50G. It's too big for the 48GII and currently there's no version for the HP48.

CNFIG program (optional)

Allows Restart on Exit, Minifont Index, and/or Port 2 instead of the SD card. You can press it at any time to display the current config settings.



See the zip file for ways to transfer the datafiles to port 2. (See link below if you don't have it)

UNINSTALL program

Lets you uninstall any or all of the hidden config flags, Port 2 datafiles, or the HLP49 library. It can't delete the HLP49 folder on the SD card, which must be removed with a cardreader.

APPENDIX

The [Appendix](#) contains cross-references to programs and utilities that are in other entries. The HP49/50 index will jump to it if you press the Space key in alpha mode, or press RS-down cursor.

► The HLP49 library for the HP49/50 series is a free download from hpcalc.org.

USING THE EXAMPLES

The larger programs use ASCII symbols for HP symbols — they begin with the %%HP header. The program brackets, for instance, will be \<< and \>>. The smaller examples don't have these because it's faster to type them in. *If there's no %%HP header, it's meant to be typed in.*

The easiest way is to install HLP49 on your calculator. Programs can be copied and pasted to the stack by marking with BEGIN and END and then COPY; on the stack, press PASTE ENTER and they're ready to run. See the previous page for the download link.

► To copy into the Emu **emulator** on the same computer - *omit the %%HP header*. **Copy** and open the emulator - press **EDIT>Paste Stack**. Run the IN program from the [Translations](#) entry.

► To copy and transfer files from this PDF, *include the header* and paste the program into a textfile (Notepad will do), give it any name, and use the [Conn4x](#) transfer program to send it to the calculator. Symbols will be translated in either text or binary modes.

► To use the SD card, *omit the %%HP header*. Copy the file to the card, and on the calculator, enter :3:FILENAME RCL (include any extension, but SD card filenames are not case-sensitive). Run the IN program from the **Translations** entry. STO with your choice of name.

TERMS FOR KEYS

RS	–	rightshift
LS	–	leftshift
LS+	–	hold leftshift
RS+	–	hold rightshift
ON+	–	hold ON key

► IMPORTANT

A plus sign means press and hold the shift key as you press the other key, as in LS+MODE.

RECOMMENDED (strongly)

Set flag 117 to get softkey menus instead of Choose boxes.

INDEX KEYS

The main index can jump to letters or numbers, the same way as CAT.

THE ENTER KEY IN THE INDEX

In the index, pressing ENTER is the same as OK. In an entry, ENTER is the same as ON (exits).

ITEMS NOT COVERED

The GAMMA, PSI, and Psi commands in MTH-NXT-SPECIAL have Help entries in CAT.

► Some of the key functions are self-evident and don't have an entry.

SIN, COS, etc

All these are in [TRIG keys](#).

KNOWN BUG

Rarely, the CHOOSE command (the main index) will not exit properly and will put 'EXITED' on the stack if it exists. The header may not be restored. This does not happen enough for proper tracing.

Special thanks to the members of Usenet's comp.sys.hp48 for all the tips and advice, especially John H Meyers, who is widely quoted in HLP49.

HP49G/49G+/50G

The HP49G is compatible with the HP49G+/50G except that it lacks the SD card and the extra screen area for the header. The 49G+ and the 50G are functionally identical except for the 50G's added serial port (the 50G is also powered by the USB cable when it's connected).

The HP48GII is really a subset of the 49G+, but its memory is too small for large programs like this one.

BASIC SHORTCUT KEYS

For more, see the entry for [Shortcut keys \(Shifthold\)](#).

LS+MODE: Mode softkeys
 LS+TOOL: Real/Complex toggle
 LS+NXT: last menu
 LS+VAR: Home dir
 RS+ENTER: Exact/Approx toggle
 LS+F1-F6: Plot screens
 LS-ALPHA: toggle upper/lower case in alpha mode



CURSOR Shortcuts

For more, see the entry for [Cursor keys](#).

In stack mode:

Right: SWAP levels 1 and 2
 Left: PICTURE (graphics)
 Up: Interactive stack
 Down: EDITB (best editor)
 Backspace: DROP Lev 1

NXT: next menu page
 LS-NXT: prev. menu page
 LS+NXT: prev. menu
 LS+VAR: Home dir
 LS-down: EDIT (text editor)

THE STACK

The stack has as many levels as available memory will allow. Level 1 is called X, Level 2 is Y, and so on. Example: the Y^X key returns Level 2 raised to the power of Level 1:

5 3 → 125 (ie, 5^3)

For manipulating the stack, see the entry for [Stack Commands](#).

COMMAND LINE

In stack mode, any new entry will begin at the lower left below the stack. This is the command line, and it's also an editor - you can make any changes before pressing ENTER to put the entry on the stack (TOOL has more editing commands).

You can have multiple entries on the command line:

```
11 "AB" 'CD' 22 ENTER → 11
                        "AB"
                        'CD'
                        22
```

ALPHA mode

The **ALPHA mode** is locked on with two presses of the ALPHA key (flag -60 clear) or a single press (flag -60 set). Alpha mode is cancelled if you press ENTER or ON.

Upper/lower case is toggled with LS-ALPHA (in alpha mode). This is cancelled whenever you press ENTER or ON. (The default is upper case.)

For a single character in upper or lower case, press LS once in alpha mode.

VARs

The term for both the variables in algebraics (A,B,X,Y, etc) and the named, stored object using the STO command.

Note that varnames are case-sensitive – 'x' and 'X' are two different variables. (Note that SD card filenames are not case-sensitive.)

VX

STOVX stores one or more characters in the VX variable in the Home dir and displays it (them) in the header. This becomes the solving var for commands such as SOLVEVX or INTVX. The default is 'X' and can be restored by running **CASCFG** or 'X' **STOVX**. See also **SOLVING basics**.

MATH COMMAND STRUCTURE

In stack mode, commands act directly on the object(s) on the stack or command line. In programming, command keys insert the command name: « SIN »

In algebraic expressions (single quotes ' '), the command is entered in algebraic form: 'SIN()'. It then waits for you to enter the desired variable between the brackets.

GRAPHICS

Pictures, plots, etc, are called grobs (graphic objects). See **Graphics menu**, **PICT/PICTURE**, **Plots**.

ON+C

IMPORTANT: Reboot (warmstart). Use for exiting diags/format, installing new libraries, etc. See **Reset** for more detail. See also **ON+ Codes**.

MODES

There are several main entry modes, Numeric (flags -2,-3 set) and Symbolic (flags -2, -3 clear). Symbolic is required whenever there's an unknown var (such as X). "Undefined name" is the usual error message for the wrong mode. See **Numerical/Symbolic modes**.

There are two main solving modes, Exact (= sign in the header) and Approximate (~ in the header). In exact mode decimals appear as fractions if Symbolic is set. RS+ENTER toggles these, or you can use MODE-CAS-Approx. See [APPROX mode](#), [EXACT mode](#).

There are examples below under the **Number Type** examples.

FLAGS

In the **MODE-Flags** menu, a flag is clear if there's no checkmark, and set if there is one. System flags on the stack are always negative (this isn't shown in the MODE flag browser). See also the entry for [FLAGS - list of flags](#).

STORING AN OBJECT

With the obj on Lev1, enter 'name' **STO**. If the name exists, you can also press LS-namekey. To store a library in a port, put the lib on Lev1 and enter the port number, **STO**. Port numbers are 0 (HomeDir - RAM), 1 (RAM), 2 (flash), and 3 (SD). See [MEMORY basics](#).

To store any other object in a port, put it on the stack and enter :portnum:name. Example for port2:

```
Lev2: << your program >>
Lev1: :2:myprog
```

Note that quote marks are not required after a port number.

RECALLING

If the name of the variable is on the stack, press **RCL** to put the contents on the stack.

If the name is showing in a menu, press RS-name. RS-name is the same as RCL-name.

To recall from a port:

```
:portnum:varname
```

Example: :3:MYVAR **RCL** recalls 'MYVAR' from the SD card.

WILDCARD

The ampersand & (char 38, alpha-LS-ENTER) can be used to search all ports if you know the varname:

```
:&: MYVAR → searches ports 0-3
```

For more on the wildcard, see the entries for [MATCH](#) and [RCL](#).

QUOTING

To put an existing name on the stack, press the ' key (single quotes), then type the name or press its namekey. You can't quote a calculator command or a library name.

If the name doesn't exist, the calculator will add singlequotes. If it does exist, the contents of the var are recalled.

```
X ENTER → 'X'
```

SYMBOLS/ASCII

See **Translations**.

EDITOR COMMANDS

See the **TOOL/EDIT** entry.

PLOT COMMANDS

For plot commands above keys A-F, press leftshift and the key at the same time. Each plot key has its own entry. See also **Plots**.

OBJECT TYPES

See the **TYPE** entry.

SD CARD

See **MEMORY basics** and **SD card basics**.

NUMBER TYPE EXAMPLES (See also **Brackets**)

INTEGERS: 5 123

REALS: 5. 5.1

COMPLEX: (5.,1.)

APPROXIMATE/EXACT MODES:

You can toggle these with RS+ENTER.

EXACT, flags -2,-3 set:

5 Enter 8 / → .625

EXACT, flags -2,-3 clear:

5 Enter 8 / → 5/8

5/8 →NUM → .625

APPROX, flags either way:

5. Enter 8. / → .625

RESOURCES

The best resource for programs, docs, etc, is Eric Rechlin's amazing **hpcalc.org**. There are thousands of free downloads.

The best resource for general advice and troubleshooting is the HP48/49/50 community at Usenet's comp.sys.hp48. You can get there from **Google Groups**. See also **www.hp.com/calculators/**.

HLP49 and its associated files are copyright © 2011 by Bill Markwick v6.0 April, 2011
bmark37@hotmail.com

ABS

Returns the absolute value of an argument. Program syntax is ABS(n).

$$n \rightarrow |n|$$

With a complex number or numeric array, returns the root of the sum of the squares:

$$\begin{aligned} (3., 4.) &\rightarrow 5. \\ [3. \ 4. \ 12.] &\rightarrow 13. \end{aligned}$$

To be sure of getting a numeric result instead of a symbol, use real numbers rather than integers. Flag 3 should be clear for symbolics. Be sure that flag 119 is clear (Rigorous on). If Rigorous is off, ABS(X) has no effect on X. (Flag 119 also affects the **SIGN** command.)

Results are dependent on the settings of **COMPLEX mode** and **APPROX. mode**.

Cmplx, Approx:

$$'X^2' \rightarrow |X^2|$$

Cmplx, Exact:

$$'X^2.' \rightarrow '\sqrt{(\text{RE}(X^2)^2 + \text{IM}(X^2)^2)}$$

Real, Exact:

$$['A' \ 'B'] \rightarrow '\sqrt{(A^2+B^2) '}$$

Cmplx, Exact:

$$['A' \ 'B'] \rightarrow '\sqrt{(A \rightarrow \text{CONJ}(A) + B * \text{CONJ}(B)) '}$$

Real, Approx:

$$['A' \ 'B'] \rightarrow '\sqrt{(B*B+A*A) '}$$

Some experimenting is required.

Access: LS-Z,
 LS-MTH-NXT-CMPLX,
 RS-CMPLX

ACK/ACKALL

ACK acknowledges the oldest past-due alarm. ACKALL acknowledges all past-due alarms.

If flag -43 is clear (the default), unacknowledged repeat alarms are rescheduled.

If flag -44 is clear (default), acknowledged alarms are deleted.

Access: RS-TIME-TOOLS-ALRM,
 RS+TIME-ALRM

ADD

Adds two lists of equal size, element to corresponding element, or adds a number to each element of a list.

```
{1 2 3} {2 4 6} → {3 6 9}
```

```
{1 2 3} 5 → {6 7 8}
```

Note the difference from the plus sign:

```
{1 2 3} {4 5 6} + → {1 2 3 4 5 6}
```

A shortcut:

+/- - is the same as ADD.

See also AUGMENT in CAT.

Access: LS-MTH-LIST

Alarms

Single or repeating alarms can be set for notification or to run a program. For example, if you find that the realtime clock is fast by two seconds a day, you can set a daily alarm to adjust it.

Press RS-TIME and select Set Alarm. With Message selected, enter:

```
« -2 CLKADJ "Clock adjusted" OFF »
```

Select Time and enter an alarm time when the calc is unlikely to be used. Select Date and enter the next day's date. Select Repeat-CHOOS and select 1. A time period field will appear. Select CHOOS and Day. Once a day the alarm will adjust the clock and put the confirmation string on the stack (you can delete the string from the program if you prefer).

For alarms that sound a tone, flags -56 and -57 should be clear to ensure that the beeper is turned on. The beeper is almost inaudible. See **BEEP** for a program to make it more noticeable.

The alarms can be checked with Browse Alarms.

Repeating Alarm:

It's possible to specify an alarm repeat time short enough that you can't cancel it. Press and hold ON and then press 9 to break out of the loop. The softkey alarm commands can be accessed with Tools or RS+TIME.

See also **ACK/ACKALL**, **STOALARM**, **RCLALARM**, **DELALARM**, **FINDALARM**. The alarm number can be counted with Browse Alarms from RS-TIME.

Access: RS-TIME (msgbox),
RS+TIME (softkeys)

ALPHA mode

The alpha mode is locked on with two presses of the ALPHA key (flag -60 clear) or a single press (flag -60 set). Alpha mode is cancelled if you press ENTER or ON.

Upper/lower case is toggled with LS-ALPHA (in alpha mode). This is cancelled whenever you press ENTER or ON. (The default is upper case.)

For a single character in upper or lower case, press LS once in alpha mode.

USER mode is turned on with LS-ALPHA (when not in alpha mode).

In the **FILER**, ALPHA twice turns on the **Search** mode - the index will jump to any letter of the alphabet. Another press of ALPHA turns it off. See also **ENTRY**.

Access: ALPHA key

ALOG

Antilog - the program version of 10^X . Expression syntax is 'ALOG(X)'.

```
'ALOG(2)' EVAL → 100
« 2 ALOG » EVAL → 100
```

Access: LS-EEX

AMORT/AMOR

Amortizes an amount based on the Time Value of Money settings. See **TVM** for the other required values.

$n \rightarrow$ principal interest balance

where:

n = # of payments to date

principal = amt paid toward principal

interest = amt of interest paid

balance = amount to pay remaining

Access: RS+7-TVM (softkeys),
LS-9 (INFORM style)

AND

Logical AND of two args. The ones at the left side below can be any non-zero value.

```
1 1 → 1.
0 0 → 0.
1 0 → 0.
0 1 → 0.
```

Reals or integers are taken as zero or non-zero:

```
5.0  →  0.
5 3   →  1.
```

In bitwise calculations with user binary integers, processing is done bit by corresponding bit, so that

110001b # 101001b → # 100001b is the same as

```
110001
101001
-----
100001
```

BASE should be in the BIN mode. Leading zeros are not displayed on the stack.

It can be used in an algebraic:

```
'X>2 AND Y<4'
```

This will return 1 or 0 when evaluated, depending on the values stored in X and Y.
It can also be used as an operator in UserRPL programs. The syntax is (for example):

```
IF <test1> <test2> AND THEN <procedure> END
```

If both test1 and test2 evaluate to non-zero, the procedure is evaluated.

The following checks for a string on Lev 2 and an integer on Lev 1:

```
%%HP: T(3)A(D)F(.);
\<<
  IF DUP TYPE 28. == ROT DUP
TYPE 2. == ROT AND
  THEN SWAP "<program>"
  ELSE SWAP
  END
\>>
```

Access: LS-PRG-TEST-NXT,
 RS-BASE-NXT-LOGIC

ANIMATE

Displays grobs in sequence. Grobs remain on stack.

```
L2,etc: grobn ... grobl
L1: number of grobs
```

The above provides a full-screen animation with a fixed display rate that cycles until cancelled. To change this, use the following list on Level 1 instead:

```
{ n { # X # Y } d rep }
```

where n = number of grobs

X # Y = the upper left

start point in pixels

d = grob duration, seconds

rep = number of repetitions (0 = continuous, maximum one million)

If you use the list, all of the parameters must be present.

See the entry for [YSLICE](#) for an example of animating.

0 ANIMATE displays [PICT/PICTURE](#) without cursor or menu, the same as {} [PVIEW](#).

Access: LS-PRG-NXT-GROB-NXT

ANS

Same as LASTARG when used from the keyboard. Doesn't give nth argument in RPN. Not a programmable command in RPN - use LAST or [LASTARG/LAST](#) if needed.

→ arguments from last cmd

The ARG (LASTARG) function must be enabled with LS+Mode-Misc or by clearing flag -55 (the default is enabled).

► (Note that there is a keyboard ARG. This is for angles and is unrelated to LASTARG.)

Accidental Overwrites: if you act before running any other command, you can correct for accidental use of STO by pressing ANS. Assume that VAR1 containing "Original" was overwritten with "Overwrite":

```
ANS → "Original" 'VAR1'
STO ANS → "Overwrite" 'VAR1'
```

Drop the unneeded 'VAR1' name and you're back where you began.

Access: LS-ENTER

APPLY

Creates functions. Flags -2, -3 should be clear (symbolic mode).

```
L2: {symb1 ... symbn} L1: 'name'
```

```
{ vars } 'A' → 'A(vars)'
{'X' 'Y'} 'F' → 'F(X,Y)'
```

For another way of creating functions, see [DEF/DEFINE](#).

Access: 93 MENU (old SYMB)

APPROX. mode

Approximate mode can be toggled with MODE-CAS or RS+ENTER. The header shows a ~ symbol (Exact mode shows a = symbol). In Approx mode, operations return decimal numbers (Exact returns rational fractions in symbolic mode). While Exact mode is best for the CAS, Approx is preferred for complex inputs. See also [REAL Mode](#).

In Approx mode, entered numbers appear on the stack as reals (a decimal point is added).

5 → 5.

Approximate mode uses numerical results in calculations. Exact mode and CAS produce a symbolic result in algebraic calculations.

If you're solving a symbolic expression that requires the approximate mode, the calc will usually switch automatically - it will ask first, and saying no will cancel the operation. Mode switching can be disabled by setting flag -123.

Access: MODE-CAS-Approx,
RS+ENTER

ARC

Draws arc (or circle) CCW in [PICT](#). Set radian or degree mode as required.

```
L4: (x,y) (center)
L3: real (radius)
L2: start, deg/rad from horiz
L1: end, deg/rad from horiz
```

You can also use pixels:

```
L4: { #m #n } (center)
L3: #r (radius in user binary)
L2: start, deg/rad from horiz
L1: end, deg/rad from horiz
```

Example: to draw a circle 1.5 in radius with center at (1.,1.) in radian mode:

(1., 1.) 1.5 0 6.28 →

Press the left cursor to see it. Optional: [ERASE](#) before running ARC. Stack graphics do not affect [EQ](#).

The same example in pixels:

```
{ # 75d # 30d } # 15d 0 6.28
```

Note that 1.5 units (tick marks) is 15 pixels. See also [C→PX](#).

Access: LS-PRG-NXT-PICT

ARCHIVE

Creates backup object of Home dir and its subdirs. It does not archive libraries or anything else stored in port 0 (which is where the Home dir is). It does not affect ports 1, 2 or 3. See [MEMORY basics](#). See also [RESTORE](#).

Note that flag settings are not archived and the flaglist from [RCLF](#) (see also [Mode-Flag menu](#) entry) should be stored in a variable. Key assignments and alarms are archived in the hidden dir.

```
:port#:name → backup stored in port
```

eg: :2:BAK49 →

Port 2 (flash memory) is best for storing backups because it isn't affected by power loss. Note that quotes are not required around the name, and that the lefthand colon does not show on the stack.

If the name already exists, it will say "Object in use". You can delete the original or rename the new object.

To create and send the backup object via transfer (ARCHIVE uses binary Kermit regardless of settings):

```
:IO:name →
```

This sends the backup object via the selected IO method (USB, IR, etc). See [TRANSFERRING](#) for more.

Better IO methods: use the SD card and a reader to transfer the backup obj to a PC, or use the USB port and HP's [Conn4x](#) program. Conn4x has the advantage that it creates a var in the Home dir ('fLaG') that will restore your flags once you install the Conn4x backup (see [RESTORE](#)).

Access: LS-PRG-MEM-NXT

ARG

Returns the polar angle of a complex number (coordinate).

```
(x, y) → θ  
(x,  $\angle$  y) → y
```

The formula used is:

$\text{ATAN}(y/x)$ for $x \geq 0$
 $\text{ATAN}(y/x) + \pi$ for $x < 0$, RAD
 $\text{ATAN}(y/x) + 180$ for $x < 0$, DEG

A single argument x is taken as $(x,0)$. Since this is a constant, ARG always returns 0 for $x > 0$ and 180 (DEG) or π (RAD) for $x < 0$.

► Note that there is another ARG in LS+MODE-MISC - this is a system setting and not a command. See the entries for [Mode-Misc-ARG](#) and [LASTARG/LAST](#).

Access: RS-Z,
LS-MTH-NXT-CMPLX,
CMPLX

ARRAY-LIST

The HP48 and 49/50 can do arrays in the same way as BASIC, with no need for dimensioning. Place the data in a list:

```
{ A B C D E F }
```

and STOrE it with any name, such as 'LST'. To retrieve any data, use the format: 'name(element)'

eg, to retrieve the D:

```
'LST(4)' EVAL → 'D'
```

It's slower than [GET/GETI](#), but may be useful in certain applications.

To retrieve data one element at a time in sequence, the program below can be used:

```
« 1 6 FOR X 'LST(X)' EVAL NEXT »
```

To add to the list, use normal list processing; *eg*, to add a value to the end of list **lname**:

```
'lname' RCL value + 'lname' STO
```

You can also use STO+ (from PRG-MEM-Arith), but the value will be added to the beginning of the list:

```
value 'lname' STO+
```

Access: type in

Arrays/vectors

See also [RECT](#) and [CYLIN](#) for more information on rectangular and polar modes.

ARRAYS

Arrays use square brackets and can be used by themselves, as vectors, or as matrices.

[2 3 4]	numeric array/vector
['A' 'B']	symbolic array/vector
[10 \angle 45]	vector in polar mode
[(1.,2.) (3.,4.)]	complex array
[[1 2] [3 4 4]]	numeric matrix
[['A' 'B'] ['C' 'D']]	symbolic matrix

See also [Brackets](#).

Arrays can contain any number of elements. The default is horizontal - vertical arrays can be made with the Matrix Writer or by setting flag -98. (The MTRW uses the matrix format - flag -98 mainly affects the display.)

Arrays with the same number of elements can be added and subtracted - processing is vertical, element by element. Multiplication of arrays requires the [DOT](#) and [CROSS](#) product commands.

An array and a scalar quantity can use any arithmetic operator except subtraction. To subtract a scalar, put it in an array with the same number of elements:

$$\begin{array}{ccc} [2 & 3 & 4] & [1 & 1 & 1] & \rightarrow & [1 & 2 & 3] \\ [5 & 6 & 7] & [0 & 0 & 1] & \rightarrow & [5 & 6 & 6] \end{array}$$

VECTORS

See also [v→](#), [-v2](#), [-v3](#). Vector tools are in MTH-VECTR. The angle symbol is alpha-RS-6.

Vectors are usually 2D (two elements) or 3D (three elements), sometimes called phasors.

Type Difference

Arrays and vectors appear to be the same and even have the same type number (3). The main difference is that the vector changes when the angular mode changes:

	RECT	POLAR (CYLIN)
Array	[10. 20.]	[10. 20.]
Vector	[10. 20.]	[22.361 \angle 1.107]

Arrays can contain reals or integers (the type number is 28 for integers/symbolics). Vectors convert integers to reals.

3D vectors have another angle symbol (the Z-axis) in spherical mode. They're made with $\rightarrow\mathbf{V3}$:

```
3.16 2. 32. → [3.16 ∠2. ∠32.]
```

Note that vectors are always in rectangular mode internally - the polar mode is only on the display:

```
[5. ∠53.1301] OBJ→ → 3. 4. {2.}
```

However, polar coords can be kept by using $\mathbf{V}\rightarrow$ in polar mode:

```
[5. ∠53.1301] V→ → 5. 53.1301
```

Note that even in polar mode $\mathbf{OBJ}\rightarrow$ returns rectangular coords.

Making Vectors

Arrays can be entered by typing them in or using $\rightarrow\mathbf{ARRY}$, but vectors require one of several methods:

1. Be sure flag -19 is clear for vector output. Enter the elements and press MTH-VECTR $\rightarrow\mathbf{V2}$ (or $\rightarrow\mathbf{V3}$ in spherical mode for 3D):

```
3 5 →V2 → [3. 5.] (Rect)
3 5 →V2 → [3. ∠-1.2831] (Polar)
```

The above is in **radian** mode.

2. In polar mode, start square brackets, type the first element, the angle symbol (alpha-RS-6 or CHARS symbol 128) and then the angle:

```
[ ] 3 ∠ 25 ENTER → [3. ∠25.]
```

The above is in **degree** mode.

If the angle is greater than 180 degrees or π radians, -360 or -2π radians is added:

```
Deg: [3 ∠250] → [3. ∠-110]
Rad: [3 ∠4] → [3. ∠-2.2831]
```

Vector Algebra

The resultant of two vectors can be found by addition or subtraction:

```
[ 3. ∠25. ] [ 3. ∠7. ] + → 5.9261 ∠16. ]
```

A vector and a scalar quantity can use division or multiplication. For addition or subtraction, put the scalar in a vector with the same number of elements (see Arrays, above).

The **magnitude** of the vector in programming can be found with **ABS** (which is the root of the sum of the squares).

The **unit vector** is the vector divided by its absolute value:

```
[x ∟ y] DUP ABS / → [1 ∟ y]
```

The **angle** of the vector in programming can be found with

```
[x y] V→ SWAP / ATAN
```

Rotating a Vector

A vector (x,y) can be rotated to (x1,y1) through the angle θ degrees by:

```
x1 = x*COS(θ) - y*SIN(θ)
y1 = x*SIN(θ) + y*COS(θ)
```

Rotation is CCW - for CW, make θ negative.

The program below automates this:

```
(x, y) ±θ → (x1, y1)
(1., 2.) 30 → (-.134, 2.232)
```

```
%%HP: T(3)A(D)F(.);
```

```
\<<
```

```
IFERR PUSH DEG RECT SWAP V\-> \-> \Gh x y
\<< 'x*COS(\Gh)-y*SIN(\Gh)' EVAL 'x*SIN(\Gh)+y*COS(\Gh)' EVAL R\->C POP
\>>
THEN POP "[x y] or (x,y),\177\Gh"
END
```

```
\>>
```

(The Greek θ appears as \Gh and \pm becomes \177.)

Vector Plots

Plotting vectors is simplified if you use the complex form as explained below. The angle symbol is alpha-RS-6.

If flag -19 is set for complex output from **→V2**, the resulting ordered pairs behave just like the array-type vectors for simple operations. A drawback is that they won't work with **DOT** and **CROSS** commands. However, they're ideal for plotting, since no conversion to coordinates is needed.

```
5 45 →V2 → (5., ∟45.)
```

```
(5., ∟45.) (0, ∟0) LINE DRAX →
```

The above is in polar and degree mode. To split a complex vector and retain its polar values, use **V→**, not **OBJ→**.

Vector Plot program

The program below will plot 2D vectors. Input can be polar, rectangular, or a complex-type vector.

```

[4 ∠30] →
[3.4641 2.] →
(4., ∠30.) →

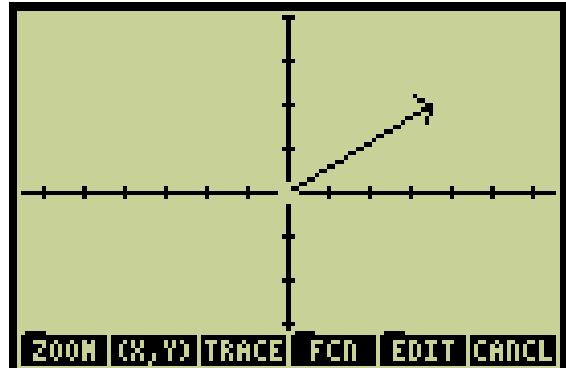
```

For best results, the calculator should be in Degree mode and Polar mode. For radians, adjust the plot parameters to suit. The above values are for the default plot settings (ZOOM-ZOOMDFLT).

```

%%HP: T(3)A(D)F(.);
\<<
IFERR PUSH DEG -19. SF CYLIN
DRAX
  IF DUP TYPE 3. ==
  THEN RECT V\-> R\->C CYLIN
  END \-> vec
  \<< vec (0., \<)0.) LINE vec V\->
  5. + SWAP .95 * SWAP \->V2 vec
  LINE vec V\-> 5. - SWAP .95 *
  SWAP \->V2 vec LINE POP PICTURE
  \>>
  THEN POP "Array,cmplx"
END
\>>

```



Be sure the plot range suits the size of your vector - use LS+WIN to set H-View and V-View if necessary.

The arrowheads will change their size and angles slightly depending on the vector angle.

ARRY→/→ARRY

For vectors, see **V→,→V2,→V3**, or **Arrays/vectors**.

ARRY→

Splits arrays and matrices into elements and list of size(s). Same as **OBJ→**.

```

[x y z] → x y z {3.}
[[1 2] [3 4]] → 1 2 3 4 {2. 2.}

```

Access: CAT

→ARRY

Creates array or matrix from elements on stack. Elements can be reals, integers, or symbolics. Arrays can contain any number of elements. The default is horizontal - vertical arrays can be made with the **Matrix Writer** or by setting flag -98. (The MTRW uses the matrix format - flag -98 mainly affects the display.)

Array:

L2-up: elements

L1: number of elements

```
'A' 'B' 'C' 'D' 4 → [ A B C D ]
```

Matrix:

L2-up: elements

L1: {m n} where m=rows, n=cols

```
'A' 'B' 'C' 'D' {2 2} → [ [A B] [C D] ]
```

Complex array:

L2-up: complex numbers

L1: number of elements

```
(1.,2.) (3.,4.) 2 → [ (1.,2.) (3.,4.) ]
```

Complex matrix:

As above, but Lev1 has a list: {rows cols}.

Access: PRG-TYPE

ASM

The 49/50 series provide a built-in compiler for SysRPL and ML source code, ASM (also called MASD).

SysRPL basics: Be sure flag -92 is set for the SysRPL mode. The source code should be a string on the stack, and must have this format:

```
"::
pgm code (comments)
;
@"
```

Note that the @ symbol must be on a line by itself and nothing must follow it other than the string's double quote.

Comments are removed during compiling. You can also add comments by putting an asterisk (times key) in the left margin.

Run ASM and the source code should compile to a program if there are no program errors. The stack will show one or more External labels unless you set flag -85 for SysRPL stack display.

There is also ASM2 from 257 MENU. It has the advantage of better error trapping and display. See also [256 MENU](#).

The above assumes a good knowledge of programming. The Extable library should be installed to allow naming of the commands. The Emacs programming editor is highly recommended, as is the SDIAG stack diagram library and the Nosy decompiler (all from hpcalc.org).

Access: 256 MENU, 5 NXTs

ASN

Assigns user functions to keys.

```
obj keynumber →
```

eg, « PURGE » 45.1 →

This assigns the **PURGE** command to the Backspace key. In **USER** mode, Backspace becomes PURGE:

```
LS-alpha-Bksp → same as PURGE
```

After one use, the assigned key returns to normal (unless you lock USER on - flag -61).

See also the entries for [STOKEYS](#), [RCLKEYS](#), [USER](#), [Key Numbers](#).

You can obtain the keynumber with 0 WAIT, which returns the keynumber of the next keypress.

► You can assign a quoted name, a program calling another program, or a full program. Examples:

```
'pgmname '  
« pgmname »  
« program »
```

Note that you can't quote an HP command; *eg*, 'COS' will cause an error. You also can't quote a library name. Use the program mode for this: « COS » or « HLP49 ».

For removing assignments, see [DELKEYS](#). A new assignment will overwrite any existing one without the need for DELKEYS.

► Note that key assignments do not work in the editor or other programs unless the assignment has the SysRpl command TakeOver. An easy way to do this is the TO? function from the Keyman library (www.hpcalc.org). See the entry for [TakeOver](#) for more detail.

```
« pgmname » →TO? keynum ASN
```

Key number is row-col.plane, where plane is the shift number. There is no apparent change to the program after →TO? (set flag -85 if you want to see it).

If .01 is added to a shifted key number's plane, the shift key must be held down to obtain the user mode function.

Example: If the Purge command is assigned to the rightshift P key (44.3), the keynum becomes 44.31 and RS must be held down at the same time as P.

Keyshift Nos:

- .1 norm**
- .2 LS**
- .3 RS**
- .4 alpha**
- .5 alphaLS**
- .6 alphaRS**

Access: LS+MODE-KEYS,
LS-PRG-NXT-MODES-KEYS

ASR

Shift bit right except for MSB.

1001b → # 1000b

Access: RS-BASE-NXT-BIT

Assign (→)

► Assign is not a command, but HLP49's name to assist index searches (the calculator's index doesn't respond to the right arrow).

For key assigning, see [ASN](#).

See also [Programming basics](#).

Assigns stack objects to local vars in program. Must be followed by program delimiters or algebraic expression. Vars are cleared on exit from program delimiters or evaluation of algebraic. See also [LOCAL](#) for permanent versus local vars.

Example (see [FOR-NEXT](#) for another):

Stack:

Lev 3: 33

Lev 2: 22

Lev 1: 11

Program:

```
<< → a b c << program >> >>
```

OR

```
<< → a b c 'expression in a,b,c' >>
```

Result:

a=33 b=22 c=11 (*ie*, Lev 1 is the rightmost var in the assignment sequence)

► If the assigning is not followed by an expression or a program, a syntax error results.

Access: RS-zero key,
CHR 141

ATICK

Sets axes tickmark spacing in **PPAR**, which it creates if needed. Default is 1 unit (10 pixels). Follow with a redraw or **DRAX**. To reset the defaults, delete PPAR.

- A single real or integer sets both axes to the same value in units.

```
n →
```

- A list with two reals (not integers) allows different X and Y unit values.

```
{ X. Y. } →
```

- A user binary digit sets X and Y ticks at a spacing of n pixels.

```
#n →
```

- A list with two binary digits sets X and Y ticks to different pixel spacing.

```
{ #X #Y } →
```

User binary digits are easier to use if the base is set to decimal (see **BASE**).

Access: 81 MENU-PPAR

ATTACH

Attaches library to current directory. External libraries are usually auto-attaching, but some built-in libraries require this command.

```
libnum →
```

Example: 256 →

The above attaches menu 256 (the "hacker's toolbox"). It appears in APPS as Development lib. For more detail, see **LIBRARY basics**.

Some programs (such as OT49) automatically attach lib 256.

The Home dir can have any number of libraries attached, but subdirs can have only one at a time (each ATTACH overwrites the last).

To make a library's commands available only from one dir, go to the Home dir and run **DETACH**, then go to the desired dir and run ATTACH.

This change will be undone by the next reboot unless you put it in **STARTUP**. See also **PATH**.

Access: 110 MENU

AUTO

Sets autoscaling for plots - adjusts the axes to display the largest possible plot. Functions also create a variable in the current dir (default name 'X' - you can safely purge this). No input is required.

If the results aren't suitable, you can either adjust manually with the Zoom features in the graphics editor, or you can delete **PPAR** and **ZPAR** - they'll be regenerated with default values on the next plot and zoom.

Access: LS+WIN,
81 MENU-NXT,
CAT

AXES

1. Sets coordinates of plot axis intersection - can be used to change the origin. To change only the plot center, see **CENTR**. See also **PPAR**. The change takes place with the next plot.

$(x, y) \rightarrow$

2. Writes axis labels into PPAR (follow with **LABEL**).

{ "hlabel" "vlabel" } \rightarrow

The above plus the argument for ATICK (optional) can be combined:

{ (x,y) ATICK "hlabel" "vlabel" } \rightarrow

Follow with **LABEL**. From the graphics editor, it's EDIT-NXT-LABEL.

3. In the LS+2D/3D window, adds axes to the next plot if a white square is showing in the softkey box.

Access: 81 MENU-PPAR

Backquotes (left quotes)

The backquote pair (``) is from RS+' and allows entry and evaluation of algebraic format expressions in RPN that would otherwise cause a syntax error.

$$\text{`CURL}([\text{SIN}(Z), X+Y, Z^2], [X, Y, Z]) \text{`} \rightarrow [0 \text{ 'COS}(Z) \text{ ' } 1]$$

Since the single quotes (') will also accept most algebraic formats, this isn't something you'll use a lot, but it may be useful under certain conditions. Single backquotes are also available from the [CHARS](#) map, character 96.

Access: RS+'

BAR

Sets the plot type to Bar and if necessary, creates [ΣPAR](#), [PPAR](#). [ΣDAT](#) must contain a vertical, single-column matrix.

See also [Plots](#) and [PLOT command](#) for more on general plotting.

A sample barplot: store the following matrix as ΣDAT (you can use STOΣ from CAT):

```
[ [ 2.5 ]
  [ 2.75 ]
  [ 3.9 ]
  [ 4.1 ]
  [ 5.5 ]
  [ 4.23 ]
  [ 3.17 ]
  [ 2.95 ] ]
```

Press LS+2D/3D and select **Bar** as the Type. Turn off **Axes**.

Press LS+WIN and press Auto, Erase, and Draw.

Access: LS+2D/3D-Type,
96 MENU-PLOT,
CAT

BARPLOT

A CAT command that executes a Bar plot (see **BAR** above). ΣDAT must contain a vertical, single-var matrix - the BAR entry provides an example. The plot does not remain on the screen - press [PICTURE](#) (left cursor) to return to it. See the entry for [ΣDAT](#) for more on making ΣDAT.

BASE

A menu that controls the current number base; the default is HEX (hexadecimal). The default base is selected from HEX, DEC, OCT, or BIN. For changing the base of a number, numbers are entered as binary integers, with the form

XB

where X is one or more integers (and/or characters A-F) and B is the desired base: h=HEX, d=DEC, o=OCT, and b=BIN (it's case-sensitive). The space after the number sign is optional.

If B is omitted, the system uses the current base, which shows in the header. Leading zeros are not displayed on the stack.

See also **AND**, **OR**, **XOR**.

Example: convert FFF hex to decimal, assuming that the current base is hex.

Enter:

#FFF → # FFFh

Press:

DEC: → # 4095d

The decimal binary integer can be converted to a real with B→R (in BASE menu):

4095d B→R → 4095.

Reals or integers are converted to binaries with R→B:

8192 → # Nn

where N is numbers (and/or A-F), and n is h, d, o, or b, according to the current base.

In programs, the base can be selected by any one of HEX, DEC, OCT, or BIN. It can also be selected with flags -11 and -12 (see **FLAGS - list of flags**).

The default size for binaries is 64 bits (see **STWS**). If you use binaries a lot, see **Custom Enter** for a way to automatically create them.

BASE also includes binary logic operations using bits or bytes - these commands have their own entries.

Access: RS-3

Batteries

Needless to say, the calculator must be off when changing batteries. Removing a battery when it's on will cause a system reset and/or data loss.

Batteries should be AAA alkaline. The 49G and 49G+ use three cells; the 50G uses four. You can use the low-cost carbon types, but they won't last very long. You can also use rechargeables (see below).

A capacitor across the batteries protects the memory for several minutes while you change the cells (one at a time is best - the 3V cell will not protect the time/date).

HP49G+/50G Only

The Port 0 and 1 memory backup battery is a 2032 3V cell. It will not run the calculator. If you remove it with the main batteries out, ports 0 (the Home dir) and 1 will be cleared. Note that while the 2032 will protect the memory, the realtime clock will stop and buffers will be cleared if the main cells are out for more than a few minutes. (For more on ports, see [MEMORY basics](#)).

Contrary to what HP may say, you can use NiMH rechargeables. In fact, they're highly recommended, even if they need charging every week or two. And contrary to what you may have heard, there is no problem with the low-batt indicator. (See hpcalc.org for various battery-monitor programs that show the remaining charge.) You should have a second set of batteries of any type to protect the memory while the NiMH set is charging.

You can also use nickel-cadmiums if you have any, but their capacity is much lower than NiMH.

BATTERY LIFE

This is very difficult to determine because the current drain depends on what the calculator is doing. Intensive all-day use might require charging every two days, while casual use might extend battery life to weeks or months.

Note: writing to flash memory (port 2) or the SD card (port 3) requires high-current pulses. If the low-batt indicator is on or about to come on, you will probably be limited to saving to the Home dir (port 0) or SRAM (port 1). If you can't save at all, power down and change the batteries.

BAUD

On the HP49G, sets the bit-transfer rate, which is stored in IOPAR in the Home dir. The HP49G+/50G generally does not require this.

Choices on the HP49G are 1200, 2400, 4800, 9600, and 15360 bits/sec. The 49G+/50 in USB mode uses 115200 and in IR mode, the rate is negotiated with the other device.

n →

Access: APPS-I/O-Transfer,
 104 MENU-IOPAR

BEEP

Sounds a tone. Flag -56 must be clear.

freq,Hz duration,seconds →

Frequency range is 1-15000Hz. Any duration greater than 1200 sec is changed to 1200.

The beeper can be turned on and off with flag -56 or with MODE Beep. The alarm beep is turned on by clearing flag -57.

The beep is nearly inaudible. You can make it more noticeable by using this program instead of BEEP:

```

%%HP: T(3)A(D)F(.);
\<<
  IFERR PUSH -56 CF
    DO 500 .5 BEEP 350 .5 BEEP
    UNTIL KEY
  END DROP POP
  THEN POP
  END
\>>

```

Store it with your choice of name. It produces a sound somewhat like a klaxon siren until a key is pressed. To make it sound for 2 seconds and then stop, remove the commands DO UNTIL KEY END and change each .5 to a 1 (or other suitable number to adjust the on-time).

Access: PRG-NXT-OUT-NXT

BESTFIT

Lets the linear regression utility choose the best data fit from linear, exponential, log, or power. Requires the data variable **[ΣDAT](#)** (see also **[ΣPAR](#)**).

Access: RS-5-Fit Data-Model

Binary Integers (user binaries)

In addition to reals and integers, the HP49/50 has user binaries in one of four bases.

They have the form # XB, where X is any number and B is one of four bases, such as # 32AFh. They're used for conversions from one base to another, and some commands require them as input.

See **[BASE](#)** for more information on creating and using binaries.

See also **[AND](#)**, **[OR](#)**, **[XOR](#)**.

If you use binaries a lot, see [Custom Enter](#) for a way to automatically create them.

Access: type in,
 BASE-R→B

BINS

Sorts independent column of current statistics variable [ΣDAT](#) into (nbins+2) bins.

```
Xmin Xwidth nbins → [[nbinl ... nbinr]] [nbinl nbinr]
```

The array on Lev1 shows data points that fall outside the specified range.

Access: STAT-Frequencies,
 96 MENU-1VAR

BLANK

Creates blank grob on Lev 1 in pixels.

```
#nwidth #nheight → grob
```

With older ROMs such as v1.23, the maximum display is #131d by #64d. With newer ROMs such as v2.09, the maximum is #131d by #80d. Otherwise the blank can be any size.

When using binary pixels, it's more convenient to set the [BASE](#) to decimal with RS-3.

Access: PRG-NXT-GROB

BOX

1. Draws a box in the graphics editor with the mark (times key) and cursor as opposite corners.

Access: EDIT-BOX (in graphics editor)

2. Draws a box in [PICT](#) using opposite corners in either pixels or coordinates. Pixel numbering on the display starts at {#0 #0} for the upper left. For coords this will be (-6.5, 4.) .

```
{ #m1 # n1 } { #m2 # n2 } →
```

OR

```
(x1, y1) (x2, y2) →
```

The example draws a border around the screen:

(Older ROM such as v1.23)

```
{ #0d #0d } { #130d #63d } →
```

(Newer ROM such as v2.09)

```
{ #0d #0d } { #130d #79d } →
```

When using binary pixels, it's more convenient to set the **BASE** to decimal with RS-3.

Optional: **ERASE** first. Stack graphics do not affect **EQ**.

Access: PRG-NXT-PICT

B→R

Converts binary numbers to reals.

```
# FFh → 255.
```

See also **Binary Integers (user binaries)**, **R→B**.

Access: BASE

Brackets [] () {} <<>>

Also called parentheses or delimiters.

Note that brackets are available only in pairs; *eg*, putting a bracket around an existing number requires deleting the halves you don't want: `()XY()` → `(55)`. Separate brackets are available from **CHARS** (*eg*, characters 40 and 41) if you don't mind the keystrokes.

[] LS-times

(Square brackets) are for arrays, vectors (see **Arrays/vectors**), and matrices.

Examples:

<code>[2 3 4]</code>	numeric array/vector
<code>'A' 'B']</code>	symbolic array/vector
<code>[10 ↻45]</code>	vector in polar mode
<code>[(1.,2.) (3.,4.)]</code>	complex array
<code>[[1 2]</code>	
<code>[3 4]]</code>	numeric matrix
<code>[['A' 'B']</code>	
<code>['C' 'D']]</code>	symbolic matrix

Timesavers: The Matrix Writer adds single quotes to symbols. The command line adds square brackets to a matrix if you enter the first two pairs; *ie*:

```
[[1 2] 3 4 5 6] ENTER → [[ 1 2 ]
                           [ 3 4 ]
                           [ 5 6 ]]
```

() LS-minus

(Round brackets) are for complex numbers, grouping operations, or plot coordinates. The latter are the same as a complex number.

(1 . , 2 .) complex number
 ' (A , B) ' symbolic complex
 '1+2*(3/4)' = 2.25
 '1+2*3/4' = 2.5

Timesaver: the command line adds the comma to numeric complex numbers (but not symbolics), and integers are converted to reals.

(1 2) ENTER → (1 . , 2 .)

Symbolic complex numbers must have the comma typed in and must be enclosed in single quotes, as in '(X,Y)'.

{ } LS-plus

(Braces or curly brackets) are for lists. Any type of objects can go in a list.

{ 1 2 A B } simple list
 { {1 2} {A B} } list of lists
 { 1 { 2 { 3 } } } nested lists

Timesaver: the Matrix Writer can do a list-of-lists if flag -91 is set. Note: the calculator will add missing right curly brackets, and not always in the desired place.

For more on lists, see the entry for [LIST series](#).

«» RS-plus

(Program brackets) Also known as guillemots, these delimit UsrcPL programs. They can be nested as required. In alpha mode they're entered on one line («») and in non-alpha they have a newline between them. See also [Programming basics](#).

BUFLEN

Returns number of chars in the serial input buffer (255 bytes), plus 0 or 1 to show if an error occurred or not.

→ n 0/1

0 - overflow, frame error
 1 - no error

Press PRG-NXT-NXT-ERROR-ERRM to see the error message, if any.

Access: 104 MENU-SERIAL

BYTES

For any object, returns checksum and size in bytes.

`obj → #checksum bytes`

For the number of elements in an object, see [SIZE](#).

Access: PRG-MEM

CASCFG

A CAS command that is not explained in the manuals, but sets conditions (flags, etc) for the best usage of the CAS. No input is required.

► Unexpected results from CAS commands can often be prevented by running CASCFG first.

Some of the settings it makes: Real mode is set, flags -2,-3 are cleared for symbolic mode, and Modulo (in [CASDIR](#)) is reset to the default value of 13.

It restores VX in the Home dir to the default var 'X' - see [STOVX](#). It also clears flag -124 (CAS-COMPEVAL). The latter is not well explained, but a wrong setting results in "Non-algebraic" errors.

Access: CAT

CASDIR

A directory in the Home dir, created by the CAS for temporary storage of settings and vars. You rarely have to access this unless you're manually adding vars to REALASSUME or checking the value in MODULO.

The contents are:

SYSTEM:

Temp storage of terms used by the CAS. May not be present.

REALASSUME:

Variables in a list that the CAS will take as real numbers in Complex mode. Any can be added. Vars can be removed manually or by putting them in a list and running UNASSUME (CAT).

Default vars are {X Y t S1 S2}. See also **ASSUME** in CAT for adding more.

PERIOD:

The value for periodic functions, default 2π .

ENVSTACK:

Temporary storage of flag and path settings. This is where PUSH stores the flags and paths, which are restored by POP.

PRIMIT:

The last expression for which an anti-derivative was found.

CASINFO:

Temporary storage of grobs.

MODULO:

The current modulus, default 13. CASCFG will restore any changes to 13, as will MODE-CAS-Modulo and then NXT-RESET.

EPS:

Polynomial coefficients smaller than this will be taken as zero by the EPSX0 command in CAT. Default is 1E-10.

MATRIX:

Temporary storage of a matrix used by the CAS. May not be present.

VX:

Contrary to some of the manuals, the var VX is in the Home dir, not CASDIR. See [STOVX](#).

CASE

Conditional program structure. Since CASE terminates on the first true (non-zero) condition, it's more concise than a series of [IF-THEN](#) tests.

CASE

DUP <test1> THEN <do if ≠ 0> END

-

-

DUP <testn> THEN <do if ≠ 0> END

<optional - do if no true cond. - similar to ELSE.>

END

Typing aids: when the CASE dir is displayed (LS-PRG-BRCH), these are available:

LS-CASE →	CASE
	THEN
	END
	END
RS-CASE →	THEN
	END

The following example tests for an algebraic, a list, or a real number on Lev1; if other, it puts an error string on Lev1. The IF conditional ends the program if there's nothing on the stack.

```

obj    →    "type, typenumber"    or    "errorstring"

%%HP: T(3)A(D)F(.);
\<<
IF DEPTH THEN
CASE EVAL DUP TYPE 9. ==
  THEN "Algebraic,9"
  END DUP TYPE 5. ==
  THEN "List,5"
  END DUP TYPE 0. ==
  THEN "Real,0"
  END "Not alg,lst,real"
END
END
\>>

```

See also **AND** and **OR** operators.

Access: PRG-BRCH

CEIL

Nearest whole number greater than or equal to the argument. See also **FLOOR**. Flags -2 and -3 should be clear for symbolic use.

```

5.1    → 6.
'A'    → CEIL(A)

```

Access: MTH-REAL-NXT-NXT

CENTR

Adjusts first two **PPAR** params to make (x,y) on Lev 1 the new plot center. Origin remains (0,0) (use **AXES** to change this). Follow with a redraw.

```
(x, y) →
```

You can also use a real or integer to move the center along the X-axis; *eg*, 3 will center the graph at (3,0).

See also CENTR under ZOOM in the entry for **Graphics menu**.

Access: 81 MENU-PPAR

%CH,%

Percentage change from y to x :

$$y \ x \rightarrow 100 (x-y) / y$$

$$30 \ 15 \rightarrow -50.$$

Percent:

x percent of y :

$$y \ x \rightarrow y (x/100)$$

$$50 \ 10 \rightarrow 5$$

Access: MTH-REAL

CHARS

BASICS

The CHARS menu displays all characters in the current font from 0 to 255. Each character can be inserted at the cursor with ECHO1 - for many chars, use ECHO. CHARS also works in the text editor and other menus. It will not display the minifont. There are minifont editors available from www.hpcalc.org, but none of them work well.

KEYSTROKES

In some keystroke listings, CHARS uses the ampersand (&) to indicate that the shift key should be held down while pressing the character key.

NEWLINE CHR

The top two rows (0 to 31) are mostly blank. However, CHR 10 is the Newline (linefeed) char. CHR 13 can be used with Find/Replace to remove Windows/DOS carriage returns from imported text (CRs show as black squares in the text). Leave the Replace field blank. See the entries for [NEWLINE](#) or [TRANSFERRING](#) for more detail.

MODIF

Any character can be changed with MODIF. Cursor to any pixel - the **period key** toggles it on and off. SCAN returns to the char map.

Note that both ON and ENTER will exit with your changes intact - there is no abandon key. See next paragraph.

The modified font will last only until the next warmstart replaces it with the default. To make it permanent, put the font on the stack with [FONT](#)→ and store it in the Home dir with a unique name. Add to the [STARTUP](#) var (create it if you don't have one):

```
'fontname' →FONT
```

The modified font will be reinstalled at each warmstart. To return to the original font, delete the above from STARTUP and do an ON+C warmstart (see [Reset](#) for more on warmstarts).

SOFTKEYS

If you press RS+CHARS you'll see the softkey menu for character handling (same as PRG-NXT-CHARS).

SYMBOLS LIST

For a list of symbols and their ASCII equivalents, see the entry for [Translations](#).

Access: RS-CHARS

CHOOSE

The CHOOSE window allows choosing one item from a list of any size. It can do incremental searches for alphabetic characters or numbers, the same as CAT (and the HLP49 program).

RS up and down cursors jump to the top and bottom of the list. LS up and down keys scroll one screen at a time.

Note that scrolling a large CHOOSE list (like this program) can make the calculator pause for a memory cleanup. The pause time seems to vary with the OS version and has never been completely eliminated. If it's a problem in a program, it can be reduced by forcing a [Garbage Collection](#) before running CHOOSE with # 5F42h [SYSEVAL](#).

Minimum syntax:

```
"title" (or "" for none)
{ a b etc }
ndefault → chosen_item 1. (or just 0. if cancelled)
```

ndefault (a real, usually 1.) is the list element that will be highlighted when it starts.

The following example shows the most basic operation:

```
<< "Choose" { A B C } 1. CHOOSE >>
```

Output will be 'A', 'B', or 'C' and 1., or a 0. if cancelled.

For more objects, each entry should be of the form below

```
{ a { x y z } }
```

In this format, the name of the choice ("a") is dropped from the output. If the name is a phrase, use double quotes ("ab cd"). Example:

```
<< "Choose" { { a { x y z } }
{ b { x1 y1 z1 } } } 1 CHOOSE >>
```

The output for "a", for example, will be { x y z } and 1.

Access: PRG-NXT-IN

CHR

Returns a character from the character map for a real or integer on Lev 1. Its inverse is NUM.

65 → "A"

Chrs 32-127 correspond to the usual ASCII map. Chrs 128-255 are special HP symbols - see the **CHARS** map . See also **Translations**, **TRANSFERRING**.

A linefeed (newline) is chr 10. A carriage return is chr 13. Returns will show in text as a black square. You can remove these from imported textfiles with Search-Replace by cursoring to chr 13 in the CHARS map and echoing it to the Find field. Leave Replace blank.

Access: PRG-NXT-CHARS,
RS+CHARS

CKSM

Sets Kermit I/O checksum, default 3. (For object checksums, see **BYTES**.) Also called **Chk** in the Transfer screen. (Xmodem uses checksum rather than CRC - the remote Xmodem will revert to this after a pause during handshaking.)

n →

1 = 1-digit arith

2 = 2-digit arith

3 = 3-digit CRC

Access: APPS-I/O Transfer screen,
104 MENU-IOPAR

CLKADJ

Adds to or subtracts from the realtime clock, where +/-8192 is one second (see also **TICKS**).

+/-n →

For automatic adjustment, see **Alarms**.

The simple program below sets the clock ahead or behind by the number of seconds (positive or negative) on Lev1.

« 8192 * CLKADJ »

+/-n →

For minutes, insert 60 * after the times symbol. For hours, insert 3600 * after the times symbol.

Access: RS-TIME-TOOLS,
RS+TIME-NXT-NXT,
PRG-NXT-NXT-TIME-NXT-NXT

CLLCD

Clears screen (except for softkeys, and on older ROMs of the 49G+, the header) until a keypress. If used in a program, it can be followed by a command such as 0 **FREEZE** or *n* **WAIT**, where *n* is the number of seconds. See also **DISP**.

Access: PRG-NXT-OUT

CLOSEIO

Closes 49G serial port and clears buffer (so does off-on cycle).

Does not apply to the 49G+, and there is little information on the 50G serial port.

Access: 104 MENU

CLVAR

Clears vars and empty subdirs from current dir. There is no confirm and no undo.

If there's a non-empty subdir, vars are cleared only up to that point.

Access: CAT

CLΣ

Deletes statistics variable **ΣDAT** if it exists in the current dir. There is no confirm and no undo.

Access: 96 MENU-DATA

CMD

Displays the last four command line entries. Pressing OK or ENTER sends the selected object to the command line, and a second ENTER sends it to the stack.

Key commands are not displayed; *ie*, if you press 2 SIN, only the 2 is kept. Commands that are typed in are kept.

Lists, programs, etc, that are placed on the stack are not kept unless they are edited; CMD will then contain the edited version.

CMD must be enabled with LS+MODE-MISC-CMD (the default is enabled). You can also use LS-PRG-NXT-MODES-MISC.

► CMD can use up a lot of memory if it contains large objects. To empty it, turn it off and then on again with LS+MODE-MISC-CMD or LS-PRG-NXT-MODES-MISC-CMD.

The CMD command is not programmable.

Access: LS-HIST

CNRM

Column norm (one-norm) of array.

$$[x \ y \ \dots] \rightarrow x+y+\dots$$

$$[(x, y) \ \dots] \rightarrow \sqrt{(x^2+y^2)}+\dots$$

Access: MATRICES-OPER

COL-

Deletes column n of a matrix, returns new matrix and deleted column.

$$[[\text{matrix}]] \ n \rightarrow [[\text{new}]] \ [\text{col}]$$

Access: MTH-MATRIX-COL

COL+

Inserts column into an array or matrix.

$$[\text{array}] \ \text{obj} \ \text{pos} \rightarrow$$

$$[1 \ 2 \ 3] \ 88 \ 3. \rightarrow [1 \ 2 \ 88 \ 3]$$

If obj is an integer, pos should be a real to prevent integer-to-real conversion.

$$[[\text{matrix}]] \ [\text{array}] \ \text{pos} \rightarrow$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \ [5 \ 6] \ 3 \rightarrow \begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{bmatrix}$$

Access: MTH-MATRIX-COL

→COL

Transforms matrix rows into separate column vectors, plus number of vectors.


```

[[ 11 12 13 ]
 [ 14 15 16 ]
 [ 17 18 19 ]] → [[ 11 14 17 ]
                   [ 12 15 18 ]
                   [ 13 16 19 ]
                   3.

```

Inverse of **COL**→ below.

Access: MTH-MATRIX-COL

COL→

Inverse of →**COL**. Converts columns of arrays and number of arrays to a matrix.

```

arrays  n  →  [[matrix]]

[ 11 14 17 ]
[ 12 15 18 ]
[ 13 16 19 ]
3. → [[ 11 12 13 ]
      [ 14 15 16 ]
      [ 17 18 19 ]]

```

Access: MTH-MATRIX-COL

COLCT

Factorizes polynomial/integer. Same as COLLECT.

```

'X^2+2*X+1' → '(X+1)^2'
72 → '2^3*3^2'
72. → error

```

Flags -2 and -3 should be clear.

Access: 93 MENU (old Symb)

COLLECT

Factorizes polynomial/integer. Same as COLCT. Flags -2 and -3 should be clear.

```

'X^2+2*X+1' → '(X+1)^2'
72 → '2^3*3^2'
72. → error

```

Access: RS-4 (ALG)

COLΣ

Specifies independent-var and dependent-var cols of statistics variable **DAT**. Values are used by the

plotter, which stores them in **Σ PAR** or Scatter and LR plots. See also XCOL, YCOL.

$$Xcol \ Ycol \rightarrow$$

The default is 1 for Xcol and 2 for Ycol.

The format of **Σ PAR** is: { indep depen Y-intercept slope }.

Access: CA

COMB

Combination of n items m at a time.

$$n \ m \rightarrow C_{n,m}$$

$$6. \ 3. \rightarrow 20.$$

Algebraic syntax in programs is 'COMB(n,m)'. In the **EQW**, enter n , press COMB, then enter m . Press ENTER to put this on the stack.

On the stack, convert this algebraic to a number with \rightarrow NUM or EVAL.

See also PERM.

Access: MTH-NXT-PROB

COMPLEX mode

The 49/50 has two basic modes for CAS calculations, Real and Complex. Real is the most-used mode (the annunciator in the header is |R). See also **Complex numbers** below.

Simple calculations with complex numbers can be done on the stack in Real mode, but complex calculations using CAS commands often require Complex mode.

The annunciator is **C**.

Modes can be toggled with LS+TOOL, or you can check/uncheck the Complex field in MODE-CAS. If you're solving a symbolic expression that requires the Complex mode, the calc will usually switch automatically - it will ask first, and saying no will cancel the operation. If flag -123 is set, auto mode switching is prevented.

Access: MODE-CAS-Complex,
LS+TOOL

Complex numbers

Numerical complex numbers are entered as an ordered pair with the format '(real,imaginary)'. The calculator will add the required comma and convert to reals:

$$(1\ 2) \rightarrow (1.,2.) \quad [\text{Single quotes don't appear on the stack with textbook display}]$$

Symbolic complex numbers must be fully entered as an algebraic expression:

$$'(X,Y)' \rightarrow (X,Y)$$

Note that the tic marks are not displayed with the stack in textbook mode. See [RE](#), [IM](#).

For changing symbolics into a complex number in programs, see the entry for [i](#).

► Algebraic complex numbers can take one of two forms on the stack, depending on flag -27:

Flag -27 clear:

$$'(X,Y)' \rightarrow (X,Y)$$

Flag -27 set:

$$'(X,Y)' \rightarrow X+Y*i$$

Numeric complex numbers are not affected by flag -27.

Most complex stack calculations are straightforward:

$$1.,2.) (2.,3.) * \rightarrow (-4.,7.)$$

The CAS with complex numbers may perform better in Complex mode. If the calc doesn't switch modes automatically, toggle this with LS+TOOL or MODE-CAS-Complex.

CON

Creates constant array or matrix with identical elements.

$$\{\text{cols}\} \ n \rightarrow [n \ n \ n \ \dots]$$

$$\{\text{rows cols}\} \ n \rightarrow [[\text{matrix of } n]]$$

Access: MTH-MATRIX-MAKE

COND

1-norm (col. norm) condition number of a square matrix.

$$[[\text{sq. matrix}]] \rightarrow \text{cond\#}$$

Access: MTH-MATRIX-NORM

CONIC

Sets the plot type to Conic and creates or modifies **PPAR**. **EQ** should contain an expression in two variables (default X and Y), although single-var will work if they're in the form $Y=f(X)$ (eg, to plot a straight line, store 'Y=A*X+B', where A and B are optional constants).

See also **Plots** for more on general plotting.

The following will plot an ellipse. Gaps in the plot can be reduced by increasing the resolution with LS-WIN-Step (see **RES** for values - try .05). If only half the plot appears, clear flag -1 to get General Solutions. If a previous PPAR affects the plot, run ZOOM-ZDFLT.

```
%%HP: T(3)A(D)F(.);
\<< '2*X^2+3*Y^2-12' STEQ CONIC ERASE DRAX DRAW PICTURE \>>
```

Access: 82 MENU

CONJ

Returns conjugate of complex number or array. See also **Complex numbers**.

Flag -3 should be clear for symbolic arguments.

```
(1.,2.) → (1.,-2.)
'(X,Y)' → 'X+Y*-i'
[ (1.,2.) (3.,4.) ] → [ (1.,-2.) (3.,-4.) ]
```

Access: CMPLX,
MTH-NXT-CMPLX-NXT

CONLIB

Displays library of constants. Menu allows selection of SI or English units, or no units. The Value key replaces the names with numerical values. The →STK key writes the selected constant to the stack with or without units. See the **CONST** entry for control of constants display of units with user flags 60 and 61.

Access: APPS-Constants lib,
115 MENU CONLIB

Conn4x basics

The Conn4x communication program for the PC is shipped with the calculator. The latest version is usually available from hpcalc.org or the HP website.

The following applies to Version 2.3 Build 2439. (Currently hpcalc.org lists this as v2.2, but the program says v2.3.)

See also [Reflashing ROM](#), [TRANSFERRING](#), [Translations](#).

Quick start

1. Plug the cable into the calc and computer.
2. Start the calc and press RS-right cursor for the Xmodem server.
3. Start Conn4x.
4. On Conn4x, click the Connect icon (2 red arrows) or File>Connect.

The contents of the Home dir should appear in Conn4x. For some reason, they're sorted alphabetically. If it errors, try unplugging the USB, plugging in again, and starting over. Be sure that you're not in the Kermit server, RS+right, (which says only "Server" onscreen rather than "Xmodem Server").

The left window is for changing dirs and the right displays the vars.

Objects can be moved or copied between windows or the PC using drag and drop or the Edit menu - see below.

Note that Conn4x currently cannot access port 1, the flash memory (port 2) or the SD card (port 3).

Quick transfer

Select the desired mode, either binary (icon shows ones and zeros) or text (icon shows A to D). Drag and drop in either direction.

Transferring

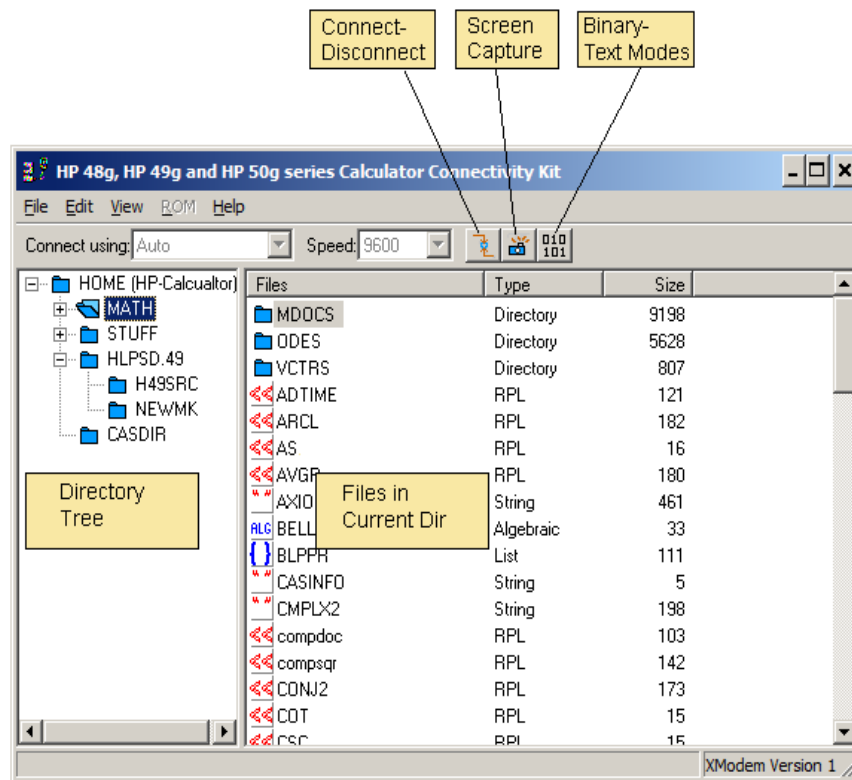
Text (or ASCII) is the best method for files meant to be edited on the PC. The HP symbols will be automatically converted to their ASCII equivalent and back again. Drag and drop in either direction.

Transfers in Text/ASCII will have a header added:

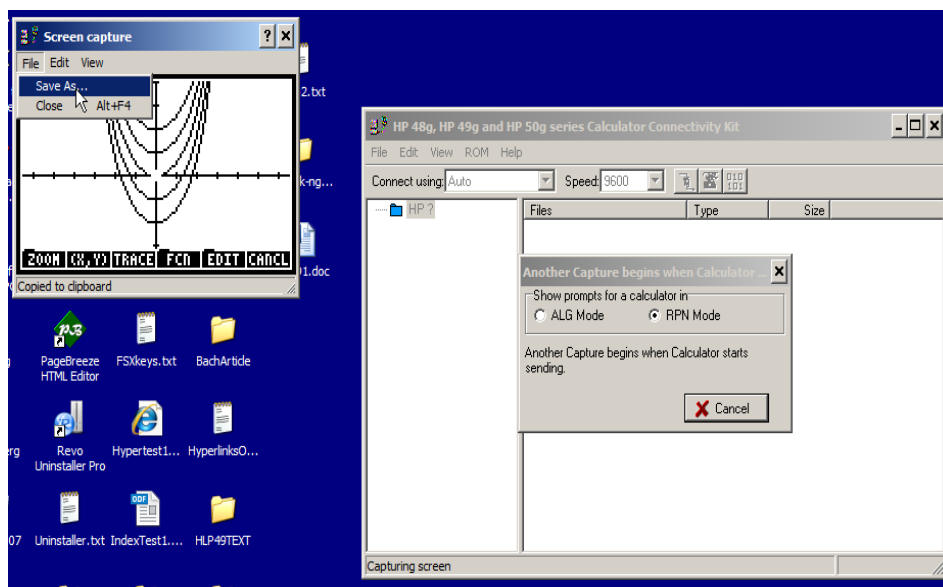
```
%%HP: T(3)A(R)F(.);
```

and the extension .txt on the computer. The .txt extension is removed when the item is transferred back to the calc.

The header on binaries is 13 chars, beginning HPHP49, and the extension is .hp. Note that Conn4x's version of Xmodem can do symbol→ASCII conversions.



Above, the Conn4x Main Screen. The **Auto** connect mode is suitable for all but reflashing the ROM, which requires the **USB Calculator** mode. The speed settings are usually automatic.



Screen Captures: set up the calculator screen and connect the cable. Select either the icon or Files> Capture Calculator Screen. On the calculator, hold ON and briefly press the Up Cursor. Save as BMP or JPG - BMP is larger, but often has better clarity for line art. Images are B&W, 137 by 86 pixels.

Conn4x menus

Connect using:

With the calc connected via USB, the choices are Auto, COM1, and USB Calculator. With the HP49G's serial connection, the choice is Auto or COM1.

Auto is adequate for most purposes (except reflashing the ROM - see below).

Speed

The HP49G uses 9600, the default for COM1. The USB will set itself to 115200.

Toolbar Icons

The leftmost (two red arrows) is Quick Connect and Disconnect. It assumes that you've started the Xmodem server on the calc (RS- right cursor).

The camera icon is for screen captures - see below.

The rightmost icon shows either ones and zeros (binary transfer, the default) or A to D (text or ASCII transfer).

File menu

New Folder

Creates a new dir in the current calc dir.

Backup

Creates a backup of the calc in your choice of computer folder. The default filename is 49_ followed by the date, with the extension "hpbackup". Only the Home dir is backed up; objects in ports are not saved.

Restore

Existing backups can be used to restore the calc to a previous state. Note that these backups are not compatible with the calc's **RESTORE** command.

Delete

Deletes the selected item. There is no confirm and no undo. You can also press the Delete key.

Rename

Allows renaming the selected item.

Properties

Mostly the item's byte size.

Connect...

Same as the Quick Connect icon except for a menu with details on starting the calc. Changes to Disconnect when the connection is made, same as the Disconnect icon.

Text/Binary

A duplicate of the toolbar text/binary icon.

Edit menu

Cut, Copy, etc

Note that Cut, Copy, Copy as Text and Paste work only within the calc's dirs. They do not use the computer's clipboard. Use drag and drop for transfers to the computer.

Edit as Text

Opens the selected item in Notepad. The computer's editor can be changed in View>Options.

View menu

The Toolbar, Status, and other options control the look of Conn4x on the screen. The Options menu controls the translation status - the defaults are ideal. It can also select the desired text editor.

Calculator Command

Opens a small window that can send commands and display the results. Note that it's case sensitive.

Screen captures

The icon showing a camera is for screen shots. Images are B&W, 137 by 86 pixels. See the text under the screen captures above.

Reflashing the OS (ROM)

Note that with the current version, the "ROM" menu is not active unless you start Conn4x with the "Connect Using USB Calculator" option.

The new ROM file with the extension .bin must be in the same folder as update.scp. Click on ROM>Download ROM. Go to and select the appropriate folder. Follow the onscreen instructions that appear.

Note that an OS update can be lengthy and uses high-current pulses to write to flash memory. Be sure the batteries are in good condition (the HP50G is powered from the USB cable).

CONST

Defines character(s) as constant or returns value.

Example:

```
CONST(g)  (accel. of gravity)
OR
'g' CONST → '9.80665_m/s^2'
```

The argument for CONST() must exist in the Constants library (**CONLIB** command, APPS-EqnLib-COLIB, or 115 MENU).

Units can be toggled between SI and English with the **CONLIB** menu.

User flags 60 and 61 also control unit display (user flags have positive flag numbers.)

60s: English units

60C: SI units

61s: no units

61C: units

Units can sometimes cause errors in user programs, especially in the Multiple Equation Solver. The units can be removed with **UVAL**:

UVAL (CONST (var))

Access: APPS-EqnLib-COLIB
115 MENU

CONT

Resumes operation after suspension by **HALT**, **PROMPT**, etc. Annunciator shows HLT during halts. Also resumes editing sessions suspended by the editor's HALT. CONT in a program cancels the program, the same as **KILL**.

Access: LS-CONT

CONVERT

1. Converts one type of **Unit** to another - the programmable version of pressing LS and the new unit key. Units must be from the same category (eg, "LENG") or the error message will be "Inconsistent Units".

source-units target-units → new_target-units

12_cm 1_in → 4.724_in

Any numerical value of target-units is discarded.

You can also use the + key, but the numerical value of target- units is not discarded:

12_cm 2_in → '6.72440944882_in'

(12_cm is converted to inches and added to 2_in.)

Access: UNITS-TOOLS

2. The CONVERT menu (LS-6) is covered under the individual command entries.

COPY

Some undocumented features (on the 49G, these might apply only to ROM 1.19-6):

Clearing the Clipboard: in the edit mode with nothing selected, press RS-COPY. From the stack: press RS-PASTE and then RS-COPY. Press ON to clear the command line.

Copy All: press RS-down to go to the end of the file. Press RS-END, RS-COPY (or CUT).

Copying a Stack Level: press the up-arrow (interactive stack), cursor to the desired level (if required) and press RS-COPY.

Copying a Stack Level From a Program: as above, but press HIST instead of the up-arrow.

Access: RS-VAR

CORR

Correlation coefficient of independent and dependent columns of the statistics variable **ΣDAT**, which must exist in the current dir. See also **Sigma stats**, **XCOL**, **YCOL**,

→ n

Access: 96 MENU-FIT

COV

Sample covariance of independent and dependent columns of statistics variable **ΣDAT**. See also **COLΣ**, **PCOV**, **XCOL**, **YCOL**.

→ n

Access: 96 MENU-FIT

C→PX

(Note that on the HP49G+/50G later ROM versions, the graphics area includes the header for a total of 131 by 80 pixels. Older ROMs have 131 by 64 pixels.)

Coordinate to pixel conversion - the inverse of **PX→C**. Default upper left is (-6.5,4) ({#0d #0d}) and lower right is (6.5,-3.9) ({#130d #79d}) with newer ROMS.

(x, y) → {#m #n}

The conversion value depends on the settings in **PPAR** - coords can be anything you want, but pixel values are absolute and based on the 131 * 64 or 131 * 80 display size.

Setting the **BASE** to DEC will simplify pixel values.

Access: PRG-NXT-PICT-NXT

CR

Prints buffer contents, if any. Also creates the variable of print parameters **PRTPAR** in the Home dir.

If the calculator is connected to a terminal or printer, it sends a carriage return with CR.

The MS-DOS carriage return shows in calculator text as a black square. It can be removed by Finding and Replacing **CHR 13** (use the **CHARS** map to enter CHR 13; leave the Replace field blank).

Access: 104 MENU PRINT

C→R

Splits complex number or array into reals. See also [Complex numbers](#).

```
(1.,2.) → 1. 2.  
[(1.,2.) (3.,4.)] → [ 1. 3. ] [ 2. 4. ]
```

If you prefer an integer output, afterwards you can run [R→I](#) on numbers or the CAT XQ on arrays.

Access: PRG-TYPE-NXT,
MTH-NXT-CMPLX

CRDIR

Creates new subdir in current dir.

```
'name' →
```

For removing a dir, use the [FILES/FILER](#) command or see [PGDIR](#).

Access: PRG-MEM-DIR

CRLIB

Before making your own library, it's a good idea to use a library splitter from hpcalc.org to see how others did it.

Creates a library from a suitable directory. See also [LIBRARY basics](#).

```
(run from inside dir) → library
```

► Note that you must run CRLIB from your directory or the error message will be "Invalid \$ROMID".

The library object is then stored in any port:

```
portnum STO →
```

Example: 2 STO →

An ON+C warmstart then installs it (see [Reset](#)).

Since HP filenames cannot begin with a number, any libnums will have an L added when stored, as in L1758.

Basics

To create the dir, enter its name, which can be the same as your program (or not), and press CRDIR from CAT. Enter the following minimum variables (those marked optional can be omitted):

Main Program

Your main program - UserRPL, SysRPL, or ML. The name of this program is the one that appears in the RS-LIB program menu (not the libname - see \$TITLE below).

Subprograms

Subprograms are optional: your program can call any number of subroutines (or there can be none).

About (optional):

Useful for the version number, date, or other information . A typical way of doing it:

```
<< "Your text" SCROLL >>
```

STOre this in the dir as 'About' or other title.

\$CONFIG

This contains a small program to attach the lib on startup so it can run from anywhere in the path:

```
<< libnum ATTACH >>
```

Choosing A Libnum

Choosing a library number: it's up to you, but HP recommends restricting numbers to the range 769-1792. Numbers outside this range are for internal use. If there are conflicts with other libs, you can always renumber it by changing \$CONFIG and \$ROMID.

\$HIDDEN

A list containing the names of variables you don't want to show when you press RS-LIB-libname. These are usually everything but your program and About (if used).

You can also have any number of other files in the dir - if they aren't listed in \$HIDDEN or \$VISIBLE, they'll be ignored.

\$VISIBLE

A list containing the names and order of variables you want to show in RS-LIB-libname. The minimum is usually your main program name and About (if used).

\$ROMID

Contains only the libnum, such as 1758. It can be a real or an integer.

\$TITLE

A string with the label you want to appear in the listings and the RS-LIB library name, such as

```
"HLP49 v6.0 AP11".
```

The program name at the left (HLP49) becomes the main name of the library (not the main program, which stays the same) in the RS-LIB menu. Since space is limited, limit the string to 14-15 chars.

Finally...

You can now run CRLIB. If all goes well, the library will be on the stack and will look like:

```
Library 1758: HLP4...
```

Storing

Enter a portnum (0 to 2, 2 recommended) and press STO. Do an ON+C warmstart to attach your library and you should see it in RS-LIB. Libraries can be stored in port 3 (SD card), but won't run.

Any problems preventing creation will be in the dir, usually one of the config vars. The error messages should point these out.

Overwriting

If you make further versions of the lib, you'll need to delete the previous version before storing:

```
:portnum: libname PURGE →
OR
:portnum: Llibnum PURGE →
```

In the second one above, libnums in a port will have an L added.

Miscellaneous

To see or run your library's commands, you can also enter libnum MENU.

There are various library tools available from hpcalc.org. These can convert libs back to dirs and vice versa. OT49+ is a good one. There are also guides detailing more of the available library features.

Note that if you use these tools to convert your lib back to a dir, your subprograms in \$HIDDEN will be renamed to numbers in hex; *eg*, if you have MYPROG, MYPROG1, and MYPROG2, they'll become MYPROG, x002, and x003.

```
Access:    CAT,
           256 MENU
```

CROSS

Cross product of two vectors.

```
[ 'A' 'B' ] [ 'C' 'D' ] → [ 0 0 'D*A-B*C' ]
```

```
[2 2 3] [3 2 4] → [2 1 -2]
```

Numbers can be reals or integers, with the result in the same format.

```
Access:    MTH-VECTR
```

CSWP

Swaps matrix columns i and j.

$[[\text{matrix}]]\ i\ j \rightarrow [[\text{new}]]$

Access: MATRICES-CREATE-COL

Cursor keys

Arrows represent the cursor keys. **A + symbol indicates that the shift key is held down while pressing the cursor key.**

Stack mode:

→	SWAP (swaps Lev1, Lev2)
←	PICTURE (displays PICT)
↑	Interactive stack
↓	EDITB
Backspace	DROP (deletes Level 1)

RS →	Xmodem server
RS+ →	Kermit server
RS ←	not used
RS ↑	Interactive stack
RS ↓	displays softkey vars

LS →	SWAP
LS ←	not used
LS ↑	not used
LS ↓	EDIT
LS+ ↓	see Shortcut keys for EDIT versus VISIT

Edit mode:

(See also the entry for **TOOL/EDIT** for more editing commands.)

RS →	right end of line
RS ←	left end of line
RS ↑	top of file
RS ↓	bottom of file

LS →	right of screen
LS ←	left of screen
LS ↑	top of screen (equiv. to Page Up)
LS ↓	bottom of screen (equiv. to Page Down)

Command (entry) line:

RS →	right end of entry
RS ←	left end of entry
LS →	right of screen
LS ←	left of screen

CUSTOM

Displays a user-defined menu. Names of favorite variables are placed in a list and stored with the reserved name 'CST'. The **MENU** command can also be used (LS+ MODE-MENU or LS-PRG-NXT-MODE- MENU). See also **STARTEQW** for custom commands in the **EQW**.

```
{ A B C } 'CST' STO →
```

OR

```
{ A B C } MENU → stores in CST, displays new custom menu
```

Note that names in a list don't need to be in quotes.

Commands

Commands can also be entered:

```
{ SIN SQ EVAL }
```

If a directory does not have a CST, CUSTOM will display any CST higher in the dir tree. Note that any changes made to these CST variables will be stored in the *current dir*.

CST Programs

The custom menu can also contain programs that are not stored separately in variables. The program and its name should be in a sublist.

Example

The following CST displays A and B, which are stored vars, and M, a program within CST that returns the free memory count:

```
{ A B { M << MEM >> } }
```

After this is stored in CST, pressing CUSTOM displays the menu:

A	B	M
----------	----------	----------

A and **B** operate according to their contents, and **M** displays the free memory.

Changing Dirs/Paths

To run the program PROG2 from DIR2 and return to DIR1:

```
{ { PROG2 << { HOME DIR2 } EVAL PROG2 {DIR1} EVAL >> } }
```

Shifted Keys

CST menus can also have direct, left-shifted and right-shifted functions for each name:

```
{{ name {<< "prg1">> << "left">> << "right">> } } } 'CST' STO →
```

When CUSTOM is pressed, only **name** shows. Results:

```
name -      "prg1"
LS-name -   "left"
RS-name -   "right"
```

Access: LS-MODE

Custom Enter

Custom Enter, also called "vectored Enter", changes any command line entry to a string on level 1 and then runs user-written programs α ENTER and β ENTER if they exist in the path. The Enter key can then change inputs in whatever way the user programs. A few limitations apply:

1. Flag -63 must be set.
2. User mode must be locked on (annunciator USR showing), either with LS-alpha twice or flag -62. This means that all user key assignments will be active unless disabled first with DELKEYS (the keylist from RCLKEYS must be stored in a var).
3. The entry-handling program ' α ENTER' must be stored in the path. The program ' β ENTER' can be added for further processing after α ENTER completes. (α = alpha-RS-A, β = alpha-RS-B)

The following example converts entered numbers into hex binary integers.

Store the program below as α ENTER'

```
<< "#" SWAP + "h" + STR→ >>
```

Enter 0 DELKEYS to clear any assigned keys and restore any unassigned keys. Be sure that the BASE is set to Hex (and be sure to store your key assignments from [RCLKEYS](#)).

Turn on Custom Enter and User modes (*ie*, flags -62 and -63 set).

Entered numbers now become hex integers.

```
56D → # 56Dh
```


To turn this off, clear flags -62 and -63. Put the saved keylist on the stack and press **STOKEYS**.

Access: type in

CUT

Note that on the HP49G, the Cut and Copy features changed slightly with different ROMs. This is for v119-6 (or v2.09 on the HP49G+/50G).

To cut a line and append it to the clipboard, cursor to the line without selecting anything and press RS-CUT.

Note that it does not include linebreaks. A paragraph cut this way pastes as one long line.

Access: RS-STO

CYLIN

Sets cylindrical angular mode (same as Polar in the - note that the **POLAR** command is a plot type). Annunciator is $R\angle Z$. See also **RECT**, **Rect/Polar**, **SPHERE**.

Note that if you're using the MODE-Coord System field, the +/- key will step through the choices faster than the CHOOS box.

Values are entered as a vector (see [Arrays/vectors](#) for vector entry): $[x \angle y]$ or $[x y]$. In the case of $[x y]$, it will be converted to polar notation if cylindrical mode is set. The angle symbol is alpha-RS-6.

For example, in Rectangular mode, enter:

$(3 \ 4) \rightarrow (3., 4.)$

OR

$3 \ 4 \rightarrow V2 \rightarrow [3 \ 4]$

Enter **CYLIN** or change to Polar in MODE and see:

$(5., \angle 53.1301)$

OR

$[5. \angle 53.1301]$

The angle symbol \angle is alpha-RS-6 (chr 128). $\rightarrow V2$ is in MTH-VECTR.

The program below toggles between rectangular and cylindrical (polar) mode. You can store it as RP or similar or assign it to a key (see [ASN](#)):

```
%%HP: T(3)A(D)F(.);  
\<< -16 FS? IF THEN -16 CF  
ELSE -16 SF END \>>
```

$[3. \ 4.] \leftrightarrow [5. \angle .9273]$ (radians)

Entering a lot of polar notation is simplified with the program below. Flag -19 should be clear.

```
<< DEG CYLIN →V2 >>
```

magnitude angle → vector

```
7.5 45 → [7.5 ↗45]
```

Substitute RAD for degrees as required or remove DEG and set it externally.

You can also use the complex number format as vectors (within limits - see Arrays/vectors):

```
<< DEG -19 SF CYLIN →V2 >>
```

```
7.5 45 → (7.5, ↗45)
```

See the also the entry for **V→,→V2,→V3**.

Access: MODE-Coord System,
 LS+MODE-ANGLE,
 PRG-NXT-MODES-ANGLE,
 LS-MTH-VECTR-NXT,
 65 MENU

DARCY

Calculates the Darcy fluid friction factor.

```
xe/D yRe → xDarcy
```

Access: 117 MENU,
 APPS-EqnLib-UTILS

DATE/DATE+

DATE returns mm.ddyyyy (flag -42 clear) or dd.mmyyyy (flag -42 set).

DATE+: past/future date, given number of days.

```
date +/-days → newdate
```

Date should be mm.ddyyyy (flag -42 clear) or dd.mmyyyy (flag -42 set).

Access: RS-TIME-TOOLS,
 RS+TIME,
 LS-PRG-NXT-NXT-TIME

→DATE

Sets new date into system date.

```
(flag -42 clear)  mm.ddyyyy →
(flag -42 set)   dd.mmyyyy →
```

Access: RS-TIME-TOOLS,
 RS+TIME,
 LS-PRG-NXT-NXT-TIME

dB

Decibel, a typing aid for user-defined units.

```
             → 1.
5.1_        → 5.1_dB
```

Access: 117.02 MENU,
 APPS-EqnLib-UTILS

DEBUG

Single-steps through program. **IFERRs** should have the **HALT** command inserted after each IFERR or the error section will be executed as one step (remove these when done).

```
<any parameters> 'pgmname' →
OR
<any parameters> << pgm >> →
```

DEBUG	starts debugging
SST	single-steps program
SST↓	includes subroutines
NEXT	see next 2 steps (no evaluation)
HALT	suspends operation; CONT resumes
KILL	abandons Dbug

The stack remains active during debugging - you can alter the results of the steps to see what happens.

Access: PRG-NXT-NXT-RUN,
 41 MENU

DDAYS

Days between dates, where the date is mm.ddyyyy (flag -42 clear) or dd.mmyyyy (flag -42 set). The most recent date should be on Lev 1 to avoid negative results.

```
date1 date2 → days
```

Access: RS-TIME-TOOLS,
RS+TIME,
LS-PRG-NXT-NXT-TIME-NXT

DECR (Decrement)

Subtracts 1 from named var stored in current dir or local var within a program; returns new value. The var should contain a real or integer.

'name' → newval

See also [INCR \(Increment\)](#).

Access: PRG-MEM-ARITH

DEF/DEFINE

Both store the expression on the right side of an equation in the variable specified on the left, or create a simple solver program from an equation.

Ex. 1: 'X=(a+b)' → stores '(a+b)' in 'X'

Ex. 2: given equation of form 'F(X)=SIN(X)^2', creates var 'F' containing program:

<< → X 'SIN(X)^2' >>

The F var now returns the sine squared of any number. You can also use the F(n) format:

F(2) EVAL → .826821810432

See also [APPLY](#).

Access: DEF key,
CAT (DEFINE)

DELALARM

Deletes specified alarm. The alarm number can be obtained from RS-TIME-Browse alarms.

0 deletes all alarms.

nindex →

Access: TIME-ALRM

DELAY

The pause in seconds between lines sent to the printer. The default is 0. See also **PRTPAR**.

n →

Access: 104 MENU-PRINT-PRTPA

DELKEYS

Deletes key assignment. 0 DELKEYS deletes all key assignments, and if normal key functions have been disabled with 'S' in the keylist (see **STOKEYS**), restores the normal functions to unassigned keys.

See also **Key Numbers**, **ASN**.

Key#nn.n →
0 →
'S' →

Using DELKEYS with the 'S' option allows locking out all unassigned keys - see **STOKEYS**.

Access: LS+MODE-KEYS,
PRG-NXT-MODES-KEYS

DEPND

Sets the dependent variable name in plotting. This appears as the last element in the **PPAR** list. Also sets the range for **TRUTH** plots.

The HP48 permitted arrays as well as reals in DEPND; the 49G/G+/50G allow only reals. Certain HP48 plots might not work on the 49 or 50 if DEPND requires arrays.

name →
{name low high} →
low high →

Access: 81 MENU-PPAR

Derivative (∂)

Finds derivative of algebraic expression on Level 2 with respect to variable on Level 1.

expr var → deriv
'X^3' 'X' → 'X^2*3'

In ASCII algebraics, ∂ stands for the complete derivative - not only partials. The format is '∂name(symb)'. For example, the derivative of X with respect to t is '∂t(X)'.

You can also enter ' $\partial X(X)$ ' (on the stack with textbook display this is $\frac{\partial}{\partial X}(X)$).

`' $\partial X(X^2)$ ' EVAL → ' $2*X$ '`

The second derivative can be expressed as ' $\partial X(\partial X(X))$ '.

`' $\partial X(\partial X(16*X^2))$ ' EVAL → NUM → 32.`

When writing an equation for the DESOLVE command (LS-S.SLV) for solving ODEs, the syntax for the first derivative is:

`'d1X(t)=expr' 'X(t)'`

This is the derivative of X with respect to t. You can use any vars as long as you keep the format d1Symb(Symb); eg, d1X(X).

The second derivative is:

`'d1d1X(t)=expr' 'X(t)'`

Note that DESOLVE creates (or overwrites) the var ODETYPE in the current dir. It contains the type of equation, such as "1st order linear". Note also that the DESOLVE command will overwrite VX with one of the variables it's using.

After rearranging terms, DESOLVE integrates both sides to solve differential equations.

Example 1: a simple ODE

`'d1Y(X)=X' 'Y(X)' → 'Y(X)=SQ(X)/2+cC0'`

where cC0 is the arbitrary constant.

Example 2: second-order ODE

Textbook format: $a = \frac{d^2 s}{dt^2}$

DESOLVE format:

`'d1d1s(t)=a' 's(t)' → 's(t)=(2*cC0+(2*t*cC1+t^2*a))/2*EXP(0)'`

Simplifying and substituting conventional terms for acceleration, velocity, etc, in place of constants:

$$s = \frac{1}{2}at^2 + v_0t + s_0$$

where v_0 = initial velocity

s_0 = initial height or distance

See also the derivative and DESOLVE sections of the **EQW** entry.

Access: RS-COS

DET

Determinant of square matrix, numerical or symbolic:

```

[[sq matrix]] → n

[[ 'A' 'B' ]
 [ 'C' 'D' ]] → 'D*A-C*B'

```

Access: MATRICES-OPER

DETACH

Detaches library from current directory. See also [ATTACH](#).

```
libnum →
```

To make a library's commands available from only one dir, go to the Home dir and run libnum DETACH, then go to the desired dir and run libnum ATTACH. This change will be undone by the next reboot unless you put it in [STARTUP](#) (see also [PATH](#)).

For another example, see the ATTACH/DETACH section of the [LIBRARY basics](#) entry.

Access: 110 MENU

DIAG→

Creates a diagonal matrix given an array and the dimension.

```

[ 1 2 3 ] 3 → [[ 1 0 0 ]
                [ 0 2 0 ]
                [ 0 0 3 ]]

```

Access: MATRICES-CREAT-NXT

→DIAG

Inverse of DIAG→ – splits diagonal matrix into array.

```

[[ 1 0 0 ]
 [ 0 2 0 ]
 [ 0 0 3 ]] → [ 1 2 3 ]

```

Access: MATRICES-CREAT

DIFFEQ (Plots, Numerical)

1. Sets the plot type to differential equations. ► For numerical solving, see part 2 below plots.

See **Plots** for more on general plotting. See also **Derivative (∂)** for brief details on solving DEs.

The following two examples were adapted from the HP48 User's Guide (better than the HP49 manuals for basics, and a free download if you search hpcalc.org).

If a previous plot has changed the parameters, delete **PPAR**.

1. Plot $Y' = Y+T$ for $Y(0)=2$ over the interval $[0,1]$.

Press LS+2D/3D (Plot Setup window) and enter:

Type: Diff Eq **Angle:** RAD
F: 'Y+T'
H-Var: 0 **V-Var:** 1 ___Stiff
Indep: 'T'
H-Tick: 1 **V-Tick:** 1 ___Pixels

Press LS+WIN to change to the Plot Window and enter:

H-View: -1 2
V-View: -2 8
Init: 0 **Final:** 1
Step: Default **Tol:** .0001
Init-Soln: 2

(Pressing 0 for Step enters the default setting.)

Press **ERASE** (optional), **DRAW**.

Values that will be stored in the current dir by the plotter:

Y=6.15477759086
T=1

2. Plotting of **stiff diff eqs** can be made faster by entering partial derivatives:

Plot $Y' = -1000*(Y-SIN(T))+COS(T)$, $Y(0)=1$

Press LS+2D/3D for Plot Setup. Enter:

F: '-1000*(Y-SIN(T))+COS(T)'
Angle: RAD

Check the **Stiff** field for the two partials $\partial F/\partial Y$ and $\partial F/\partial T$.

A note on the partials: the easiest way is storing them before plotting. Put the main expression on the stack and dupe it with ENTER. Enter 'Y' and press **RS-∂** to get the partial. Store this as 'FY'.

Repeat using the var 'T' and store this partial as 'FT'. In the **2D/3D** screen, enter the name **FY** into $\partial F \partial Y$ and the name **FT** into $\partial F \partial T$ - the stored values will appear in the fields (don't use single quotes, which will insert the name instead of the value).

Alternative method: enter the main expression into **F:** of the 2D/3D DiffEq screen. Press **RS-COPY**. Check the **Stiff** field and move the highlight to $\partial F \partial Y$.

Press **NXT-CALC** for the stack.
 Press **RS-PASTE** and enter 'Y'.
 Press **RS-∂** for the partial and then **OK** to return.
 The value will transfer to the field.
 Repeat for $\partial F \partial T$ using 'T' as the var.

For reference:

$$\partial F \partial Y = -1000$$

$$\partial F \partial T = '-1000 * -\cos(T) - \sin(T)'$$

Enter:

H-Var:0 **V-Var:**1
Indep:'T'
H-Tick:10 **V-Tick:**10
 Check the **Pixels** field

Press **LS+WIN** for the Plot Window and enter:

H-View: -1 2
V-View: -1 1
Init: 0 **Final:** 1
Step: Default **Tol:** .0001
Init-Soln: 1

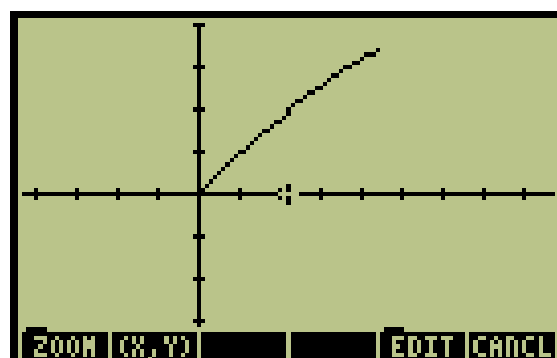
(Pressing 0 for Step enters the default setting.)

Press **ERASE** and **DRAW**. Plot time is typically 45 seconds. (With **Stiff** unchecked, plot time is over 5 minutes.)

Values stored in current dir:

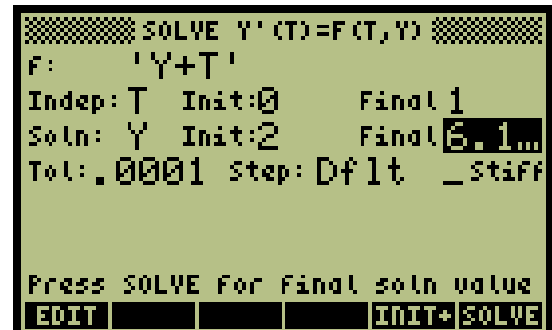
T: 1.
Y: .841569099036

Access: **LS+2D/3D-Type**,
82 MENU



2. Numerical Solving. Use RS-NUM.SLV-DIFFEQ. Example: solve the equation from the first plot above, $Y' = Y+T$ for $Y(0)=2$ over the interval $[0,1]$. Enter:

f: 'Y+T'
 Indep: T Init: 0 Final: 1
 Soln: Y Init: 2 Final:
 Tol: .0001 Step: Dflt Stiff



Highlight the lower **Final** and press SOLVE. The full answer will be on stack Level 1 with the Tolerance on Level 2. INIT+ will move the new values into the INIT fields so you can solve from these points.

There is little in the 49/50 manuals about this feature. A much better discussion is in the HP48G Series User Guide, hpcalc.org.

DIR

Labels a group of variables as a directory. For navigating the dirs, see [PATH](#). The syntax is:

```
DIR name var1 ... name varn END
```

If you're making a dir manually, the names between DIR and END do not need single quote marks.

Access: CAT

DISP

Displays a string on line **n** (1 to 7 for 49G+/50G Font 8) until a keypress. Must be followed by x FREEZE (see the [FREEZE](#) entry for values of x - 7 freezes all). It can't write to the menu area.

Example:

```
<< "string" 2 DISP 7 FREEZE >> →
```

CLLCD can be used first to clear the screen. Any key restores the display. To display for a fixed number of seconds, replace x FREEZE with s **WAIT**, where *s* = seconds.

Access: PRG-NXT-OUT

DISPXY

Similar to DISP, but allows a string in any desired screen location in pixels, plus different font sizes. Must be followed by [FREEZE](#) in the same way as DISP. It can't write to the menu area.

```
<< "string" {#m #n} size DISPXY x FREEZE >> EVAL →
```

Size: 1 = minifont
 2 = current font
 x: see FREEZE entry, else use 7
 {#m #n}: the upper left corner of the string in pixels

Upper left screen is { #0 #0 } - for ROM v1.23, lower right is { #130d #63d }. Later versions such as v2.09 have a lower right of { #130d #79d }.

Access: 2269 MENU

DMS (degrees, minutes, seconds)

Oddly, the 49/50 series has no direct →DMS command, but the →HMS command can do the same thing, though it can be improved.

The program below converts decimal degrees to dd.mmss and also corrects for improper formatting (such as returning 1.60 degrees instead of 2.0). You can store it as DMS or any name you prefer.

decimal_degrees → dd.mmss

```
%%HP: T(3)A(D)F(.);
```

```
\<<
```

```
IFERR
```

```
IF DUP TYPE 0. ==
```

```
THEN DUP IP SWAP
```

```
FP 3600. * 0 RND 60.
```

```
DUP2 MOD 10000. / 3
```

```
ROLLD / IP 100. / + +
```

```
IF DUP FP .6 \>=
```

```
THEN CEIL
```

```
END
```

```
END "dd.mmss" \->TAG
```

```
THEN "Real\->DMS"
```

```
END
```

```
\>>
```

The tag on the result can be removed by pressing **EVAL**.

To convert back to decimal degrees, use the HMS→ command in RS-TIME-Tools-NXT (167.02 MENU).

dd.mmss → decimal_degrees

An alternative to the above is to store the program below as DMS→ or similar:

```
<< HMS→ >>
```

ACCESS: Copy, paste

DOERR

User-specified error trip for **IFERR** (any message displays in **MSGBOX** - but see **STARTERR**). Often used with an **IF-THEN** statement: IF <condition> THEN 0 DOERR END

This will activate IFERR if <condition> is non-zero (and the program is enclosed by IFERR-THEN-END).

Must be preceded by n, where n = errm# or user string. If none, use "" or 0.

```
n →  
"user string" →
```

Can also be used to find an error message if you know the number:

```
# errnum → "error msg"
```

The format below lets you write your own error message. It can be used anywhere in a program as often as needed:

```
IFERR <procedure> THEN "Your msg" DOERR END
```

Access: PRG-NXT-NXT-ERROR

DOLIST

Applies program, command, or function to each element of list on Lev 3. Results are merged with any other lists (see examples below).

```
list1 ... listn  
number of lists  
    << program >> → newlist  
  
{ 1 2 3 }  
  1  
<< 2 * >> → { 2 4 6 }  
  
{ A B C }  
{ D E F }  
  2  
<< 2 * >> → { A 'D*2' B 'E*2' C 'F*2' }
```

See also MAP in CAT.

Access: PRG-LIST-PROC

DOSUBS

Applies obj (a program, command, or name) to groups of elements in a list. To put a command on the stack without evaluation, see [LIST series](#) or →[TAG](#). Any named var should contain a pgm or cmd.

```
    {list}  
group (a real)  
    obj → {list}
```

```

{ A B C D }
  2.
<< SQ >>   →   { A 'SQ(B)' B 'SQ(C)' C 'SQ(D)' }

{ A B C D E }
  3.
<< SQ >>   → { A B 'SQ(C)' B C 'SQ(D)' C D 'SQ(E)' }

```

In the first above, SQ is applied to every second element of groups of two. In the second example, it's applied to every third element of groups of three.

The program below was adapted from a post on comp.sys.hp48 by John H Meyers. Given a list of coefficients and a varname, it returns an algebraic expression.

```

%%HP: T(3)A(D)F(.);
\<< -3. CF 0. ROT 1. \<< 3. PICK
  ENDSUB NSUB - ^ * + \>> DOSUBS
  SWAP DROP SIMPLIFY \>>

{ 3. 4. -5. } 'X' → '3.*X^2.+4.*X-5.'

```

The program uses ENDSUB-NSUB (total minus position) for the exponent of each term, which is always one less than its position. See also [ENDSUB](#), [NSUB](#).

Access: PRG-LIST-PROC

DOT

Dot product of two real, integer or symbolic vectors (arrays).

```

[1 2] [3 4] → 11
[ 'A' 'B' ] [ 'C' 'D' ] → 'B*D+A*C'

```

Access: MTH-VECTR

DO-UNTIL

Conditional loop repeats procedure until test1 is non-zero.

```
DO <procedure> UNTIL <test1> END
```

If test1 never returns a non-zero value, you can stop the loop with ON (sometimes repeatedly).

Typing aid: when the DO dir is displayed (LS-PRG-BRCH), these are available:

```

LS-DO → DO
        UNTIL
        END

```

The example below clears the stack of all but the last object:

```
<< DO DROP UNTIL DEPTH 1 == END >>
```

Access: PGM-BRCH

D→R

Converts degrees to radians. Its inverse is **R→D**.

degrees → radians

The formula is $\text{deg} * (\pi/180)$.

Access: MTH-REAL-NXT-NXT

DRAW

Plots expression or list of exprs in EQ. Use **DRAX** to draw axes or select AXES key in the LS+2D/3D window (press ENTER after). It will use the existing plot type in **PPAR** unless you specify another.

If PPAR doesn't exist, DRAW will create the default, a function in X.

Note that DRAW will merge the new plot with any existing image unless you first use ERASE.

This program can be assigned to a key for convenient plotting:

```
<< DRAX DRAW PICTURE >>
```

Optional: add **ERASE** before DRAX.

Plots: each side of an equation is plotted separately; *ie*, for 'X^2=5', it will plot X^2 and then 5. 'X^2-5' is the preferred form.

See the **Plots** entry for more information.

Access: LS+2D/3D,
81 MENU

DRAW3DMATRIX

Plots a 3D image from a matrix without using the **EQ** or **PPAR** vars. It will overwrite **PICT**.

```
[[matrix]] vmin vmax →
```

Operation is similar to **FAST3D** and the plot can be rapidly rotated with the cursor keys. One difference is that TRACE lacks the coordinate display feature. The following example can be copied and pasted to the stack to show a typical use:

```
[[ 3 8 8 1 ]
 [ 8 4 1 8 ]
 [ 2 1 1 5 ]
 [ 1 1 5 7 ]] 1 10
```

The matrix does not remain on the stack. To retrieve it, press UNDO immediately after exiting.

Access: CAT

DRAX

Draws axes in **PICT** - the program version of the Axes toggle in the LS+2D/3D window. See **AXES** or **ATICK** for labels or tick spacing.

This program can be assigned to a key for convenient plotting of the expression in **EQ**:

```
<< DRAX DRAW PICTURE >>
```

Optional: add ERASE before DRAX.

Access: 81 MENU

DTAG

Removes tag from obj (so does multiplying by 1 and most other operations, such as **EVAL**). **OBJ→** removes the tag and keeps it as a string on Lev1.

```
:tag:obj → obj
```

```
:tag:obj OBJ→ → obj "tag"
```

Access: PRG-TYPE-NXT

e

Base of natural logs. Returns e (symbolic, flag -3 clear) or 2.1828182846 (numeric, flag -3 set).

Access: alpha LS-E

E/EEX

E is the symbol for exponent notation, 10^n .

```
2.E15 = 2.*10^15
```

It's used by the SCI and ENG number formats, the EEX key, (and any number too large or small for normal display). See the entry for **Mode-FMT menu**.

EEX (Enter Exponent) allows entering very large or small numbers, such as Boltzmann's constant ($1.380658 * 10^{-23}$):

1.380658 EEX 23 +/- → 1.380658E-23

It's also useful for entering large numbers such as 1 million:

EEX 6 ENTER → 1000000.

Thousands separators for reals are available in fixed modes (see FMT link, above).

Access: EEX key

EDIT/EDITB

Commands for EDIT/EDITB are in the TOOL/EDIT entry.

On the 49G+/50G EDIT and EDITB with later ROM versions such as v2.09 can use the full 12-line screen; earlier versions such as v1.23 display a blank header.

EDIT opens the text editor for editing most types of objects. If the stack is empty, start a new editing session with a null (empty) string ("" - see the entry for **NOT** to automate this).

See also the entries for **STARTED** and **EXITED**. EDIT displays a string's double quotes.

To embed double quotes inside a string, precede each with a backslash, alpha-RS-5, char 92. These display in EDIT but not in EDITB, the stack, or **SCROLL**.

Access: LS-down cursor (*continued below*)

EDITB selects the best editor for the object type. Examples:

string, list, pgm:	text editors EDIT or EDITB
algebraic:	Equation Writer
matrix:	Matrix Writer
grob:	graphics editor

EDITB doesn't display a string's double quotes and doesn't display the backslash used to embed quote marks (see EDIT above).

On the 49/50 series, LS+Down is similar to Down (which is EDITB), but can act as **VISITB** in that it edits the *contents* of a named var rather than the varname:

	<u>Down</u>	<u>LS+Down</u>
name	edits name	edits content
object	edits obj	edits obj

Access: down cursor,
TOOL-EDIT (calls EDITB)

EGV

Eigenvectors and right eigenvalues for a square matrix.

$$[[\text{matrix}]] \rightarrow \begin{matrix} [[\text{evectors}]] \\ [\text{evalues}] \end{matrix}$$

Access: MATRICES-NXT-EIGEN

EGVL

Eigenvalues for a square matrix.

$$[[\text{matrix}]] \rightarrow [\text{evalues}]$$

Access: MATRICES-NXT-EIGEN

ENDSUB

Total number of sublists or elements in list used by DOSUBS.

$$\begin{matrix} \{ 22. 33. 44. 55. \} \\ 1. \\ \ll \text{ENDSUB NSUB} \gg \text{ DOSUBS} \rightarrow \{ 22. 4. 1. 33. 4. 2. 44. 4. 3. 55. 4. 4. \} \end{matrix}$$

In the above simple demonstration, ENDSUB returns 4 (no. of elements) for each element, while **NSUB** returns each element's position (1, 2, 3, or 4).

See the entry for **DOSUBS** for more.

Access: PRG-LIST-PROC

ENTRY

Starts the command line below the stack for data entry. It can also change the editor's type of entry as explained below:

Changes the command line or editor text entry mode to Algebraic (no spaces added), or Program (spaces added).

Annunciators show only when the header is displayed.

Editor result when inserting a hyphen (minus key) into "AB":

<u>ANNUN.</u>	<u>ALPHA</u>	<u>SPACES</u>	<u>RESULT</u>
Prg	off	yes	"A - B"
Prg	on	no	"A-B"
Alg	off	no	"A-B"
Alg	on	no	"A-B"

In ALG mode, command keys take arguments: SIN appears as SIN(). In PRG mode only the command name appears.

This also applies inside the editor, although the annunciators don't appear in the header.

Access: RS-ALPHA

EQ

Reserved name for the variable that holds an equation or expression for the plotter or solvers; multiple equations or expressions should be in a list. EQ is created by the plotter or solvers. There can be an EQ in each directory.

It will be created when you enter a formula into LS+**2D/3D**-EQ. See also LS+**WIN**.

To create it manually:

'expr/eqn' {or list} STEQ →

OR

'expr/eqn' {or list} 'EQ' STO →

Examples:

'X^2+2*X' →
{ 'A^2-B^2' '2*A^3' } →

EQ can be recalled with **RCEQ** if it exists in the current dir.

Plots: each side of an equation is plotted separately; *ie*, for 'X^2=5'; it will plot X^2 and then 5. 'X^2-5' is the preferred form.

See also **Plots**, **STEQ**.

Access: LS+2D/3D,
LS+WIN,
STEQ (CAT) or type in

EQ→

Returns split equation.

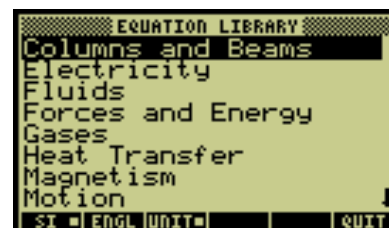
'A+5=B' → 'A+5' 'B'

Returns 0 if it errors; otherwise the same as **OBJ**→DROP2. EXLR is similar (CAT).

Access: PRG-TYPE-NXT

EQLIB Equation Library

The HP48 Equation Library was removed from the HP49G and then came back as separate libraries in OS v2.09 (available from the 49G+ or 50G sections of www.hp.com/calculators/ or from hpcalc.org). It should appear in the list when you press **APPS**.



Installing: (if needed) If it isn't built-in, install by transferring libraries 226 and 227 to the Home dir (or SD card) and then use **FILES** to move them to port 1 or 2. Do an ON+C warmstart and "Equation Library" will appear in the APPS box.

Press this and then EQLIB, EQNLI. You can also enter 114 MENU or use the EQNLIB command (CAT, program, or typed in).

Basic Operation

At the main menu, you can select SI or English units, or no units. These appear with VARS-NXT.

An annoyance: the QUIT and EXIT keys are where you expect to find OK. Use ENTER to run the selected item.

SOLV: uses the Multiple Equation Solver. See the SOLVR entry for details. In short, enter a value and press its unit key. To solve for an unknown, press LS-varname.

EQN/NXEQ: displays the equations. NXEQ appears after EQN if there's more than one equation.

VARS: displays all the vars in the eqn(s). If units were selected at the main menu, the NXT key toggles them.

PIC: displays a picture if there is one.

→**STK:** sends the eqn list to the stack.

EXIT: returns to previous menu.

PICT Tip: A useful aid: if the selected item has a picture, press PIC and then →**PICT** (which will appear when any pic is displayed). The pic can be viewed at any time during stack calculations by pressing the left cursor.

See also **SOLVEQN** for extracting and solving equations.

Access: APPS-Equation Library, then EQLIB, EQNLI
114 MENU,
EQNLIB command

==/= (equals)

The keyboard = is used in equations in the usual way:

'x=5' 'a+b=42'

It's also used as "**equate**" and makes an equation from two algebraics:

'X' 'Y+2' → 'X=Y+2'

Access: RS-W

The double == is used in programming as the usual "equals". IF X 5 == reads "If X equals 5". See also [SAME](#).

Also used for testing equality:

obj1 obj2 → 0/1

Access: PROG-TEST,
 alpha-RS-W twice

EQW

The Equation Writer makes it easy to enter a complicated algebraic formula. Editing that formula, however, is often easier with ASCII text (EDIT in EQW or LS-down arrow from the stack).

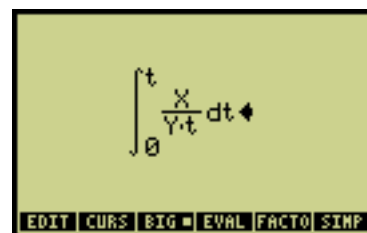
OPENING:

You can open the EQW directly by pressing its key. You can also use an algebraic, a global name, or a complex number on Lev 1:

obj EQW HIST ECHO →

OR

obj EQW (typed in or CAT) →



► Note that pressing EQW and typing it in are not the same thing.

If the object is a variable name, pressing [EVAL](#) replaces it with its contents.

If the object is on other than Lev 1, [HIST](#) allows cursoring to it.

If the command EQW is used in a program, it will open with any valid object on Lev1 (and errors with Too Few Arguments otherwise).

CURSORS:

The right cursor moves to the next step for entering values.

The right cursor applies highlighting to any or all of the expression. The down cursor undoes the highlighting a step at a time. To apply an operator to the entire expression, highlight all of it with the right cursor.

CUSTOM COMMANDS:

You can use custom commands in EQW - see [STARTEQW](#). Your choice of commands appears when you press the **CUSTOM** key.

UNITS:

Currently (ROM v2.09), the EQW does not support units.

KEY ACTIONS:

Some of the EQW keys behave differently from stack mode:

Y^X inserts superscript in the EQW and the $^$ symbol in Edit.

Σ inserts the sigma symbol; for the syntax, see the entry for Sigma (Σ).

Other math keys insert the corresponding symbols.

INTEGRALS:

\int inserts the integral sign and the closing d. The sequence is below (\blacktriangleright = right cursor):

\int lower limit \blacktriangleright upper limit \blacktriangleright integrand \blacktriangleright variable of integration

Example: The indefinite integral of X^2 :

\int 0 \blacktriangleright X \blacktriangleright X^2 \blacktriangleright X ENTER

EVAL or SIMPLIFY returns ' $X^3/3$ '.

The ASCII form is ' \int (0,X, X^2 ,X)' (\int (lower, upper, integrand, variable of integration)).

DERIVATIVES:

∂ starts the derivative notation; *eg*, the derivative of X with respect to t:

$$\partial t \blacktriangleright X \rightarrow \frac{\partial}{\partial t}(X)$$

In ASCII this will be ' $\partial t(X)$ '. You can also enter $\partial X(X)$.

DESOLVE: When writing an equation for the DESOLVE command (LS-S.SLV), the syntax for the first derivative is:

'd1X(t)=expr' 'X(t)'

and the second derivative is:

'd1d1X(t)=expr' 'X(t)'

Example: 'd1Y(X)=X'
'Y(X)' → 'Y(X)=SQ(X)/2+cC0'

where cC0 is the arbitrary constant. DESOLVE integrates both sides to solve ODEs.

See the entry for **Derivative (∂)**.

COMB/PERM:

For combinations and permutations of the form m things taken n at a time, enter m, press **COMB** or **PERM** (from MTH-PROB), then n. If desired, press the right cursor to highlight the expression and EVAL or →NUM to convert it to a number. If on the stack, convert this algebraic to a number with EVAL or →NUM.

LISTS,ARRAYS,UNITS:

Some functions, such as lists, arrays, and units are currently not supported in the EQW.

CUT/PASTE:

Copy, Cut, and Paste work as usual, though cursor highlighting is used instead of Begin/End.

CURS:

The CURS softkey allows rapid cursoring to any part of the expression; press Enter to return to editing with that part highlighted. Enter returns to the stack; ON abandons any changes.

To use the EVAL, FACTOR, or SIMPLIFY softkeys, first highlight the desired term(s).

Access: EQW(49G),
RS-EQW(49G+/50G),
Type in (not the same - see above)

ERASE

Erases the graphic in **PICT**. There is no confirm and no undo.

To save the graphic first, use the graphics editor EDIT-NXT-NXT-PICT→ to put a copy on the stack.

Access: WIN, LS+2D/3D,
81 MENU,
graphics EDIT-NXT

ERR0

Clears error number and error message. Next ERRN returns #0h and next ERRM returns "".

Access: PRG-NXT-NXT-ERROR

ERRM

Returns last error message.

→ "message string"

It can be used anywhere in programs to specify a particular error; use it as many times as needed:

```
IFERR <procedure> THEN ERRM END
```

For changing messages to the header or controlling display time, see [STARTERR](#). For writing your own error message, see [DOERR](#).

Access: PRG-NXT-NXT-ERROR

ERRN

Returns number of last error message.

→ # n

Error numbers are user binaries in the current base. Most of the common ones start at # 201h.

Given the error number, the message can be found with [DOERR](#):

```
errn → "error message"
```

Access: PRG-NXT-NXT-ERROR

EVAL

Evaluates an object.

<< pgm >>	→	runs program
'pgmname'	→	runs program
'globalname'	→	contents of global variable
{a b c}	→	'a' 'b' 'c'
{1 2 3}	→	1 2 3
'2*3'	→	6

An algebraic expression whose symbols have stored values (global or local) will have its symbols replaced by the stored values. If 'x' contains 3 and 'y' contains 2, then

$$'x+y' \rightarrow 5$$

Some expressions are simplified:

$$\begin{aligned} \text{ABS}(-5) &\rightarrow 5. \\ '1/(\text{SQ}(X-2)+1)' &\rightarrow '1/(X^2-4*X+5)' \end{aligned}$$

Lists will be split into elements in the same way as **OBJ**→ but without the number of elements appearing on the stack. One difference is that programs in a list will run.

Strings, arrays, and numbers are not affected (unless numbers are in algebraic form: '2*3' → 6).

EVAL works only on the outside of composite objects; eg, if you eval this program

```
<< 2 << 3 4 + >> >>
```

the stack will contain:

```
      2
<< 3 4 + >>
```

To place a program name on the stack without evaluation, use quotes: 'MYPROG'

To place a command, etc, on the stack without evaluation, you can use lists or tags:

```
{SIN}
::SIN
::MYPROG
```

With tags, only a single colon will appear on the stack.

A program with large algebraics to be EVALed may be very slow due to EVAL's CAS extensions. Using stack operations instead will be faster.

Access: RS-P (49G),
EVAL (49G+/50G)

EXACT mode

Exact and approximate modes are toggled with MODE-CAS or RS+ENTER. The annunciator in the header is = for Exact and ~ for Approximate.

In Exact mode with Symbolic mode (flags -2,-3 clear), operations result in exact results (such as 5/3). If flags -2,-3 are set for Numeric mode, results will be approximate decimals (such as 1.6666).

Exact/Numeric is a good general-purpose setting for the calc.

In Exact mode, numbers entered from the keyboard appear on the stack as integers (if you don't enter a decimal fraction). In Approx mode, numbers will be reals with an added decimal point.

While Exact mode is best for the CAS, Approx is preferred for complex inputs.

Approximate mode uses numerical results in calculations. Exact mode and CAS produce a symbolic result in algebraic calculations.

If you're solving a symbolic expression that requires the approximate mode, the calc will usually switch automatically - it will ask first, and saying no will cancel the operation. If flag -123 is set, auto mode switching is prevented.

Access: MODE-CAS,
 RS+ENTER

EXITED

If this variable contains a program in the Home dir, it will automatically reset the edit mode as desired on exit. A typical usage:

```
<< 2 →HEADER -73 CF >>
```

This restores the header if it's been changed and resets the editor to the current font.

Sometimes this doesn't always work if you exit with ON, depending on the OS version - it sometimes leaves 'EXITED' on the stack. Press **EVAL** to run it if this happens.

See also **STARTED**.

Access: type in

EXP

The algebraic expression equivalent of exponential notation:

$$'EXP(X)' = e^X$$

The program << 1 EXP >> returns 2.71828182846 when EVALed.

Access: LS-Q

EXPAN/EXPAND

Expands and simplifies algebraic or numeric expressions. For EXPAND, flags -2 and -3 should be clear for symbolic mode. EXPAN works in either mode.

$\text{expr} \rightarrow \text{newformat}$

$'(X+1)^2' \rightarrow 'X^2+2*X+1'$

Access: ALG,
SYMB-ALG,
93 MENU (EXPAN)

EXPFIT

Stores EXPFIT in **Σ PAR** for exponential fitting by **LR**. Creates Σ PAR if required.

Access: 96 MENU- Σ PAR-MODL

EXPM

Returns e^x-1 (ie, $(e^x)-1$).

$x \rightarrow e^x-1$

Access: MTH-HYP-NXT,
LS-8

EYEPT

Eyepoint in 3D plots.

$x \ y \ z \rightarrow$

Real numbers that appear in LS+WIN as XE, YE, and ZE and are stored in **VPAR** (see also the entry for this). YE must be 1 unit less than Ynear of **YVOL**.

Access: LS+WIN,
81 MENU-3D-VPAR

FACT

Factorial - same as ! in MTH-PROB.

$n \rightarrow n!$

Access: CAT

factorial (!)

Factorial - same as FACT from CAT. Expression syntax is '*n*!'.

$$\begin{array}{lcl} n & \rightarrow & n! \\ 5 & \rightarrow & 120 \end{array}$$

Access: MTH-NXT-PROB

FANNING

Returns the Fanning fluid friction factor.

$$Xx/D \ YRe \rightarrow XFanning$$

Access: 117 MENU,
APPS-EqnLib-UTILS

FAST3D

Sets plot type in PPAR (spelled Fast3D in LS+2D/3D). **EQ** should contain an expression in two variables (typically X and Y). It creates the 3D variable **VPAR** as required.

The plot can be rotated around the X-axis with the up/down cursor keys and around the Y-axis with the left-right keys.

If you press TRACE, the cursor can be stepped around the image with the cursor keys while the coordinates appear in the upper left.

The View Volume and steps can be set with LS+WIN. To return to the defaults, press NXT-RESET-Reset All.

' $2 * X^2 + Y^2 - 2$ ' gives an example. Place it on the stack and type **STEQ**. Type or select FAST3D. Press LS+2D/3D and DRAW.

See also **DRAW3DMATRIX**.

See also **Plots** for more on general plotting.

Access: CAT,
LS+2D/3D-Type

FFT

1- or 2-dim. FFT of an array. Number of terms must be a power of 2.

array1 → array2

Access: MTH-NXT-FFT

FILES/FILER

Opens the file manager with LS-G or the **FILER** command (CAT or typed in).

SLOW SD CARD

Note that the filer can take a very long time opening the SD card menu. It depends on the size of the card and, especially, the number of files. You can speed it up considerably by using a cardreader to move files into a few folders.

NAVIGATING

The right cursor displays the contents of a highlighted dir or the menu(s) of a highlighted library. The left cursor steps back to the previous screen (The UPDIR key (LS-VAR) is the same as the left cursor). The TREE key returns to the Filer's root dir. **HPDIRs on the SD card cannot be opened.**

SEARCHING

Press the alpha key twice - it will say "Search Mode On". Press one or more alphabet keys. To exit Search, press the alpha key again. (Flag -60 doesn't affect this.)

CASE SENSITIVITY

Files (varnames) made on the calculator (Home dir and subdirs, ports 0, 1, 2) are case sensitive. Files on the SD card (port 3) are not.

CHDIR

Select the desired dir with the up/down cursor keys and press CHDIR. The filer will exit to that dir.

FORMATTING

Newer ROM versions such as v2.09 have a formatting command for the SD card. Cards less than 32MB are formatted to FAT (actually FAT12 or FAT16). 32MB or larger are formatted to FAT32.

The fastest turn-on and file operations are with the smallest card that suits your needs - 8MB to 32MB recommended (although perhaps hard to find).

MULTIPLE FILES

Multiple files can be selected (or deselected) by pressing ENTER or the +/- key. Movement up or down will be the same as the last time the cursor was moved.

CURSORS

Besides the obvious up/down, the right cursor displays the contents of the selected dir or port; the left cursor returns to the previous screen.

MEMORY (lack of)

The Filer sometimes returns odd "Insufficient memory" errors when copying to the flash ROM. This is because the memory is in 128K blocks that need to be defragmented. Free up the required memory in the Home dirs for the defragger and try again. See the entry for **MEMORY basics** for more on this.

The following commands are available once a port or dir is selected:

VIEW:

Uses the SCROLL viewer for port objects. It sometimes fails for large objects, especially those on the SD card, but will return to the filer after a long pause. It might say "Insufficient Memory".

If flag 81 is clear, VIEW will display algebraics with the current font. If it's set, it uses the minifont.

To view the filenames in the SD card's DIRs, highlight the DIR and press the right cursor. The left cursor returns. HPDIRs cannot be opened.

EDIT:

This appears only in the Home dir and its subdirs (OS v2.09 or later). EDITB can be used for viewing objects in Port 2 flash or the SD card, but it will not save changes to these two ports. **EDIT/EDITB** may be very slow when run from the filer with large objects.

COPY:

Copies selected object(s) to any other dir or port. Asks for confirmation if the file already exists.

MOVE:

Copies the selected object(s); deletes from the source dir or port. Asks for confirmation if the file already exists.

RCL:

Copies highlighted object to the stack (so does the **RCL** key).

EVAL:

Copies highlighted object to the stack (so does the **EVAL** key). Programs will run.

TREE:

Displays the ports and dirs; *ie*, returns to the root display.

PURGE:

Deletes objects, dirs. If flag -76 is clear, asks for confirmation. Can do mass purging if files are tagged with the Enter key. Cannot purge SD card dirs made with a card reader - these are labelled DIR rather than HPDIR.

RENAME:

Opens an editor to allow renaming a dir or object - **Home dirs only**. It doesn't work in ports 1-3 (OS v2.09 or earlier).

NEW:

Allows copying an existing object to a new name, or creating a new, empty dir. With new dirs, leave the "Object" field empty.

ORDER:

Select any or all objects in a directory (not a port) with ENTER in the preferred display order - **ORDER** will then rearrange them. Changes to both Filer and the dir's var order remain after exiting - use **SORT** (see below) for temporary ordering.

SEND:

Transfers selected object(s) using Kermit and the settings from 104 MENU or APPS-I/O- Transfer. See also Transferring.

RECV:

Receives transfer using Kermit. Objects are stored in the current dir. See also **TRANSFERRING**.

HALT:

Returns to the stack. **CONT** returns to the Filer.

EDITB:

Uses the best editor for the selected object. It often fails for large objects and SD card files, but will return to the filer after a long pause.

HEADER:

Switches the header display from free memory to the name of the current dir or port. The MODE key does the same.

LIST:

Toggles the listing of object type and size on and off. The VAR key does the same.

SORT:

Opens a choose box for the type of sort. "INV." means descending order. The sort is only for the current session - ORDER can make changes permanent if in a dir.

XSEND:

Transfers selected object using Xmodem. Multiple objects can be sent if the receiver is in server mode.

CHDIR:

Changes to the selected dir and exits.

Access: LS-G,
 FILER (CAT or typed in)

FINANCE Solver

This computes basic loans, investments and mortgages using an **INFORM**-style window (for the softkey version, see **TVM**).

The variables are:

N - # of monthly payments
I%YR - interest in %/year (*ie*, 4.5% is 4.5, not .045)
PV - present value; principal
PMT - monthly payment
PYR - payments/yr (typ. 12)
FV - future value

PMTs are negative for withdrawals or pmts against a loan; a positive value indicates extra money being deposited per period.

By default the interest is calculated at the end of the period. To change this, use the Beg/End field or flag -14. It can also be set from the stack with 79 MENU-BEG.

(All values are stored in the current dir when you exit, and must be deleted manually.)

Example of short-term investment: \$18000 is invested for 5 years at 4.5% annual interest. Find the amount that can be withdrawn monthly to make the future value zero. Find the total gain.

Enter the values. If you need to do any calculating, highlight the desired field and press NXT-CALC. When you press OK, the Lev1 value is copied to the field. Press NXT again. (Note that CALC wants reals, not integers, but you can ignore the error message.)

N - 60 (*ie*, 5*12)
I%YR - 4.5
PV - 18000
PMT - ignore
PYR - 12 (the default)
FV - 0

Highlight PMT and press SOLVE to see the value -335.57 (this is tagged and placed on the stack - if you press +/-, the tag and minus sign will be removed).

The gain is $335.57 * 60 = 20134.20$.

Using **AMOR**: a \$150,000 mortgage is to be paid off in 25 years at 6%. Find the monthly pmt, and the status of the mortgage after 5 years.

Enter values as shown above (years should be in months) and solve for PMT - 966.45.

Status: Press AMOR and enter 60 into Payments. Press AMOR to get:

:Principal: -15101.87
:Interest: -42885.26
:Balance: 134898.13

Unfortunately, you can't enter dollar amounts into AMOR to find Payments, but it's easy to try various Payments to get near the desired value.

Access: LS-9

FINDALARM

Returns index of first alarm due after specified date.

```
mm.ddyyyy → n (flag -42 clear)
```

Access: TIME-TOOLS-ALRM

FINISH

Terminates remote Kermit server.

Access: 104 MENU-SRVR

FLAGS - list of flags

See also [RCLF](#), [STOF](#) and [Mode-Flag menu](#) entries.

Flag tools: LS+MODE-FLAG or PRG-MODES-FLAG

Flag Toggle

A program that toggles any flag on and off, where **-n is the flag number** (substitute the number before running the program):

```
%%HP: T(3)A(D)F(.);  
\<< -n FS? IF THEN -n CF ELSE  
-n SF END \>>
```

USER FLAGS:

User flags have positive numbers from 1 to 128. Note that flags 60 and 61 are reserved for system use in controlling constants (see the entry for [CONST](#)). User flags are only for user-set indicators and do not affect the system. They can be set, cleared and tested like system flags.

RESET:

MODE-NXT-RESET resets the flags to their default state (algebraic mode, etc). To reset the calc to your preference, store the flaglist from RCLF in a program such as 'MYFLAGS':

```
<< {flaglist} STOF >>
```

Since flag states are not stored by [ARCHIVE](#), this can be used after a [RESTORE](#).

PUSH/POP

In programming you can alter the flags without affecting the current settings - PUSH stores the flags and paths; POP restores them. See [Programming basics](#).

MAIN FLAGLIST:

Please note that HP information about flags above 64 is often wrong or missing. The definitions below are as accurate as circumstances allow.

All system flags are actually negative numbers. Positive numbers are user flags.

MASD in the flaglist refers to the built-in compiler, called ASM in 256 menu. It can compile either SysRPL code (flag 92 set) or machine code (flag 92 clear).

Note that some of the system flags do not appear in the MODE-FLAGS browser.

► Upper-case C (clear) or S (set) indicates the **default** setting. Note that the defaults may vary with the ROM version.

1s: Symbolics yield principal solutions

1C: Symbolics yield general solutions

General solutions return all possible results. In plots, use general solutions if only half a graphics plot appears.

2s: Symbolic constants eval to numbers

2C: Symbolic constants remain constants if flag -3 is clear

3s: Symbolic args eval to numbers

3C: Symbolic args remain symbolic

User-assigned keys can select Symbolic mode ({-2 -3} CF) or Numeric mode ({-2 -3} SF). See the entry for **Numerical/Symbolic** for a single-key flag toggle.

4: Unused

5-10: set binary wordsize from 1 to 64 bits. 5 is bit one (clear= 0, set=1), 10 is bit six. Add one to the result to get the wordsize (default 64, all flags set). See **STWS**.

11-12 BASE

11c,12c: DECimal base

11c,12s: BINary base

11s,12c: OCTal base

11S,12S: HEXadecimal base

13: unused

14s: TVM beginning of period

14C: TVM end of period

TVMEND will clear this flag; **TVMBEG** will set it.

15s: Spherical coord mode

15C: Cylindrical (polar) (if flag -16 set)

16s: Polar coordinate mode

16C: Rectangular coordinate mode

See also **RECT**, **CYLIN**.

17S: radians

17c,18C: degrees

17c,18s: grads

Same as **RAD**, **DEG**, and **GRAD** commands.

19s: $\rightarrow V2$ produces complex number

19C: $\rightarrow V2$ produces 2D vector

See **CYLIN** for an example of using $\rightarrow V2$. See also **Arrays/vectors**, **V \rightarrow , $\rightarrow V2$, $\rightarrow V3$**

20s: Underflow \rightarrow error

20C: Underflow \rightarrow 0 and sets flag -23 or -24

Underflow means that the calculation returned a negative non-zero result less than -MINR.

21s: Overflow \rightarrow error

21C: Overflow \rightarrow +/- **MAXR** and sets flag -25

Overflow means that the calculation returned a positive non-zero result greater than MAXR.

22s: Infinite result \rightarrow +/- MAXR and sets flag -26

22C: Infinite result \rightarrow error

23s: Negative underflow if flag -20 is clear

23C: No negative underflow

24s: Positive underflow if flag -20 is clear

24C: No positive underflow

25s: Overflow if flag 21 clear

25C: No overflow

26s: Infinite result if flag -22 is set

26C: No infinite result

The corresponding flag (-23 through -26) is set only if it is not treated as an error.

27S: Algebraic complex \rightarrow 'x+yi'

27c: Algebraic complex \rightarrow '(x,y)'

See the entry for **i**.

28C: Sequential plot of multiple equations

28s: Simultaneous plot of multiple equations

29s: No plot axes

29C: Draw plot axes

Same as the **DRAX** command.

30: unused

31s: No plot points connected

31C: Connect plot points

32s: Inverse graphics cursor

32C: Graphics cursor always black

33s: Transfer via IR port

33C: Transfer via wire

Leave cleared on HP49G; see flag -78 for HP50G USB.

34s: Print via wire

34C: Print via I/R

Leave set on HP49G; see flag -78
for HP50G USB.

35s: Kermit transmits in binary

35C: Kermit transmits in ASCII

36s: Received files overwrite

36C: Received files change name

Mainly affects Kermit commands.

37s: Double-spaced printing

37C: Single-spaced printing

38s: No linefeed added to print

38C: Linefeed added to print

39s: No I/O messages

39C: I/O messages

Mainly affects Kermit commands.

40s: Clock in header

40C: No clock

41s: 24hr clock

41C: 12hr clock

42s: DD.MM.YY date

42C: MM/DD/YY date

43s: Unacknowledged repeat alarms are not rescheduled

43C: Unacknowledged repeat alarms are rescheduled

44s: Acknowledged alarms retained

44C: Acknowledged alarms deleted

45-48: no. of dec. digits in FIX, SCI, ENG modes. 45 is bit 1, 46 is 2, 47 is 4, and 48 is 8. The binary nibble gives the number of decimal digits, max. 11.

49C,50C: STD mode

49s,50c: FIX

49c,50s: SCI

49s,50s: ENG

51s: Fraction mark is a comma

51C: Fraction mark is a period

52s: Obj displayed on single line

52C: Obj displayed on many lines

Flags -52 and -65 must be clear for textbook display (flag -79). With -52 set, textbook is off.

53s: Show all parentheses

53C: Remove extra parentheses

54s: Small matrix values not set to 0 and DET doesn't round

54C: Small matrix values set to 0 and DET rounds

55s: LASTARG disabled

55C: LASTARG enabled

56s: No beep tone

56C: Beep tone enabled

57s: No alarm tone

57C: Alarm tone enabled

58s: Param and var INFO disabled

58C: Param and var INFO enabled

eg, 106 MENU shows a **MSGBOX**; flag -58 set turns this off.

59s: Show varnames only

59C: Show varnames and contents

60s: One press locks Alpha

60C: Two presses lock Alpha

61s: One press locks User

61C: Two presses lock User

62s: User mode on

62C: User mode off

63s: Custom (vectored) ENTER

63C: Normal ENTER

64s: last GETI or PUTI wraps index to 1

64C: no wrap

65s: Only Level1 is multiline

65C: All levels are multiline

Flags -52 and -65 must be clear for textbook display (flag -79) of all levels - with -65 set only
Lev1 is textbook.

66s: Strings with linefeeds are displayed on a single line

66C: Strings with linefeeds are displayed on multiple lines

67s: Analog clock

67C: Digital clock

The analog clock is nearly unusable since it's only updated every 5 minutes.

68s: command line and editor indent same as previous line

68C: no indent

69s: Full-screen editing

69C: Single-line editing

70s: →GROB accepts multiline

70C: →GROB accepts single-line

71s: Labels, not addresses in ASM

71C: Add addresses to commands in ASM

72C: current font for stack numbers

72s: minifont for stack numbers

Flag -72 also affects the size of the stack level numbers.

For character font size, see flag 80. Flag -72 does not affect algebraics (such as 'X^2') but affects global names (such as 'X').

73s: Cmd line, Edit uses minifont

73C: Cmd line, Edit uses current font

74s: Left-justified stack

74C: Right-justified stack

75s: Keyclick if flag 56 clear

75C: No keyclick

76s: No Purge confirm in **FILES/FILER**

76C: Purge confirm in **FILES/FILER**

There is no confirm for the stack **PURGE** command.

77: internal use

78s: HP50G only - Serial I/O if flag -34 is set

78C: HP50G only - USB I/O if flag -34 is set

79s: Standard algebraic display

79C: Textbook algebraic display

Flag -52 must be clear for textbook display. Also, flag -65 must be clear for textbook display of all levels.

80C: current stk font

80s: minifont stk font

The flag browser calls it EQW Stk Font. It only controls the stack character font size - the EQW font is -82. Doesn't seem to affect global names ('X') or stack level numbers. See also flag -72.

81s: algebraic converted to grob using →GROB with n=0 uses the minifont

81C: as above, but uses current font

Flag -81 also controls whether the Filer's VIEW uses the current font (clear) or the minifont (set) for algebraics.

82s: EQW uses minifont

82C: EQW uses current font

The flag browser calls it EQW Edit Font. It only controls the main EQW font - the EQW edit function uses the current font.

83s: display grobs on stack as "Graphic *m* times *n*"

83C: display full grob

84s: MASD screen off

84C: MASD screen during compiling

When clear, may provide useful information during [ASM](#).

85s: SysRPL stack display

85C: normal stack display

Normal stack display shows SysRPL objects only as one or more "External".

86s: program prefix off

86C: program prefix on

For SysRPL purposes. This flag is said to auto-attach 256 menu when set and after an ON+C. It doesn't seem to be necessary.

87s: recursive stack for embedded composite pointers that are unsupported (not in Extable)

87C: non-recursive stack

For SysRPL purposes. Flag -85 should be set.

88 - not used

89 - not used

90S: [CHOOSE](#) boxes use minifont

90c: CHOOSE boxes use current font

91s: [Matrix Writer](#) uses list of lists

91C: Matrix Writer uses arrays

92s: MASD SysRPL mode

92C: MASD assembler mode

93 - not used

94s: Result of programs such as MTWR or EQW is not in LASTCMD

94C: Result of programs such as MTWR or EQW is in LASTCMD

LASTCMD is actually the [CMD](#) window (LS-HIST).

95S: Algebraic mode

95c: RPN mode

96: never fully implemented

97s: Lists are vertical

97C: Lists are horizontal

98s: Vectors are vertical

98C: Vectors are horizontal

99s: CAS verbose mode

99C: CAS quiet mode

Computation info appears in the header area

100s: Step by step on

100C: Step by step off

101: internal use

102s: no GCD calculations

102C: GCD calculations allowed

103s: Complex mode

103C: Real mode

LS+TOOL toggles this

104 internal use

105s: Approximate mode

105C: Exact mode

RS+ENTER toggles this

106s: TSIMP not allowed in SERIES

106C: TSIMP allowed in SERIES

107,108: internal use

109s: Numeric factorize

109C: Symbolic factorize

110s: Large matrices

110C: Normal matrices

111s: No recursive simplification in TSIMP, EXPAND

111C: Recursive simplification in TSIMP, EXPAND

112s: i is not simplified

112C: i is simplified to $\text{SQRT } -1$

113s: No linear simplify in CAS integration commands

113C: Linear simplify in CAS integration commands

114s: Polynomials in increasing power order

114C: Polynomials in decreasing power order

115s: Square roots are not simplified

115C: Square roots are simplified (as irrational square)

116s: Sine preferred to cosine in CAS simplifying

116C: Cosine preferred to sine in CAS simplifying

117s: Softkey menus (using function keys)

117C: CHOOSE list menus

118s: INT command not simplified

118C: INT command simplified

This flag has no apparent effect on **INT**. Both **INTVX** and **RISCH** clear it if it's set.

119s: Rigorous mode off ($|X|$ is simplified to X)

119C: Rigorous mode on ($|X|$ is not simplified to X)

This particularly affects **ABS** and the way the **SIGN** command will plot. 119 should be clear for general use.

120c: Silent mode off - prompts before mode changes

120S: Silent mode on - no prompts before mode changes

121,122: internal use

123s: No mode switching

123C: Allow mode switching

124s: No non-algebraic CASCOMPEVAL

124C: Non-algebraic CASCOMPEVAL allowed

There is no information available as to what this flag is for. If accidentally set, it returns "Non algebraic in expression" when the CAS tries to solve. **CASCFG** clears it.

125s: Fast sign determination. Also cancels SQRT simplification

125C: Accurate sign determination using Sturm sequences

126s: Row reduction (rrf) done without last column

126C: Row reduction (rrf) done with last column

127: unused

128s: all vars are reals

128C: Complex vars allowed

User (positive) flags in use:

60s: CONST uses English units

60C: CONST uses SI units

61s: no CONST units

61C: CONST returned with units

FLASHEVAL

Given a flash ROM address as a hex integer, runs built-in commands. Example:

```
#47002h → I/O transfer window
```

Since a wrong integer can corrupt memory, doublecheck before using.

FLASHEVALs are not addresses of commands (as are [SYSEVALs](#)), but bank and function numbers. (The Getname command of SDIAG won't find them.)

Access: CAT

FLOOR

Nearest whole number \leq the argument. See also [CEIL](#).

```
5.1 → 5.  
'A' → FLOOR(A)
```

Flags 2 and 3 should be clear for symbolic use.

Access: MTH-REAL-NXT-NXT

FONT series

FONT→ returns current system font.

```
→ font_obj
```

→**FONT** sets a new font as the current system font.

```
font_obj →
```

Fonts can also be changed with the [MODE-DISP](#) window.

Fonts other than the minifont can be also be edited with RS-CHARS-MODIF (if other than the default Font8, install first with MODE- DISP-FONT). See also [CHARS](#).

FONTn returns any one of the system fonts 6,7,8 (except the **MINIFONT** - see its entry). Put 6, 7, or 8 in place of n.

```
FONTn → font_obj
```

Font changes are reset to the default Font8 by warmstarts. To make a change permanent, recall the new font to the stack with FONT→ and store it with a unique name in the Homedir. Add:

```
varname →FONT
```

to the program in the var **STARTUP** in the Home dir, or if it doesn't exist, enter:

```
<< varname →FONT >> 'STARTUP' STO →
```

To undo this, edit or delete the STARTUP var.

Access: CAT,
MODE-DISP

FOR-NEXT

For-next and for-step definite loop. See also **START**.

```
start end FOR x <procedure> NEXT
```

OR

```
start end FOR x <procedure> stepsize STEP
```

x is the index var (which can be any other var) and counts from start to end - it can be recalled for use by the program. If this isn't needed, START is somewhat faster.

The numerical value for STEP can be replaced with the name of a stored value, providing external control.

Typing aids: when the FOR dir is displayed (LS-PRG-BRCH), these are available:

```
LS-FOR → FOR  
NEXT
```

```
RS-FOR → FOR  
STEP
```

The following example displays multiples of $\pi/2$ (ie, 90 degrees apart in a plot). For symbolic results, clear flags -2 and -3; for numeric, set both.

```
<< .5 2 FOR X X  $\pi$  * .5 STEP >>
```

Controlling termination: the loop can be stopped when a condition is met, as shown by this example:

```
%%HP: T(3)A(D)F(.);
\<< 10 → x loop
  \<< 1 x
    FOR X
      IF X loop >
        THEN 0 'x' STO
        ELSE X
        END
      NEXT
  \>>
\>>
```

$n \rightarrow n_numerals$ where $n = 1$ to 10

The program puts 1 to 10 numbers on the stack. Enter a number and run the program. That number of numerals will appear. Your value is stored in the 'loop' local var. The program will continue to output X until your number is reached, and then x (the end value) is set to zero to end the loop.

See also [LOCAL](#) and [Programming basics](#) for more on using variables.

Access: PRG-BRCH

FOURIER

Nth coefficient of complex Fourier expression. The period T of a T-periodic function should be stored in the variable PERIOD in [CASDIR](#) (the default is 2π).

Symb1 n → coeff

Access: CALC-DERIV

FP

Returns fractional part of real number. See also [IP](#).

$5.1 \rightarrow .1$

Access: MTH-REAL-NXT

FREE

An obsolete port-management function from the HP48 command set. Not used on the HP49/50.

FREEZE

Freezes an area until a keypress.

`n →`

- 1** freezes lines 1 and 2 of the header (status area)
- 2** freezes the stack area and command line
- 4** freezes the menu area

n can be the sum of any of the above numbers. 0 or 7 will freeze all.

The actions of FREEZE depend on the application - some experimenting is required.

Example with **DISP**:

```
<< "string" 2 DISP 3 FREEZE >>
```

Example with **PICT** and a menu:

```
%%HP: T(3)A(D)F(.);
\<< { list of vars } TMENU { # 0h # 0h } PVIEW 3 FREEZE \>>
```

PICT should contain a plot or graphic for best results.

To display for a fixed number of seconds, replace `x FREEZE` with `s WAIT`, where *s* = seconds.

See **STARTERR** for a WAIT example.

Access: PRG-NXT-OUT

FUNCTION

Sets the plot type to Function and if necessary, creates **PPAR** (the list of plot parameters).

This is also the default if no plot type is specified. **EQ** should contain an expression that's a function of one variable, default X. Other variables can be included if they have stored values (constants). It then plots $Y=f(X)$.

See the entry for **Plots** for more on $f(X)$ and object types.

A simple example that plots a parabola:

```
%%HP: T(3)A(D)F(.);
\<< 'X^2' STEQ FUNCTION ERASE
DRAX DRAW PICTURE \>>
```

If a previous plot has set the wrong parameters, either purge PPAR from the stack or press ZOOM-ZDFLT from the graphics menu.

Access: LS+2D/3D-Type,
82 MENU

F0λ

Returns the fraction of total black-body emissive power at temp Xt between wavelengths 0 and Yλ.

$Y\lambda \text{ } X_t \rightarrow X_{\text{power}}$

Access: 117 MENU,
APPS-EqnLib-UTILS

Garbage Collection

The operating system must pause from time to time to clear unused objects from memory. This will suspend whatever you're doing for a few seconds. The GC improves with each OS upgrade and is no longer much of a problem. See also [Pauses](#).

If necessary, a GC can be forced with

5F42h SYSEVAL (also works on the HP48)

[MEM DROP](#) in a program will also force a GC.

GET/GETI

Returns element n from array, matrix ({row col}) or list. GETI also returns index of next element and does not drop original object.

obj n → element (GET)

obj n → obj index element (GETI)

{ 22 33 44 55 } 3 GET → 44

{ 22 33 44 55 } 3 GETI → { 22 33 44 55 } 4. 44

For strings, see [SREPL](#).

Access: PRG-LIST-ELEM

gmol

A typing aid for user-defined units.

→ 1_mol
55_ → 55_gmol

Access: 117.02 MENU,
APPS-EqnLib-UTILS

GOR

Superimposes one grob on another with no changes to target grob. Opposite **GXOR**, which inverts coincident pixels.

targetgrob {#m #n} grobl → newgrob

OR

targetgrob (x,y) grobl → newgrob

{#m #n} (or coords (x,y)) is the upper left corner of grobl.

Access: PRG-NXT-GROB

GRAD

Sets angle measure to grads - the grad is 1/100 of a right angle. 360 degrees is 400 grads. One degree is approx. 1.1111 grad.

Access: MODE-Angle Measure

GRAPH

Plots the expression(s) in variables Y1, **EQ** according to the settings in **PPAR**. Y1 need not exist (see **Y=**). The axes can be toggled on or off with the softkey in LS+**2D/3D**.

Creates the variable Y1 as required. Similar to **DRAX DRAW** (axes will be drawn only if they're selected in 2D/3D).

Access: LS+F3

Graphics menu

Pictures, plots, etc. are called grobs (graphic objects). See also **PICT**, **Plots**. Enter the graphics editor with the left cursor. If a grob is on the stack, you can edit it by pressing the down cursor (**EDITB**).

Note that not all plot types have all the following menu items.

Shortcut Keys:

*	(times key) places a mark at the cursor
-	toggles menu off and on
+	toggles coordinate display
ENTER	places cursor position coordinates on the stack
ON	exits (left-cursor returns)
RS-cursors	jump to screen edge
LS-down	displays current eqn
LS-left	toggles menu/cursor
RS-CLEAR	erases screen

Any function key restores the softkey menu

ZOOM Menu:

► Note that not all plot types have the zoom function.

ZFACT

Sets V and H zoom factors (4 is the default, and 2 works well).
Also allows centering of the display at the cursor after a zoom.

One way to restore the normal display is to remove the checkmark from ZFACT and press ZDFLT.

BOXZ

Draws zoom box starting with a corner at the cursor position - choose this corner first with the cursor keys, press BOXZ, choose the other corner, and then press ZOOM

ZIN/ZOUT

Zooms in or out by factors in ZFACT

ZSQR

Redraws with equal V and H tick values (undoes a V or H zoom)

ZDFLT

Redraws with PPAR reset to default values

HZIN/HZOUT

Zoom the horizontal axis in or out by the H-factor in ZFACT

VZIN/VZOUT

Zoom the vertical axis in or out by the V-factor in ZFACT

CENTR

Redraws to place the plot center at the cursor position. The origin remains (0,0)

ZAUTO

Zooms and relocates the origin to display the largest image of the current equation

ZDECI

Redraws with the default X-tick value of 1 (0.1 per pixel)

ZINT

Redraws with the X-tick value of 1 per pixel (10 per tick)

ZTRIG

Redraws with the X-tick value of $\pi/2$ per tick

ZLAST

Redraws with the values of the previous image (stored in **ZPAR** and **PPAR**)

(X,Y)

Displays cursor coordinates at the bottom of the screen. Any function key (or the - key) restores the menu

TRACE

L-R cursors move along function; Up-Dn cursors select other functions, if any

FCN menu: (Function plots only)

Calculated results are also tagged and placed on the stack

ROOT

Jumps to and displays value of root of current equation; for multiple roots, jumps to root closest to cursor

ISECT

Jumps to X intercept closest to cursor

SLOPE

Returns the slope of the current function at the cursor

AREA

Finds the area under the function between the mark and the cursor. Also writes IERR to the current dir

SHADE

Shades the area under the function between the mark and the cursor

EXTR

Jumps to and displays value of extremum nearest the cursor

F(X)

Finds value of current function for X value of cursor

F'

Finds and plots the derivative of the current function and appends it to EQ

TANL

Plots tangent line at the cursor position on the function and displays its equation. If you get an error message, the cursor probably wasn't exactly on the function - move the cursor to the desired X value and use TRACE and up-arrow to move it to the plotted line

NXEQ

Displays next equation for multiple equations in EQ

VIEW

Displays current equation in upper left

EDIT menu:

DOT+/-

Turns pixels on or off under the cursor when a white square is showing in the softkey label

LINE

Draws line from mark to cursor

TLINE

As above, but turns off any coincident pixels

BOX

Draws box with opposite corners at mark and cursor

CIRCLE

Draws circle CCW with center at mark and cursor at end of radius

MARK

Sets mark at cursor, same as the * (times) key

+/-

Inverse cursor when white square shows in softkey label

LABEL

Labels axes with default values from **PPAR** - see **AXES** and **LABEL** for user labels

DEL

Deletes area in imaginary box with opposite corners at mark and cursor

ERASE

Erases everything - and there's no Undo. If you press ZOOM-ZLAST it will redraw EQ, but any edits you made will be lost - see PICT→ below for saving plots

MENU

Turns menu off/on, the same as the - key; any function key also restores it

SUB

Writes a subgrob to the stack, bordered by an imaginary box with the mark and cursor at opposite corners

REPL

Inserts a subgrob from the stack, with the upper left corner at the cursor

PICT→

Writes a copy of the entire screen to the stack as a grob (without the softkey menu - if a menu appears, it's usually from **EDITB**; try **SCROLL** or VIEW in **FILES/FILER**)

X,Y→

Writes cursor coordinates to the stack in the form (x,y), same as the Enter key

GRIDMAP

Sets the plot type to Gridmap and modifies or creates **PPAR.EQ** should contain a complex expression in two variables. For example, '(X,Y)' draws a 7 x 9 linear grid with the default settings.

See also **Plots** for more on general plotting.

The following plots a curved gridmap with default settings.

```
%%HP: T(3)A(D)F(.);  
\<<RAD { PICT PPAR VPAR } PURGE  
GRIDMAP 'SIN((X,Y))' STEQ DRAW  
PICTURE \>>
```

Use LS+WIN to redraw with new values; *eg*, to decrease the image size slightly, use

```
X-Left -1.5  
X-Right 1.5  
Y-Near -1.5  
Y-Far 1.5
```

Detail increases with higher settings of Step Indep and Depnd, but drawing time is longer.

A much better discussion of plotting than in the HP49 manual is in the HP48 Series User's Guide, available for free downloading by searching www.hp48.org.

Access: LS+2D/3D-Type
85 MENU

→GROB

Converts obj (usually a string or algebraic) to grob (graphic object), where n = 1 (minifont) or 2 (current font).

obj n → grob

Algebraics

If n = 0, algebraics will become grobs using the current font if flag 81 is clear, or the minifont if it's set. HP49/50 manual documentation on flag 81 can be disregarded.

Grob to PICT

To insert the grob into **PICT** in stack mode, see **GOR** or **GXOR**. To insert in graphics mode: the upper left corner of the grob on Lev1 appears at the cursor position when you press EDIT-NXT- NXT-REPL.

Adding Text to Grob

For axis labels, see **LABEL**. To add any text to any point on a graphic in PICT, exit the graphic mode to the stack. Place the text string on the stack, enter 1 for the minifont or 2 for the current font and run **→GROB**.

Press the left cursor to return to the graphic mode. Move the cursor to where you want the upper left of the string and press EDIT-NXT-NXT-REPL. It's an imprecise method and some experimenting is required.

Access: PRG-NXT-GROB

GROBADD

Merges 2 grobs (stacked vertically - two 131 x 64 = 131 x 128). Grob1 is at the top.

grob1 grob2 → grob3

Grob1 will be at the top of Grob3.

Access: CALC-GRAPH

GXOR

Superimposes one grob on another, inverting coincident pixels. Opposite **GOR** (no changes to target grob).

targetgrob {#m #n} grob1 → newgrob

OR

targetgrob (x,y) grob1 → newgrob

{#m #n} (or coords (x,y)) is the upper left corner of grob1.

Access: PRG-NXT-GROB

HALT

Suspends operation when encountered in a program. Suspends editing session and displays stack if called from Tool menu. Suspends **DEBUG**.

HALT after an **IFERR** allows Debug to single-step through the program. Press SST to continue past the halt.

CONT cancels the halt.

See also **LOCAL** for another usage.

Access: PGM-NXT-NXT-RUN,
Edit mode TOOL-NXT-NXT

HEAD

Returns first element of a string or list. See also **TAIL**.

```
"abc"  → "a"  
{ a b c } → 'a'
```

To return any character(s) or element(s), see **SUB**.

Access: PRG-NXT-CHARS-NXT,
PRG-LIST-ELEM-NXT,
RS+CHARS-NXT

HEADER series

On the HP49G+/50G, the header is a separate screen area, although →HEADER still controls the number of lines displayed.

HEADER→ returns size of header, where n is the number of lines (0-2).

→ n

→**HEADER** sets the number of lines displayed (0-2).

n →

Access: CAT,
MODE-DISP

Helptext (in pgms)

Help text can be inserted into programs in various ways:

- 1. Comments:** although sourcecode comments using @ are removed, you can insert text strings followed by DROP.
- 2. IFERR:** use the error loop to put a reminder string on the stack (if the program errors without arguments).

```
<< IFERR < program > THEN "Helptext" END >>
```

- 3. Doubleclick:** the following runs a program normally on one click, but puts helptext on the stack if the name is doubleclicked.

```
%%HP: T(3)A(D)F(.);
\<< .3 WAIT KEY
  \<< DROP "Helptext"
  \>>
  \<< "main program"
  \>> IFTE
\>>
```

Some experimenting might be required with the .3 value depending on the calculator or emulator speed. It works well on the HP49G+/50G.

Hidden dir

The hidden directory in the Home dir is used by the operating system to store the key assignments and alarms. It's also used by programs to store counters, data, etc. It has a nullname (") and since it isn't in the path, its variables can't be recalled from other dirs.

To see it, type

```
# 3714Ah SYSEVAL      (49/50 series only!)
```

Use caution - the wrong **SYSEVAL** can corrupt memory. Don't change any system vars - you can add or delete other vars as in any dir.

Another way is to use a nullname:

```
"" S~N → '' EVAL
```

See **256 MENU** for the S~N (symbolic-name toggle) command.

Since the hidden dir is not in the path, programs in other dirs will not run.

To leave it, press UPDIR.

HIST

Allows returning to the stack from within a program to retrieve any object from any level. Select the desired object and Echo inserts a copy at the cursor.

Use ON or Enter to leave HIST mode if Echo doesn't cancel it.

In stack mode, it will echo any level to the command line, and provides a number of stack commands, such as Edit, Roll and Keep. (See the entry for [Stack Commands - Interactive Stack](#).)

You can also copy to the clipboard. Cursor to the object and press RS-COPY.

The stack is not available for calculating in HIST. **In edit mode** use the TOOL menu's HALT for this, or CALC if it appears in a function's softkey menu.

Access: HIST key

HISTOGRAM

Sets the plot type to Histogram and modifies or creates [PPAR](#), [ΣPAR](#). [ΣDAT](#) should contain a one-column matrix. It displays the number of occurrences vertically and the magnitude according to horizontal position. See also [Plots](#) for more on general plotting.

The following plots a sequence of random numbers between 1 and 10:

Store as ΣDAT (you can use **STOΣ** from CAT):

```
[[ 4.5 ]  
[ 2.35 ]  
[ 2.78 ]  
[ 6.61 ]  
[ 1.73 ]  
[ 7.89 ]  
[ 9.01 ]  
[ 8.05 ]  
[ 1.86 ]  
[ .68 ]]
```

Press LS+2D/3D and select Histogram as the Type.

Press LS+WIN and press Auto, Erase, and Draw.

A much better discussion of plotting than in the HP49 manual is in the HP48 Series User's Guide, available for free downloading by searching www.hp48.org.

Access: LS+2D/3D-Type,
88 MENU

HISTPLOT

A CAT command that executes a **HISTOGRAM** plot. The variable **ΣDAT** must exist in the current dir. The plot does not remain on the screen - press **PICTURE** (left cursor) to return to it.

HMS series

See also for more on working with degrees.

→**HMS:** decimal hours/degrees to hh.mmss format.
HMS→: hh.mmss format to decimal.
HMS+: adds 2 hh.mmss.
HMS-: subtracts 2 hh.mmss. The earlier time should be on Lev 1 to avoid negatives.

Access: RS-TIME-TOOLS-NXT,
 RS+TIME-NXT,
 LS-PRG-NXT-NXT-TIME-NXT

i

Returns the square root of -1.

→ i (symbolic mode)
 → (0.,1.) (numeric mode)

Flags -2,-3 set is numeric mode; clear is symbolic. Complex mode should be on (flag -103 set).

To make symbolics into a complex number in programming (flag -27 clear):

'symb1' 'symb2' i * + → (symb1,symb2)
 'X^2' 'Y' i * + → (X^2,Y)

Access: LS-TOOL,
 CMPLX

IDN

Creates identity matrix n by n.

n →
 3 → $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Access: MATRICES-CREAT

IF-THEN

Starts If-Then or If-Then-Else conditionals.

```
IF <test> THEN <procedure> END
```

OR

```
IF <test> THEN <proc1> ELSE <proc2> END
```

If test is non-zero, procedure or proc1 is evaluated. If test is 0 and the ELSE clause is present, proc2 is evaluated. The result of the test is always dropped.

The test can precede the IF command:

```
<test> IF THEN <proc> END
```

As long as there's a non-zero number on Lev1, THEN will run the procedure (IF is really only included for readability).

Typing aids: when the IF dir is displayed (LS-PRG-BRCH), these are available:

```
LS-IF  → IF
        THEN
        END
```

```
RS-IF  → IF
        THEN
        ELSE
        END
```

The example below finds the number of items on the stack and displays it if ≥ 1 , or if 0 it displays "Empty stack".

```
%%HP: T(3)A(D)F(.);
\<< IF DEPTH DUP 0 \=/
  THEN R\>I \>STR " Item(s)" +
  ELSE DROP "Empty stack"
  END \>>
```

See also **AND** and **OR** operators.

See also **IFT** and **IFTE** for more compact forms, and **FOR-NEXT** for an IF-THEN example.

Access: PRG-BRCH

IFERR

An error in a program (or section of program) enclosed by IFERR runs procedure; if no error and ELSE is present, runs proc2.

```
IFERR <program> THEN <procedure> END
```

OR

```
IFERR <program> THEN <procedure> ELSE <proc2> END
```

Since flag -55 (Save Last Args) affects the return of objects to the stack, it should be clear when using IFERR.

The error procedure can be as simple or as complex as desired. The minimum is IFERR <pgm> THEN END, which suppresses error messages.

IFERR is useful for clean exits; *ie*, removing unwanted objects from the stack after a cancel.

```
IFERR "" "" INPUT THEN DROP2 END
```

The above converts any input to a string for further processing. If cancelled, the two null strings are dropped.

See also [Helptext \(in pgms\)](#) for adding usage reminders to a program.

IFERRs can be nested:

```
IFERR <pgm section 1> IFERR <pgm section 2> THEN <proc2> END  
THEN <procl> END
```

When single-stepping a program, DEBUG executes IFERRs as a single step. To avoid this, follow each IFERR with HALT. Remove when done.

Error-handling can be user-invoked - see [DOERR](#).

Access: PGM-NXT-NXT-ERROR-IFERR

IFFT

1- or 2-dim. inverse discrete FFT of an array. Number of terms must be a power of 2.

```
[array1] → [array2]
```

Access: MTH-NXT-FFT

IFT

A compact IF-THEN branching test. Evaluates obj if test is non-zero; otherwise drops both.

```
test  obj  → result
```

```
<< MEM 50000 ≥ "OK" IFT >> EVAL
```

The simple example checks to see if there is at least 50000 bytes of free memory. If so, the string is put on the stack. If not, there is no output. See also **NOT** for another use.

There is no algebraic syntax - see IFTE below.

Access: PRG-BRCH-NXT

IFTE

A compact IF-THEN-ELSE branching test. Evaluates obj1 if test is non-zero, else evaluates obj2.

```
test obj1 obj2 → result
```

Algebraic syntax is:

```
' IFTE (test, obj1, obj2) '
```

```
' IFTE (X>0, X^2, 1) '
```

The example reads "If X is greater than zero, then return X^2, else return one." (Plot it as a Function to see the effect.)

The example from IFT expanded:

```
<< MEM 50000 ≥ "OK" "<50K memory" IFTE >>
```

If mem is below 50000, the second string is put on the stack.

See the end of the **Helptext (in pgms)** for another example.

Access: PRG-BRCH-NXT

IM

Returns imaginary part of complex number. For the real part, see **RE**.

```
(1., 2.) → 2.
```

Access: CMPLX,
MTH-NXT-CMPLX

INCR (Increment)

Adds 1 to contents of named local or global var; returns new contents. The var should contain a real or integer. See also **DECR (Decrement)**.

```
'name' → newval
```

Access: PRG-MEM-ARITH

INDEP

Specifies the independent var for plotting and/or its plot range. See also **PPAR**, **XRNG**, **WIN**.

```
'name' →
{n m} →
{'name' n m} →
{ 'X' -10 10 } →
```

The LS+2D/3D field can only change the name. The others are stack commands, although you can set the range with LS+WIN's H-View.

Access: LS+2D/3D,
81 MENU-PPAR

Infinity (∞)

Returns the $+\infty$ infinity symbol in symbolic mode (flag -3 clear and Exact mode selected) or 9.999999999999999E499 in numeric mode (flag -3 set). Set to $-\infty$ with the +/- key (or **NEG** in program). See also **MAXR**. The infinity symbol in the minifont is somewhat strange due to the low res.

Access: LS-0

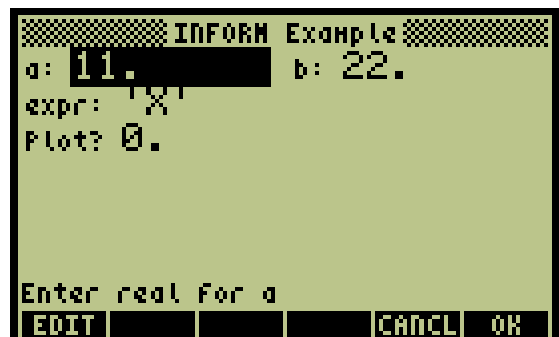
INFORM

Inform dialog box.

```
"maintitle" (" " for none)

{ {"fieldtitle1" "helptext"
TYPES} ...{ etc } }
{format}
{reset values - can be empty}
{default values - can be empty}

→ {list of values} 1.
```



Returns 0. if cancelled.

Format is { cols tabs }. Columns is the number of fields across. Tabs controls the spacing between the name and the field, and can be omitted. If you have only one column, the list can be empty.

Note that INFORM allows only four rows in UserRPL.

In reset and default lists, use NOVALs if there are no values and you don't want to leave them empty - INFORM will often put NOVALS on the stack from empty lists.

Type numbers allow only desired objects to be entered - see the entry for **TYPE**. Typical use:

```
{"fieldtitle" "help" 2 5 }
```

The field now accepts only strings and lists.

Note that there are differences from the HP48. Some Types may behave unexpectedly (eg, Type 28 (integer) is not accepted in the list, although the fields themselves accept integers).

To increase the width of a field, put an empty list after it:

```
{"title" "help" TYPES} {}
```

If you do this, there must be at least two columns in the Format list.

To reset any or all of Inform's field values to those in the Reset list, press NXT, RESET.

Note that the checkmark feature is not available in UsrRPL. It requires a SysRPL system program.

EXAMPLE INFORM

The following shows the basic features. Note that expression's field is full-width and allows both algebraics and names. The output is the four-element list {11. 22. X 0.} and 1., or 0. if cancelled.

```
%%HP: T(3)A(D)F(.);
\<< "INFORM Example" { { "a:" "Enter real for a" 0 } { "b:" "Enter real for b" 0 } {"expr:" "Enter
algebraic" 6 9 } } { { "Plot?" "Draw Graph? 1=Yes 0=No" 0 } } {2. 2. } { 11. 22. X 0. } { 11. 22. X 0. }
INFORM
\>>
```

LOOPING INFORM

The program below opens with the vars that were last used. It does this by making an external var (called LST1) in the current dir. It contains a four-element list. The INFORM section is in a list so the **PUT** command can overwrite the default values list with any existing values from LST1.

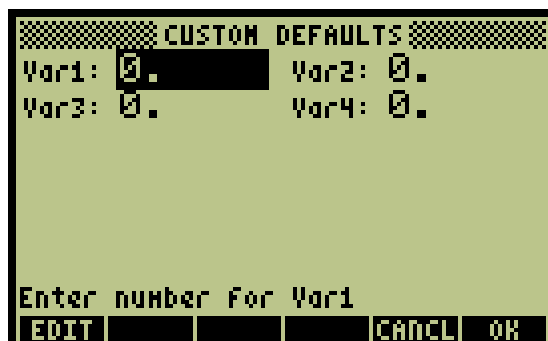
The actual calculating does nothing but add the inputs. Enter any numbers and press OK. These will be displayed the next time you run the program.

You can reset the values to zero by deleting LST1, or by pressing NXT-RESET-Reset_all from inside the program.

```

%%HP: T(3)A(D)F(.);
\<<
  IFERR -55. CF IFERR 'LST1' RCL THEN { 0. 0. 0. 0. }
DUP 3. ROLL SWAP STO
  END
  { "CUSTOM DEFAULTS" { {"Var1:" "Enter number for Var1" 0. }
{"Var2:" "Enter number for Var2" 0. }
{"Var3:" "Enter number for Var3" 0. }
{"Var4:" "Enter number for Var4" 0. } }
{ 2. 1. } { 0. 0. 0. 0. } { 0. 0. 0. 0. } INFORM }
SWAP 5. SWAP PUT EVAL
  IF
  THEN DUP 'LST1' STO EVAL \->
var1 var2 var3 var4 'var1+var2+var3+var4'
"Sum" \->TAG
  END
  THEN
  END
\>>

```



Access: PRG-NXT-IN

INPUT

Allows user input during program execution; up to four lines of objects can be entered (HP49G) or up to seven (49G+/50G) if the prompt is omitted.

Multiple objects can be entered on each line by putting spaces between them.

Tags and default values in the main string are optional. The output is a string of objects for further processing.

Minimum syntax is two null (empty) strings:

```
"" "" INPUT → "datastring"
```

If cancelled with ON, the output is the two null strings. See **IFERR** for adding the DROP command.

Full syntax:

```

<< "Prompt" { :Tag: "Defaultvalues"
  α ALG V {0 0} }
INPUT >>

```

→ objects inside a string

The following are options and can be omitted (except Prompt):

Prompt: any string for the user. If none, use a null string ("").

Defaultvalues (a string): can be tagged objects, numbers, etc, or the strings and/or tags can be omitted. There can be up to 4 lines for the 49G, and up to 6/7 for the 49G+/50G.

α: if present, turns on alpha mode (α char is alpha-RS-A).

ALG: Algebraic entrytype (see [ENTRY](#) for more detail).

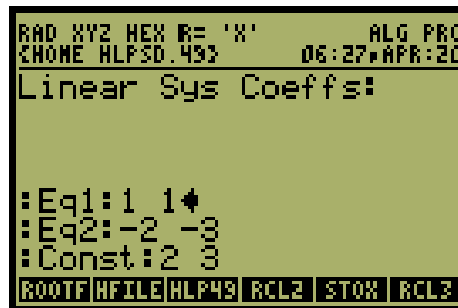
V: verify UserRPL syntax in the main string. If this is absent, it can contain anything.

{ 0 0 } is {cursor line,cursor position}. {0 0} or omitting the list puts the cursor at the end of the last line.

The example below demonstrates an Input program that finds the roots of a linear system when you press Enter. Values can be changed if you like. Press ON twice to cancel.

```
%%HP: T(3)A(D)F(.);
```

```
\<< IFERR "Linear Sys Coeffs:" {
:Eq1:1 1
:Eq2:-2 -3
:Const:2 3"
ALG V { 1. 9. } } INPUT OBJ\-> 6.
\->LIST 1. * EVAL 2. \->ARRAY 5.
ROLLD { 2. 2. } \->ARRAY / "Roots"
\->TAG THEN DROP2
END \>>
```



Access: PRG-NXT-IN

INT

1. Returns antiderivative of expression for var at n. n can be real or symbolic. It also creates the error value variable IERR.

```
expr var n → integral
```

```
'X^2' 'X' 2 → 2.6666...
```

```
'X^2' 'X' 'X' → '1/3*X^3'
```

Flag -3 should be clear (symbolic mode). Note that as of ROM v2.09, INT can produce odd results for definite integrals - it also lacks auto mode switching. An example:

Set Exact mode, flag 3 clear

```
'SIN(X)' 'X' 3.14159265359 → 1.
```

However, INT returns 2. in Approx mode. It isn't clear whether the integral is measured at a point or over the interval from zero. It seems to do both depending on the settings.

► INT seems to be an add-on command that was never fully implemented. If in doubt, use the full integral command (RS-TAN), RISCH, or INTVX.

Access: CAT

2. If an integration command returns a result that looks like 'INT(algebraic)', it means that it cannot find the closed form of the algebraic.

Integers

The 49/50 series has three basic types of numbers: reals, integers and user binary integers. (See **Binary Integers** or **BASE** for more on binaries.)

Integers have no decimal point (unless you enter a decimal fraction, which makes it a real).

A real always has a decimal point: 5., 5.123, etc.

The calc should be in **EXACT mode** for entering integers. (Exact's annunciator in the header is = and Approximate's is ~ - RS+ENTER toggles this.)

In Exact and symbolic modes (flags -2,-3 clear), integer operations such as division return rational fractions. In numeric mode, they return decimal fractions.

In general, reals and integers can be mixed without problems. To convert reals to integers or vice versa, use **R→I** and **I→R** from LS-CONVERT-REWRITE. Note that reals that have decimal fractions (such as 5.123) cannot be converted to integers.

You can also convert an integer to a real by multiplying it by 1. or other real number.

However, there are some integer manipulations that will unexpectedly convert integers to reals. If this happens, the easiest fix is to run the XQ command (from CAT) on the results. This converts reals back to integers.

To use integer CAS menu commands, enter the INTEGER command (typed in or CAT).

Access: MODE-CAS-Approx,
RS+ENTER

Integral (I)

Finds definite or indefinite integral of algebraic expression.

```
lower upper expr var → integral
```

For definite integrals, it also creates the error value variable IERR in numerical mode (flag -3 set - see below).

For indefinite integrals, the upper limit should be the same variable as Lev 1.

eg: 0 'X' 'X^2' 'X' → 'X^3/3'

If the upper limit is different, the result is:

```
0 't' 'X^2' 'X' → 't^3/3'
```

Note that it's up to the user to insert any arbitrary constants.

The ASCII form is '∫ (0,X,X^2,X)' (lower, upper, integrand, variable of integration)).

If it returns a result in the form INT(expression), it means that it cannot find the closed form of the integral.

Note that →**NUM** will create IERR if used to evaluate an integral, either in ASCII or **EQW** format.

The accuracy can be controlled by the number format (STD, FIX, etc): more digits give better accuracy but longer times, and vice versa. Some experimentation is required.

See also INTVX, INT, and RISCH in CAT, and **EQW**-INTEGRALS.

See also the DESOLVE section of the entry for **Derivative (D)**. DESOLVE integrates both sides of its eqn to solve ODEs. It has the added advantage of inserting the arbitrary constant.

Access: RS-TAN

INV

Inverse of object; program version of 1/X.

```
n → 1/n
'name' → 1/'name'
[[sq matrix]] → 1/[[matrix]]
'INV(X)' EVAL → '1/X'
<< 5 INV >> EVAL → .2
```

The inverse of a 2x2 matrix is:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \rightarrow \begin{bmatrix} D/W & -B/W \\ -C/W & A/W \end{bmatrix}$$

where $W = (D*A - C*B)$

Access: 1/X

IOPAR

A variable automatically created in the Home dir with parameters for I/O transfers. (Note that the HP49G+/50G USB ignores IOPAR.)

The HP49G default is: { 9600 0. 0. 0. 3. 1. }

The HP49G+/50G default is: { 115200. 0. 0. 0. 3. 1. }

Parameters are baud, parity, Recv pacing, Xmit pacing, Kermit checksum, Kermit/printing translation. There are no controls for pacing (Xon/Xoff flow control) and the feature is not implemented on the HP49. Better translation is obtained with:

{ 9600 0. 0. 0. 3. 3. } (9600 is 115200 on the 49G+.)

Parameters such as baud, parity, checksum (**CKSM**) and translate level (**TRANSIO/XLAT**) are in 104 MENU-IOPAR or APPS-I/O-Transfer. See also **TRANSFERRING**.

IP

Returns integer part of real number. See also **FP**.

5.1 \rightarrow 5.

Access: MTH-REAL-NXT

I \rightarrow R

Integer to real number conversion. Its inverse is **R \rightarrow I**.

5 \rightarrow 5.

The algebraic expression version is 'I \rightarrow R (X) '.

'X' \rightarrow 'I \rightarrow R (X) '

You can also convert integers to reals by multiplying by 1. or other real number.

Access: CONVERT-REWRITE

ISOL

Arranges equation to isolate first occurrence of var.

eq1 var → eq2

'X^2+2*X+Y^2=25' 'X' → { 'X=-(1+\sqrt{-(Y^2-26)})'
'X=-(1-\sqrt{-(Y^2-26)})' }

Not quite the same as the HP48, which allows only one occurrence.

See also the entries for **n1**, **n2** and **s1,s2**.

Access: S.SLV,
93 MENU (old SYMB)

KERRM

Returns text of last Kermit error packet.

Access: 104 MENU

KEY

Detects keypress during program execution, returns keynumber, 1; otherwise returns 0. See also **Key Numbers**.

keypress → rc.p 1.
no keypress → 0.

rc.p - see Key Numbers

Pressing KEY in stack mode does nothing but put 0 on the stack.

Typical use:

<< DO 400 1 BEEP UNTIL KEY END >>

Sounds tone until keypress; returns keynumber. Operation is not precise with older ROM versions.

See also **Helptext**, section 3 for another KEY example.

Access: PRG-NXT-IN

Key Numbers

Key numbers are row-col.plane, where plane is a shift number. A typical example: LS-STO is 32.2.

Shift Nos:

- .1 norm (unshifted)
- .2 LS
- .3 RS
- .4 alpha
- .5 alphaLS
- .6 alphaRS

Counting the row/col numbers is straightforward except for the **four cursor keys**. Unshifted, these are:

25.1
34.1 36.1
35.1

To find a keynumber, run

<< 0 WAIT >>

and press the desired key(s).

If .01 is added to a shifted key number, the shift key must be held down to obtain the user mode function.

Example: If the Purge command is assigned to the rightshift P key (44.3), the keynum becomes 44.31 and RS must be held down at the same time as P.

See also [ASN](#), [DELKEYS](#), [STOKEYS](#), [TakeOver](#), [USER](#).

KEYTIME

Sets a time delay after a keystroke to eliminate contact bouncing (*ie*, multiple entries per keypress). Older OSs for the HP49G+ used hardware debouncing. Keytime was restored to v2.09 for the HP49G+/50G.

KEYTIME→ returns key delay setting in ticks (see [TICKS](#)).

→KEYTIME sets new value (max. 4096).

Keytime should be set to the lowest number that prevents key bouncing . Try 250 as a start.

Access: CAT

KGET

Gets file(s) from remote Kermit server.

```
'name' →  
{list of names} →
```

Access: APPS-I/O-Transfer,
 104 MENU-SRVR

KILL

Cancels **DEBUG**, all halts, and any program calling it (including any calling main program in the path).

Because of its aggressive action, it should be considered a last resort in programming an exit.

Access: PRG-NXT-NXT-RUN

LABEL

Labels axes with current range and vars from **PPAR**. (For the text editor's LABEL, see the entry for **TOOL/EDIT menu**. For labelling a matrix, see **Matrix Writer**.)

For user-defined labels:

```
{ "hlabel" "vlabel" } AXES LABEL →
```

Access: Graphics menu-EDIT,
 81 MENU

LANGUAGE

Languages apply only to error messages (and on the 49G+/50, the Filer labels). This may be extended in future OS upgrades.

Language→ returns n, the current setting.

→Language sets new value, 0-2.

- 0 - English
- 1 - French
- 2 - Spanish

Access: CAT

LASTARG/LAST

Returns arguments used by last command. In RPN, same as [ANS](#) on the keyboard. Can be abbreviated to LAST.

Note that LASTARG must be enabled by the ARG key in LS+MODE-MISC (the default is enabled) or by clearing flag -55. See the entry for [Mode-Misc menu](#). (Note that there is a keyboard ARG - this is for angles, not LASTARG.)

LASTARG (flag -55) affects commands such as [RCL](#) if a var doesn't exist; the varname is dropped if LASTARG is off.

LASTARG should be enabled in programs using [IFERR](#).

Access: LS-ENTER,
PGM-NXT-NXT-ERROR

lbmol

A typing aid for user-defined units.

→ 453.59237_gmol
55_ → 55_gmol

Access: 117.02 MENU,
APPS-EqnLib-UTILS

LCD series

LCD→ captures the stack display to a Lev 1 grob.

→LCD sends a grob (such as one in a program or on Lev 1) to the display; it must be followed by 7 [FREEZE](#).

To preserve the current softkey menu or to capture screens such as the Eq Writer, use CAT LCD→.

To capture the main screen of the [FILER](#), run:

<< FILER LCD→ >>

Exit from the Filer and the grob will be on Level 1. To capture directories, you have to store

<< LCD→ >>

in the desired dir. In the Filer, cursor to this and press EVAL. There doesn't seem to be a way to avoid having the LCD→ program show in the menu, unless you set up [Conn4x](#) and do a screen shot with ON+up-cursor.

To send a Lev1 grob to the graphics editor, enter **PICT STO**. You can also capture screens with Conn4x.

Access: PRG-NXT-GROB-NXT

LIB

(Note that LIB does not apply to the HP49G+/50G SD card.)

Displays (a) a softkey menu of installed libraries, and (b) directory-like displays of ports 0-2.

Note that the Home dir does not display in port 0, although that's where it is (*ie*, objects stored in Home dirs subtract from port 0 memory).

To put a library on the stack, press LIB and then NXT until you see the ports, 0 to 2. Press the port key, find the libnum, and press RS-libnum.

Libraries on the stack display as

```
Library 1758: HLP4...
```

or similar.

Access: RS-2

LIBEVAL

A binary integer with LIBEVAL calls built-in library routines:

```
# A300Eh → softkey listing  
OR  
# B4001h → Numeric Solver
```

Since a wrong integer can corrupt memory, doublecheck before using LIBEVAL. The integers are not the same for the 48 and 49 - cross-references are available from the Entry Points section of hpcalc.org.

Access: CAT

LIBRARY basics

Libraries are groups of related programs (a directory) in a single object, using UserRPL, SysRPL and/or ML.

See **CRLIB** for details on making your own library, and **LIB** for accessing libraries.

HP48 and 49G+/50G

HP48 Libs are not compatible with the 49/50 because of differences in memory locations. HP49G+ and HP50G libs are compatible with each other unless a 50G library accesses the serial port (which the 49G+ doesn't have).

49G

All 49G libs will run on the 49/50 series.

Libs for the 49G+/50G will usually run on the 49G unless the lib accesses the SD card or the 49G+/50G ARM processor (the 49G uses a Saturn).

The 49G also has a 131 by 64 pixel screen (the 49G+/50G series uses 131 by 80).

HP48GII

The HP48GII is a subset of the 49G+/50G but has memory fixed at 128K (port 0 only, and recent models have 256K). Libraries must be installed in port 0 and will subtract from the Home dir's free memory.

Note that library numbers stored on the 49/50 have an L added (*eg*, 1758 becomes L1758 in the Filer). You can recall (or purge) a library from ports 1 or 2 without the L:

```
:2: 1758 RCL → library
```

but it's required for the SD card (port 3):

```
:3: L1758 RCL → library
```

Storing Only

If you're only storing a lib in a port and don't want it to run, use 1 →**LIST** on it first (large libraries may cause insufficient-memory errors), or on the 49G+/50G, store it on the SD card.

Transfer/Installation:

Quick Method:

Transfer from the PC - the library will be in the current dir. Use the **FILES/FILER** (LS-FILES) to move it to port 1 or 2, and then do an ON+C warmstart - see **Reset** for more on ON+C.

For the HP48GII, or if you get low-memory errors, use the full method below.

On the 49G+/50G, you can also use a card reader to transfer the lib into the SD card. Use the Filer to move it to the desired port and do an ON+C warmstart - see **Reset** for more detail about ON+C.

Full Method:

1. Transfer the library from the PC into the Home dir using HP's Conn4x comm program in binary mode (the default).

2. Put the library on the stack with RS-name and enter the port number as 0, 1, or 2 (2 is best - see below).
3. Press STO. If you get an insufficient-memory error, purge the library from the dir (so that it's now only on the stack) and press STO again.
If you still get memory errors, see **MEMORY basics**.
4. To attach the library and have its name display in RS-LIB, do a warmstart with ON+C - see Reset for more on warmstarts.
5. You can now delete the lib from the Home dir if you haven't already done so. You can use **PURGE** or the Filer.

Almost all libs are auto-attaching and will run from any dir. A possible exception to auto-attach is the internal lib 256. If you type 256 MENU and see it, it's attached. If not, type 256 **ATTACH** first.

ATTACH/DETACH

To make a library's commands available only from one dir:

Assume you want the CAL calendar library (lib # 879) to run only in your DAYTIMER dir. (**ATTACH** and **DETACH** are in 110 MENU as well as CAT.)

Substitute whatever dir you want for the fictional DAYTIMER.

From the Home dir: 879 DETACH →
From the DAYTIMER dir: 879 ATTACH →

Typing CAL from any dir but DAYTIMER or its subdirs will result only in 'CAL' on the stack.

This change will be undone by the next reboot unless you put it in **STARTUP**. You would add this to the program:

```
HOME 879 DETACH DAYTIMER 879 ATTACH
```

If STARTUP doesn't exist yet, enter from the Home dir:

```
<< HOME 879 DETACH DAYTIMER 879 ATTACH >> 'STARTUP' STO →
```

Other Information:

- port 2 (flash) is the usual port for libs. Storing in 0 will reduce the memory available for the Home dirs. Port 1 (RAM) is fast, but not as safe as port 2 (which is non-volatile ROM).
- port 3 (49G+/50 only) is the SD card. Libraries can be stored there but won't run. Note that library numbers stored on the SD card have an L added (eg, 1758 becomes L1758).

- libs are not self-modifying and can't even be read without special tools (OT49+ from [hpcalc](#) is good).
- any lib or lib command can be assigned to a key.
- to see or run a library's commands, you can also enter <libnum> MENU.
- if a defective lib causes continuous rebooting, do an ON+C, then quickly hold down the **Backspace** key (the left arrow to the right of SYMB) to prevent libs from attaching. Then remove the offending lib with the Filer. Getting this to work is a matter of the timing of the BKSP press. Several tries might be required.
- if you have copies of the same lib in different ports, only the lib with the lowest portnum will run. This means you can have a safety copy in port 2 and a working copy in faster port 1.

LIBS

Returns list of libraries in ports 0, 1, and 2.

```
→ { title libnum portnum ... }
```

Access: 110 MENU

LINE

1. In stack mode, draws a line in **PICT** using either coordinates or pixels.

```
(x1,y1) (x2,y2) →
```

OR

```
{#m1 #n1} {#m2 #n2} →
```

If a drawing already exists, coincident pixels are left on. To turn them off, use **TLINE**.

Access: PRG-NXT-PICT

2. In graphics mode, draws a line from a mark (times key) to the cursor.

Access: Graphics-EDIT

LINFIT

Stores LINFIT into **ΣPAR** so that **LR** uses linear curve fitting. Creates ΣPAR as required.

Access: 96 MENU-ΣPAR-MODL

LININ

Tests whether an algebraic is linear for a given var.

```
'algebraic' 'var' → 0/1
'X^2+5*Y' 'Y' → 1.
```

Access: PRG-TEST-NXT-NXT-NXT

LIST series

List-handling commands are in LS-PRG-LIST and LS-MTH-LIST. All their commands have their own help entries.

Basic Lists:

Elements can be any type of object, or a mix. Using \rightarrow LIST: $ob1\ ob2\ \dots\ obn\ n \rightarrow \{ob1, etc\}$ See also [GET](#), [PUT](#), [SUB](#), [REPL](#).

Single quotes are not displayed inside lists:

```
'AB' 'CD' 2 →LIST → { AB CD }
```

They're redisplayed when the list is split:

```
{ AB CD } OBJ→ → 'AB' 'CD' 2
```

[OBJ](#)→ is in PRG-TYPE.

Adding Elements:

```
{a b c} x + → {a b c x}
```

OR

```
x {a b c} + → {x a b c}
```

See also AUGMENT in CAT.

Removing Elements

Edit the list with the text editor, or in a program, use LIST→ to separate the elements, drop the unwanted ones, and then reassemble them with \rightarrow LIST. See also [HEAD](#), [TAIL](#).

Adding Lists, Numbers:

To add two lists of equal size element to element or add a number to each element, use [ADD](#) (see its entry).

To add two lists end to end, use the plus key:

```
{a b} {c d} + → {a b c d}
```

→LIST

Creates list from elements, size.

$$'a' \ 'b' \ 'c' \ 3 \ \rightarrow \ {a \ b \ c}$$

Access: PRG-LIST
 PRG-TYPE

LIST→

Splits list into elements, size. Same as **OBJ→**.

$${a \ b \ c} \ \rightarrow \ 'a' \ 'b' \ 'c' \ 3$$

Access: CAT

EVAL:

will also split the list into elements without the size, but programs are evaluated, which might cause unexpected results. Lists containing programs should be split with **OBJ→** (in PRG-TYPE) or **LIST→** (CAT).

Command Entry:

In programming, a list allows stack entry of commands that would otherwise error. See **→TAG** for another method.

$$\begin{aligned} \{SIN\} \ OBJ\rightarrow & \rightarrow SIN \ 1. \\ \{SIN\} \ EVAL & \rightarrow error \end{aligned}$$

The following three are in MTH-LIST:

ΔLIST

First differences - *ie*,

$$\{A \ B \ C \ D\} \ \rightarrow \ {'B-A' \ 'C-B' \ 'D-C' \ }$$
ΣLIST

Sum of elements.

ΠLIST

Product of elements.

LNP1

Returns $\ln(x+1)$.

$$x \ \rightarrow \ \ln(x+1)$$

Access: MTH-HYP-NXT,
 LS-8

LOCAL

The LOCAL command creates local vars (see the definition below) that remain in memory for use by other programs. They're somewhat different from the *compiled variable* of the HP48 (which also works on the 49/50 series).

'←A=55' → same

OR

{ '←A=55' '←B=65' } → same

To test:

'←A' EVAL → 55

OR

{ ←A ←B } EVAL → 55 65

The left-arrow is chr 142, available only via the **CHARS** map.

Once the vars have been created by LOCAL, **STO** and **RCL** work as usual. It's possible that LOCAL will interfere with UNDO. If so, UNDO can be restored by pressing LS+MODE-MISC and turning STK off, then on again - note that this may clear the local variables.

To remove:

A warmstart (**Reset**) or **UNBIND** (which didn't always work on the 49G, but the 49G+ seems stable, though you might have to turn off UNDO with LS+MODE-MISC- STK).

A way around the oddities of LOCAL and UNBIND is to use the right-arrow assign method:

<< obj → varname << HALT >> >>

The variable will be in memory until **CONT**, **KILL** or a warmstart. However, the HLT annunciator will be on in the header.

Access: CAT

Definition: *local variables* are usually created by programs and are cleared when the program exits.

eg: << → V1 V2 I << program2 >> >>

V1, V2, and I remain in memory for calculating until program2 terminates.

The opposite is the *global variable*, which is stored in a dir with STO. Global vars are never cleared until purged by the user.

obj 'varname' → obj stored in current dir

LOGFIT

Stores LOGFIT into **ΣPAR** so that **LR** uses logarithmic curve fitting. Creates ΣPAR as required.

Access: 96 MENU-ΣPAR-MODL

LQ

LQ factorization of matrix.

$$[[\text{matrix}]]a \rightarrow \begin{matrix} [[\text{matrix}]]l \\ [[\text{matrix}]]q \\ [[\text{matrix}]]p \end{matrix}$$

Access: MATRICES-FACT

LR

Uses the data in **ΣDAT** to return the Y intercept (**b**) and the slope (**m**) of the linear regression line.

$$\rightarrow \quad b \quad m$$

The type of fit is linear by default (LINFIT) and can be changed by entering LOGFIT, EXPFIT, PWRFIT, or BESTFIT from STAT-Fit-data or 96 MENU-ΣPAR-MODL. This is stored in **ΣPAR** (see below or its entry).

For detail on entering data, see **SCATTER/SCATR** or **STAT**, Fit Data section.

To obtain a plot and the LR equation written into the EQ var, use LS+2D/3D to draw a Scatter plot, then press STATL. STAT-Fit-data will also give the equation, plus the correlation and covariance.

The ΣLINE command will also return the equation of the regression line.

The plotter takes column 1 as X values and column 2 as Y values. To change this, see **XCOL** or **YCOL**.

The format of ΣPAR is: { indep depen y-int slope }.

Access: 96 MENU-FIT

LSQ

Minimum norm least squares solution to linear system where $A \times X = B$.

$$[\text{array}]B \quad [[\text{matrix}]]A \rightarrow [\text{array}]X$$

Use when a system is over-determined (more equations than variables).

Access: MTH-MATR,
MATRICES-OPER-NXT,
RS+7-SYS (78 MENU)

LU

LU decomposition of square matrix.

$$[[\text{matrix}]]A \rightarrow \begin{matrix} [[\text{matrix}]]L \\ [[\text{matrix}]]U \\ [[\text{matrix}]]P \end{matrix}$$

Access: MATRICES-FACT

MANT

Returns mantissa of argument that's using E notation.

$$5.E312 \rightarrow 5.$$

For other exponent notation, returns the E equivalent mantissa:

$$'13^{16}' \rightarrow 6.65416609183$$

since $13^{16} = 6.65416609183E17$. See also **XPON**.

Access: MTH-REAL-NXT

MATCH

Replaces a symbol or pattern in an expression. ↑MATCH works bottom-up; ↓MATCH top-down. It also returns 1. if there was a successful match and 0. if not.

$$\begin{matrix} \text{expr} \{ \text{old new} \} & \rightarrow & \text{newexpr n} \\ 'X^2+Y' \{ 'X' 'A' \} & \rightarrow & 'A^2+Y' 1. \end{matrix}$$

It can match only one variable - see the **WHERE (|)** command for multiples.

It can also be used for numeric solving:

$$'X^2+5' \{X 3\} \rightarrow 3^2+5 1.$$

Drop the 1 and **EVAL** or →**NUM** will return 14.

Wildcard:

You can also use the ampersand & (char 38, alpha-LS-ENTER) to match all occurrences. For example, &Z means "match anything" and SIN(&Z) means "all occurrences of sines" (the Z var can be any character):

'TAN (X) ^2 ' { &A 'COS (X) ^2 ' } → 'COS (X) ^2 ' 1 .

'SIN (X) +SIN (Y) ' { 'SIN (&Z) ' 'COS (W) ' } → 'COS (W) +COS (W) ' 1 .

For another use of the wildcard, see [RCL](#). See also SUBST in CAT.

The two MATCH commands are hard to find in CAT because they begin with arrows - they're the 8th and 9th items from the end of CAT.

Access: 93 MENU (old SYMB)

Matrix Writer

► Note: if the MTRW menu is overwritten, TOOL restores it.

To start with an existing matrix, put it on Lev 1 and select EDITB (down-cursor).

When you start a new matrix, the cursor moves to the right (or down, depending on GO). After you enter the first row, move the cursor manually - it will be automatic for the rest.

EDIT	current cell
VEC	sets 1-row matrix to vector
WID→	increases col width
←WID	reduces col width
GO→	cursor proceeds right
GO↓	cursor proceeds down
+ROW	inserts row of zeros at cursor, except last row (press zero, ENTER)
-ROW	deletes row at cursor
+COL	inserts col of zeros at cursor
-COL	deletes col at cursor
→STK	current cell to stack
GOTO	prompts for row, col
DEL	fills selected cell or highlighted range with zero

A range of cells can be selected with Begin and End keys. The entry line allows entering an entire line (across or down, depending on the GO setting):

1 2 3 ENTER ENTER → [1 2 3]

To enter a value from any stack level, use [HIST](#)-ECHO.

The Matrix Writer can use a list-of-lists if you set flag 91.

Labelling a Matrix: from a comp.sys.hp48 post by John H Meyers - you can add labels by using a list-of-lists. The program below is an example:

```
%%HP: T(3)A(D)F(.);
\<< PUSH -91. SF { { "" "Pull" "Push" } { "Mon" 210. 155. } { "Tue" 213. 131. } } EDITB POP \>>
```

Access: LS-MTRW

MAX

Returns greater of 2 inputs.

5 3 → 5

Inputs can be real or integers. Algebraic form is 'MAX (x, y) ' . See also [MIN](#).

Access: MTH-REAL

MAXR

The largest number possible on the calculator. Returns 9.9999999999E499 (numeric mode, flags -2,-3 set) or 'MAXR' (symbolic mode, flags -2,-3 clear).

See also [Infinity](#), [MINR](#).

Access: MTH-NXT-CONST-NXT

MAXΣ

Finds the maximum value in the column(s) in the current statistics variable [ΣDAT](#).

→ n (single column)
→ [array] (multiple columns)

The Maximum function in STAT-Single-Var returns only the maximum for the selected column.

Access: STAT-Single-var,
96 MENU-1VAR

MCALC

Assigns 'name' (or list of names or all MES vars) as var for the **Multiple Equation Solver (MSOLVR)** rather than user-defined (see [MUSER](#), its opposite, for more details).

'name(s)' must be in Mpar, which must be in the current dir (see [MINIT](#)).

'name' →
{list of names} →
"ALL" →

In the MES, you can select the var by pressing the single-quote key and then the var key.

When a variable is not user-defined, the MES can change its value during calculations. The var's softkey will return to white.

Access: MES screen-NXT-NXT,
116 MENU

MEAN

Arithmetic mean of the column(s) in the current statistics variable Σ **DAT**.

- n (single column)
- [array] (multiple columns)

STAT-Single-Var returns only the mean for the selected column.

Access: 96 MENU-1VAR,
STAT-Single-Var

MEM

Free memory available in Port 0, (*ie*, the Home dir) in bytes. MEM DROP MEM is more accurate since MEM forces a garbage collection to remove unused objects from memory.

→ n

Use **PVARS** for ports 0-2. The **FILES/FILER** command will display free memory in port 3 (SD card).

Access: PRG-MEM

MEMORY basics

See also **STO**, **RCL**, **PURGE**, **FILES/FILER**, **PVARS**, and **Recovery**. For memory backups, see **ARCHIVE** and **RESTORE**.

For information on choosing a port for a library, see below, and also **LIBRARY basics**. For storing in ports 1-3, see the entries for **STO** and **SD card basics**. A short summary of the SD card follows the Port 2 section below.

Programs and Libraries

In general, UserRPL programs (the ones inside << >> delimiters) run best from the Home dir and its subdirs (port 0). They can also be run from a port using RS-LIB or :portnum:progname EVAL. Libraries will run from ports 0, 1 or 2. Choosing a port is described below.

Low Memory

If you get low memory errors (eg, when trying to save a file), it's usually because the flash memory needs more room in the Home dir for defragmenting files. A warmstart (ON+C) will clear various buffers and free some memory.

Or, try pressing LS+MODE-MISC and then turning off STK, ARG and CMD. This will provide some extra memory; the features can be turned on again afterwards.

You can also use the Filer to move unneeded files from the Home dir to another port.

See also **Pauses**.

Low Battery

Writing to flash memory (port 2) or the SD card (port 3) requires high-current pulses. If the low-batt indicator is on or about to come on, you will probably be limited to saving to the Home dir (port 0) or SRAM (port 1) as described below. If you can't save at all, power down and replace the batteries.

Viewing Ports

The FILES command (LS-APPS) is the best way to view the contents of the ports. Using the stack, *n* **PVARS** returns a list of the port contents, where *n* is 0, 1, or 2 (it will not access Port 3, the SD card).

Not counting the SD card on the 49G+/50G, memory is divided into three ports:

PORT 0 (IRAM in Filer)

256K of SRAM (about 240K is available to the user). Contains the Home dir and its subdirs. Can contain libraries, but they subtract from Home dir's free memory. Cleared or corrupted by programming crashes or power loss. Cleared by pressing ON+A+F-NO. MEM returns the amount of free space.

MEM DROP MEM is more accurate because it forces a **Garbage Collection**.

Path

Programs (and **RCL**) will operate if called from a lower dir. All programs in the HOME dir can be run from any dir - see **PATH**. Library commands will run from any dir. This can be changed - see **LIBRARY basics**. Note that anything recalled from a higher dir is saved to the **current** dir.

Maximum Memory

The 240K in port 0 that's available to the user is the maximum workspace. Because of segmented memory, objects to be saved in flash memory must be less than 128K.

For maximum available user memory, temporarily turn off Laststack (STK), Lastarg (ARG) and CMD with LS+Mode-Misc. Restore these as needed. You can also temporarily move Home dir vars to the SD card (port 3).

PORT 1 (ERAM in Filer)

256K of SRAM (127K on HP49G+/50G). For libraries or storage - it's faster than flash ROM for libraries (although not much, since the ARM processor is fast). Usually safe from crashes, but cleared by power loss. Not affected by ON+A+F. The 49G+ and 50G have a smaller port 1 because the Saturn processor emulator uses half this memory space.

PORT 2 (FLASH in Filer)

2M of flash ROM, 1M for the OS, 1M for user storage (~ 700K on the HP49G+/50G - the OS uses part of this). For libraries or storage. Storing and retrieval is slower than RAM. Usually safe from crashes. Not affected by ON+A+F or power loss.

Backup objects should be stored here or on the HP49G+/50G SD card - see **ARCHIVE**.

If you're only storing a lib in a port and don't want it to run, use →**STR** or →**LIST** on it first (both are in PRG-TYPE), or use the SD card (port 3).

Insufficient Memory

This error message might occur when you try to save to flash memory (port 2) even when it's not full. This is because the flash is in 128K blocks that need to be defragmented. The defragger moves files in and out of port 0 (the Home dir) to do this. If you get this error message, free up space in the Home dir and try again. See Maximum Memory above.

PORT 3: SD Card

(49G+/50G only - see also [SD card basics](#))

Formatting the card

► *Formatting erases all existing data on the card.*

Note: there is much conflicting information about the SD card, esp. formatting. If in doubt, format on the calculator. It will format cards less than 32K using FAT16 (FAT in Windows). 32K or larger cards will be formatted to FAT32.

Note that FAT16 is required if you want to reflash the ROM from the SD card - see [Reflashing ROM](#).

Older ROMs (v1.23,etc): If the card isn't formatted, press ON+D and then 9 (Format). Follow the onscreen instructions.

Newer ROMs (v2.09,etc): The Filer opening screen has an easy Format command.

SD Card Quick Review

Note that as late as v2.09, the HP4G+/50G has poor SD handling. You can't purge dirs (folders) made with a PC. A card reader is essential for anything but basics.

The Filer can open folders (directories) marked DIR when you press the right cursor. It can't open folders made on the calc and marked HPDIR, but it can delete them.

The Filer can be very slow at displaying the SD card listings if you have a lot of files. You can speed it up considerably by using a cardreader to move files into a few folders.

Long Filenames

In older OS versions, long filenames will be truncated to ABCDEF~1 in the File Manager display. For best readability, use the DOS convention of 8-and-3: FILENAME.EXT. V2.09 of the OS will accept long filenames in the Filer (FILES key). Press VAR for maximum display width.

SD STORING

```
obj :3:varname →  
OR  
obj :3:{dirname varname} →  
OR  
obj :3:"dirname/varname" →
```

(Use the third format for vars made with a card reader)

Note that library numbers stored on the 49/50 have an L added (eg, 1758 becomes L1758).

Storing in a Port

Port numbers are 0 (RAM), 1 (RAM), 2 (flash), and 3 (SD). Port 2 is best for libraries and general storage.

Put the obj on Lev1 and enter

```
:portnum: objname
```

(for example, :2: MYPROG) and press STO.

Another way is to enter:

```
'name', port number, →TAG STO
```

Example: store the program MYPROG on Lev1 in flash ROM:

```
'MYPROG' ENTER 2 →TAG STO →
```

(→TAG is in PRG-TYPE)

If the object is to go in an existing directory, enter

```
:portnum: { dirname objname }
```

OR

```
:portnum: "dirname/objname"
```

If the object already exists in the port, it will say "Object in Use".

If you already have the :portnum:name on Lev1, press ENTER to duplicate it, press PURGE, and then STO. Otherwise:

To delete an existing obj, use the Filer, or enter

```
:portnum:objname
```

and press PURGE (in PRG-MEM).

Example: to delete MYPROG from flash memory:

```
:2:MYPROG PURGE →
```

No quotes are needed around the objname. For objects inside a stored dir, use

```
:portnum:{ dirname objectname }
```

OR

```
:portnum: "dirname/objectname"
```

An Oddity

If you enter :portnum:{name}, you can RCL this object but you can't use STO because it's expecting a directory inside the list.

SD RCL

```
:3:varname → obj  
OR  
:3:"dirname/varname" → obj  
OR  
:3:{dirname varname} → obj
```

There can be false results if the var doesn't exist in the specified dir - see the [RCL](#) entry.

SD PURGE

```
:3:varname →  
OR  
:3:"dirname/varname" →  
OR  
:3:{dirname varname} →
```

(Note that dirs made with a card reader cannot be deleted on the calculator.) The SD card is mainly for mass storage. Libraries will not run. Programs can be evaluated under some conditions, such as:

```
<< :3:MYPROG EVAL >> EVAL →
```

SD General

OS versions as late as 2.09 can't create true DOS-type directories on the SD card - use a card reader instead. The dirs made on the calculator will be called HPDIR by the Filer.

The Filer's features do not work well with SD files. Expect memory and Undefined Name error messages on many vars.

Vars moved back and forth between Home and the SD appear unchanged, but if you copy them to a PC with a card reader, they're binaries beginning with HPHP49, the same as a binary transfer. See the entries for [TRANSFERRING](#) and [Translations](#) .

If you want readability when copying a UsrcPL program to a PC using the SD card, make it into a string before copying it to the SD (LS-PRG-TYPE, then →STR). Use ASCII transfer mode.

Names

Note that stored library numbers have an L added (eg, 1758 becomes L1758). This is because HP doesn't allow names to begin with a number.

MEMORY viewer

► (49G only)

From the HP49 FAQ, hpcalc.org, condensed for HLP49:

Access the memory viewer with On+D, then A. This starts you at address 40000. The left and right cursor keys move by one nibble (00001), the up and down keys by sixteen (00010), the add and subtract by 01000, and the multiply and divide by 10000.

The ON key exits (and reboots). The 0-9 and A-F keys do not modify memory, but select which bank of the flash memory is visible from 40000 - 7ffff.

MENU

For n, displays corresponding softkey menu. (If any menu is already displayed, RCLMENU will return its number for you.) See also [WAIT](#) for displaying any menu in a program. See also [TMENU](#).

n →

Some useful values of n:

0	Last menu
1	CST menu
2	Same as pressing VAR
93	Old HP48 SYMB menu
145, 1792	All branching programming commands
2050	All commands

If the menu number has a decimal part, such as 2.01, it indicates the softkey page number.

To see or run a library's commands, enter <libnum> MENU.

Creating CST

MENU can also create var CST, the custom menu, from a list of desired vars (see the entry for [CUSTOM](#)):

{ varnames, etc } → new CST stored in current dir

Access: LS+MODE-MENU,
 PRG-NXT-MODES-MENU

MENUXY

Displays the softkey menus of 141 CAS commands, 0 to 140.

y x →

y = which of the 141 begins the menus (0 gives the first, EXPAND)

x = the number of the last command displayed, 0-140

Example: 0 11 starts at the beginning and displays 12 commands, which is two softkey pages. 0 140 displays all 24 pages, EXPAND to VER.

Access: CAT

MERGE

An obsolete port-management function from the HP48 command set. Not used on the HP49/50.

MIN

Returns lesser of 2 inputs. See also [MAX](#).

5 3 → 3

Inputs can be real or integers. Algebraic form is 'MIN(x,y)'.

Access: MTH-REAL

MINEHUNT

The Minehunt game seems to come and go, depending on the OS version.

If you create a variable called 'Nmines' with a negative number in it, Minehunt will reveal that many mines. This seems pointless.

If you want to save the game and continue later, press STO. The game will be stored in a variable named MHpar. If you want to begin a new game instead, delete MHpar.

Access: 117 MENU

MINIFONT

Minifont→ returns the current minifont.

→ fontobj

→Minifont sets the font used as the minifont.

fontobj →

The calc has no facility for editing the minifont. Minifont editors are available from hpcalc.org (and some mini characters don't display properly). None of them are particularly good.

Font changes are reset to the default by warmstarts. To make a change permanent, store the new font with a unique name in the Home dir. Put:

```
varname →MINIFONT
```

in the var **STARTUP** in the Home dir. To undo this, edit STARTUP.

Access: CAT

MINIT

Initializes Multiple Equation Solver (**MSOLVR**); creates Mpar data variable.

Syntax is:

```
{list of eqs} STEQ MINIT MSOLVR
```

See also **MITM**, **MUSER** and **MROOT**.

Access: 116 MENU

MINR

In numeric mode, flags -2,-3 set, returns 1.E-499, the smallest number > 0. In symbolic mode, flags -2,-3 clear, returns 'MINR'.

See also **MAXR**.

Access: MTH-NXT-CONST-NXT

MINΣ

Finds the minimum value in the column(s) in the current statistics variable **ΣDAT**.

```
→ n      (single column)
→ [ array ] (multiple columns)
```

The Minimum function in STAT-Single-Var returns only the minimum for the selected column.

Access: 96 MENU-1VAR,
STAT-Single-Var

MITM

Changes Multiple Equation Solver title and var order. Mpar must exist - see [MSOLVR](#) or [MINIT](#). In program, place after MINIT and before MSOLVR. The full syntax is:

```
{eqn list} STEQ MINIT "title" {list of vars} MITM MSOLVR →
```

If there's no title, use a null string ("").

Note that the title bar might only flash for a short time (it depends on the OS version). If so, follow MSOLVR with 2 [FREEZE](#). It will disappear with the first keystroke.

Access: 116 MENU

MOD

Returns remainder where $x \bmod y = x - (y \cdot \text{floor}(x/y))$.

```
x y → x mod y
7 3 → 1.
```

Access: MTH-REAL

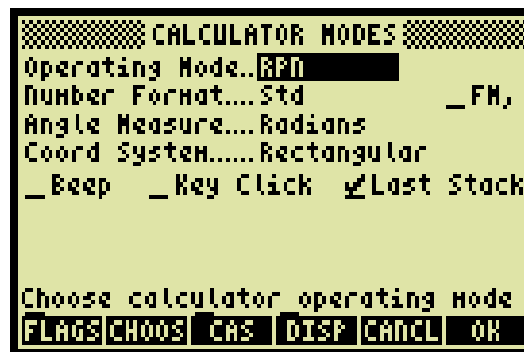
MODE screens

Note that in the MODE displays, the +/- key cycles through the choices in fields, and may be faster than the CHOOS box.

More detail on flags is in [FLAGS - list of flags](#). Stack commands have their own entries.

MAIN SCREEN

NAME	Pgm/Stk Version
Operating Mode	Flag -95
Number Format	LS+MODE-FMT <i>Note 1</i>
Angle Measure	DEG, RAD, GRAD
Coord. System	RECT, CYLIN, SPHERE
Beep	Flag -56
Key click	Flag -75
Last Stack	LS+MODE-MISC-STK <i>Note 2</i>



Note 1. See also flags 45-50 in the [FLAGS](#) entry. Decimal places for formats other than STD are entered into the small field that opens to the right, or use CHOOS.

Note 2. Last Stack enables or disables the [UNDO](#) function, which can use a lot of memory with large objects.

CAS MODES

<u>NAME</u>	<u>Pgm/Stk Version</u>
Indep. Var.	STOVX
Modulo	See CASDIR
Numeric	Flag -3
Approx.	Flag -105, RS+ENTER
Complex	Flag -103, LS+TOOL
Verbose	Flag -99 <i>Note 3</i>
Step-by-step	Flag -100 <i>Note 3</i>
Incr. Power	Flag -114
Rigorous	Flag -119 <i>Note 4</i>
Simp non-rat.	Flag -111

Note 3. Verbose and SBS display very small amounts of calculating information in the header.

Note 4. Rigorous can be left on for most calculating. See Flag -119 in **FLAGS**.

DISPLAY MODES

<u>NAME</u>	<u>Pgm/Stk Version</u>
Font	Fonts 6-8 <i>Note 5</i>
Edit	Flag -73 <i>Note 6</i>
Indent	Flag -68
Stack	Flag -72 <i>Note 7</i>
Textbook	Flag -79 <i>Note 8</i>
EQW (minif.)	Flag -82
Sm. Stk Disp	Flag -80 <i>Note 9</i>
Header lines	→HEADER (see HEADER series)
Clock	Flag -40
Analog	Flag -67 <i>Note 10</i>

Note 5. Allows fonts 6-8. The browser allows manual searches for fonts but doesn't load them.

Note 6. Usage is not clear, since all edit modes are full-screen.

Note 7. Minifont stack except for arrays, lists, etc. See *Note 9*. For full stack minifont, set both Stack and Small Stack Disp.

Note 8. Flag -52 must be clear for textbook display. Also, flag -65 must be clear for textbook display of all levels.

Note 9. Minifont stack for all objects other than text or numbers. The help line says EQW, which is incorrect, as is the flag browser.

Note 10. Analog clock updates only every 5 min.

Mode-Flag menu

The Mode-Flag menu is for controlling the flag settings. For the meaning of the flags, see the [FLAGS - list of flags](#) entry.

The softkey version is accessed with LS+MODE-FLAG or PRG-MODES-FLAG. Commands apply to both system flags (*negative* flag number) and user flags (*positive* flag number).

There are 128 system flags (negative) and 128 user flags (positive). Except for 60 and 61 (which control the [CONST](#) command), user flags have no function other than user-set indicators.

Archiving Flags:

Since flags are not archived by the [ARCHIVE](#) command, it's a good idea to store the flaglist from [RCLF](#) (see below) in a var. Flags can then be restored with STOF. The HP [Conn4x](#) comm program makes a variable containing flags ('fLaG') and stores it in the Home dir. When run, it restores the flag settings and deletes itself.

Flag Toggle:

A program that toggles any flag on and off, **where -nn is the flag number** (for user flags, omit the minus sign):

```
<< -nn FS? IF THEN -nn CF ELSE -nn SF END >>
```

SF:

Sets flag: n →

Use a list for setting multiple flags. Example: { -2 -3 -105 } SF →

Flags that are set have a checkmark in the flag browser (MODE-Flags).

CF:

Clears flag: n →

Use a list for clearing multiple

flags. Example: { -2 -3 -105 } CF →

Flags that are clear have no checkmark in the flag browser (MODE-Flags).

FS?:

Tests for set flag: n → 0/1

FC?:

As above for clear flag

FS?C:

Returns 1 if flag is set, then clears flag, else returns 0

FC?C:

As above for clear flag

STOF:

Stores list of flag settings {see RCLF below}

RCLF:

Recalls list of flags with four binary integers:

```
{ # system 1-64
  # user 1-64
  # system 65-128
  # user 65-128 }
```

The binary integer format depends on the setting of **BASE**.

RESET:

Resets all flags to the HP default (algebraic mode, etc). Note that the defaults vary with the ROM version and may not agree with those in **FLAGS - list of flags**.

PUSH/POP

You can set and clear flags in a program without affecting the current settings. PUSH stores the flags and paths in the ENVSTACK var (see **CASDIR**), and POP will restore them:

```
<< PUSH your_program POP >>
```

Push and Pop are in CAT.

Mode-FMT menu

The softkey version of Format is accessed with LS+MODE or PRG-NXT-MODES-FMT. Most features are available via the MODE window (and certain system flags).

Features such as STD are enabled when the softkey menu box is showing a white square.

For number formats, internal accuracy is maintained in all formats; to reduce the precision of a number, use the **RND** or **TRNC** commands. Note that E (exponent) notation is powers of 10; *ie*, E2=10².

STD:

Displays numbers in standard mode (maximum decimal places, no thousands separator).

FIX:

Displays with decimals fixed to n places; reals have thousands separator.

```
4 → 3,141.5927
```

SCI:

Displays with **E** (exponent) notation and *n* decimal places.

```
4 → 3.1416E0
```

ENG:

Displays with E (exponent) engineering notation with E in multiples of three and n decimal places (the E and exponent count as places).

4 → 314.16E3

FM,:

Sets comma instead of decimal point as fraction mark, same as flag -51.

ML:

Displays multiline objects with many lines instead of one; maximum 7 with header and font 8 (49G+/50G); same as flag -52.

► Note that single-line display (flag -52 set) turns off the textbook mode, even if it's checked in MODE-DISP.

Mode-Misc menu

The softkey version is accessed with LS+MODE-MISC or PRG-NXT-MODES-MISC. Most features are also available via the MODE window or system flags. Features are enabled when the softkey menu box is showing a white square.

BEEP:

Toggles the normal beep on and off, same as flag -56

CLK:

Enables clock display in the header, same as flag -40

SYM:

Enables symbolic mode instead of numeric, same as flag -3

► The following four commands can take up a lot of memory when large objects are kept. They can be turned off to free up memory if required. Empty by turning them off and then on again.

STK:

Saves last stack (*ie*, enables **UNDO**).

ARG:

Saves last arguments, same as flag -55; args are returned with the ANS key or LASTARG command - note that there is a keyboard ARG for angles.

ARG/LASTARG (flag -55) affects **RCL** if a var doesn't exist; see the end of the RCL entry. Flag -55 should be clear when a program uses **IFERR**.

CMD:

Saves last four command line entries (displayed with **CMD** key - see the help entry for this)

INFO:

Enables display of the Info variable in certain dialog boxes

MROOT

Uses Multiple Equation Solver to solve for one or more variables using the equations in [EQ](#). This lets you solve from the stack without running MSOLVR. Mpar and the variables must exist in the current dir - see [MSOLVR](#), [MINIT](#).

MROOT then solves for var(s):

```
'varname' → result  
{varnames} → {list of results}
```

The result(s) will also be stored in the corresponding var(s).

MROOT is best used after the MES has created Mpar and the vars. If you want to do it manually, store the list of equations in EQ and store the known values in their vars (MROOT will create the unknowns). Run MINIT to create Mpar.

Put the name of an unknown on the stack and run MROOT.

When using MROOT, it may say "Too many unknowns" even though all needed values exist. Reassign a list of these values with [MUSER](#).

Note that the MES commands do not apply to single equations - there must be a list with at least two in EQ. (You can put two copies of an equation in a list in EQ - that will work.)

See also [MCALC](#).

Access: 116 MENU

MSGBOX

Displays string in message box. Maximum 15 chars by 6 lines with Font 8 (23 chars with minifont stack - flag -72 set).

```
"string" →
```

Error messages use the message box - to change them to the header HP48 style and adjust the time that they're on, see the entry for [STARTERR](#).

Access: PRG-NXT-OUT

MSLV

The Multivariate Solver can solve for multiple equations. It's best suited to CAS expressions - [MSOLVR](#) is faster and more convenient for most uses.

The RPN format is:

```
[ 'eqn1' 'eqn2'... ]
[ vars to solve for ]
[ initial guesses ]    →    [ solutions ]
```

Note that MSLV switches to Approximate, radian, and symbolic modes. The closer the initial guesses to the answer, the faster it converges to the result.

A simple example:

```
[ 'I*R=6' 'I^2*R=12' ]
      [ 'I' 'R' ]
      [ 2 4 ]    →    [2. 3.]
```

The CAT Help example in RPN:

```
[ 'SIN(X)+Y' 'X+SIN(Y)=1' ]
      [ 'X' 'Y' ]
      [ 0 0 ]    →    [ 1.8238 -.9681 ]
```

Access: NUM.SLV-MSLV

MSOLVR

Multiple Equation Solver. A list with ≥ 2 equations must be in **EQ**. **MINIT** must be run to create the Mpar data variable.

```
{list of eqs} STEQ MINIT MSOLVR    →    inverse menu of EQ vars
```

Enter known values and press the appropriate softkey to store (not LS-varkey). Solve for unknown with LS-varkey. RS-varkey recalls value as usual.

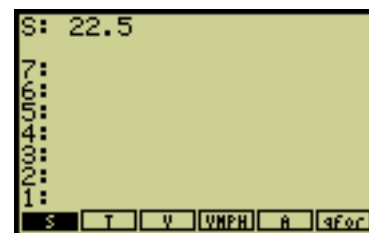
Using ALL:

ALL - undefines all variables so you can start over. Dark keys return to white.

LS-ALL - solves for all unknowns.

RS-ALL - returns an info screen with the last results.

MUSER, MCALC - see their entries.



Units can sometimes cause errors in user programs, especially with constants in the equation solvers. Units can be removed with **UVAL**:

```
UVAL (CONST (var) )
```


MSOLVR is not meant for single equations. In this case store the eqn in EQ and use **one** of the following:

1. Use RS-7 (NUM.SLV) and its Solve equation option, **or**:
2. Use 30 MENU, which is the softkey version of NUM.SLV and works much like MSOLVR, **or**:
3. Duplicate the equation with ENTER, put the two copies in a list, store it in EQ, and run MINIT MSOLVR.

#3 has the advantage that the key labels become dark when a value is stored.

See also **MITM**, **MUSER** and **MROOT**. See also **SOLVEQN** for a similar solver for the equations in the Equation Library, and **SOLVING basics** for **Clean Solver**, easier to use than 30 MENU.

Access: 116 MENU

MUSER

Specifies a name (or list of names or all MES vars) as user-defined var(s) for the Multiple Equation Solver (**MSOLVR**). 'name(s)' must be in Mpar, which must be in the current dir (see **MINIT**).

```
'name' →  
{list of names} →  
"ALL" →
```

In the MES, you can select the var by pressing the single-quote key and then the var key.

When a variable becomes user- defined, the MES will not change its value during calculations unless you solve for it directly. The var's softkey will turn dark. To undo this and return the var to normal MES usage, use **MCALC** as above. The var's softkey will turn to white.

When using **MROOT** it may say "Too many unknowns" even though all needed values exist. Reassign these values with **MUSER** as above.

The following example shows how to use MUSER to insert optional user values into the MES in case you need them. The program solves for frequency, velocity, or wavelength (G1). The velocity of light is included in two vars, SI m/s (VSI) and English ft/s (VEng) values. These two vars will be dark because they contain a value.

To use it, enter two of F, V and G1 and use the leftshift to solve for the third. To use the velocity of light, use the rightshift to put the desired value on the stack, then press V to store it as an MES value.

```
%%HP: T(3)A(D)F(.);
\<< 299792458. 'VSI' STO
983571056.43 'VEng' STO
{ 'F=V\GI' 'VEng=983571056.43'
'VSI=299792458.' } STEQ MINIT
{ VEng VSI } MUSER MSOLVR \>>
```



Note that VSI and VEng appear twice - once for storing and again in the list so that MUSER can find them. To exit, press the VAR key. Stored values must be deleted manually.

Access: MES screen-NXT (as reqd),
116 MENU

n1, n2

In results from **ISOL** and **QUAD** solutions, the variable n1 represents an arbitrary integer 0, ±1, ±2, etc. Additional arbitrary integers are represented by n2, n3, etc.

If flag -1 is set, then ISOL and QUAD return principal solutions, in which case the arbitrary integer is always zero.

See also **s1,s2**.

→NDISP

Number of lines over which a program is displayed on the stack. Default and maximum is 5 for the HP49G and 7 for the HP49G+/50 (for Font 8, header 2).

n →

Access: CAT

NDIST

Normal probability distribution at x based on **MEAN** m and variance v . Statistical matrix **ΣDAT** must exist - see **STAT**-Single variable.

m v x → n

Access: MTH-NXT-PROB-NXT

NEG

Changes sign; program version of the +/- key.

n → -n
-n → n

Note that the +/- key will also toggle checkmarks on and off in dialog boxes.

Access: +/-,
 MTH-NXT-CMPLX-NXT,
 CMPLX

NEWLINE

The newline command is a linefeed only - **CHR 10**. The symbol is ↵ in files.

If you import text from a PC, the newline is usually CR/LF - CHR 13 and CHR 10. The carriage returns show as a black square in the text editors.

One way to remove the squares is Find/Replace in the text editor. Use **CHARS** to enter CHR 13 into Find; leave Replace blank.

To automate this, see the IN/OUT programs in the **Translations** entry.

Access: RS-period

NEWOB

Changes existing object into new object.

obj with name → new unnamed obj (with same content)

Creates a new copy of an object in memory so that the original can be deleted; creates a separate copy of an object in a composite such as a list so the composite can be deleted.

Flag -55 should be set so the copy is not saved in Lastarg.

Access: PRG-MEM

NEXT

NEXT in the editor's TOOL-SEARCH FIND/REPL menu finds the next occurrence of the Find/Replace entry. See the entry for TOOL menus.

It's also used in definite structures **FOR-NEXT** and **START-NEXT** - it begins the next loop. It can be replaced by:

stepsize STEP

for fractional increments in loop values.

Access: PRG-BRCH

NXT

NXT displays the next six softkeys (if there are more).

LS-NXT displays the previous six softkeys.

LS+NXT displays the previous menu.

Access: NXT key

NOT

Returns ones complement or logical inverse; makes non-zero zero and makes zero into 1. The one in the left of the table below represents any non-zero value.

1	→	0.
0	→	1.

Reals or integers are taken as zero or non-zero:

5	→	0.
0	→	1.

In bitwise calculations with user binary integers, each bit is inverted:

1010101010b → # 101010101b

Note that NOT works with the current binary wordsize - the above is reset with [STWS](#) to 10 bits from the default 64. Leading zeros are not displayed on the stack, so the result has 9 bits.

The following lets the editor start whether the stack is empty or not. If DEPTH returns zero, NOT makes it a one and [IFT](#) puts a null (empty) string on the stack. Store it as ED or EDT or similar, or assign it to a key (see [ASN](#)).

```
<< DEPTH NOT "" IFT EDIT >>
```

Access: PRG-TEST-NXT,
 BASE-NXT-LOGIC

NOVAL

Placeholder for empty fields in dialog boxes, positions in lists, etc. Can be placed on the stack; its [TYPE](#) is 19 (command).

Access: PRG-NXT-IN

NSUB

Access to sub-list or element position during program iteration using **DOSUBS**.

```

      { 22. 33. 44. 55. }
              1.
<< ENDSUB NSUB >> DOSUBS →
      { 22. 4. 1. 33. 4. 2. 44. 4. 3. 55. 4. 4. }

```

In the above simple demo, **ENDSUB** returns 4 (no. of elements) for each element, while NSUB returns each element's position (1, 2, 3, or 4).

Access: PRG-LIST-PROC

NUM

Returns character code number of char (or first char in string). Its inverse is **CHR**.

```

"A" → 65.
"ABC" → 65.

```

Chrs 32-127 correspond to the usual ASCII map. Chrs 128-255 are special HP symbols - see the **CHARS** map (RS-CAT on HP49G, RS-EVAL on HP49G+).

Access: PRG-NXT-CHARS,
RS+CHARS

→NUM

Converts symbolic expression to decimal equivalent, or converts rational number to decimal approximation.

```
'Σ(X=1,10,X^2)' → 385.
```

```
π → 3.14159...
```

```
5/9 → .555555555556
```

If you have an equation on the stack, **→NUM** may say "Undefined variable". Remove the variable and the equals sign to make an expression. **EQ→** from PRG-TYPE-NXT does this easily:

```

'X=2*9' → "Undefined var"
'X=2*9' EQ→ → 'X' '2*9'
'2*9' →NUM → 18.

```

Note that if **→NUM** is used to evaluate certain expressions (such as integrals), it may reset numerical mode (flag -3 set). With integrals, it may create the IERR error variable in the current dir.

Access: RS-ENTER

Numerical/Symbolic modes

These are controlled by flags -2 and -3 (MODE-FLAGS).

Flag -2

Clear:

Constant → symbol

Set:

Constant → number

Flag -3:

Clear:

Function → symbolic

Set:

Function → numeric

If you try to solve an algebraic and get "Undefined Variable", the calculator is likely in numeric mode. Clear flags -2 and -3 for symbolic mode.

If you try to use numeric operators and get incomplete results (such as fractions when you divide), the calculator is likely in symbolic mode. Set flags -2 and -3 for numeric mode.

One way to automate this is to assign simple programs to keys:

```
<< { -2 -3 } SF >> (numeric)
<< { -2 -3 } CF >> (symbolic)
```

You can also toggle the modes with a single assigned key:

```
%%HP: T(3)A(D)F(.);
\<< -2 FS?
  IF
  THEN -2 CF "Symbolic"
  ELSE -2 SF "Numeric"
  END -3 FS?
  IF
  THEN -3 CF
  ELSE -3 SF
  END
\>>
```

The two indicator strings are optional and can be removed.

Access: MODE-CAS-Numeric
 MODE-FLAGS

NUM.SLV

Numerical solvers of various types using dialog boxes. More information is in entries as shown below:

<u>FUNCTION</u>	<u>ENTRY</u>
1. Solve eqn	SOLVING basics
2. Solve diff eq	DIFFEQ (part 2)
3. Solve poly	---
4. Solve lin sys	---
5. Solve finance	FINANCE Solver
6. MSLV	MSLV

Choose the desired type and enter the known values. You can also store an eqn in **EQ** - it will appear in the Solve Equation solver's "Eq:" field. Also - anything you enter in this field is stored back to EQ. Results are also stored in vars with the same name as the variables used in the eqn.

In Solve Equation, the EXPR= key on the EQ field places both sides of the eqn on the stack once solving is done.

If a matrix or array is required, press LS-MTRW from the appropriate field and create the object - it will appear in the solver's field when you press ENTER.

If the obj is on the stack, press NXT-CALC-OK to send it to the selected field in the solver.

When STS (Status?) is selected, it's supposed to replace system messages with CAS information during calculations.

You can also press CALC to return to the stack if you need to calculate values. Press OK to return and insert the value into the selected field.

Units can sometimes cause problems in solvers. If so, remove them first with **UVAL**.

Access: RS-7

NUMX/NUMY

Number of X or Y steps in 3D perspective plots and special-purpose plots like **SLOPEFIELD**. They're stored as elements 14 and 15 in **VPAR**. Called Step Indep, Depnd in LS+WIN.

nx →
ny →

Access: LS+WIN,
81 MENU-3D-VPAR

NΣ

Number of rows in current statistics variable **ΣDAT**.

→ n

Access: STAT-Summary,
96 MENU-SUMS

OBJ→

Separates object into elements. Any added number or list is size or dimensions (rows,cols).

(x,y)	→	x y
{x y}	→	x y 2
(2.,3.)	→	2. 3.
[5 6]	→	5 6 {2}
[[1 2]		
[3 4]]	→	1 2 3 4 {2 2}
'X-Y=10'	→	'X-Y' 10 2. =
'(X,Y)'	→	'X' Y*i 2. +
"ABC"	→	'ABC'
'ABC'	→	N/A
'4*5'	→	4 5 2. *
"ABC5"	→	'ABC5'
"5ABC"	→	5 'ABC'
<< pgm >>	→	N/A

The example for "5ABC" shows that varnames cannot begin with numbers. For a way around this, see the S~N command in the entry for **256 MENU**.

Access: PRG-TYPE
RS+CHARS-NXT

OFF

Turns the calculator off. Useful in alarm programs that turn the calculator on.

The calculator also turns itself off after 5 minutes without a keystroke; this feature is disabled if a program is running.

There is no loss of any data when the calculator is turned off, either manually or automatically; just turn it on again.

For controlling the auto-shutoff, see **TOFF, STARTOFF**.

Access: PRG-NXT-NXT-RUN-NXT (pgm)
RS-ON (direct)

Old SYMB menu

This is from the HP48 command set and is often still useful on the 49/50 series. It contains the commands **COLCT**, **EXPAN**, **ISOL**, **QUAD**, **SHOW**, **TAYLR**, \uparrow **MATCH**, \downarrow **MATCH**, \rightarrow **Q**, \rightarrow **Q π** , **WHERE ()**, **APPLY**, **QUOTE**. Each of these has its own entry.

Access: 93 MENU

ON+ Codes

► Hold ON while pressing other key(s).

F1+F6	memory clear (port 0 only - see MEMORY basics)
F2	cancel unreleased keys
F3	warmstart/reboot (see below)
F4	interactive tests, SD format
F5	continuous test (ON+F3 exits)
F6	the F4 test plus IR,EMI (49G+/50G)
Up-curs	screen dump to I/O
9	cancels repeating alarm
+ or -	contrast adjust

The ON+C (F3) reboot or warmstart cancels any program that's running and clears the stack. It also clears any buffers such as Undo, CMD, and Lastarg. It does not affect the main memory or stored vars.

It's required for completing the installation of new libraries and for exiting diagnostics. It can be used at any time to restart the OS if it appears to have been corrupted by a crash or unstable program. See **Reset** for more detail.

The ON+F4 (ON+D) diagnostic screen also returns a more complete OS number than the **VER/VERSION** command, and includes the build number.

ON+F2+F3+F4 is said to restart the calculator in RPN mode with menu 256 attached. This probably doesn't get used much.

See also **Shortcut keys**, **Recovery**.

OPENIO

Opens the serial port (HP49G only) using the parameters in the variable **IOPAR**.

Access: CAT

OR

Logical OR operator for two arguments. The ones at the left below can be any non-zero value. **BASE** should be in the BIN mode. Leading zeros are not displayed on the stack.

```
1 1  → 1.
1 0  → 1.
0 1  → 1.
0 0  → 0.
```

Reals or integers are taken as zero or non-zero:

```
5 0  → 1.
5 3  → 1.
0 0  → 0.
```

In bitwise calculations with user binary integers, processing is done bit by corresponding bit, so that

```
# 110001b # 101001b  →  # 111001b  is the same as

110001
101001
-----
111001
```

It can be used in an algebraic:

```
'X>2 OR Y<4'
```

This will return 1 or 0 when evaluated, depending on the values stored in X and Y.

It can also be used as an operator in programs. One syntax is below. If either test1 or test2 evaluates to non-zero, the procedure is evaluated.

```
IF <test1> <test2> OR THEN <procedure> END
```

The following checks for a real or an integer on level 1:

```
%%HP: T(3)A(D)F(.);
\<<
  IF DUP TYPE 0. == SWAP
  DUP TYPE 28. == ROT OR
  THEN "<program>"
  END
\>>
```

Access: PGM-TEST-NXT,
 BASE-NXT-LOGIC

ORDER

Reorders vars in a directory.

```
{edited list from VARS cmd} →
```

You can process the list with various commands, such as **[SORT](#)** for alphabetical order. Use with caution, since ORDER will delete vars with missing names.

You can also use ORDER by itself with an appropriate list. This sequence puts the dirs first in alphabetical order (a dir is type number 15). For descending order, insert **[REVLIST](#)** after SORT.

```
15 TVARS SORT ORDER
```

You can also use ORDER from the Filer - see the **[FILES/FILER](#)** entry.

Access: PRG-MEM-DIR-NXT

PARAMETRIC

Sets the plot type to Parametric and creates **[PPAR](#)** if required. See also **[Plots](#)** or **[PLOT command](#)** for more on general plotting. The expression in **[EQ](#)** should be complex in form:

```
' 3*SIN(3*X)+i*2*SIN(4*X) '
```

OR

```
' (3*SIN(3*X), 2*SIN(4*X)) '
```

Either of the above draws the same Lissajous figure. If a previous plot has set the wrong display parameters, either purge PPAR using the stack or press ZOOM-ZDFLT from the graphics menu.

Access: 82 MENU

PARITY

On the HP49G, sets **[IOPAR](#)** to the desired parity. The default is none. Note that IOPAR is ignored by the 49G+/50G USB, and the IR uses the default 0 (none).

```
n →
```

n	Parity
—	—
0	none
1	odd
2	even
3	mark
4	space

Access: APPS-I/O-Transfer,
104 MENU-IOPAR

PATH

The PATH command returns the path to the current dir.

→ { path }

A list with a new path changes to a new dir when evaluated. The path has the form:

{ HOME dir } → to dir

OR

{ HOME dir1 dir2 } → to dir2

DIRNAME

You can also use only Dirname ENTER if moving sideways but not down (see below).

UPDIR

The UPDIR command steps up one dir in the path. If you're in HOME-dir1-dir2, it steps to HOME-dir1.

Access is LS-VAR, 2050.06 MENU, or typed in.

Note that LS+UPDIR returns to HOME from anywhere (see also [Shortcut keys](#)).

PUSH-POP

PATH is best used in programs. To change dirs easily, use the softkeys as noted below, or select using the Filer and press CHDIR.

PUSH and POP save and restore the current paths (and flags) - see the entry for [Programming](#).

RUNNING A PROGRAM

Running a program that's in another dir:

{dirname pgmname} EVAL →

OR

{dir1 dir2 ... pgmname} EVAL →

RECALLING

Recalling an object that's in another dir:

{dirname objname} RCL →

OR

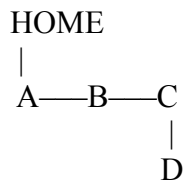
{dir1 dir2 ... objname} RCL →

Programs (and RCL - but see the caution in [RCL](#)'s entry) will operate if called from a lower dir. All programs in the HOME dir can be run from any dir.

{ newpath } EVAL → change to newdir

NAVIGATING THE TREE

You can change to another dir by typing its name, but only if it's on the same level or above. In the tree below, if you're in A you can move across by typing B or C, but to get to D you have to enter C, then D. From D, you can go to A, B, C, or HOME by entering a single name.



In these examples, the PATH command is not required.

From A to C: C →
 From A to D: { C D } **EVAL** →
 From D to A: A →

In the 2nd example, { C D } **EVAL** can be replaced in a program by C D. Either format can be used as long as the commands are evaluated when placed on the stack.

Access: PRG-MEM-DIR

Pauses

The 49/50 series are noted for slight pauses that suspend operation for a few seconds. This is caused by the GC or garbage collection (the removal of unneeded objects from memory). It improves with each OS upgrade, but has never been completely eliminated.

Pauses during operations such as saving to the stack can be caused by a low memory condition. See Memory basics for ways to clear more memory. You can also move some unneeded vars from the Home dir to a port using the Filer.

If the HLP49 index (which uses **CHOOSE**) coincides with a Garbage Collection, the program can pause. This effect was much reduced with v5.0, which forces a GC when the program starts.

PCOEF

Returns coefficients of a polynomial, given the roots. Its inverse is **PROOT**.

```

[roots] → [coeff]

[ -1. -1. ] → [ 1. 2. 1. ]

```

To change coefficients to a symbolic polynomial, see **PEVAL**.

Access: ARITH-POLY-NXT-NXT
 RS+7-POLY (77 MENU)

PCONTOUR (Ps-Contour)

Sets the plot type to Ps-Contour and modifies or creates **PPAR**. The plot is a lattice of line segments, each tangent to a contour of $F(X,Y)=\text{constant}$.

► Note that the command PCONTOUR is called Ps-Contour in the Plot Type fields.

See also **Plots** or **PLOT** for more on general plotting.

Example: Plot the surface contour of $'(X^2.-1.) / (Y^2.-1.)'$.

Press LS+2D/3D and enter:

Type: Ps-Contour **Angle:** RAD
EQ: '(X^2.-1.)/(Y^2.-1.)'
Indep: 'X' **Depnd:** 'Y'

Press LS+WIN and enter:

X-Left: -2. **X-Right:** 2.
Y-Near: -2. **Y-Far:** 2.
Step **Indep:**16. **Depnd:** 16.

Press Erase and Draw. The plot will be very slow (> 5min).

Access: LS+2D/3D-Type,
 85 MENU

PCOV

Population covariance of independent and dependent columns of statistics variable **ΣDAT**, which must exist in the current dir. See also **COVE**, **XCOL**, **YCOL**.

→ n

Access: 96 MENU-FIT

PDIM

Replaces PICT with blank of specified dimensions.

(xmin,ymin) (xmax,ymax) →

OR

{#mmin #nmin} {#mmax #nmax} →

Example: (-3,-2) (3,2) → screen with upper left of (-3,2) and lower right of (3,-2)

Access: PRG-NXT-PICT

%

x percent of y . See also [%CH,%](#).

$$y \ x \rightarrow y(x/100)$$

$$50 \ 10 \rightarrow 5$$

Access: MTH-REAL

%T

Percentage of total on Lev 2 represented by Lev 1.

$$x \ y \rightarrow 100y/x$$

$$50 \ 5 \rightarrow 10. \text{ (Flag -3 set)}$$

$$50 \ 5 \rightarrow '1/10*100' \text{ (Flag -3 clr)}$$

Access: MTH-REAL

PERM

Permutations of n things m at a time.

$$n \ m \rightarrow P_{n,m}$$

$$6. \ 3. \rightarrow 120.$$

Algebraic syntax is 'PERM(n,m)'; in the [EQW](#), enter n , press PERM, then m . On the stack, convert the algebraic to a number with \rightarrow [NUM](#) or [EVAL](#).

See also [COMB](#).

Access: MTH-NXT-PROB

PEVAL

Array of coefficients to symbolic polynomial.

$$[\text{coeffs}] \ \text{var} \rightarrow \text{expr}$$

$$[1 \ 1 \ -1] \ 'X' \rightarrow \ '-1+(1+X)*X'$$

SIMPLIFY returns 'X^2+X-1'

See also [DOSUBS](#), or AXQ in CAT.

Access: RS+7-POLY (77 MENU),
CAT

PGDIR

Purges named dir and its vars, plus any subdirs. There's no confirm and no undo. It does not work with dirs on the 49G+/50G SD card - use :3:dirname **PURGE**, the **FILES/FILER**, or a card reader. (See the entry for **SD card basics** for more on SD syntax).

'name' →

As of ROM v2.09, **dirs made with a card reader** cannot be deleted on the calculator.)

Access: PRG-MEM-DIR

PICT/PICTURE

PICT is the reserved name for the grob (graphics object) that PICTURE displays and is used by plot and graphics functions. The softkey PICT is a typing aid. See PICTURE below for more. See also **Graphics menu** for editing commands.

To put a grob in the graphics editor, STO it with the name PICT. Press the left cursor. If a grob is on the stack, you can display it in the editor by pressing the down cursor (EDITB), but it will not overwrite the existing PICT.

To put the graphics editor grob on the stack, use PICT RCL from the stack, or EDIT-NXT-NXT-PICT→ from the editor.

Access: PRG-NXT-PICT (typing aid)

PICTURE displays PICT with cursor and menu. In stack mode, the left cursor is PICTURE and displays the graphic in PICT.

To obtain the grob, use left-cursor-EDIT-NXT-NXT-PICT→ or use PICT RCL from the stack (or in a program).

PICTURE is often used at the end of a graphics program to keep PICT on the screen.

To display PICT without menu or cursor, use { } **PVIEW**.

To display PICT with a custom menu, use the format in the program below:

```
<< { list of vars } TMENU
{ # 0h # 0h } PVIEW 3 FREEZE >>
```

Access: left cursor, CAT

PINIT

Initializes active ports. From the HP48 command set, where it was used to initialize external RAM cards on the 48GX.

On the 49/50 series, it can be used to remove corrupted objects from ports 0 to 2 - this might depend on the OS version. If you get "Invalid Card Data", it's worth a try. Note that there's always the possibility of data loss.

$n \rightarrow$

where n is the port number from 0 to 2.

Access: 110 MENU

PIX series

PIXON and PIXOFF turn **PICT** pixels on or off. PIX? returns 1 or 0 if pixel is on or off.

$(x, y) \rightarrow$

OR

$\{ \#m \#n \} \rightarrow$

In the graphics editor, PIXON is DOT+ and PIXOFF is DOT-.

Access: PRG-NXT-PICT

PKT

Sends command "packets" to (and receives requested data from) a Kermit server.

"datatype" \rightarrow "response"

Access: 104 MENU-SRVR

Plots

(See the next entry for the PLOT command.)

See also **Graphics menu** for plot editing commands, and LS+**2D/3D**.

The plotter draws the object in the var EQ (see **EQ**, **STEQ**, **RCEQ**). There are no error messages. The object type depends on the type of plot selected - usually it's an algebraic expression.

EQ is the reserved name for the var that's plotted. There can be one in each dir.

To see the variety of plots available, press LS+2D/3D. High- light the **Type** field and press CHOOS. Each of these has its own entry.

See also **RES** for controlling the plot resolution and speed. For plotting vectors, see **Arrays/vectors**.

Plotter Reserved Variables

The plotter creates variables containing values for the current setup - see **PPAR**, **VPAR**, **ZPAR**. If these exist from a previous plot, they can affect the current plot - if you get unexpected results, check these and delete if necessary.

If your plot type has zooming, you can press ZOOM-ZDFLT to set the three control vars to their default values.

Function Plots

The Function plot is used as an example below:

The Function mode plots $Y=f(X)$, where the *Independent* variable (typ. X) is the horizontal axis and the *Dependent* (typ. Y) is the vertical axis.

In Function mode, you can plot:

- An expression in one variable, eg, ' X^2-2 ', which has no equals sign. The plotter solves this as ' $Y=X^2-2$ ' to find the vertical (Y) values. This is the preferred method.
- An equation in one variable, eg, ' $X^2=2$ '. It draws the left side, then the right - ' $X^2=2$ ' will plot as a parabola and a horizontal line.
- A program using the specified var, eg, `<< 'X^2-2' >>`. It must return only one value when evaluated by the plotter. See **IFTE** for another example.

Saving Your Plots

If you have special conditions set up, you can enter PPAR to put it on the stack and save it with a new name ('PPAR1' or similar). To reproduce your plot later, put the contents of this var on the stack and STORe it into 'PPAR'. Store your expression back into EQ, or use a different name.

To save a picture of the plot, press EDIT-NXT-NXT-PICT→ from the plotter. Store the stack grob with a new name. To see it, put it on the stack and press the down cursor. To put it in the plotter, STORe it as PICT. It will use the existing PPAR unless you replace it as described above.

Resetting PPAR

If a previous plot has changed the parameters, you can delete PPAR. An easier way is to store the program below in the HOME dir as NP (for Normal Plot). Typing NP from any dir resets the existing PPAR to default values.

Note that this is for the 49G+ or 50G. For the 49G, change -3.9 to -3.1 and 4. to 3.2.

```
%HP: T(3)A(D)F(.);  
\<< { (-6.5,-3.9) (6.5,4.) X 0.  
      (0.,0.) FUNCTION Y } 'PPAR'  
      STO  
\>>
```

All-purpose Sine Wave

The sine below can be controlled in plotting by changing any of its coefficients.

$$A * \sin(B * X \pm C) \pm D$$

- X is the independent variable.
- A sets the peak amplitude.
- B sets the period inversely; eg, 10 gives one-tenth the period or ten times the frequency.
- C is the phase shift in radians or degrees.
- D controls the offset above or below the X-axis.

Obviously A and B cannot be equal to zero.

No Plot or Unexpected Results:

- Check PPAR, VPAR and ZPAR; purge them if necessary, or press ZOOM-ZDFLT if your plot type supports zooming.
- Check the plot size - it may be too big for the screen's default ± 6.5 horizontal units. Use AUTO from LS+WIN to scale the axes, or set them manually.
- Check that the expression's var and the Indep var listed in 2D/3D are the same.
- Check for radians versus degrees - their plot parameters are widely different.
- Check that the plot type and expression agree; eg, 'X^2+Y^2-5' is a conic, not a function (other vars can be included in a function if they have stored values).
- For gaps in the plot, see [RES](#).
- If only half a curve plots, clear flag -1 for General solutions.

Adding text: see → [GROB](#), [LABEL](#).

Related Commands: [ANIMATE](#), [ATICK](#), [AXES](#), [DRAW](#), [DRAX](#), [ERASE](#), [LABEL](#).

PLOT command

Given an object on Level 1, creates [EQ](#), [PPAR](#) and opens the LS+[2D/3D](#) plot setup screen. The plot type depends on the type of object and must be set manually in 2D/3D.

obj → same; obj stored in EQ

The plot type and independent var will be the one previously set if PPAR exists; if not, the default is Function in X.

See **Plots** above for more on object types. Press LS+2D/3D to change the plot type. See also the entry for the specific plot type.

► Note that PLOT erases before plotting, even if you don't press the Erase key.

A much better discussion of plotting than in the HP49 manual is in the HP48 Series User's Guide, available for free downloading by searching hpcalc.org.

Access: CAT

PMAX/PMIN

Set the upper right (PMAX) and lower left (PMIN) corners of the display; *ie*, they modify the first two elements of PPAR (and will create it if necessary).

$(x, y) \rightarrow$

To reset the default plot size without changing anything else (49G+/50G only - Y values for the 49G are 3.2 and -3.1):

$(6.5, 4.)$ PMAX \rightarrow
 $(-6.5, -3.9)$ PMIN \rightarrow

To reset the default plot size completely from the graphics screen, press ZOOM-ZDFLT (assuming that the plot type supports zooming).

Access: CAT

POLAR

Sets the plot type to Polar and modifies or creates **PPAR**. The independent variable is the angle and the dependent is the radius.

For polar angle mode, see [CYLIN](#). See also [Plots](#) for more on general plotting.

The textbook formula for a polar plot is usually $r = f(\theta)$. In this case r is Y and $f(\theta)$ is some function of X . Y is implicit and isn't needed.

$\cos(X)$ plots a circle around the X axis and $\sin(X)$ plots a circle around the Y axis. X plots a spiral. The following example will plot a rose-like figure:

```
%%HP: T(3)A(D)F(.);  
\<< '4*COS(4*X)' STEQ POLAR ERASE  
DRAX DRAW PICTURE \>>
```

If a previous plot has set the wrong parameters, either purge PPAR using the stack or press ZOOM-ZDFLT from the graphics menu.

Access: LS+2D/3D,
82 MENU

POS

Returns:

- start position of substring
- position of char within string
- position of object within list.

```
"string" "substr" → n
"string" "char"   → n
{list} obj       → n
```

POS returns 0 if it errors. See also [SREPL](#), [SUB](#), [REPL](#).

Access: PRG-NXT-CHAR,
 PRG-LIST-ELEM,
 RS+CHARS

PPAR

The list of plotting parameters created by various plot commands. There can be one in each dir. If an existing PPAR is deleted, the plotter rewrites it using the default settings or the user values set in LS+2D/3D and LS+WIN.

If a previous plot's PPAR causes unwanted views, it can be deleted - the plotter will then generate a default PPAR. See [Plots](#) for the NP method of resetting PPAR to the defaults. You can save copies of PPAR - see *Saving Your Plots* in the entry for Plots.

The default settings for Function are below; the commands in brackets can change the parameter(s) from the stack. Other plot types may add more elements.

```
{
(-6.5,-3.9) upper left corner
(6.5,4)    lower right corner      (49G is (-6.5,3.2),(6.5,-3.1))
```

In the above, the X values (1st no.) can be set with [XRNG](#). Y values (2nd no.) are set with [YRNG](#).

X	independent var	(INDEP)
0.	resolution	(RES)
(0.,0.)	origin	(AXES)
# n	see note below	(ATICK)
FUNCTION	plot type	(varies)
Y	dependent var	(DEPND)
}		

The number # n appears if the tick spacing has been changed. It may also be two numbers in a list if X and Y ticks are different values.

PREDV,X,Y

Predicts values using current statistics variable **ΣDAT**, which should contain a 2-column matrix of the form [[X Y]]. You can use the Matrix Writer, STAT single-variable, or Σ+ (see Sigma stats).

PREDV is the same as PREDY

PREDX: $y \rightarrow x$

PREDY: $x \rightarrow y$

Access: 96 MENU-FIT
RS-5-Fit Data-PRED
(PREDV is CAT only)

PRINT series

All commands create the variable **PRTPAR** in the Home dir. This was originally meant for use with an HP accessory printer for the calculator.

PR1 prints the object on level 1. The object is not dropped.

PRVAR prints a named variable.

'name' \rightarrow

PRST prints the contents of the stack. The objects are not dropped. PRSTC prints in compact form (one line each).

PRLCD prints the display as a grob.

CR prints buffer contents, if any; sends a newline command.

Access: 104 MENU-PRINT

Programming basics

The "native" programming language is UserRPL, which is common to the HP48/49/50 series. It's limited, but it's easy to use and can't cause any damage from crashes. There is also SysRPL, more difficult but more versatile, and assembly language (machine code). These last two are not covered in HLP49.

Programs are enclosed in double angle brackets (RS-plus key):

<< 5 6 7 + + >>

The program is then stored with your choice of name in single quotes:

'MYPROG' STO \rightarrow

The program will run when you press the MYPROG key. You can also put the program itself on the

stack with RS-MYPROG. It can then be edited by pressing the down cursor, or run with **EVAL** if desired.

To save it back again, press LS-MYPROG.

When run, anything other than operators or commands will be placed on the stack. In the above, 5, 6, and 7 go on the stack and are then added by the two + operators.

If the program delimiters are nested, the internal set is placed on the stack without evaluation:

```
<< 1 2 << 3 4 + >> >> EVAL → 1 2 << 3 4 + >>
```

EVAL can be inserted later in the program to run the secondary program at the appropriate time.

Simple Example

Programs are useful when you have to do the same calculations repeatedly. For example, if you often multiply a number by 20 times its log, you could store the program below as 'LOG20':

```
<< LOG 20 * >>
```

To enter a keyboard command while you're inside the angle brackets, just press its key.

Var Assigning

Most simple programs are made up of the same keystrokes that you'd normally use for stack calculations.

However, they become even more useful when you assign variables and use algebraic solving. (See the entry for **Assign**, and **FOR-NEXT** for another example.)

Var assignments take one of two forms:

```
<< → A B << program >> >>
```

OR

```
<< → A B 'algebraic expr' >>
```

In both cases vars A and B, which are called local variables, are cleared when the internal program or expression completes (see the entry for **LOCAL** for permanent local vars).

You can assign as many vars as you want. If there are more than a few, you can use names instead of characters (in the example below, **I** could be replaced with **current**, and so on).

Example: power= $I^2 \cdot R$. The two forms would be:

```
<< → I R << I SQ R * >> >>
```

OR

```
<< → I R 'I^2*R' >>
```

Store either with a name such as 'PWR'. Place the value of I on the stack, then the value of R.

Example: 3 amps, 2 ohms:

3 2 PWR → 18

The first version is more flexible since you can put anything into the internal program. The second version is better suited to short calculations.

If you want, you can also store the values I and R in the current dir with **STO**. These are global variables and the program can be reduced to:

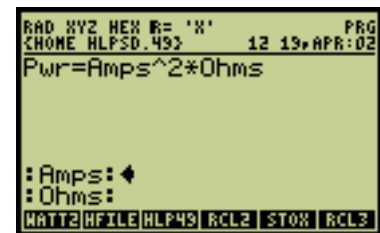
<< 'I^2*R' >>

STO the program, then press its name and **EVAL** to solve it for the stored values. The disadvantage to this simple version is having to store global variables that you'll have to delete later. Also, it's not a good idea to use a single character as a varname - there will almost certainly be future conflicts.

Getting Input

You can prompt for input in various ways. The program below shows the user the required arguments (see **INPUT** for more detail). The single-character varnames are now full names.

```
%%HP: T(3)A(D)F(.);
\<< "Pwr=Amps^2*Ohms"
{ ":Amps:
:Ohms:"
{ 1 7 } }
INPUT OBJ\-> SWAP SQ SWAP *
"Watts" \->TAG
\>>
```



Type the values and press ENTER (you can leave in extra spaces - they'll be removed), or press ON twice to exit. Some objects will be left on the stack after a cancel unless you use **IFERR** and DROP (see Error Handling below).

See **Helptext (in pgms)** for other ways of reminding the user. See also Error Handling in this entry.

The program below uses the 30 MENU softkey solver (see **SOLVING basics**). Enter the numbers and press the related keys. To solve for the unknown, press LS-var. To exit, press VAR. You can change the vars to full names if you prefer.

<< 'Watts=I^2*R' STEQ 30 MENU >>

The entered vars will be stored in the current dir.



Key Assignments

Often-used programs are most useful when assigned to a user key - see also [ASN](#).

Hint: if you store the program in the Home dir and just assign its name, you can then easily make changes without changing the keylist. Assign names like this:

```
'MYPROG' keynumber ASN →
```

OR

```
<< MYPROG >> keynumber ASN →
```

MYPROG will now run when you press LS-ALPHA and the assigned key ([USER](#) mode).

Programs are more flexible if you use decision commands like [IF-THEN](#) and loop commands such as [FOR-NEXT](#). See the related entries.

PUSH/POP

You can set and clear flags in a program without affecting the current settings. PUSH stores the flags and paths in the ENVSTACK var (in [CASDIR](#)), and POP will restore them:

```
<< PUSH <your pgm> POP >>
```

Error Handling

You can correct usage errors with the [IFERR](#) command. A simple typical use:

```
<< IFERR LOG 20 * THEN "Number" END >>
```

If this is run without input, it puts "Number" on the stack to remind you. You can also use DROP, DROP2, etc (see [Stack Commands](#)), to remove unwanted objects after a cancel if needed. See IFERR for more examples.

Debugging

If a program errors and you can't find the mistake, [DEBUG](#) (41 MENU or PRG-NXT-NXT-RUN - see its entry) lets you single-step it for tracing. Put [HALT](#) after each IFERR or DEBUG will treat the entire IFERR section as a single step. Remove the HALT when done.

Conditionals, tests

For conditional loops, see the entries for [DO-UNTIL](#), [FOR-NEXT](#), [START](#) and [WHILE](#). For tests, see [CASE](#), [IF-THEN](#), [IFT](#), and [IFTE](#).

PROMPT

Used in program to display string in header. Halts the program. [CONT](#) continues. Maximum chars with Font 8 is 21 (although there's room for 22).

```
"string" →
```

The need to use CONT is annoying. For alternatives, see [MSGBOX](#).

Access: PRG-NXT-IN-NXT

PROOT

Given polynomial in coefficient-array form, returns roots. Its inverse is **PCOEF**.

```
[coeffs] → [roots]
[ 1. 2. 1. ] → [ -1. -1. ]
```

To change coefficients to a symbolic polynomial, see **PEVAL**.

Access: ARITH-POLY-NXT-NXT,
RS+7-POLY (77 MENU)

PARSURFACE (Pr-Surface)

Sets the plot type to Pr-Surface and modifies or creates **PPAR**. Note that the command is called PARSURFACE, but the plot type is called Pr-Surface.

See also **Plots** for more on general plotting.

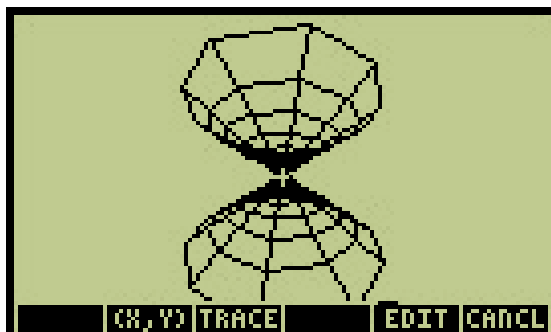
From the HP48 Series User's Guide: the Pr-Surface plot type draws an oblique-view, perspective, 3D plot of a wire-frame model of a surface determined by:

$$F(u, v) = x(u, v) i + y(u, v) j + z(u, v) k$$

where u and v are drawn from the sampling grid (XX- and YY-) ranges. Pr-Surface combines the coordinate-mapping approach of Gridmap with the 3D perspective plotting of Wireframe.

The following plots an example. Plot time is about 2 minutes.

```
%%HP: T(3)A(D)F(.);
\<<RAD { PICT PPAR VPAR } PURGE
PARSURFACE { 'X*COS(Y)' 'X*SIN(Y)
)' X } STEQ 16. NUMX 16. NUMY
-3.14159 3.14159 DUP2 DUP2 4.
DUPN XVOL YVOL ZVOL XXRNG YYRNG
0. -4.2 0. EYEPT DRAW PICTURE \>>
```



A much better discussion of plotting than in the HP49 manual is in the HP48 Series User's Guide, available for free downloading by searching www.hpcalc.org.

Access: LS+2D/3D-Type,
85 MENU

PRTPAR

Print parameters variable created in the Home dir by print commands. This was originally meant for use with an HP accessory printer for the calculator.

The format and defaults are:

```
{ 0. "" 80. "■←" }
```

0. is the delay in seconds between lines

"" is any special remapping

80. is chars per line

"■←" is the end-of line character, in this case CR-LF

PSDEV

Population standard deviation from **ΣDAT**.

```
→ n or [array]
```

Access: 96 MENU-1VAR

PURGE

Deletes a named variable or a list of named variables. Will not delete directories unless they're empty (use **FILES/FILER** or **PGDIR**).

Stack mode has no confirm and no undo. The Filer has Confirm if flag -76 is clear.

```
'name' →  
{ name1 name2 ... } →
```

If a name does not exist, there is no error message - you can use PURGE just to make sure of deletions.

You can do mass purging of vars in a dir by starting an empty list with the LS-Plus key and then pressing the softkey for each var to be purged. When done, press PURGE.

For the HP49G+/50G SD card, use

```
:3:name →  
{ :3:name1 :3:name2 ... } →
```

Purge cannot delete SD card directories made with a card reader. In the Filer, these are labelled DIR instead of HPDIR.

It can still delete vars:

```
:3:{dirname varname}
```

OR

```
:3:"dirname/varname"
```

Access: TOOL (stack mode),
LS-PRG-MEM

PUT/PUTI

Overwrites position or {row col} in list, array, or matrix. PUTI also returns index of next obj. See also GET/GETI.

For strings, see [REPL](#), [SREPL](#).

```
list,array pos obj → result (plus index if PUTI)

[ 11 22 33 44 ] 3 99 → [ 11 22 99 44 ] (and 4. if PUTI)

[matrix] {row col} obj → result (plus index if PUTI)

[[ 1 2 3 ]
 [ 4 5 6 ]
 [ 7 8 9 ]] {3 1} 99 → [[ 1 2 3 ]
                        [ 4 5 6 ]
                        [ 99 8 9 ]] (and {3. 2.} if PUTI)
```

Access: PRG-LIST-ELEM

PVAR

Population variance from statistics variable [SDAT](#).

→ n (single column)

OR

→ [array] (multiple columns)

For sample variance, see [VAR](#).

Access: 96 MENU-1VAR

PVARS

Returns list of port objects and port free memory. Works only with ports 0, 1, or 2. Note that the Home dir does not show in the port 0 list, although it's in port 0.

For port 3 (SD card), use [FILES/FILER](#).

```
port# → {list} portmem
```

An empty port returns an empty list; *eg*:

```
0 → { } 125649.
```

Access: 110 MENU

PVIEW

{#0h #0h} PVIEW in a graphics program displays **PICT** during execution. {} PVIEW at end of program displays PICT without cursor or menu. To display with them, use **PICTURE**. To display with a custom menu, see the program below.

(X,Y) or {#m #n} PVIEW displays the upper left of PICT at the specified coordinates, but PICT must be large enough to fill the display. In this case PVIEW should be followed by 0 **FREEZE**.

With older ROMs such as v1.23, the screen is 131 by 64. With newer ROMs such as v2.09, the screen is 131 by 80. The 49G is fixed at 131 by 64.

To display PICT with a custom menu, use the format in the program below. See **TMENU**.

```
%HP: T(3)A(D)F(.);
\<< { list of vars } TMENU { # 0h # 0h } PVIEW 3 FREEZE \>>
```

Access: PRG-NXT-PICT-NXT

PWRFIT

Stores PWRFIT into **ΣPAR** so that **LR** uses power curve fitting. Creates **ΣPAR** as required.

Access: 96 MENU-ΣPAR-MODL

PX→C

Pixel to coordinates - inverse of **C→PX**. Default upper left corner is {# 0d # 0d}. Bottom right is {#130d #63d} with older ROM versions and {#130d #79d} with newer ones.

$$\{ \#m \#n \} \rightarrow (x, y)$$

Note that with older ROMs, the header is separate from the graphics screen. The user area is the same as the HP48/49G.

The conversion value depends on the settings in **PPAR** - coords can be anything you want, but pixel values are absolute and based on the display size. Setting the **BASE** to DEC simplifies pixel values.

Access: PRG-NXT-PICT-NXT

→Q/→Qπ

Returns the rational form of argument (→Qπ also factors out π). See also XQ, XNUM in CAT.

→Q: .5 → '1/2'

→Qπ: 1.5707963268 → '(1/2) * π'

Access: LS-CONVERT-REWRI-NXT
93 MENU (old SYMB)

QR/qr

QR factoring of matrix:

$$[[\text{matrix}]]A \rightarrow \begin{matrix} [[\text{matrix}]]Q \\ [[\text{matrix}]]R \\ [[\text{matrix}]]P \end{matrix}$$

Q is $m \times m$ orthogonal matrix

R is $m \times n$ upper-trapezoidal m.

P is $n \times n$ permutation matrix

The QR matrices can be converted to rational form with XQ from CAT. This sets the calculator to Exact mode.

Access: MATRICES-FACT,
MTH-MATRX-FACTR

qr factoring of matrix:

$$[[\text{sq. matrix}]] \rightarrow \begin{matrix} [[\text{matrix}]]Q \\ [[\text{matrix}]]R \end{matrix}$$

Q is $m \times m$ orthogonal matrix

R is $n \times n$ triangular matrix

The outputs of qr are rational in form. They can be converted to reals if desired with $\rightarrow\text{NUM}$.

Access: MATRICES-FACT

QUAD

Solves an expression for zeros or solves an eq. Identical to SOLVE except that it follows flag -1 in returning either principal or general solutions.

`expr var → result`

`'X^2-5' X → 'X=√5' (principal)`

`'X^2-5=0' X → { 'X=√5*-1' 'X=√5' } (general)`

See also the entries for [n1](#), [n2](#) and [s1,s2](#).

Access: 93 MENU (old SYMB)

QUOTE

See also **Backquotes (left quotes)**.

1. A name for the single-quote function, ('). It allows program and variable names to be placed on the stack ("quoted") without evaluation; *eg*, if a var is named X, then entering X will recall its contents or run a program, while 'X' will be placed on Lev 1.
2. Double quotes (RS-times) denote a string of characters. Strings are not evaluated when placed on the stack. To embed double quotes in a string, precede each with a backslash, alpha-RS-5.
3. The QUOTE command returns unevaluated arguments.

'QUOTE (ABC) ' → QUOTE (ABC)

Access: 93 MENU

4. Embedded Quotes: if you write a string like this:

"This is an "embedded" quote."

it will cause a syntax error because the calculator sees it as two strings and an unknown word (embedded). To prevent this, precede each embedded quote mark with a left backslash (\, alpha-RS-5).

"This is an \"embedded\" quote."

RAND

Returns pseudo-random number between 0 and 1; updates seed.

→ n

→ .298877175677

For shorter numbers, follow with *n* **RND**, where *n* is the desired number of digits. (Formats such as FIX shorten only the display, not the number itself.)

Access: MTH-NXT-PROB

RANK

Returns rank of rectangular matrix.

[[matrix]] → n

Access: MATRICES-OPER-NXT

RANM

Creates matrix using random numbers from -9 to 9.

`{rows cols} → [[matrix]]`

`{2 3} → [[6 -7 -5]
[5 -8 6]]`

Access: MATRICES-CREATE-NXT-NXT

RATIO

Algebraic version of divide.

`x y → x/y`

Access: CAT

R→B

Real (or integer) to user binary integer. See also [Binary Integers](#).

`5. → # 5h (HEX)
5. → # 5d (DEC)
5. → # 5o (OCT)
5. → # 101b (BIN)`

Access: BASE

R→C

Two reals or integers to a complex number.

`1 2 → (1., 2.)`

For symbolics to complex, see the entry for [i](#).

Access: PRG-TYPE-NXT,
MTH-NXT-CMPLX

RCEQ

Same as 'EQ' [RCL](#) - returns contents of var [EQ](#) if it exists in the current dir. See also [STEQ](#).

`→ contents of EQ`

Returns "No current equation" if EQ doesn't exist in the current dir.

Access: CAT or 2050.42 MENU

RCI/RCIJ

RCI:

Multiplies row n of a matrix or element n of a vector by a constant x .

`matrix/vector x n → newmatrix/vector`

Access: MATRICES-CREATE-ROW

RCIJ:

Multiplies row i by a constant x and adds this to row j .

`[[matrix]] x i j → newmatrix`

Access: MATRICES-CREATE-ROW

RCL

Recalls the contents of a stored variable to the stack. See the cautions at the end of the entry.

`'varname' → contents`

If the varname is showing in the softkey menu, the contents can be recalled by pressing RS and the name key.

If the variable is in another dir higher in the tree, its contents can be recalled, but if they're stored back, they're stored in the current dir.

See also [PATH](#).

CASE SENSITIVITY

Recalls from ports 0, 1, and 2 are case-sensitive, but not from port 3 (the SD card).

PORT RECALLS

It can also recall from a port:

`:portnum:varname`

Example: `:3:MYVAR RCL` recalls 'MYVAR' from the SD card. Note that single quote marks are not required after port numbers.

RECALL FROM PORT DIR

If the var is to be recalled from an existing port dir, there are two ways:

`:portnum: {dirname varname}`

OR

`:portnum: "dirname/varname"`

WILDCARD:

The ampersand & (char 38, alpha-LS-ENTER) can be used to search all ports if you know the varname:

```
:&: MYVAR → searches ports 0-3
```

Note that it will not search inside directories.

LIBRARIES:

Because HP names can't begin with a number, libnums will have an L added - *eg*, 1758 becomes L1758. You don't need the L when recalling from ports 1 or 2, but it's required for port 3 (the SD card).

```
:3: L1758 RCL → library
```

CAUTIONS WITH RCL:

If you enter the name of a port var (such as :3:varname) or any varname from a subdir and it doesn't exist, there is no error message.

Instead, it drops the tag and searches the path. It can be misleading if it finds and recalls a var of the same name in another dir. This applies to all HP49/50s, and also the HP48.

Second, RCL drops the name of a non-existent file if LASTARG is off (flag -55 set), but not if it's on (flag -55 clear). See the entry for [LASTARG/LAST](#) or [Mode-Misc menu](#).

Access: LS-STO

RCLALARM

Returns a list of parameters for a specified alarm (numbered from 1 upwards). Alarm numbers can be obtained from RS-TIME-Browse Alarms. This command and [STOALARM](#) are meant for programming.

The alarm list is stored in the [Hidden dir](#) by STOALARM.

```
n → {date time action repeat}
```

The action is usually a reminder string to be placed on the stack or a program to be executed. If no action is specified, this will be an empty string.

Repeat is the time between repeats in ticks (see [TICKS](#)) - one tick is 1/8192 second. For example, if the repeat is once per 24hr day, the value is:

$$8192 * 24 * 60 * 60 = 707788800.$$

Access: RS+TIME-ALRM,
 95 MENU

RCLF

Recalls list of system flags (negative 1-128) and user flags (positive 1-128). The list is stored with **STOF**. The format is

```
{ # system 1-64
  # user 1-64
  # system 65-128
  # user 65-128 }
```

The binary integer format depends on the setting of **BASE**.

The flaglist can be recalled by a program to determine if the program is running on an HP48 series or the 49/50 series. The 48 list has 2 elements, the 49/50 has 4 (use SIZE on the list).

For more flag information, see the entries for **Mode-Flag menu** and **FLAGS - list of flags**.

Note that flag settings are not stored by **ARCHIVE**. You can store the RCLF list in a variable for use by **STOF** after a **RESTORE**, or you can use HP's **Conn4x** program.

If you use the Conn4x restore feature, flag settings are in a separate program ('fLaG') that will appear in the Home dir. It's self-purging.

Access: LS+MODE-FLAG-NXT,
 PRG-NXT-MODES-FLAG-NXT

RCLKEYS

Recalls the list of user key assignments. This can be stored in a variable as a backup or as a way to have multiple key assignments with **STOKEYS**.

This command can be slow if a lot of keys are assigned.

```
→ { keylist }
```

The list has the format

```
{ S function1 keynum function2 keynum ... }
```

The S indicates that unassigned keys retain their normal function in User mode.

See also **ASN**, **DELKEYS**, **Key Numbers**, **USER**.

Access: LS+MODE-KEYS,
 PRG-NXT-MODES-KEYS

RCLMENU

Returns current menu number. User menus such as VAR return 2.01 or similar; ".01" is the menu page number. See also [MENU](#).

→ nn.nn

Some menus, especially CAS menus, return only .01. See [MENUXY](#) for more on CAS menus.

Access: LS+MODE-MENU,
PRG-MODES-MENU

RCLΣ

Recalls the contents of the statistics data variable [ΣDAT](#) if it exists in the current dir.

→ [[matrix]]

Access: CAT

RCWS

Returns binary integer wordsize; default 64 bits. See also [STWS](#).

→ n

Access: BASE-NXT

R→D

Radians to degrees.

The formula is radians $\times (180/\pi)$.

Access: MTH-REAL-NXT-NXT

RDM

Redimension - changes size of vector or redimensions matrix.

[vector] {size} → new
[1 2 3 4] {2} → [1 2]

[[matrix]] {rows cols} → new
[[1 2 3] [4 5 6]] {2 2} → [[1 2] [3 4]]

Access: MATRICES-CREATE-NXT-NXT

RDZ

Specifies seed for random numbers.

$n \rightarrow$

Access: MTH-NXT-PROB

RE

Returns real part of complex argument.

$(x, y) \rightarrow x$
 $[(x, y) (a, b)] \rightarrow [x \ a]$

For the imaginary part, see **IM**.

Access: MTH-NXT-CMPLX,
CMPLX-NXT

REAL Mode

The 49/50 has two basic modes for numbers, Real and **COMPLEX**. Real is the most-used mode (the annunciator in the header is **R**).

Simple calculations with complex numbers can be done on the stack, but complex calculations using CAS commands often require Complex mode. The annunciator is **C**.

Modes can be toggled with LS+TOOL, or you can check/uncheck the Complex field in MODE-CAS.

If you're solving a symbolic expression that requires a different mode, the calc will usually switch automatically - it will ask first, and saying no will cancel the operation. If flag -123 is set, auto mode switching is prevented.

Access: MODE-CAS-Complex,
LS+TOOL

Real numbers

The 49/50 series has three basic types of numbers: reals, integers and user **Binary Integers**.

A real always has a decimal point: 5., 5.123, etc.

Integers have no decimal point (unless you enter a decimal fraction, which makes it a real).

The calc should be in Approximate mode for entering reals. (Exact's annunciator in the header is = and Approx's is ~.)

In general, reals and integers can be mixed without problems. To convert reals to integers or vice versa,

use **R→I** and **I→R** from LS-CONVERT-REWRITE. Note that reals that have decimal fractions (such as 5.123) cannot be converted to integers.

► You can also convert an integer to a real by multiplying it by 1. or any other real number.

However, there are some integer manipulations that will unexpectedly convert integers to reals. If this happens, the easiest fix is to run the XQ command (from CAT) on the results. This converts reals back to integers (assuming that there's no decimal part - fractions become rationals).

Access: MODE-CAS-Approx,
 RS+ENTER

RECN

Receives a file from a Kermit source and stores it with the name on Level 1. Similar to **RECV**, but send and receive names can be different.

```
'name' → file stored in 'name'
```

Access: 104 MENU

Recovery

See also **ON+ Codes** for info on multiple-key combinations. See also **Reset** for more detail about ON+C.

Stack Errors:

Press **ANS** to retrieve the arguments or press UNDO to replace the stack with a previous copy.

You can also retrieve certain arguments from **CMD**.

All three must be enabled with LS+MODE-MENU - see the entry for **Mode-Misc menu**.

Overwrites:

If you act before running any other command, you can correct for accidental use of **STO** by pressing **ANS**.

Assume that VAR1 containing "Original" was overwritten with "Overwrite":

```
ANS → "Original" 'VAR1'  
STO ANS → "Overwrite" 'VAR1'
```

Drop the unneeded 'VAR1' name and you're back where you began.

You may also retrieve objects from the CMD buffer. Note that ARG and CMD must be enabled in LS+MODE-MISC.

Repeating Alarm:

It's possible to specify an alarm repeat time short enough that you can't cancel it. Press and hold ON and then press 9 to break out of the loop.

Low Memory:

If you get low memory errors (eg, when trying to save a file), press LS+MODE-MISC and press STK, ARG and CMD to remove the white square showing that they're active. This frees at least some memory - reset them when done.

Another way is an ON+C warmstart, which clears all the temporary buffers. See [Reset](#).

You can also try using the Filer to move unneeded files from the Home dir to another port.

Worst case: if you *really* need lots of memory, make sure you have a backup (see [ARCHIVE](#)), and then press ON+A+F-NO to clear the Port 0/Home memory. Restore the backup when done.

Program Freezes:

If pulsing ON repeatedly doesn't work, ON+C will terminate the program and reboot. The stack and undo buffers will be cleared. If this doesn't work, see below.

Lockup:

Sometimes the calc will lock up and ignore the keyboard. Try a mechanical reset: insert a paperclip into the small hole in the back and press for several seconds, then try ON (you might have to do this multiple times).

It may return to normal, or it may likely reboot or say Try to Recovery Memory? Press Yes - see Memory Clear below.

It may restart in Algebraic mode (use MODE to return it to RPN) and require a RESTORE (keep a backup - see [ARCHIVE](#)). Be sure to change back to RPN before restoring.

(Contrary to a statement in an HP manual, holding the reset contact for an extra length of time has no effect on the type of reset.)

Note: after crashes, you'll need to check:

1. Time and date
2. Alarms, esp. dates
3. Flags (keep an RCLF backup)
4. User keys (keep a RCLKEYS backup)
5. Contrast setting (it's often too dark - hold ON and press the + or – keys).

Reboots Repeatedly:

This is usually caused by a defective library. Do an ON+C reboot and then quickly hold down the Backspace key. This prevents the libraries from attaching -delete the offending library with the Filer. This is a matter of timing - you might have to try it a number of times. See [Reset](#).

Memory Clear:

A last resort, since it clears port 0 (though not the others). Press ON+A+F (3 keys at the same time). It will say Try to Recover Memory? - press Yes. Chances are not good; parts of your data will likely be in dirs called D.01, D.02, etc. Doing frequent backups using [ARCHIVE](#) is essential.

If you answer NO, port 0 (and the Home dir) will be cleared. Flags will be default settings (algebraic mode, etc).

OS Failure:

It's possible, if very rare, for a program fault to corrupt the operating system, which then has to be replaced. Follow the instructions about reflashing the system for your particular model. Full instructions are available at hpcalc.org.

The 49G+/50G can be reflashed using the SD card if it was formatted to FAT16 (called FAT in Windows); see [Reflashing ROM](#).

RECT

Sets rectangular angular mode. The annunciator is XYZ. For polar mode, use either the MODE window (where it's called Polar) or the [CYLIN](#) command. (The [POLAR](#) command is a plot type.) See also [SPHERE](#).

Values are entered as a vector (see [Arrays/vectors](#) for more about vector entry): $[x \angle y]$ or $[x \ y]$. In the case of $[x \angle y]$, it will be converted to $[x \ y]$ notation if rectangular mode is set.

► The angle symbol is alpha-RS-6 (chr 128).

If you're using the MODE-Coord System field, the +/- key will step through the choices faster than the CHOOS box.

The program below toggles between rectangular and polar mode and is more convenient than the many keystrokes required by MODE. Store it as RP or similar, and/or assign it to a key (see [ASN](#)):

```
%%HP: T(3)A(D)F(.);  
\<<-16 FS? IF THEN -16 CF  
ELSE -16 SF END \>>
```

$[3. \ 4.] \leftrightarrow [5. \angle .92729521]$

If spherical mode is set, this program toggles between rectangular and spherical.

Access: MODE-Coord System,
 LS+MODE-ANGLE,
 PRG-NXT-MODES-ANGLE,
 LS-MTH-VECTR-NXT,
 65 MENU

Rect/Polar

Note that the polar conversion command is called **CYLIN**, but the MODE menu uses Polar (the actual **POLAR** command is for plotting).

MODE-Coord-System or flag -16 can be used to change modes. Values are entered as a vector:

$$[3 \ 4] \leftrightarrow [5. \angle 53.1301]$$

Vectors are entered in one of several ways:

$$\begin{aligned} x \ y \rightarrow V2 &\rightarrow [x \angle y] \\ [\] \ x \ By \ ENTER &\rightarrow [x \angle y] \end{aligned}$$

Where x,y are numbers and the angle symbol is alpha-RS-6. The above are in polar mode with flag -19 clear. (The second example works only when typed, not pasted in.)

To automate vectors with a key assignment (and other useful utilities), see **RECT** and **Arrays/vectors**.

Access: MODE menu

RECV

Receives a file sent by a device using Kermit, such as a computer or calculator. The file name is not required at the receiving end. To receive and rename a Kermit file, see **REC�**. For Xmodem, see **XRECV**.

Access: APPS-I/O-Transfer,
 104 MENU

REF

Reduces a matrix to row echelon form.

$$[[matrix1]] \rightarrow [[matrix2]]$$

Access: MATRICES-LIN S

Reflashing ROM

Note that an OS update can be lengthy and uses high-current pulses to write to flash memory. Be sure the batteries are in good condition (the HP50G is powered from the USB cable).

HP49G+/50G

When new ROM editions become available, the easiest way to reflash the ROM is to use the Conn4x computer program (see the entry for **Conn4x basics**). The ROM download contains an extra file holding only the version number (update.scp). This should be in the same dir as the ROM file.

Note that the ROM menu item is only active if you start Conn4x with "Connect using USB calculator". Click on ROM and follow the onscreen instructions.

OLDER HP49G

The early HP49G did not have the Xmodem server built in, so Conn4x can't be used. In this case, use the older HPComm program on the PC. This program, newer ROMs, and reflashing information are available from www.hpcalc.org.

SD CARD

You can also reflash from the SD card if the card has been formatted to FAT16 (not FAT32). Cards formatted with the Filer will be FAT16 for cards smaller than 32K and FAT32 for those above. For large cards, use a cardreader and a computer (FAT16 is just called FAT in Windows).

Be sure both the ROM file and update.scp are in the root of the card (*ie*, not in a folder).

1. With the SD card out, press and hold the plus and minus keys.
2. Press the Reset with a paperclip or similar.
3. Release the Reset, wait 2 seconds, then release the plus and minus keys. The update screen should appear.
4. Insert the SD card with the ROM and .scp files.
5. Press 1 (Update ROM code).
6. Press 2 (Update from SD).
7. When the update is finished, press the Reset.

RENAME

Renames a variable or directory in the Home dir and subdirs. *It will not rename objects stored in a port.*

```
'newname' 'oldname' →
```

If 'newname' already exists, the error message is "Name Conflict".

The Filer (FILES key) is easier to use, but it also only works in the Home dir.

There is no easy way to rename an object in a port, other than to recall it, store it with the new name, and purge the old. See [RCL](#), [PURGE](#), [SD card basics](#).

► Objects on an SD card cannot be renamed with this command or the [FILER](#).

Access: CAT

REPL

Note that REPL is available from many different menus - they're all the same command.

See also **SUB**, **SREPL** for strings. See also **PUT/PUTI** for a list, array, or matrix. See also **AUGMENT** in **CAT**.

1. Replaces part of a matrix with a new one, starting at {row col}.

```
[matrix] {row,col} [new] → [newmatrix]
[[ 7 -8 0 ]
 [ 2 -9 6 ]
 [ 0 1 6 ]] { 2 1 } [ 11 22 33 ] → [[7 -8 0]
                                     [11 22 33]
                                     [0 1 6]]
```

Access: MATRICES-CREATE-NXT-NXT,
MTH-MATRX-MAKE-NXT

2. Replaces element(s) in a list, starting at the position on Lev2.

```
{list} position element/list →
{ 1 2 3 4 } 2 { 5 6 } → {1 5 6 4}
```

Access: PRG-LIST,
PRG-NXT-CHARS,
RS+CHARS

3. In graphics mode, inserts a grob from stack Level 1 with upper left at cursor.

Access: EDIT-NXT-NXT

4. Replaces text in string with character or substrings starting at the position on Level 2.

```
string pos char/string →
"ABCDE" 2 "XX" → "AXXDE"
```

Access: PRG-LIST,
PRG-NXT-CHARS,
RS+CHARS

5. In stack graphics, superimposes Level 1 grob into Level 3 grob with upper left corner at pixel/coord location on Level 2. Level 1 grob overwrites Level 3 pixels.

```
grob1 {#m #n} grob2 → grob3
```

OR

```
grob1 (x,y) grob2 → grob3
```

Access: PRG-NXT-GROB

RES

Sets plot resolution; default in LS+WIN-Step (which is the same thing) is 0, which is a point every two pixels. Can use either binary hex integers or reals - the table below gives the number of plotted points.

n →
OR
n →

You can also type the value into LS+WIN-Step. To return to the default, enter a zero.

If you check Pixels in the LS+WIN-Step display, you can enter the resolution directly in pixels.

BINARY	REAL	POINTS
# 4h	.4	every 4th pixel
# 3h	.3	3rd
# 2h	.2	2nd (default)
# 1h	1	every pixel
# 0h	4	4th
# 2h	0.	2nd (default)

Note that #2h and 0 are the same thing - #2h speeds up the HP48, with its default of one point per pixel, but has no effect on the HP49/50, which uses a default of every second pixel. #3h or .3 give better speed without too much loss of image quality.

► .05 or smaller can be used to close gaps in certain plots (but it will be much slower).

Access: LS+WIN-Step
81 MENU-PPAR

Reset

See also [Recovery](#) for other methods. More about ON+C is below.

1. A rogue program or glitch can sometimes make the calculator freeze. Try an ON+C warmstart first. If the keyboard is locked out, you can do a system restart by inserting a straightened paperclip or pin into the small hole in the back. This will cancel any running program and reload the OS. It usually does not affect memory unless it was corrupted to begin with.
2. The RESET key from MODE-NXT resets the flags to their default state (see [FLAGS - list of flags](#)).
3. Many of the built-in solvers have RESET in the softkey menu (press NXT if necessary). This allows you to reset any or all of the field values.

ON+C Warmstart

Also called reboot, reset, restart. To run it, press and hold ON and then press and release the C key. The screen will blank, a splash screen will go by too fast to read ("Powered by Metakernel"), and you'll be in the Home dir with the Tool menu showing.

► It has no effect on stored memory and is safe to use. (It does clear the stack and some buffers.)

It does the following:

1. Cancels any program and reloads the operating system commands - sometimes they're corrupted by a program or glitch.
This resets system defaults such as fonts, binary wordsize, etc (flag settings are not changed).
2. Runs the program in **STARTUP** if it exists. This sets defaults to your preferences.
3. Attaches all libraries and completes the installation of new ones. Libnames now appear in the RS-LIB softkey menu.
4. Clears buffers such as Undo, ANS (LastArg), Find/Replace, etc. Clears the stack.
5. Cancels a program that freezes. If this doesn't work, see **Recovery**.

When to use it:

- After installing a new library
- If commands fail for no reason
- To reset conditions as listed in STARTUP
- If a program freezes or won't stop

Starting Without Libraries

If a lib causes startup problems, such as repeated reboots, do an ON+C - as soon as you release them, press and hold Backspace during the reboot. It's a matter of key timing. Libs will not be attached (nothing in RS-LIB).

Remove the offending lib with :portnum: libnum **PURGE** or use the **FILER**. Finally, restore RS-LIB with a normal ON+C.

RESTORE

Overwrites Home dir and subdirs with backup object created by **ARCHIVE**; causes warmstart (reboot). Note that RPN mode is required - if the calculator is in algebraic, use MODE to return it to RPN.

:portnumber: name →

eg: :2: BAK49

Note that the name does not require quotes. Only the right-hand colon appears on the stack, although both appear in edit mode.

If you use this in a program, the warmstart means that RESTORE must be the last command in the program.

Note that flag settings are not archived by the calculator and the flaglist from **RCLF** (see **Mode-Flag menu**) should be stored in a variable before **ARCHIVE** is used; flags can later be restored with **STOF**. The user keylist and alarms are automatically stored in the backup object by **ARCHIVE**.

The **Conn4x** program for the PC has both backup and restore, but it's not compatible with backup objects made with **ARCHIVE** (you can get your files back, but they'll be in a dir named \$\$\$t and there's no flag restore).

If you use the Conn4x restore feature, flag settings are in a separate program ('fLaG') that will appear in the Home dir. It's self-purging.

If you're transferring the backup object from an external source, use binary Kermit or Xmodem. The backup object should be stored in a port, such as 2. **RESTORE** can then be run.

:2: backupname

Note that if you have a daily alarm set, **RESTORE** will run it and advance the date of the next alarm. If necessary, use **RS-TIME** to correct the alarm(s).

Access: PRG-MEM-NXT

REVLIST

Reverses list order.

{1 2 3} → {3 2 1}
{A B C} → {C B A}

Access: PRG-LIST-PROC

R→I

Real to integer conversion.

5. → 5
5.1 → error
'X' → 'R→I (X) '

XQ from **CAT** will also convert reals to integers. Decimal fractions become rationals. It will also work with entire expressions and equations.

Access: CONVERT-REWRITE-NXT-NXT

RKF series

RKF

Finds solution to IVP using Runge-Kutta-Fehlberg (4,5) method.

{list} Xtol Xtfinal → {list} Xtol

RKFERR

Finds abs. error estimate for step h for IVP.

$$\{\text{list}\} \ h \rightarrow \{\text{list}\} \ h \ Y_{\text{delta}} \ \text{err}$$
RKFSTEP

Finds next solution step (hnext) for IVP.

$$\{\text{list}\} \ X_{\text{tol}} \ h \rightarrow \{\text{list}\} \ X_{\text{tol}} \ h_{\text{next}}$$
RRK

Finds solution to IVP for DE with known partial derivatives.

$$\{\text{list}\} \ X_{\text{tol}} \ X_{\text{tfinal}} \rightarrow \{\text{list}\} \ X_{\text{tol}}$$
RRKSTEP

Finds next solution step for an IVP and displays method used.

$$\{\text{list}\} \ X_{\text{tol}} \ h \ \text{last} \rightarrow \{\text{list}\} \ X_{\text{tol}} \ h_{\text{next}} \ \text{current}$$
RSBERR

Returns error estimate for step h for IVP.

$$\{\text{list}\} \ h \rightarrow \{\text{list}\} \ h \ Y_{\text{delta}} \ \text{err}$$

Access: RS+7-DIFFE (76 MENU)

RL/RLB

Rotate bit/byte left.

$$\begin{aligned} \# \ 101011h &\rightarrow \#0202022h \\ \# \ Ah\# &\rightarrow \ 14h \end{aligned}$$

Access: BASE-NXT-BIT/BYTE

RND

Rounds number to n places and removes extra digits.

To remove digits without rounding, use [TRNC](#).

To round while retaining internal accuracy, use n FIX (see [Mode-FMT menu](#)).

$$\text{real} \ n \rightarrow \text{real to } n \text{ places}$$

Access: MTH-REAL-NXT-NXT

RNRM

Row norm (infinity norm) of an array.

`[array] → n`

Access: MATRICES-OPER-NXT

ROMUPLOAD

(HP49G only) Transfers the 49g calculator's operating system to another HP49G. Connect the calculators and press ROMUPLOAD on the source HP49. Follow the onscreen instructions.

Access: CAT

ROOT

Returns value for expression that mostly nearly approximates 0. Stores result in variable 'var'. See also [PROOT](#).

`expr 'var' guess → result, also stored in 'var'`

`'X^2-5' 'X' 2 → 2.2360679775` plus this result stored in 'X'

The root solver can be steered to other roots by using a positive and then a negative guess. For instance:

`'X^2-5' 'X' -2 → -2.2360679775`

Access: RS+NUM.SLV-ROOT,
75 MENU

→ROW

Transforms matrix into vectors; returns row count. See also ROW→ below.

`[[matrix]] → vectors count`

`[[[1 2]
[3 4]] → [1 2] [3 4] 2`

Access: MATRICES-CREATE-ROW

ROW→

Transforms vectors and row count into matrix; inverse of →ROW above.

```

vectors  count  →  [[matrix]]

[ 1 2 ] [ 3 4 ] 2  →  [[ 1 2 ]
                      [ 3 4 ]]

```

Access: MATRICES-CREATE-ROW

ROW+/-

Row+ inserts array into matrix at row *n*.

```

[[matrix]] [array] n  →  [[newmatrix]]

```

Row- deletes matrix row *n* or vector element *n*; returns deleted obj.

```

matrix/vector n  →  newobj  obj

```

Access: MATRICES-CREATE-ROW

rpm

A typing aid for user-defined units.

```

      →  1_/min
550_ →  550_rpm

```

Access: 117.02 MENU,
 APPS-EqnLib-UTILS

RR/RRB

Rotate bit/byte right

```

# 101011h#  →  80000000000080808h
# A00hh     →  Ah

```

Access: BASE-NXT-BIT/BYTE

RREF series

rref reduces matrix to row-reduced echelon form plus pivot points.

RREF reduces matrix to row-reduced echelon form.

`[[matrix]] → result`

RREFMOD reduces matrix to row-reduced echelon form modulo the current modulo (CAT only).

`[[matrix]] → result`

Access: MATRICES-LIN S

RSBERR

Error estimate at step h when solving an IVP DE. See also **RKF series**.

`{list} h → {list} h y err`

Access: 76 MENU

RSD

Residual $B-AZ$ of matrices B , A , and Z .

`[[B]] [[A]] [[Z]] → [[B-AZ]]`

Access: MATRICES-OPER-NXT,
RS+7-SYS (78 MENU)

RSWP

Swaps rows i and j of matrix.

`[[matrix]] i j → [[newmatrix]]`

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow{2 \ 3} \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \end{bmatrix}$$

Access: MATRICES-CREAT-ROW-NXT

s1,s2

In results from the **ISOL** and **QUAD** commands, the variable s1 represents an arbitrary + or - sign. Additional arbitrary signs are represented by s2, s3, etc.

If flag -1 is set, then ISOL and QUAD return principal solutions, in which case the arbitrary sign is always +. See also **n1**, **n2**.

SAME

Tests if two objects (any type) are identical.

```
obj1 obj2 → 0/1
```

SAME and == (two equals) are not the same thing:

```
'X=5+5' 'X=10' SAME → 0.
'X=5+5' 'X=10' == → '1=1'
```

That is, the expressions are not the same, but they're equal algebraically.

Access: PRG-TEST-NXT

SBRK

Interrupts Kermit serial transmission or reception.

Access: 104 MENU-SERIAL

SCALE series

These are the programmable versions of some of the plotter's zoom functions. A SCALE of 8 and 8, for example, multiplies both X and Y by 8.

SCALE adjusts first two **PPAR** objects to control hor. and vert. scale.

```
Xscale Yscale →
```

SCALEH adjusts height (vertical or Y).

```
Yscale →
```

SCALEW adjusts width (horizontal or X).

```
Xscale →
```

Access: 83 MENU-PPAR

SCATTER/SCATR

SCATTER:

Sets the plot type to Scatter and modifies or creates **PPAR**, **ΣPAR**. **ΣDAT** should contain a 2-column matrix of ordered pairs (col 1 is X; col 2 is Y). See also **LR**, **SCLΣ**.

See also **Plots/PLOT** for more on general plotting.

SCATRLOT:

Plots the data in **ΣDAT** as a scatterplot and returns to the stack. Press the left cursor to see it.

To plot an example, store the following matrix in **ΣDAT** (you can use **STOΣ** from CAT):

```
[[ 7.6 3.2 ]  
 [ 2.7 2.2 ]  
 [ .7 .9 ]  
 [ -1.3 .3 ]  
 [ -3.5 -.5 ]  
 [ -6. -1.4 ]]
```

You can then run SCATRLOT. To see the plot, press the left cursor. To redraw with axes, select them in LS+2D/3D and press DRAW.

If you press STATL in the graphic mode, it draws the regression line, stores its eqn in **EQ**, and resets the plot type to Function.

Full Scatter Method:

Store the matrix as above. Press LS+2D/3D and select Scatter as the Type. Optional: turn off Axes.

Press LS+WIN and press Auto, Erase, and Draw. (The Auto key modifies **PPAR** so that the plot fills the screen.)

Optional: after plotting, press STATL to draw the linear regression line. The line's equation will be in EQ and the plot type will be Function.

STAT-Fit_data (RS-5) will also give the equation, plus the correlation and covariance. The **ΣLINE** command (96 MENU FIT) returns the equation as well.

A much better discussion of plotting than in the HP49 manual is in the HP48 Series User's Guide, available for freedownloading by searching www.hpcalc.org.

Access: LS+2D/3D-Type,
 88 MENU,
 (SCATRLOT is CAT only)

SCHUR

Schur decomposition of a square matrix.

$$[[matrix]]A \rightarrow \begin{matrix} [[matrix]]Q \\ [[matrix]]T \end{matrix}$$

If A is complex, Q is unitary and T is upper-triangular.

If A is real, Q is orthogonal and T is upper quasi-triangular.

In both cases, A is equal to (or approximately equal to) $Q \times T \times \text{TRN}(Q)$.

Decimal fractions can be converted to rational form with XQ from CAT.

Access: MATRICES-FACT,
MTH-MATRX-FACTR

SCLΣ

Adjusts (xmin,ymin) and (xmax,ymax) in **PPAR** so scatterplots fill **PICT**. The statistics variable **ΣDAT** must exist. No input is required.

The Auto command in the Plot Window (LS+WIN) does the same thing.

Access: CAT

SCONJ

Conjugates complex number contents of a var.

$$'name' \rightarrow$$

If 'test1' contains (1.,2.), SCONJ changes it to (1.,-2.). This does not appear on the stack.

Access: PRG-MEM-ARITH-NXT

SCROLL

Full-screen scrollable display of the object on Lev 1. Maximum text width is 33 char; 9 lines display vertically. Grobs in Text mode are displayed as one continuous line of ASCII.

$$obj \rightarrow$$

► Note that SCROLL drops the obj after displaying it. To retain the object, either duplicate it first with ENTER or press UNDO immediately after exiting SCROLL.

Text to be displayed can be one long line, or you can insert linebreaks (RS-period) as desired.

Embedded double quotes should be preceded by a backslash (\, alpha-RS-5). Styles from the **TOOL menu** work only in SCROLL's Graphic mode - in text mode the style codes show as black squares.

Access: CAT

SD card basics

Valid up to OS v2.09. See also **MEMORY basics** and **FILES/FILER**.

The SD card is flash memory that occupies port 3. HP is not clear on the maximum size, but 512MB cards will work. Large cards tend to slow down the calc turn-on time, as do large numbers of files.

FILE STRUCTURE:

The file structure is based on DOS. Filenames should be no more than 8 characters plus a 3-character extension. Longer filenames will work but will be truncated.

► Filenames are not case-sensitive, unlike the other ports.

The Filer can be very slow at displaying the SD card listings if you have a lot of files. You can speed it up considerably by using a cardreader to move files into a few folders.

PROGRAMS:

The card is mainly for storage, but you can run programs with EVAL:

```
:3: pgmname EVAL →
```

DIRS, HPDIRS

► Dirs made with a cardreader cannot be deleted on the calculator.

On the SD card, the Filer can look into dirs made with a cardreader (labelled DIR) if you press the right cursor, but cannot open the calc's own dirs (labelled HPDIR).

LIBRARY NAMES:

Libraries stored by their numbers in ports 0-2 will have an L added because HP filenames can't begin with a number. For example, library 1010 will become L1010.

STO, RCL, PURGE:

The syntax for storing, recalling or purging is

```
:portnum: {dirname varname}
```

eg, :3: {HLPD H16}. If there is no dir, it can be :3: H16.

Another format for accessing dirs is

```
:portnum: "dirname/varname"
```

such as :3: "HLPD/H02".

► Dirs made with a cardreader cannot be purged by the calculator.

All files are stored in binary form. To keep text files readable on a computer, use ASCII transfers (see [TRANSFERRING](#)).

Strings are usually readable on the PC, but calculator symbols (such as chars 128-255) will not display correctly (see the entry for [Translations](#)).

RENAME:

The Filer's RENAME and the [RENAME](#) command don't work with port objects. A cardreader is required for renaming files on the SD card.

FORMATTING:

Cards can be formatted on the calc with the Filer. Cards smaller than 32K will be FAT16; others will be FAT32. If in doubt, cards can be formatted on the PC (FAT16 is just called FAT in Windows).

SD cards used for reflashing the ROM should be formatted to FAT16. See the entry for [Reflashing ROM](#) for details.

SDEV

Returns sample standard deviation using [SDAT](#), which must exist in the current dir.

```
→ n or [array]
```

Access: 96 MENU-1VAR

SEND

Sends a named file using Kermit to another Kermit device.

```
'name' →
```

For Xmodem, use [XSEND](#).

Access: APPS-I/O-Transfer,
104 MENU

SEQ

Repeatedly executes using **var** between **start** and **end** in steps of **incr**.

```
obj var start end incr → {list}
```

Works well for multiple solving of symbolics. Example: solve X^2+X for values of X from 2 to 5:

```
'X^2+X' 'X' 2 5 1 → { 6 12 20 30 }
```

Access: PRG-LIST-PROC-NXT

SERVER

Starts Kermit server mode (CAT or typed in).

From keypad:

Kermit:	RS+right cursor 104 MENU SRVR APPS-I/O-Start server
Xmodem:	RS-right cursor, XSERV (CAT)

See also [Conn4x basics](#), [TRANSFERRING](#).

Shortcut keys

► (Shifthold) The + sign means hold shift while pressing the other key.

LS+MODE	Mode softkeys
LS+TOOL	Real/Complex toggle
LS+NXT	Previous menu
LS+VAR	Home dir
LS+Down	<i>See note below</i>
RS+ENTER	Exact/Approx toggle
RS+9	Time softkeys
RS+right	Kermit server
RS+SPC	Semicolon
RS+CAT	(49G) CHARS softkeys
RS+EVAL	(G+/50G) CHARS softkeys
RS+'	Left quotes (` chr 96) (see Backquotes entry)
RS+7	Root/Diff/Poly/Sys/TVM softkey menu

LS+Down is similar to Down (which is **EDITB**), but can act as **VISITB** (from the HP48SX command set) in that it edits the contents of a named var rather than the varname:

	<u>Down</u>	<u>LS+Down</u>
name	edits name	edits content
obj	edits obj	edits obj

Holding LS while pressing keys A-F gives six plotting functions; these have their own entries.

See also [ON+ Codes](#), [Recovery](#).

SHOW

1. Implicit references to the variable 'name' are made explicit.

symb 'name' →

If 'Y' contains 'Z=5', then 'X+Y=10' 'Z' → 'X+(Z=5)=10'

2. Prevents specified vars from being evaluated.

symb {varlist} → evaluation except listed vars

Assume 'A' contains 'X=1'

'B' contains 'Y=2'

'C' contains 'Z=3'

Then 'A+B+C' {A} → 'A+Y+Z=A+2+3'

Access: 93 MENU (old SYMB)

SIDENS

Returns the intrinsic density of silicon versus temp Xt.

Xt → Xdensity

Access: 117 MENU,
APPS-EqnLib-UTILS

Sigma (Σ)

(Not to be confused with SIGMA or SIGMAVX, which are CAS commands. For stats commands $\Sigma+$, $\Sigma-$, etc, see [Sigma stats](#).)

Summation function. Easiest to use in the Equation Writer, but the stack syntax is:

```
var start end function
```

eg: the sum of the squares of the numbers from 1 to 10:

Stack:

```
'X' 1 10 'X^2' → 385
```

Eq Writer:

(▶ is rt curs, ^ is Y^X key)

```
 $\Sigma$  X ▶ 1 ▶ 10 ▶ X ^ 2 Enter →NUM → 385
```

ASCII algebraic:

```
' $\Sigma(X=1,10,X^2)$ ' →NUM or EVAL → 385
```

If flags -2 and -3 are clear, it can be used in symbolic mode:

```
' $\Sigma(A=1,X,A^2)$ ' EVAL → '(2*X^3+3*X^2+X)/6'
```

The result will give the sum of the squares from 1 to X when solved for the desired value of X. (See the entry for [WHERE \(|\)](#).)

Access: RS-SIN

SigmaDAT (Σ DAT)

The variable that holds the matrix (one or two columns) required by the stats functions. There are 3 ways to create it:

1. Press RS-STAT and then any of the first four options. Press LS-MTRW to open the [Matrix Writer](#). Enter the data (one or two columns). Press ENTER twice to exit. The matrix will be in Σ DAT.

OR

2. Enter the data using the Matrix Writer and store the matrix with $\text{STO}\Sigma$ from CAT.

OR

3. Use $\Sigma+$ from CAT or 96 MENU DATA to enter values - it creates Σ DAT in the current dir (to start a 2-col matrix, make the first entry an array:

[X Y] (following pairs can be on the stack). Use $\Sigma-$ to correct wrong entries.

ΣLINE

Returns the eqn of the regression line based on the data in **ΣDAT**, which must exist in the current dir. See **LR**, **Sigma stats**, **SCATTER/SCATR**.

→ eqn

Access: 96-MENU-FIT

SigmaPAR (ΣPAR)

ΣPAR is the variable that holds stats plot data. It's created by stat plot commands. The default is:

```
{ 1. 2. 0. 0. LINFIT }
```

The format of ΣPAR is:

```
{ independent column
  dependent column
  y-intercept
  slope
  type of LR fit }
```

To change the default columns, see **COLΣ**, or **XCOL** and **YCOL**. To change the type of regression line, see the entries for **LINFIT**, **LOGFIT**, **EXPFIT**, **PWRFIT**, or **BESTFIT**.

For the regression line, see **LR** or **SCATTER/SCATR**.

Sigma stats

The following are for stack Σ processing of statistics rather than the dialog boxes of RS-STAT.

Σ+ adds element(s) to the matrix **ΣDAT**, which it creates. For a 2-col matrix, *make the first entry an array*, [X Y]. The following entries can then be on the stack:

```
X Y → stored in ΣDAT
```

Σ- removes last Σ+ entry from ΣDAT, puts it on the stack - used for checking or correcting entries.

Access: 96 MENU-DATA

ΣX sums X column

ΣY sums Y column

ΣX2 sum of squares in X

ΣY2 sum of squares in Y

ΣXY sums products of X,Y pairs

NΣ number of rows in ΣDAT

Access: 96-MENU-SUMS,
STAT-SUMMARY

ΣLINE returns the linear regression line equation in the form 'mX + b' from the data in ΣDAT, where m = slope and b = y-intercept.

Access: 96-MENU-FIT

SIGN

1. Returns sign of number using ± 1 .

-5 → -1.
5.1 → 1.

2. SIGN(f(X)) is the signum function - it outputs 1 or -1 as a periodic function changes sign. Plot 'SIGN(SIN(X))' or similar as a function to see the effect.

Be sure flag -119 is clear (Rigorous on). If Rigorous is off, SIGN may not return negative values, especially when plotting.

Access: MTH-REAL-NXT,
CMPLX-NXT

SINV

Replaces content of var with its inverse. Var may contain numerics or symbolics (if symbolics, flags -2 and -3 should be clear).

'name' →

Access: PRG-MEM-ARITH-NXT

SIZE

For the size in bytes, see [BYTES](#).

SIZE returns:

- number of chars in a string
- number of elements in a list or array
- matrix dimensions {row col}
- width (Lev2) and height (Lev1) of a grob in pixels
- number of digits in an integer number (reals return 1.).
- number of elements in an algebraic object

With algebraic objects, operators are included, so that X^2+1 would return 5.

obj → n

Access: PRG-NXT-CHARS,
PRG-LIST-ELEM,
RS+CHARS

SL/SLB

Shift bit/byte left.

```
# 101010h → # 202020h
# Ah → # A00h
```

Access: BASE-NXT-BIT/BYTE

SLOPEFIELD

Sets the plot type to Slopefield and modifies or creates **PPAR** and **VPAR**.

See also **Plots** for more on general plotting.

The Slopefield plot draws a series of line segments whose slopes represent the slope of $F(X,Y)$ at their midpoints. It's plotting segments of the anti-derivative of the entered function so you can see the family of curves resulting from integration. Single-var $F(X)$ will also work.

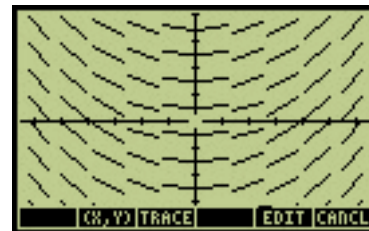
Note that it does not use all the elements of PPAR, so changes should be made using LS+2D/3D first to set Slopefield and then LS+WIN to set view values.

The following simple example shows the family of segment curves resulting from plotting the slopes of 'X', which has the antiderivative ' $X^2/2+C$ ', where C is the arbitrary constant. Different values of C result in the curve set.

A possible drawback to Slopefield is that it sets the four view values to +/-1 in LS-WIN. The example edits LS+WIN to allow a larger view.

If a previous plot has changed PPAR, delete it first to ensure default values.

- Press LS+2D/3D and change the plot type to SLOPEFIELD and \angle to Rad.
- Highlight EQ, type X as the variable. ENTER sends it to EQ.
- Press LS+WIN and enter
H-left: -2 H-right: 2
V-left: -2 V-right: 2
- Press ERASE and DRAW.



The result is segments showing a family of parabolic curves as determined by the Step setting in LS+WIN (try changing the steps to 15 and 12 for more curves).

If you want to see an exact curve, press LS+2D/3D and change the Type to FUNCTION. Change EQ's 'X' to its antiderivative ' $X^2/2$ '.

Without pressing Erase, press DRAW.

Some functions, such as e^{-X^2} , do not have a closed-form integral. However, a slopefield plot shows that it can be handled with numeric methods.

The example below uses F(X,Y) to show an ellipse's anti-derivative set.

```
%%HP: T(3)A(D)F(.);  
\<<-1. CF 'VPAR' PURGE  
  '1.2*X^2+1.5*Y^2-.55' STEQ  
  ERASE SLOPEFIELD DRAX DRAW  
  PICTURE  
\>>
```

Access: LS+2D/3D-Type,
82 MENU

SNEG

Negates contents of var. Var may contain numerics or symbolics. Flag -3 should be clear for symbolics to avoid "Undefined name" errors.

'name' →

Example: if 'X^2' is in 'var1', then 'var1' SNEG makes it '-X^2'

Access: PRG-MEM-ARITH-NXT

SNRM

Spectral norm of array - the root of the sum of the squares of the elements.

[array] → n

Access: MATRICES-OPER-NXT-NXT

SOLVEQN

Extracts and solves any equation from the [EQLIB Equation Library](#) using solvers similar to the Multiple Equation Solver (MES).

The Equation Library was dropped with the HP49G and reintroduced as separate libraries (226 and 227), starting with ROM v2.09, Build 92.

SOLVEQN requires 3 numbered arguments: the main Subject (such as Electricity), the Title (such as Ohm's Law and Power) and the Picture number. If the Picture# is 1, then any pic is stored in PICT (view it with the left cursor key). If Picture# is zero, the pic is ignored (use zero if there's none).

Equations are stored in [EQ](#).

Level3: Subject#
Level2: Title#
Level1: Picture# (0/1)

Units are controlled by user flags 60 and 61 (positive numbers):

60s: English units
60C: SI units
61s: no units
61C: units



The result of 2 3 1 SOLVEQN.

If you use the same equation a second time and want to change units, be sure to delete any existing EqLib variables - once created, these determine the units regardless of flag settings.

A table of the EqLib numbers is available on p. 5-2 of the HP49G+ Advanced User Reference, or you can count down manually.

As of Aug/09, SOLVEQN is not in the HP50G manuals. The two 49G+ PDF manuals can be downloaded from the HP site or hpcalc.org.

Example: the Voltage Divider from Electricity. Electricity is the 2nd subject and the divider is the 3rd title. There is a pic.

2 3 1 → runs solver, stores pic in PICT, stores eqn(s) in EQ

Press the left cursor to see the pic. The solver works the same way as the MES: enter a value and press the associated softkey. Press LS-softkey of any unknown to solve for that value. VAR exits.

Note: be sure there are no stored vars with the same name as the EqLib vars in the path, or errors such as Bad Guess(es) or Inconsistent Units can result.

Access: APPS-Equation_Lib-EQLIB
114 MENU

SOLVING basics

In solving, expressions are always equated to zero: 'X-5' is the same as 'X-5=0'.

VX:

If you solve for the same var repeatedly, you can store it in the VX var in the Home dir using **STOVX**. Your var will be used by CAS commands such as SOLVEVX and INTVX. The default var is X.

You can also use compound vars, such as 'Xt'.

Contrary to some of the manuals, the var VX is in the Home dir, not **CASDIR**.

STACK SOLVING:

Uses commands such as SOLVE, SOLVEVX, QUAD, ISOL, etc. The equation or expression is placed on the stack with the desired var and solving command, such as:

'X^3-27' X SOLVE → 'X=3'

OR

'X^3-27' SOLVEVX → 'X=3'

The calc will switch to required modes, such as symbolic. It should be in Real mode for non-complex inputs.

If solving returns "Undefined Name", clear flags -2,-3 (*ie*, remove checkmarks) for symbolic instead of numeric mode, or use MODE-CAS-Numeric and uncheck this field.

With any type of solving, results may not be in the desired form. SIMPLIFY (CAT) will be useful, as will **COLLECT**, **EXPAN/EXPAND**, and FACTOR from RS-ALG.

EXCO:

HP's EXCO is a simple program for reducing and simplifying:

```
%%HP: T(3)A(D)F(.);
\<<
DO DUP EXPAN
UNTIL DUP ROT SAME
END
DO DUP COLCT
UNTIL DUP ROT SAME
END
\>>
```

SOLVING COMMANDS:

For some of the many solving commands, see **ISOL**, **MATCH**, **QUAD**, **ROOT**, **SHOW**, **WHERE (|)** and **ZEROS**. See also RS+NUM.SLV below.

RS-NUM.SLV:

A number of useful solver types. NUM.SLV has its own entry with more detail.

Some solver dialog boxes require an expression or equation. You can entire it directly into the field, or you can store it in **EQ** first.

Coefficient arrays should be entered into the field, or you can have them on the stack and press NXT-CALC-OK. There may be a syntax error message that you can ignore.

You can also cursor to the input field(s) and press LS-MTRW for the Matrix Writer. When you exit this, the array will be in the solver's selected field.

If you need to calculate, press NXT CALC to return to the stack. Press OK to return.

INTEGRALS:

See **Integral (∫)** and **EQW-INTEGRALS**. See also INTVX, INT, and RISCH in CAT.

See also the DESOLVE section of [Derivative \(\$\partial\$ \)](#). DESOLVE is a powerful integrator for solving ODEs. It has the added advantage of inserting the arbitrary constant.

DERIVATIVES, ODEs:

See [Derivative \(\$\partial\$ \)](#), [DIFFEQ](#), [EQW](#).

FINANCIAL:

For more on financial, see the entry for [FINANCE Solver](#).

MULTIPLE-EQUATION SOLVER:

A powerful solver for any number of equations or vars. EQ must contain a list with a minimum of two equations. MINIT must be run first to initialize it. For details, see the entry for [MSOLVR](#).

CUSTOM SINGLE-EQUATION SOLVER:

30 MENU starts a simple solver for one equation, which must be stored manually in [EQ](#). The program below demonstrates Ohm's Law.

```
<< 'V=IR' STEQ 30 MENU >>
```

Enter the two known values and press the associated varkeys. To solve for the third, press LS-varkey.

To exit, press VAR. The stored vars must be deleted manually.

CLEAN SOLVER:

In the upgraded version below, adapted from a program written by Peter Karp of [comp.sys.hp48](#), "CLR" appears in the softkeys. This will purge all the variables when pressed and the program will exit.

```
eqn  →  result

%%HP: T(3)A(D)F(.);
\<<
  IFERR PUSH -105. SF LNAME AXL
  { { "CLR"
    \<< RCEQ EVAL DROP LNAME AXL
  PURGE 0. MENU DROP -105. CF POP
    \>> } } + 2. \->LIST STEQ 30. MENU
  THEN POP "Equation"
  END
\>>
```

Operation is the same as 30 MENU (see CUSTOM SINGLE-EQUATION SOLVER, above), except that you can purge leftover vars with CLR.

UNITS:

Units can sometimes cause problems in solvers. If so, remove them with **UVAL**.

SAME-NAME VARS:

Variables with the same name as the **EQ** vars anywhere in the path can cause error messages. If so, change to unique names or do deletions.

SORT

Sorts list in ascending order. Cannot mix numbers and alphabetic.

```
{ 3 2 1 } → { 1 2 3 }
{ C B A } → { A B C }
{"C" "B" "A"} → {"A" "B" "C" }
```

For descending order, follow with **REVLIST** (LS-MTH-LIST). Useful for alphabetic ordering of dirs or similar - see **ORDER**.

Access: PRG-LIST-PROC-NXT

SPHERE

Sets spherical angular mode. Annunciator is **R**↻↻. See also **CYLIN**, **RECT**.

Note that if you're using the MODE-Coord System field, the +/- key will step through the choices faster than the CHOOS box.

Entering a 3D vector is easiest using →**V3** (in LS-MTH-VECTR):

```
5 10 6 → [ 5 ↻10 ↻6 ]
```

The above is in DEG mode. You can also enter it directly from the command line (the angle symbol is alpha-RS-6, chr 128).

If you enter a spherical vector while another angular mode is set, values will be converted according to the mode in use.

If spherical mode is set, the program below toggles between rectangular and spherical mode and is more convenient than the many keystrokes required by the 49/50. (If polar mode is set, it will toggle between rectangular and polar.)

Store it as RP or similar, and/or assign it to a key (see **ANS**):

```
%%HP: T(3)A(D)F(.);
\<< -16 FS? IF THEN -16 CF ELSE -16 SF END \>>
```

Access: MODE-Coord System,
LS+MODE-ANGLE,
PRG-MODES-ANGLE,

LS-MTH-VECTR-NXT,
65 MENU

SR/SRB

Shift bit/byte right.

```
# 101011h → # 80808h
# Ah → # 0h
```

Access: BASE-NXT-BIT/BYTE

SRAD

Spectral radius of square matrix.

```
[[matrix]] → n
```

Access: MATRICES-OPER-NXT-NXT

SRECV

Receives up to n chars from the serial input buffer with error digit.

```
n → "string" 0/1
```

Timeout is 10 seconds, or it can be set by STIME:

```
n →
```

where n is a real from 0 to 24.5. If $n=0$, it waits indefinitely.

Access: 104 MENU-SERIAL

SREPL

Finds and replaces substring in text and returns number of occurrences replaced, including 0 for none.

```
textstr findstr replstr → newstr n
                        OR textstr 0
```

```
"ABC DE FGH"
  "DE F"
  "XY Z" → "ABC XY ZGH" 1.
```

See also [REPL](#), [SUB](#).

Access: PRG-NXT-CHARS-NXT,
RS+CHARS-NXT

SREV

Reverses string order.

"abc" → "cba"

Access: 256 MENU-NXT-NXT

Stack Commands

TOOL-STACK or PGM-STACK

Shortcut keys are in brackets.

DUP	duplicates Lev 1 onto Lev 2 (ENTER)
SWAP	interchanges Lev 1, Lev 2 (Right-curs)
DROP	deletes Lev 1 (Backspace)
OVER	dupes Lev 2 to Lev 1
ROT	Lev 3 to Lev 1
UNROT	Lev 1 to Lev 3
n ROLL	Lev n to Lev 1
n ROLLD	Lev 1 to n
n PICK	Lev n duped to Lev 1
n UNPICK	replaces n+1 with Lev 1
PICK3	dupes Lev 3 to Lev 1
DEPTH	# of objects on stack. Doesn't drop any stack objects
DUP2	dupes Lev 1, Lev 2
n DUPN	dupes n levels
DROP2	deletes Lev 1, Lev 2
n DROPN	deletes n levels
DUPDUP	dupes Lev 1 twice
NIP	same as SWAP DROP
n NDUPN	dupes Lev 1 n times; returns n

INTERACTIVE STACK (Up-curs)

(Note that some of the listed commands are not available with earlier ROM versions.)

BACKSPACE	deletes selected level
ECHO	select level, press Echo, then ENTER or ON to copy to the command line
VIEW	displays selected level using SCROLL
EDIT	edits selected object with text editor
PICK	copies selected lev to Lev1
ROLL	selected level to Lev1
ROLLD	Lev1 to selected level
→LIST	list from Lev 1 to selection
DUPN	dupes selected level and everything below it
DROPN	drops selected level and everything below it
KEEP	deletes all above selection
GOTO	prompts for level, jumps
INFO	size and CRC
LEVEL	copies the number of the level to Lev 1

To undo changes: press ON to exit, then UNDO.

COPYING TO CLIPBOARD

Press the up-arrow for the interactive stack, choose the desired level, and press RS-COPY. Does not append - the copy overwrites. ON or ENTER exits.

START

Definite loop like **FOR-NEXT**, but with no index counter, which makes it faster for the same procedure.

```
start end START <proc> NEXT
```

OR

```
start end START <proc> stepsize STEP
```

The numerical value for STEP can be replaced with the name of a stored value, providing external control.

Typing aids: when the START dir is displayed (LS-PRG-BRCH), these are available:

```
LS-START  → START  
           NEXT
```

```
RS-START  → START  
           STEP
```

The following do-nothing program can be used for delays, since the **WAIT** command sometimes fails in programming - see [Help text](#) for an example. The 800 value gives a delay of about one second on the 49G+/50G.

```
<< 1 800 START NEXT >>
```

Access: PRG-BRCH

STARTED

If this variable contains a program in the Home dir, it will automatically configure the edit mode as desired whenever the editor is opened. A typical usage:

```
<< 0 →HEADER -73 SF >>
```

This removes the header (49G) or blanks it (49G+/50G) and selects the minifont. To undo these changes on exit, see [EXITED](#).

To start the editor with an empty stack, see [NOT](#).

Access: type in

STARTEQW

If this variable exists in the Home dir, it can run a program after the Equation Writer opens when the user presses [CUSTOM](#). This allows commands to be run on highlighted parts of an expression.

Example: store the following program as 'STARTEQW' in the Home dir. Open the EQW with any expression. Highlight any section and press LS-CUSTOM. A Choose box will open with various commands (your choice - the ones shown in the list are only examples).

```
%%HP: T(3)A(D)F(.);  
\<< ""  
  { SHOW SIMPLIFY COLLECT DERVX }  
  1 CHOOSE  
  IF  
  THEN EVAL  
  END  
\>>
```



Access: type in

STARTERR

If this variable contains a program in the Home dir, it will run after any error occurs.

The following STARTERR changes error messages from messageboxes to the header (HP48 style):

```
<< 2 DISP 1 DISP 7 FREEZE >>
```

To display for a fixed number of seconds, replace 7 **FREEZE** with s **WAIT**, where s = seconds.

► Some error messages, such as "Interrupted", remain in messagebox form.

Anything in this variable other than a program will suppress error messages.

Access: type in

STARTOFF

To change the time until auto-shutoff, see **TOFF**.

If the STARTOFF var contains a program in the Home or current dir, it will be evaluated after auto-off (see TOFF). If turnoff is desired, it must contain the OFF command.

The implementation varies from one ROM version to another. Use only with patience...

Access: type in

STARTUP

If this variable contains a program in the Home dir, it will configure the system as desired after an ON+C warmstart (see **Reset**), or it can be run manually. A typical usage:

```
<< 250 →KEYTIME MYFONT →FONT >>
```

This sets the key delay to 250. It then recalls a custom font from MYFONT and makes it the current font. Any other commands can be entered as desired - these are only examples. If STARTUP doesn't exist, default values are used.

Access: type in

STAT

For statistics tests in softkey format, see the entries for **Sigma stats** and **ΣPAR, ΣDAT**.

Single-variable

Entering values:

Values are stored in ΣDAT. If it exists, you can edit or continue. If not, highlight the ΣDAT field and press EDIT or LS-MTRW to open the **Matrix Writer**.

Enter the values as a single vertical column. You can have multiple columns, but only one can be processed at a time.

When done, press ENTER to store the values in ΣDAT and return to STAT.

Choose the column (for multiple columns), then Sample or Population. Checkmark the desired parameter fields and press OK. Tagged values are placed on the stack. It will also create ΣPAR , ΣDAT in the current dir.

Frequencies

If necessary, see Single Variable above for creating ΣDAT .

Select the column of ΣDAT to be divided into bins (classes) according to size. The params and defaults are number of bins (13), bin width (1), and minimum X value (-6.5).

Output is two arrays. Level 2 has the number of values that fall into each bin, and Level 1 has values that fall outside this range.

Fit Data

If necessary, see Single Variable above for creating ΣDAT .

ΣDAT should contain data in 2 columns. The values in Col1 become X values for prediction or plotting and the values in Col2 become Y.

Predicting: press PRED and enter any X value to predict Y or vice versa; press PRED again.

Plotting: exit STAT and press LS+2D/3D. Change the plot Type to Scatter. Press LS+WIN and press AUTO, ERASE (optional), and DRAW. (x,y) points will be plotted from values in cols 1 and 2 of ΣDAT .

To draw the regression line for the above points, press STATL in the graphics editor menu. The default fit is linear. For others, see the entry for [LR](#).

To plot columns other than 1 and 2, see [COL \$\Sigma\$](#) .

Summary Stats

If necessary, see Single Variable above for creating ΣDAT .

ΣDAT should have two columns. Checkmark the desired parameters, which are:

ΣX	sum of X col
ΣY	sum of Y col
ΣX^2	sum of squares of X
ΣY^2	sum of squares of Y
ΣXY	sum of the XY products
$\text{N}\Sigma$	number of entries

Values are tagged and placed on the stack. It will also create ΣDAT in the current dir.

Hypothesis Tests

Confidence Intervals

Six tests are available for each. Change defaults as required - definitions of each parameter appear at the bottom of the window as you change fields. The Help key provides both help text and examples. Output can be toggled between text and graph modes; pressing OK places tagged values on the stack.

Access: RS-STAT

STEQ

Stores an object (usually an expression, an equation, or a list of them) in the variable **EQ** in the current dir; creates EQ if it doesn't exist. Same as 'EQ' STO.

EQ is a reserved variable used by the plotter and solvers. There can be a different EQ in each dir. See also **RCEQ**.

obj → obj stored in EQ

Access: CAT or 2050.42 MENU

STO

Global vars

The STO key stores an object with your choice of name in a global variable in the current dir or in a specified port. Names are case-sensitive (except for the SD card) and should not start with a number (see below). The name appears in the softkey menu.

obj 'name' →

Local vars

In a program, the STO cmd stores a value in a local var (one created inside the program and cleared on exit - see **LOCAL**). Local names do not appear in the menus. Example:

<< → x << program using x >> >>

The above creates var x, which is used by the program. The x var is cleared when the program ends.

Limitations

The name must be in single quotes ('varname').

You can't begin a name with a number, since the HP49/50 then thinks it's an algebraic (for a way around this, see S~N in the entry for **256 MENU**).

The name can't include any math operators ('X^2' is an algebraic, not a name).

The object and name must be different - you can't store 'X' in 'X'.

Using Leftshift

If a stored object is on Level 1, it can be stored back into its varname by pressing LS, then the name softkey.

```
obj LS-varkey = obj 'varname' STO
```

Storing in a Port

To store an object in a port, use the **FILER** to copy it if it's already a stored var, or put it on the stack, enter

```
:portnum: name
```

Example: :2: BAK49 and press STO. Names after a portnum don't need quote marks. Note that names on the SD card are not case sensitive.

If the object already exists in the port, the error message is "Object in use". The Filer will overwrite it or you can first purge the original:

```
:portnum: name PURGE →
```

If the var is to be stored in an existing port dir, there are two ways:

```
:portnum: {dirname varname}
```

OR

```
:portnum: "dirname/varname"
```

Libraries

If the object is a library, it provides its own filename, so you can store it by just entering the portnum without the colon - for example, for port 2:

```
library 2 →
```

If the library is already stored in a port, use the Filer to copy or move it. See also **LIBRARY basics**.

Because HP names can't begin with a number, library numbers will have an L added - *eg*, 1758 becomes L1758. You don't need the L for commands with ports 1 and 2, but it's required for port 3 (SD card).

Undoing a STO

To undo an accidental STO, assuming that you haven't entered any other commands, see **ANS** or **Recovery**.

Low Battery

Writing to flash memory (port 2) or the SD card (port 3) requires high-current pulses. If the low-batt indicator is on or about to come on, you will probably be limited to saving to the Home dir (port 0) or SRAM (port 1). If you can't save at all, power down and replace the batteries (see Batteries). Your stack data should be safe - a capacitor protects the stack memory for one or two minutes while batteries are out. If the calculator is turned off, the backup battery will protect it. See also **Batteries**.

Access: STO key

STOALARM

Stores the list of alarms (from RCLALARM) in the hidden dir. For a description of the list, see [RCLALARM](#).

{list} →

Access: RS+TIME-ALRM,
95 MENU

STO arithmetic

(Note that there is no RCL arithmetic as there was on some previous HP calculators.)

STO+ STO- STO* STO/

These perform the operation on the var contents named on Lev 1, using the Lev 2 obj.

obj 'name' →

Example: 55 'myvar' STO+ →

This adds 55 to the value stored in 'myvar', which should contain a number or algebraic.

Access: PRG-MEM-ARITH

STOF

Stores flaglist in memory (not in a variable) and resets the flags to these values. Use [RCLF](#) (see [Mode-Flag menu](#)) to obtain the current list (see RCLF for the list structure).

Note that flag settings are not stored by [ARCHIVE](#). You can store the RCLF list in a variable for use by STOF after a [RESTORE](#), or you can use HP's [Conn4x](#) program.

If you use the Conn4x restore feature, flag settings are in a separate program ('fLaG') that will appear in the Home dir. It's self-purging after it resets the flags.

{list of settings} →

Access: LS+MODE-FLAG-NXT,
PRG-NXT-MODES-FLAG-NXT

STOKEYS

Stores list of assigned keys, format {S obj key# obj key# ...}, in the hidden dir. See also [ASN](#), [RCLKEYS](#) (which provides the list) and [Key Numbers](#). The user keylist is automatically stored in the backup object by [ARCHIVE](#).

{ S list} →

The S in the list means that unassigned keys retain their normal function.

► If you turn on user keys with flag -62 and unassigned keys are locked out, be sure to assign
 << -62 CF >> to a key or you'll be stuck in user mode (although ON+C will get you out).

It's also a good idea to save the keylist from RCLKEYS in a variable just in case.

Access: PRG-NXT-MODES-KEYS,
 LS+MODE-KEYS

STOΣ

Stores object (the matrix with stats data) in the statistics var **ΣDAT**. Note that this overwrites any existing ΣDAT. Storing is usually done by the stat functions, but this command can be used for manual editing.

obj →

Note that STOΣ will store any object - it's up to the user to ensure that the proper matrix is on Level 1.

Access: CAT

STOVX

STOVX stores one or more characters in the VX variable in the Home dir and displays it (them) in the header. This becomes the default var for commands such as SOLVEVX or INTVX. The default is 'X'.

Contrary to some of the manuals, the var VX is in the Home dir, not **CASDIR**.

Example:

'Xt' → 'Xt' stored in VX

CASCFG restores this to 'X'. Note that the DESOLVE cmd will overwrite VX with one of the variables that it's using.

Access: CAT

STREAM

The first two elements of a list are placed on the stack and obj is executed. The next element is placed with the result and obj executed again. This continues to the end of the list.

{list} obj → result

The obj is meant to be a program that takes two arguments and returns one result.

```
{A B C D} << + >> → A+B+C+D
{1 2 3 4 5} << + >> → 15
```

If the obj takes only one argument, the first element is unchanged:

```
{2 3 4} << SQ >> → 2 9 16
```

Access: PRG-LIST-PROC

STR series

STR→ removes quotes from string and evaluates result. Same as **OBJ**→. Included for older programs and available only from CAT.

```
"ABC" → 'ABC'
```

Access: CAT

→**STR** encloses obj in double quotes to make its type a string.

```
'2*X' → "'2*X'"
```

Access: PRG-TYPE,
PRG-NXT-CHARS-NXT

STWS

Sets binary wordsize in bits from 1 to 64; default is 64.

```
n →
```

Access: BASE-NXT

SUB

1. Returns part of string, array, or list:

```
obj start end → newobj
```

```
"ABCDEF" 3 5 → "CDE"
[1 2 3 4 5] 3 5 → [3 4 5]
{A B C D E} 3 5 → {C D E}
```

2. Returns section of matrix using start/end of opposite corners.

```
[[matrix]] start end → [[newmatrix]]
```

```
[[ 5 4 8 ]
 [ 9 6 0 ]
 [ 3 3 1 ]] 4 9 → [[ 9 6 0 ]
                  [ 3 3 1 ]]
```

3. In graphics mode, saves (to stack) grob with opposite corners at mark and cursor.

4. In stack graphics, returns section of grob on Lev 3 with opposite corners on Lev 2 and Lev 1 (can be pixels or coords).

```
grob {m1 n1} {m2 n2} → newgrob
```

OR

```
grob (x1,y1) (x2,y2) → newgrob
```

Access: PRG-LIST,
PRG-NXT-GROB,
RS+CHARS

SVD/SVL

Singular value decomposition, singular values of a matrix.

```
SVD:      [[matrix]]a → [[matrix]]u
                        [[matrix]]v
                        [[matrix]]s
```

```
SVL:      [[matrix]] → [array]
```

Access: MATRICES-FACT

SYSEVAL

A binary integer with SYSEVAL calls routines at that address:

```
# 3714Ah → displays hidden dir (HP49/50 only)
```

Since a wrong integer can corrupt memory, doublecheck before using SYSEVAL. The integers are not the same for the 48 and 49 - cross-references are available from the Entry Points section of www.hpcalc.org.

Access: CAT

%T

Percentage of total on Level 2 represented by Level 1.

`x y → 100y/x`

`50 5 → 10. (Flag -3 set)`

`50 5 → '1/10*100' (Flag -3 clr)`

Access: MTH-REAL

TABLE

Displays calculated values of X and Y from the last plot, or allows user-supplied values for X (Build Your Own mode - see [TBLSET](#)). The variable [EQ](#) must exist in the current dir.

The TABLE feature is not well documented and appears to work only with single-variable expressions (eg, Conic plots produce an error message). The DEFN key supplies X/Y labels for the selected values, or the current expression.

The table is look-only; there doesn't seem to be a way to export the values (except to enter them manually in the Build mode).

Access: LS+F6.

→TAG

Attaches a label to an object.

`obj "string" → :string:obj`

OR

`obj 'name' → :name:obj`

Only the right-hand colon appears on the stack; both appear in the editor.

When tagging a port number onto a varname, the colons are not required inside the string:

`"3" 'varname' → 3:varname`

Tags are removed by DTAG and most operations, including [STO](#). One way to preserve a tag is to make the object into a string with PGM-TYPE→STR. Remove the string afterwards with [OBJ→](#).

An empty tag can be used in programming to put a name or command on the stack without evaluating it:

`::pgmname or ::command`

This method saves a few bytes compared to quotes or a list. See also [EVAL](#).

Access: PRG-TYPE

TAIL

Returns all but the first element of a list or string. See also [HEAD](#).

```
obj → newobj
{ a b c d } → { b c d }
```

Access: PRG-LIST-ELEM-NXT,
PRG-NXT-CHARS-NXT,
RS+CHARS-NXT

TakeOver

If you want assigned keys to have their functions appear in the text editor, the key assignment must have the SysRPL command TakeOver added to it.

The easiest way is to use one of many toolkit programs from hpcalc.org. Keyman is a good one - its →TO? command adds TakeOver to any assignment on the stack. For example:

```
<< DATE TIME TSTR MSGBOX >> →
```

This displays the time and date when the assigned key is pressed. There is no visible difference when you run →TO?, but if you set flag -85 for SysRPL, you'll see:

```
:: TakeOver x<< xDATE xTIME xTSTR xMSGBOX x>> ;
```

Assign this to a key and it will insert a message box with the time and date into the editor.

Note that TakeOver does not work in items such as MTRW or EQW.

TAYLR

Taylor approximation of expr at zero. For non-zero, use SERIES with <expr 'var=n' order>.

```
expr var order → newexpr
'SIN(X)' 'X' 3 → '-1/6*X^3+X'
```

The TAYLOR0 cmd from CAT gives a fourth-order approximation at zero:

```
'SIN(X)' → '1/120*X^5+-1/6*X^3+X'
SIMPLIFY → '(X^5-20*X^3+120*X)/120'
```

For non-zero, see SERIES in CAT.

Access: CALC-LIMIT

TBLSET

The table setup screen. Allows entering start, step, zoom, etc. Calculated values of X and Y for the next plot can be viewed with the **TABLE** command.

The TABLE feature is not well documented and appears to work only with single-variable expressions (eg, Conic plots produce an error message).

For Type: Build Your Own, the TABLE screen allows entry of user-supplied X values; Y values are automatically calculated and stored as an array in the variable TAB (or your choice of name).

Settings in TBLSET are stored in a list in the variable TPAR.

Access: LS+F5

TDELTA

Temperature change - can mix any of the temp units. Result takes the unit type of the level 2 object.

$$x \ y \rightarrow y-x$$

$$'50_^{\circ}\text{C}' \ '10_^{\circ}\text{F}' \rightarrow 62.2222_^{\circ}\text{C}$$

Access: CAT,
117.02 MENU,
APPS-EqnLib-UTILS

TETRIS

There are two Tetris games, at least in OS versions 1.23 and later. Earlier versions might only have the first:

Type "HpMad" (including the quote marks) in any new input form command line (such as the BEEP item in the MODE window). The cursors control left-right and rotation. ON+C exits - on the HP49G+/50G, you might have to try ON+C repeatedly.

Note: the HpMad version doesn't seem to work reliably on the 49/50 series with modern ROM versions.

The other Tetris (versions later than version 1.18 only): in the **Equation Writer (EQW)**, type MINEISBETTER and highlight it with the right arrow, then press SIMP. The 4 and 6 keys are left/right and 5 is rotation. The 2 key steps the falling object a little faster. The Backspace key exits.

TEXT

Displays the stack during program execution. The demo program below (thanks to John H Meyers) will switch three times between the image in **PICT** and the stack. (If there's nothing in PICT, all you'll see is a blank screen.)

```
%%HP: T(3)A(D)F(.);
\<< 1 3 START 1 WAIT
    { # 0h # 0h } PVIEW
    1 WAIT TEXT NEXT \>>
```

Access: PRG-NXT-OUT

Text Editors

There are two almost identical text editors.

(Note that there is a text editor listed when you press APPS. This is actually the command line for entry of new material, although it does certain editing - unless you press a command key, which will be acted on directly in this mode.)

EDITB is called by the down cursor when there's a string, a list, program, algebraic, on the stack. It does not display codes such as the backslash or the string's double quotes. (For other objects, such as a matrix or grob, it loads the appropriate viewer for these.)

EDIT is from LS-down cursor, and it will load whatever's on the stack into the ASCII plain-text editor. It displays backslashes, quotes, etc.

The EDIT from the TOOL menu is actually EDITB.

For more information, see the **TOOL/EDIT menu**, **EDIT/EDITB** entries. Note that the TOOL menu's editor commands appear only when you're in the editor. See also **STARTED**, **EXITED**, **NOT**.

Text to Grob

You can insert text strings or algebraic expressions into a graphic with →GROB. (GROB = graphics object.)

```
obj  n  →  grob
```

where n=**1** for the minifont or **2** for the current font.

With the new grob on the stack, load the main graphic into the graphics editor and put the cursor where you want the upper left of the added grob from Lev1. Press EDIT-NXT-NXT-REPL.

It's not an precise system and some guesswork about placement is needed.

For more detail, see the entry for →**GROB**.

Access: PRG-NXT-GROB

TICK

1. One tick of the real-time clock is 1/8192 sec. See [TICKS](#).
2. Another name for the single- quote key, RS-EQW (49G) or the ' key (49G+/50G), also spelled "tic". It allows names of vars or algebraic expressions to be placed on the stack without evaluation.

For example, if the dir has the stored program MYPROG, the following keystrokes place the name on the stack:

```
' <MYPROG key> ENTER → 'MYPROG'
```

3. The name for the marks along the X and Y axes of a plot. For changing the spacing, see [ATICK](#).

TICKS

Returns system time as a binary integer - 8192 ticks/sec.

Example:

```
→ # 1D7BB75D4BB73h
```

Use the **B→R** command in [BASE](#) to convert this to a real.

For elapsed time in seconds:

TICKS and then TICKS SWAP - B→R 8192 /

Access: RS-TIME-TOOLS,
RS+TIME,
LS-PRG-NXT-NXT-TIME

TIME/→TIME

TIME returns time in 24hr hh.mmss format. Not affected by flag -41 (12/24 hr).

```
→ hh.mmss
```

→TIME sets system time using 24hr format (not affected by flag -41 (12/24 hr):

```
hh.mmss →
```

Access: RS-TIME-TOOLS,
RS+TIME,
LS-PRG-NXT-NXT-TIME

TINC

Temperature increment - can use any temp units. Result uses the unit type on level 2. Note in the example that 5_°F is an increment in degrees, not a temperature.

```
initial delta → initial ±delta
```

```
'20_°C' '5_°F' → '22.7777_°C'
```

Access: CAT,
117.02 MENU,
APPS-EqnLib-UTILS

TLINE

(Toggle line) Draws a line in **PICT** using **XOR** (coincident pixels are turned off). From stack:

```
(x1,y1) (x2,y2) →
```

OR

```
{#m1 #n1} {#m2 #n2} →
```

In graphics mode, draws a tline from a mark (times key) to the cursor. Coincident pixels are turned off. To keep coincident pixels on, use **LINE**.

Access: PRG-NXT-PICT
Graphics-EDIT

TMENU

Displays temporary softkey menu - built-in, user-defined, or library. The menu stays in place until another menu is selected. See also **CUSTOM** for more about menus. See also **WAIT** for displaying any menu in a program.

```
n or list →
```

Useful Menu Numbers:

0	Last menu
1	CST menu
2	Same as pressing VAR
93	Old HP48 SYMB menu
145, 1792	All branching program commands

To display **PICT** with a custom menu, use the format in the program below:

```
%%HP: T(3)A(D)F(.);  
\<< { list of vars } TMENU  
{ # 0h # 0h } PVIEW 3 FREEZE \>>
```

Access: PRG-NXT-MODES-MENU,
LS+MODE-MENU

TOFF

If the variable 'TOFF' exists in the Home (or current) dir and contains clock ticks as a binary integer, it controls the time (with no keystrokes) until auto-shutoff. The default is 5 minutes.

There are 8192 ticks/second. The stored value should be converted to a binary integer.

Example: For 1 minute, $8192 * 60 = 491520$, or as a binary integer using **R→B (BASE menu)** #491520d, or #78000h in hex. Store it as 'TOFF'. (See the BASE entry for more on binaries.)

The following simple program takes the number of minutes until off and writes it to TOFF as a binary integer. With no input, it puts a reminder on the stack.

```
%%HP: T(3)A(D)F(.);
\<<
  IFERR #78000h * 'TOFF' STO
  THEN DROP "Min. to Off"
  END \>>
```

Example for 9 minutes: 9 → #438000h stored in TOFF The value will differ in different bases.

The default auto-off without TOFF is 5 minutes. TOFF will not begin timing until any running user program completes. For more control with auto-off, see **STARTOFF**.

Access: type in

TOOL/EDIT menu

► Note that the TOOL key has two sets of commands, one for stack mode and another for the editor.

STACK MODE:

EDIT - this is actually EDITB, same as the down-cursor

VIEW - uses SCROLL to display stack objects

STACK - see **Stack Commands**

RCL - 'name' → obj recalled to stack

PURGE - 'name' → obj deleted

CLEAR - clears entire stack

CASCMD - two CAS help programs **HELP** that seem to be identical, though **CASCMD** will accept a string as a search key:

"TAN" → help for TAN2CS2 (Available only in stack mode.)

EDIT/EDITB MODE:**Cursor keys:**

► RS+ means holding the shift key while pressing the other key. Note that there is no undo for deletions.

RS-left	to left of line
RS-right	to right of line
LS-left	to left of screen
LS-right	to right of screen
RS-up	to top of file
RS-down	to end of file
LS-up	to top of screen
LS-down	to bottom of screen

Edit Menu Keys:

←SKIP	skip word left
RS+←SKIP	skip to left of line
SKIP→	skip word right
RS+SKIP→	skip to right of line
←DEL	delete word left
RS+←DEL	delete line left of cursor
DEL→	delete word right
RS+DEL→	delete line to right of cursor
DEL L	delete line
INS	insert mode on/off (insert is on when a white square is in the softkey label)

SEARCH/REPL menu:

FIND	find string
REPL	find and replace string - to delete, leave the Repl field empty
NEXT	next occurrence
R	replace, wait
R/N	replace, go to next
ALL	replace all
FAST REPLACE	(Choose boxes only, flag -117 clear) replace all while freezing display
IFIND	incremental search (finds characters as they're entered)

GOTO menu:

(Use INFO to find line number, cursor position, etc.)

GOTOLINE	jumps to desired line
GOTOPOS	jumps to desired position (characters from beginning)
LABEL	jumps to line starting with asterisk (*, times key). A scroll window displays any number of choices and can do alphabetic searches, same as CAT.
EDIT:	edits highlighted choice using EDITB
→ BEG	to beginning of highlight
→ END	to end of highlight

Stack:

See entry for [Stack Commands](#).

EXEC:	Runs highlighted command, deletes command name
HALT:	Suspends edit, displays stack - CONT resumes edit

STYLE menu:

Changes highlighted text to various styles. To remove the style, highlight and press the same style twice. Codes are embedded in the text - they show as black squares in the Graph mode of SCROLL, but don't show in the other editors.

You can see the codes if you do an ASCII (text) transfer to a PC. See [TRANSFERRING](#).

The FONT key has no apparent purpose since it won't change anything.

INFO:

Displays page of statistics relating to the current file:

#lines:	total in file
Xposition:	chars from left
Yposition:	chars from top
Position:	total chars from beginning
Line size:	chars per line
Text size:	size of file, bytes
Stk size:	size of file if on stack
Mem(KB):	total free memory (Home dir)
Clip size:	size of clipboard
Sel. size:	size of object in clipboard
SLOW:	Toggles slow/fast scrolling

TOT

Sum of column(s) in statistics variable **ΣDAT**.

→ n or [array]

Access: 96 MENU-1VAR

TPAR

Variable for settings of the Table feature, created by **TBLSET**. The default is:

{ 0. .1 0. 4. 0. TAB }

These are Start, Step, Auto, Zoom, small font, and the filename used by the Build feature.

TRACE

1. Stack: returns trace of square matrix.

[[matrix]] → n

Access: MTH-MATRX-NORM-NXT
MATRICES-OPER-NXT-NXT

2. Graphics menu key: l-r cursor moves along function; up-dn cursors select other functions, if any.

TRAN

Transpose of a matrix. See also **TRN**, which is the same but adds conjugation of complex numbers.

[[matrix]] → [[newmatrix]]

```

[[ 11 12 13 ]
 [ 14 15 16 ]
 [ 17 18 19 ]]
      →  [[ 11 14 17 ]
          [ 12 15 18 ]
          [ 13 16 19 ]]

```

Access: MTH-MATRX-NORM-NXT,
MATRICES-OPER-NXT-NXT

TRANSFERRING

See [Translations](#) for ASCII/Symbol translations. See also [Conn4x basics](#).

The 49G has only a serial port. The 49G+ has USB and infrared (IR). The 50G has all three.

IO METHOD

This is set by flags:

- 33s: Transfer via IR port
- 33C: Transfer via wire

Leave cleared on HP49G.

- 78s: HP50G only - Serial I/O if flag -34 is set
- 78C: HP50G only - USB I/O if flag -34 is set

The USB port requires the use of the Conn4x communication program, included with the calc, or from hpcalc.org.

HP50G Serial Port: there is currently insufficient information available.

Infrared

Note that the HP49G does not have infrared. There is very little documentation on using the infrared port. Transfers between calculators is straightforward: press APPS, select I/O Functions, and choose the desired method (Transfer is the most flexible). The two calculators should be very close together (the range is measured in inches) with the triangular arrows at the top aligned.

The manuals claim that transfers to and from a PC with infrared are possible, but there isn't a word on how to do it, and no one seems to have done it. The only information available is that the IR port is IrDA-compliant.

There are three methods of transferring: ASCII Kermit, binary Kermit, and binary Xmodem. Settings can be selected with APPS-I/O-Transfer or 104 MENU IOPAR.

The IR port is IrDA compliant and can be used between two HP49G+/HP50G calculators. The distance between units is limited to a few inches (<5cm).

49G Wire Transfer

Xmodem gives the best speed, but its binaries may not be readable in a PC editor. There is no translation of symbols unless you can use **Conn4x**.

Binary transfers will have a 13 char header added: "HHP49...". These are compatible with the HP50G, but not the HP48 series. There are object fixers to remove this header, such as OT49's ObFx (www.hpcalc.org).

ASCII Kermit is slow but gives the best readability on the PC. Special symbols are translated to ASCII equivalents: program delimiters << become \<< and so on. See also [TRANSIO/XLAT](#).

ASCII Header

ASCII transfers have a header added:

%%HP: T(3)A(R)F(.);

This is Translate All, Angle=radians and Fraction Mark=Decimal Point .

USB

This requires HP's Conn4x program on the computer as noted above.

It uses binary Xmodem, but also does translations of HP symbols. There is also a text mode, best for program, string, and text readability on the PC. It's the most versatile method (though it can't access the SD card or other ports).

If Conn4x can't recognize your calculator when you click on File-Connect, try unplugging and reconnecting the USB cable at either end. Try exiting and restarting. The calculator must be in the Xmodem Server mode (RS-right cursor). Be sure you're not in the Kermit server (RS+right cursor).

48 to 49 Transfers

UserRPL programs will usually run on the 49/50 series (see below for common problems).

HP48 libraries will not run on the 49/50 series.

For best results when importing an HP48 program, the 49/50 should be in Approximate mode. For 49/50 programs it should be in Exact mode.

Binary transfers from the older HP48 to the HP49 will have the header "HHP48... and will be unreadable. There are object fixers to remove this header, such as OT49's ObFx (www.hpcalc.org).

48 program Slow on the 49/50:

The 48 had only integers, while the 49/50 has both real and integer math - integers are noticeably slower. To convert the integers to faster reals on the 49/50:

- Put the 49/50 into Real mode (RS+ENTER toggles this)
- Put the program on the stack and press the Down cursor to put it in the editor
- Press ENTER to return the program to the stack; this converts integers to reals
- Save the program back to its var

Another cause of 48→49/50 slowness is EVAL, especially with large expressions and/or loops (EVAL is more complex on the 49/50). You can convert evaluations to stack operations, or you can try putting the expr in a list (1 →LST EVAL), since this will use the simpler version of EVAL.

Strings

ASCII is the best for transferring strings, but Xmodem can be used. Strings sent to the PC in binary are readable, but there's no symbol translation unless you use Conn4x.

Xmodem strings from the PC and strings transferred with the SD card will have a carriage return (CR, chr 13) added, resulting in black squares at the end of each line. Find and Replace will remove these (Use CHARS to enter chr 13 into Find; leave Replace blank), and so will the IN program in the **Translations** entry.

Objects transferred in binary from the HP48 will often appear as uneditable strings beginning with "HHP48... The ObFx program in OT49 (www.hp48.org) can convert these to HP49 objects.

ARCHIVE

ARCHIVE uses binary Kermit regardless of settings. Use the filename format:

:IO: name

for infrared or wire transfers. For Conn4x, use the File-Backup command. Note that backups from the calc's ARCHIVE command are not compatible with Conn4x's backups.

Pinouts

The four pins at the lower left of the HP49G connector are Shield, Transmit, Receive, and Signal Ground (common). This is the same order as the HP48.

If the PC serial connector is the usual 9-pin (DB9), the pins are:

2 - Receive (Rx)

3 - Transmit (Tx)

5 - Signal ground

HP49G Cable

If you make your own cable, interchange pins 2 and 3 (*ie*, Transmit goes to Receive and vice versa). Also, you can omit the Shield connection for short cables (2m or less). The little 4-pin connector for CD-ROM audio will fit the HP48, and even the HP49G if you carefully fit it over the four pins at the lower left of the 49's 10-pin connector. The same connector can often be found inside older mouse models.

TRANSIO/XLAT

TRANSIO is the softkey version; XLAT is on the dialog box from APPS-I/O-Transfer. Both select Kermit character translation of HP49 special symbols to ASCII during I/O. See [TRANSFERRING](#).

Use 3 with TRANSIO and →255 with XLAT to translate all. Zero (or NONE) prevents translation. 1 and 2 provide less translation if that's required.

Xmodem will not do character translation unless you use HP's [Conn4x](#) comm program.

Translated symbols are preceded by a backslash: ∂ becomes \d. and so on. See the entry for [Translations](#) for more detail.

n → where n is 1-3

On the PC, ASCII Kermit files will have a header: %%HP: T(3)A(R)F(.); This is translate level, Angle (Radians or Degrees), and Fraction mark (period or comma).

Access: 104 MENU-IOPAR,
APPS-I/O-Transfer

Translations ASCII ↔ Symbols

The translation of HP special symbols into ASCII or vice versa in strings and programs is usually handled by normal transfers using Kermit or Conn4x with XModem. However, some situations (such as emulators) might require manual translation. See also **TRANSFERRING, Conn4x basics**.

IN/OUT programs

The IN/OUT programs below were posted to comp.sys.hp48 by John H Meyers. They will convert ASCII to symbols or vice versa in strings and UsrRPL programs.

They will also remove carriage returns (chr 13), which show in text files as black squares at the end of lines. They work with all of the HP48, 49, and 50 series.

IN program

"ASCII string" → "HP49 symbols"

"\<< \=/ \>>" → « ≠ »

►► If there's a header similar to:

```
%%HP: T(3)A(R)F(.);
```

it should be deleted before running the program. ◄◄

Symbols can be included in strings using the 3-digit ASCII number from **CHARS** and a backslash; *eg*:

"angle \128" → "angle ∟"

```
%%HP: T(3)A(D)F(.);
\<< IFERR \->STR 3 TRANSIO RCLF
SIZE 3 > # 2F34Dh # 3016Bh IFTE
SYSEVAL + STR\->
THEN \"ASCII string\"
END \>>
```

OUT program

Can be used for translation if you're not using a normal transfer (such as Conn4x).

"HP49 symbols" → "ASCII string"

« ≠ » → "\<< \=/ \>>"

```
%%HP: T(3)A(D)F(.);
\<< IFERR STD 64 STWS \->STR 3
TRANSIO RCLF SIZE 3 > # 2F34Fh
# 2FEC9h IFTE SYSEVAL
THEN \"HP symbols\"
END \>>
```

ASCII/Symbol List 128-255

Chars below 128 don't require translation. Their ASCII numbers can be obtained from the [CHARS](#) map (RS-EVAL). **Special thanks** to the creator of the HP48 TrueType font, who signs himself only as Spiros on hpcalc.org. Not all of the characters are in it, so I've taken some liberties to approximate several of them.

The chart shows the **ASCII number**, the **calculator symbol**, and the ASCII **conversion symbol**.

128	⌞	\<)	171	«	\<<	214	Θ	\214
129	⌘	\x-	172	⌵	\172	215	×	\.x
130	⌵	\.V	173	-	\173	216	Φ	\O/
131	√	\v/	174	—	\174	217	Ù	\217
132	ſ	\.S	175	—	\175	218	Ú	\218
133	Σ	\GS	176	°	\^o	219	Û	\219
134	▶	\>	177	±	\177	220	Ü	\220
135	π	\pi	178	...	\178	221	Ý	\221
136	ð	\.d	179	³	\179	222	Þ	\222
137	≤	\<=	180	´	\180	223	ß	\Gb
138	≥	\>=	181	μ	\Gm	224	à	\224
139	#	\=/	182	¶	\182	225	á	\225
140	α	\Ga	183	■	\183	226	â	\226
141	→	\->	184	µ	\184	227	ã	\227
142	←	\<-	185	¹	\185	228	ä	\228
143	∅	\v	186	Ω	\186	229	å	\229
144	↑	\^	187	»	\>>	230	‡	\230
145	γ	\Gg	188	¼	\188	231	ç	\231
146	δ	\Gd	189	½	\189	232	è	\232
147		\Ge	190	¾	\190	233	é	\233
148	η	\Gn	191	ζ	\191	234	ê	\234
149	θ	\Gh	192	À	\192	235	ë	\235
150	ι	\Gl	193	Á	\193	236	ì	\236
151	ρ	\Gr	194	Â	\194	237	í	\237
152	σ	\Gs	195	Ã	\195	238	î	\238
153	τ	\Gt	196	Ä	\196	239	ï	\239
154	ω	\Gw	197	Å	\197	240	ð	\240
155	Δ	\GD	198	Æ	\198	241	ñ	\241
156	Π	\PI	199	ζ	\199	242	ò	\242
157	Ω	\GW	200	Γ	\200	243	ó	\243
158	■	\[]	201	É	\201	244	ô	\244
159	∞	\oo	202	Ê	\202	245	õ	\245
160	€	\160	203	Ë	\203	246	ö	\246
161	ì	\161	204	Φ	\204	247	÷	\:-
162	¢	\162	205	χ	\205	248	ø	\248
163	£	\163	206	Λ	\206	249	ù	\249
164	☐	\164	207	í	\207	250	ú	\250
165	┘	\165	208	Ð	\208	251	û	\251

‡ 230 actually looks like a backwards swastika

166	¡	\166	209	Ñ	\209	252	ü	\252
167	:	\167	210	Ò	\210	253	ý	\253
168	■	\168	211	Ó	\211	254	þ	\254
169	⌞	\169	212	Ô	\212	255	ÿ	\255
170	⌟	\170	213	Õ	\213			

TRIG keys

SIN, COS, TAN

These three work the same way. In stack mode, they directly return the value for an integer or real, in the current angle measure:

n → value

In an expression, they return funct() and wait for you to enter the desired var. Example:

'X+Y+SIN()'

Flags 2 and 3 should be clear for symbolic mode.

If the var is already on the stack, they form an expression:

'X' COS → 'COS(X)'

INVERSE FUNCTIONS

ASIN, ACOS, ATAN are also called arcsine or SIN⁻¹, etc. These return the angle for a given numerical trig value:

n → angle

.707106781187 ACOS → 45.

The above means that 45 degrees is the angle whose cosine is .707106781187.

For an algebraic expression, they work the same way as SIN, COS, TAN.

RECIPROCAL FUNCTIONS

There are no direct keys for these, but simple programs can be stored if needed for secant, cosecant and cotangent:

<< SIN INV >> 'SEC' STO

<< COS INV >> 'CSC' STO

<< TAN INV >> 'COT' STO

HYPERBOLIC FUNCTIONS

SINH returns the hyperbolic sine of a numeric angle:

`n → value`

The formula is $(e^x - e^{-x})/2$. In algebraics, it works the same way as other trig keys:

`'X' → 'SINH(X)'`

COSH returns the hyperbolic cosine of a numeric angle:

`n → value`

The formula is $(e^x + e^{-x})/2$. In algebraics, it works the same way as SINH.

TANH returns the hyperbolic tangent of a numeric angle:

`n → value`

The formula is $\sinh(x)/\cosh(x)$. In algebraics, it works the same way as SINH.

ASINH, ACOSH and ATANH return the angle in the same way as ASIN, etc:

`hyperbolic_n → angle`

`'ASINH()'`

`'X' ASINH → 'ASINH(X)'`

Access: MTH-HYP

TRN

Conjugate transpose of a matrix. See also **TRAN** which is the same without the complex conjugation.

`[[matrix]] → [[newmatrix]]`

Real number example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 1. & 4. \\ 2. & 5. \\ 3. & 6. \end{bmatrix}$$

Complex number example:

$$\begin{bmatrix} (1,2) & (2,3) \\ (3,2) & (3,4) \end{bmatrix} \rightarrow \begin{bmatrix} 1+2*-i & 3+2*-i \\ 2+3*-i & 3+4*-i \end{bmatrix}$$

In Complex and Approx modes, →**NUM** will convert this result to:

$$\begin{bmatrix} (1.,-2.) & (3.,-2.) \\ (2.,-3.) & (3.,-4.) \end{bmatrix}$$

Access: MTH-MATRIX-MAKE

TRNC

Truncates number of decimal places and internal precision of a number; that is, it actually removes digits rather than rounding off the display. Does not affect the current number format.

To round without losing the internal precision, use a format such as FIX (see **Mode-FMT menu**).

real n → real to n places

Access: MTH-REAL-NXT-NXT

TRUTH

Sets the plot type to Truth and modifies or creates **PPAR**. A pixel is plotted whenever the values supplied to the variables result in a true (non-zero) condition. Since it tests every pixel, it can be slow. Omitting pixels can speed it up - see the entry for **RES**.

See also **Plots/PLOT** for more on general plotting.

The following example is adapted from the HP48 User's Guide.

Plot the inequalities $X+Y \geq 2$, $4Y \leq X+8$, $2Y \geq 3X-6$.

Store the program below in **EQ**.

```
%%HP: T(3)A(D)F(.);
'X+Y\>=2 AND 4*Y\<=X+8 AND 2*Y\>=3*X-6'
```

Press LS+2D/3D (Plot Setup) and enter:

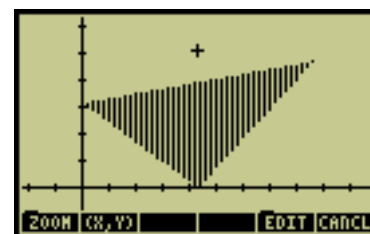
Type: Truth **Angle Mode:** any

EQ: 'X+Y2 ≥ AND etc (This will already be present)

Indep: 'X' **Depend:** 'Y'

H-Tick:10 **V-Tick:**10

Check the **Pixel** field



The pixel method for Truth plots.

Press LS+WIN (Plot Window) and enter:

H-View: -1 5

V-View: -1 4

Indep Low: 0 **High:** 4

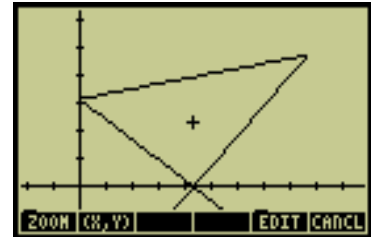
Step: Default **_Pixels**

Depnd Low: Default **High:** Default (A zero gives Default)

Press Erase, Draw. Note that this type of plot is *very* slow (about 5 minutes), even with the default Step value of every second pixel.

A much **faster method** is to replace all the inequality signs with equals signs, make a list of the equations for EQ, and plot it as a **CONIC**. Plot time is ~10s.

{ 'X+Y=2' '4*Y=X+8' '2*Y=3*X-6' }



The Conic method for truth plots.

The triangle enclosed by the lines is the solution set for the inequalities (not the set of equalities, which was only for plotting convenience).

Access: LS+2D/3D-Type,
82 MENU

TSTR

Readable string from **DATE** and **TIME** keys.

(flag -42 clear)

DATE TIME → "day mm/dd/yy hh:mm:ssA/P"

(flag -42 set)

DATE TIME → "day dd/mm/yy hh:mm:ssA/P"

Access: RS-TIME-TOOLS,
RS+TIME,
LS-PRG-NXT-NXT-TIME

TVAR

Returns list of vars in the current dir with contents that match a specified type. See also **TYPE**.

typenumber → {list}

OR

{typenumbers} → {list}

Returns {} if there are no matches.

Access: PRG-MEM-DIR-NXT

TVM

Time Value of Money solver. This uses the SOLVR format rather than the **INFORM** window of LS-FINANCE. See **FINANCE Solver** for more details.

The variables are:

N I%YR PV PMT PYR FV

These are:

N	# of monthly payments
I%YR	interest in %/year (<i>ie</i> , 4.5% is 4.5, not .045)
PV	present value; principal
PMT	monthly payment
PYR	payments/year (typ. 12)
FV	future value

PMTs are negative for withdrawals or pmts against a loan; a positive value indicates extra money being deposited per period.

By default the interest is calculated at the end of the period. To change this, use either flag 14 or **TVMBEG/TVMEND**, below.

Enter the known values and press the associated key. Solve for the desired value with LS-key. Exit with VAR; vars must be deleted manually.

See also **AMORT/AMOR**.

Access: 79 MENU-SOLVR

TVMBEG

Sets the Time Value of Money solver calculations to the beginning of the payment period (see **TVM** above for more information). Called BEG (a toggle) in 79 MENU (the solver); a white square in the key label means 'beginning'.

Flag -14 also controls this.

Access: 79 MENU

TVMEND

Sets the Time Value of Money solver calculations to the end of the payment period (see **TVM** above for more information). Called BEG (a toggle) in 79 MENU (the solver); a white square in the key label means 'beginning', and no square means 'end'. Flag -14 also controls this.

Access: RS+7-TVM (79 MENU),BEG

TVMROOT

Solves for a Time Value of Money variable using the other TVM variables. See [TVM](#) for entering required values.

'TVMvarname' → result

The variables are:

N I%YR PV PMT PYR FV

Access: RS+7-TVM (79 MENU)



2D/3D

The plot setup screen. Creates vars [EQ](#), [PPAR](#) as required. Can be used instead of [Y=](#). If you make any changes and decide not to plot, ENTER saves and exits - ON discards the changes. See also [WIN](#), [Plots](#) for more. (Hold LS while pressing 2D/3D.)

Type: sets a wide variety of plot types, which must have a compatible expression in EQ. Use the CHOOS key, or cycle through the choices with the +/- key.



Angle symbol: sets degrees, radians, or grads, the same as MODE-Angle Measure.

EQ: displays the expr in the var EQ. Pressing EDIT displays this in the Equation Writer. Pressing EDIT again allows an ASCII edit. Press ENTER as required to return to LS+2D/3D, or ON to cancel.
Multiple eqs must be in a list. Edits will be ASCII.

Indep: the independent var, which must agree with the var in the EQ expr. The default is X.

Depnd: dependent var - appears for plot types with two vars, such as Conic.

Simult: if checked, plots multiple expressions simultaneously. They must be in a list in EQ. This is the same as flag -28.

Connect: if checked, joins the plotted pixels into a smooth line. If not, the plot shows as dots according to the Step setting in LS+WIN (see also [RES](#)).

H-tick,V-tick: these set the axis tick mark spacing. The default is ten pixels apart. If **Pixels** is unchecked, the spacing is 10 times the entered value between ticks. See also [ATICK](#).

Access: LS+F4,
CALC-GRAPH-2D/3D

256 MENU

Thanks to Joe Horn, who collected these for the comp.sys.hp48 newsgroup. There has been some editing for HLP49.

NOTE: some of these tools have no error-checking and can crash the calculator if used without the proper arguments. A basic knowledge of SysRPL programming is essential.

Also note that there is much confusion about starting menu 256. It would seem that with newer ROMs such as v2.09, the menu attaches automatically. Try this: press APPS and look for "Development lib". If it's there, 256 is ready to use by pressing it or running 256 MENU.

If not, follow the instructions below.

To have its commands available from anywhere, type 256 ATTACH from the Home dir first. To make this permanent, put the Attach statement in STARTUP in the Home dir (see the entry for **STARTUP**).

Some programs, such as OT49, attach it automatically on startup. Once it's attached, run it with APPS-Development_lib or 256 MENU.

256 MENU

→H To Hex

Same as →ASC in the 48. Converts object to hex string:

"ABC" → "C2A20B0000142434"

H→ From Hex

Converts a hex string to an object.

"C2A20B0000142434" → "ABC"

→A To Address

Finds the address of an object, typically a global name or string.

obj → #address

A→ From Address

Gets the object at the specified address.

#address → obj

A→H Address to Hex

Converts #12345h to "54321".

H→A Hex to Address

Converts "54321" to #12345h.

→CD To Code

Turns a string of hex digits into a Code object.

"C2A20B0000142434" → Code

CD→ Code Out

Turns a Code object into a string of hex digits.

Code → "C2A20B0000142434"

S→H String to Hex

Changes "Hi!" into "849612".

H→S Hex to String

Changes "849612" into "Hi!".

In the next four, n = number of elements.

→LST Make List

Composite or meta-obj n → list

→ALG Make Algebraic

Composite or meta-obj n → algebraic

→PRG Make Program

Composite or meta-ob n → program

COMP→ Composite Out

Composite → meta-object n

Similar to **OBJ→** for algebraics, but separates more elements.

→RAM To RAM

Dumps any ROM object into RAM.

SREV String Reverse

Turns strings backwards.

"ABC" → "CBA"

POKE

Pokes raw hex into RAM. Use with caution.

PEEK

Extracts the raw hex as a string at any address. (Address on level 2, bytes to get as binary integer on level 1.)

APEEK

Extracts 5 bytes of raw hex as a binary number at any address.

R~SB

Real ↔ System Binary.

Converts reals to bints & back.

SB~B

System Binary ↔ Binary.

Bints to user binaries & back.

LR~R

Long Real ↔ Real.

S~N

String ↔ Name. Change "ABC" to 'ABC' & back. Use this to create "illegal" global names:

"45ABC" → '45ABC'

The Filer will delete variables with names like this - PURGE won't work because you can't put the name on the stack.

LC~C Complex type

Long Complex ↔ Complex

ASM→ Assembly Out

Disassembles code objects into source code.

BetaTesting - names of testers

CRLIB

Create Library. When dir is ready, run. Puts lib on stack. See the [CRLIB](#) entry for details.

CRC

Returns CRC of string as bint. Use SB~B to convert to user binary.

MAKESTR

Makes string n chars long, with newline after every 7.

10 → "ABCDEFGG
AB"

SERIAL

Returns internal serial number, which is not the same as the external.

ASM

Compiles a source code string into object code. The last char in the string must be @ (AT symbol) on a line by itself.

ER

Error handler for ASM. It provides error messages if ASM errors. ASM2 from menu 257 is similar to:

```
<< IFERR ASM THEN ER END >>
```

→S2

Disassembles a binary object into source code in hex. Makes SysRPL code into a string for editing.

XLIB~

Libnum cmdnum ↔ XLIB nums

eg: XLIB 1759 3 → 1759 3

or: 1759 3 → XLIB 1759 3

Note on the next three: see Chapter Six of the 49G+ Advanced User Reference for information on ARM programming. Available for download from hpcalc.org.

ARM→

The ARM→ disassembler converts ARM assembly to a source string.

Level 2

Binary integer (start address of the memory area to disassemble)

Level 1

Binary integer end address

PEEKARM

Memory read: reads nibbles from a specified address in memory in the ARM address space.

Level 2

Binary integer (address)

Level 1

Binary Integer (number of bytes to read)

POKEARM

Memory write command. Writes bytes in ARM memory address space.

Note: You cannot write data in the Flash ROM using this command.

Level 2

String (Data to write in hex)

Level 1

Binary integer (Address where to write)

TYPE

Returns type number of object on Level 1.

obj → typenumber

OBJECT	User	SysRPL
any	--	0
real number	0	1
complex	1	2
string	2	3
real array, matrix	3	4
complex array	4	4
list	5	5
global name	6	6
local name	7	7
program/seco	8	8
algebraic/sym	9	9
class (usr 6,7,9)	--	A

user binary (hxs)	10	B
grob	11	C
tagged obj	12	D
unit obj	13	E
XLIB name	14	F
ROMPTR	14	F
directory	15	2F
library	16	8F
backup obj	17	9F
function	18	8
command	19	8
sys binary (bint)	20	1F
long real	21	3F
long complex	22	4F
linked array	23	5F
char	24	6F
code	25	7F
library data	26	AF
minifont	27	DF
integer	28	FF
sym/int array,matrix	29	4
font	30	CF
extended obj	31	EF

The SysRPL values above when used with Dispatch commands should be in the form of a bint (*eg*, BINT9) if it exists, or a hex number (*eg*, #B, without the usual "h").

Access: PRG-TYPE-NXT-NXT,
PRG-TEST-NXT

UBASE

Converts unit object to SI base units.

`unit_obj` → SI base units

`'1_dyn'` → `'.00001_kg*m/s^2'`

You can convert the returned SI value to English units using **UFACT** by typing in the equivalent English compound unit:

`'1_lbf'` UBASE → `'4.4482_kg*m/s^2'`

`'4.4482_kg*m/s^2'` `'1_lb*ft'`
UFACT → `'32.1740_lb*ft/s^2'`

Access: UNITS-TOOLS

UFACT

Factors Lev 1 unit out of Lev 2 unit obj.

`A_unit1 B_unit2 → C_unit1unit2`

`'.762_m/s' '1_ft' → '2.5_ft/s'`

`'6.6726E-11_m^3/(s^2*kg)' '1_N' → '6.6726E-11_N*m^2/kg^2'`

See also [UBASE](#) for a way to convert SI base values to English using UFACT.

Access: UNITS-TOOLS

UNBIND

A CAT command that clears local variables created in memory by [LOCAL](#). Local vars should have their values in the form:

`'←A=55' → same`

OR

`{ '←A=55' '←B=65' } → { '←B=65' '←A=55' }`

It often didn't work consistently on the 49G, but the 49G+/50G seems stable, though you might have to turn off UNDO with LS+MODE-MISC-STK.

An ON+C reboot will also clear the local vars.

If there are no local variables, a "cannot unbind" error msg will appear.

Access: CAT (not typed in)

UNDO

Replaces the current stack with a copy of the stack from before the last operation.

Does not undo operations such as PURGE, settings, etc. Not a programmable command. The last-stack option must be enabled (MODE or LS+MODE-MISC-STK).

The Undo buffer can sometimes contain large objects that take up a lot of memory. You can empty the buffer by pressing LS+MODE-MISC. Press STK twice to turn Undo off and then on again. STK is also available from MODE and PRG-MODES-MISC.

Access: UNDO key (RS-M)

→UNIT

Creates unit object from real or integer and unit expression.

`x y_unit → x_unit`

`5.1 1_m → 5.1_m`

Any y value is discarded.

You can also use the times key, but y values are not discarded:

`5.1 3.5_mm → 17.85_mm`

Access: UNITS-TOOLS

Unit basics

You can attach any SI unit to a number with the UNITS menu. Enter the number and press the desired unit key.

To convert one unit to another, put the first unit on the stack and press the left-shifted new unit. For example, to convert inches to centimetres:

`'5.1_in' LS-cm → '12.954_cm'`

You can also type in any unit with the underscore (RS-minus), even if the unit is not in the menus.

`5.5_anyunit`

Units will accept any SI prefix, even if it isn't listed in the menus. For example, megohms:

`1_Ω`

Add an M for `1_MΩ`

Convert back to ohms: `1000000_Ω`

A list of SI prefixes is in the HP49G+ User's Guide, p. 3-24, or the HP50G User's Guide, p. 3-10.

Units can be removed with **UVAL**. (Units can cause unexpected results in calculating, especially with solvers and physical constants.)

► The Equation Writer does not support units.

The unit tools from RS-UNITS have their own entries. See **CONVERT**, **UBASE**, **UFACT**, **→UNIT**.

Access: RS-UNITS

USER

Changes the keyboard to User mode, allowing user-defined commands that have been assigned to keys. Unassigned keys retain their normal function (see DELKEYS for locking out unassigned keys).

In the one-time User mode, the annunciator 1US appears. The mode can be locked on - see flags -61 and -62. The annunciator is then USR.

See also [ASN](#), [Custom Enter](#), [DELKEYS](#), [Key Numbers](#), [RCLKEYS](#), [STOKEYS](#).

Example: assign the **PURGE** command to the DEL key (LS-Bksp) (before making any changes to an existing keylist, store the keylist from RCLKEYS in a var - if necessary you can then restore the original assignments with this list and STOKEYS).

The key number for DEL is 45.2.

Enter: << PURGE >> 45.2

and press ASN (PRG-MODES-KEYS).

To use, type LS-USER-DEL.

To delete this key assignment, enter 45.2 and press DELKEYS.

Note that key assignments do not work in the editor or other programs unless the assignment has the SysRpl command **TakeOver**. An easy way to do this is the →TO? function from the Keyman library (www.hpcalc.org):

```
<< pgmname >> →TO? keynum ASN
```

Access: LS-ALPHA, flag -62

UTPC

Probability that a chi-squared random variable is > x given *n* degrees of freedom.

```
n x → utpc(n, x)
```

Access: MTH-NXT-PROB-NXT

UTPF

Probability that a Snedecor's random variable is > *x*. *n1* and *n2* are the numerator and denominator degrees of freedom of the F distribution.

```
n1 n2 x → utpf(n1, n2, x) →
```

Access: MTH-NXT-PROB-NXT

UTPN

Probability that a normal random variable is $> x$. m and v are the [MEAN](#) and variance (see [VAR](#)) of the normal distribution.

$$m \ v \ x \rightarrow \text{utpn}(m, v, x)$$

Access: MTH-NXT-PROB-NXT

UTPT

Probability that a Student's random variable is $> x$, given n degrees of freedom.

$$n \ x \rightarrow \text{utpt}(n, x)$$

Access: MTH-NXT-PROB-NXT

UVAL

Returns numerical part of unit object.

$$x_unit \rightarrow x$$

Units can sometimes cause errors in user programs, especially with constants in the equation solvers. Units can be removed from constants with [UVAL](#):

$$\text{UVAL}(\text{CONST}(var))$$

They can be prevented by setting user flag 61 (positive flag number - see the [CONST](#) entry).

Access: UNITS-TOOLS

$V \rightarrow, \rightarrow V2, \rightarrow V3$

See also [Arrays/vectors](#).

$V \rightarrow$

Separates a vector or complex number. Also works with arrays.

$$\begin{aligned} [1. \ 2.] &\rightarrow 1. \ 2. \\ [10 \ \angle 45] &\rightarrow 10. \ 45. \\ (1., 2.) &\rightarrow 1. \ 2. \end{aligned}$$

Note that vectors are always in rectangular mode internally - the polar mode is only on the display:

$$[5. \ \angle 53.1301] \text{ OBJ} \rightarrow \rightarrow 3. \ 4. \ \{2.\}$$

However, polar coords can be kept by using $V \rightarrow$ in polar mode:

$$[5. \ \angle 53.1301] \text{ V} \rightarrow \rightarrow 5. \ 53.1301$$

→V2

Creates a 2D vector from two reals if flag -19 is clear. If set, it creates a complex number.

```
1. 2. → [1. 2.] (-19 clear)
1. 2. → (1.,2.) (-19 set)
```

Vectors change in the display between rectangular and polar mode, while arrays stay the same.

Rectangular mode:

```
1. 2. → [1. 2.]
```

Polar mode:

```
1. 2. → [2.2360 ∠ 63.4349]
```

See [CYLIN](#) for an example of using →V2.

→V3

Creates 3D vectors.

Rectangular mode:

```
1. 2. 3. → [1. 2. 3.]
```

Polar mode:

```
1. 2. 3. → [2.2360 ∠ 63.4349 3.]
```

Access: MTH-VECTR

VAR

1. The VAR key displays the current softkey menu. It's also used to exit from the single or multiple equation solvers. In programming, 2 MENU is the same as pressing VAR.

Access: VAR key

2. Returns sample variance using the column(s) in the statistics variable [ΣDAT](#). For the population variance, see [PVAR](#).

```
→ n or [array]
```

Access: STAT-Single-var,
96 MENU-1VAR

3. HP's term for both the variables in algebraics (A,B, X, Y, etc) and the named, stored object using the STO command.

```
obj 'varname' STO → stored var
```

Note that varnames are case-sensitive - 'x' and 'X' are two different variables.

The stored object is the same as a file in computers. It's called a *global variable*. A point of confusion: the single-quote format is used for both varnames ('MYVAR') and for algebraics ('X^2+5'). For this reason, names can't begin with a number (but see S~N in [256 MENU](#)).

Another type is the *local variable*, which is created by programs and cleared when the program exits. See also [LOCAL](#), [Assign](#) for more detail.

VARs

Returns list of variables in current dir, which can be used to reorder the dir with the [ORDER](#) command.

```
→ { list of vars }
```

Use Cut and Paste to move varnames to the desired positions in the list. Note that ORDER will delete vars with missing names.

Access: PRG-MEM-DIR-NXT

VER/VERSION

VER: CAS version number.

VERSION: system version number.

The ON+D diagnostic screen will return a more complete system number than VERSION and includes the build number. Use ON+C to exit. See also [ON+ Codes](#).

Access: CAT

VISIT/VISITB

Similar to [EDIT/EDITB](#), included for compatibility with HP48SX programs.

```
'name' →
```

The object referenced by 'name' must be stored in a variable in the current dir (not on the stack as with EDIT).

If ENTER is pressed, changes will be written to the 'name' variable, which never appears on the stack.

On the 49/50 series, the LS+Down cursor is similar to the Down cursor (which is EDITB), but can act as VISITB in that it edits the contents of a named var rather than the varname:

	<u>Down</u>	<u>LS+Down</u>
name	edits name	edits content
obj	edits obj	edits obj

Access: 2269 MENU, CAT

VPAR

Variable containing the view parameters for 3D plots and special-purpose plots such as **SLOPEFIELD**. It's created automatically in the current dir. Since VPAR is erased and rewritten each time, it appears at the beginning of the varnames in the current dir.

In the parameters list below, **numbers in brackets are the default values**. These are taken from HP49G+ v2.09 and may not agree with HP docs. All parameters may not appear in the plot window (LS+WIN).

```
{
1-2      (Xleft, Xright) (XVOL) width of the view volume. (-1, 1)

3-4      (Yfar, Ynear) (YVOL) depth of the view volume. (-1, 1)

5-6      (Zlow, Zhigh) (ZVOL) height of the view volume. (-1, 1)

7-8      (XXleft, XXright) (XXRNG) the width of the input plane (domain). Used by GRIDMAP and
PARSURFACE (Pr-Surface). (-1, 1)

9-10     (YYfar, YYnear) (YYRNG) the depth of the input plane (domain). Used by GRIDMAP and
PARSURFACE. (-1, 1)

11-12-13 (XE, YE, ZE) (EYEPT) the point from which the plot is viewed. (0, -2, 0) (Pressing Reset in
LS+WIN may show 0,0,0)

14-15     (Xstep (Indep), Ystep (Depnd)) (NUMX/NUMY) the increments between X-coordinates and
Y-coordinates. (10, 8)
          Used instead of (or with) RES.
}
```

VTYPE

Returns the object type in a specified variable. See also **TYPE**.

```
'varname' → n
```

Access: PRG-TYPE-NXT-NXT

WAIT

Suspends operation for n seconds. If n=0, waits for keypress and returns keynumber (except alpha and shift keys).

If n=-1, waits for keypress and displays the current menu. **MENU** (or **TMENU**) and WAIT can be used in a program to display any menu (menus are usually not updated until the program ends).

Follow with DROP if the keynumber isn't needed.

Example: 41 MENU -1 WAIT DROP in a program displays the **DEBUG** menu when a key is pressed.

```
n    →
0    → keynumber
-1   → keynumber, displays menu
```

See also the [Key Numbers](#) entry.

Error messages can be displayed for a specified time in seconds using WAIT - see [STARTERR](#).

The WAIT command sometimes fails in programming - see [Helptext](#) for an example and [START](#) for a substitute.

Access: PRG-NXT-IN

WHERE (|)

WHERE substitutes value(s) for symbol(s) in an expression.

```
expr { symb val ... } → newexpr
```

```
'x+y' { x 2 } → '2+y'
```

```
'A+B' { A 2 B 3 } → '2+3'
```

```
'X^2+5' { X 3 } → 3^2+5
```

EVAL or **→NUM** will simplify the numerical expressions.

It can also be used in algebraics in the textbook sense:

```
'X^3 | (X=2) '
```

The above reads "X to the third where X equals 2". The EVAL/→NUM commands will evaluate it.

Access: RS-TOOL

WHILE

Starts conditional repeat loop.

```
WHILE <condition> REPEAT <procedure>    END
```

Repeats the procedure as long as the condition returns a non-zero value. It terminates when the condition returns zero.

If the condition never returns zero, you can stop the loop with ON (sometimes repeatedly).

Typing aid: when the WHILE dir is displayed (LS-PRG-BRCH), these are available:

```
LS-WHILE  → WHILE
           REPEAT
           END
```

The example below drops all stack objects but the last.

```
<< WHILE DEPTH 1 > REPEAT DROP END  >>
```

Access: PRG-BRCH

WIN

A screen for entering vertical and horizontal display and plot ranges (the range plotted can be larger or smaller than the displayed range).

The default plot type is a **FUNCTION**, which is described below. Other plot types are similar.

H-View:

This is the plot width. The Default plot range is not shown in numbers - it's -6.5 to 6.5. The programmable version is **XRNG**.

V-View:

This is the plot height. The default is 4 (top) to -3.9. The programmable version is **YRNG**.

Indep Low,High:

This is the plot range. The default values are the same as the values in H-View. The programmable version is **INDEP**. For example, to plot only from the origin to the right side, change Indep Low to zero.

Step:

This sets the resolution - how many pixels will be plotted. The default is every 2nd pixel. The programmable version is **RES**.

Pixels:

If this is checked, the number in Step is the number of pixels plotted. A value of 3 (every 3rd pixel)

gives rapid plots with acceptable resolution.

If gaps appear in some plots, uncheck this and enter .05 or smaller. Plots will be slower.

Defaults can be restored to fields Step and Indep by entering a zero.

The NXT key shows a softkey menu with RESET (restores defaults) and CALC (returns to stack for calculating). The STS (status?) key allows full 7-line stack mode on the next entry. Pressing it again gives a brief description of which field the results will be passed to when OK is pressed.

Access: LS+F2

WIREFRAME

Sets the plot type to Wireframe and modifies or creates **PPAR**. See also **Plots** for more on general plotting.

Wireframe draws an oblique-view 3D model of the surface of $Z=F(x,y)$. The sampling grid is determined by the base of the View Volume (Xleft,Xright,Ynear, Yfar).

Example: Plot ' $X^3.Y-X*Y^3.$ ' as a wireframe.

Press LS+2D/3D and enter:

EQ: $X^3.Y-X*Y^3.$ **RAD**

Indep: 'X' **Depnd:** 'Y'

Press LS+WIN and enter:

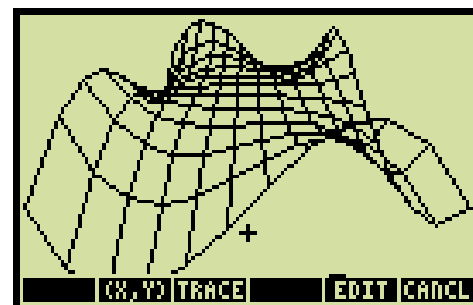
X-left: -1. **X-Right:** 1.

Y-Near: -1. **Y-Far:** 1.

Z-Low: -.4 **Z-High:** .4

XE: 0. **YE:** -2. **ZE:** 1.

Step Indep: 12. **Depnd:** 12.



Press Erase and Draw.

A much better discussion of plotting than in the HP49 manual is in the HP48 Series User's Guide, available for free downloading by searching www.hp48.org.

Access: LS+2D/3D-Type,
85 MENU

WSLOG

Returns four strings giving the time and date of the last four warmstarts (reboots) caused by anything other than ON+C.

The meaning of the first number or letter is (from a post on comp.sys.hp48, and may not be up to date):

- 0** - coma exit (HP48)
- 1** - low battery shutdown
- 2** - I/O timeout error
- 3** - run through addr 0
- 4** - corrupt time
- 5** - port data corrupt
- 6** - not used
- 7** - corrupt word in RAM
- 8** - not used
- 9** - alarm list corrupt
- A** - RPL jump to addr 0
- B** - card removed (HP48)
- C** - hardware reset
- D** - RPL ERRTRAP not found
- E** - corrupt config table
- F** - RAM card pulled (HP48)

Access: CAT

XCOL

Sets the independent-var column for the matrix in [ΣPAR](#) (the default for X is column 1). The values are used by the plotter - see the entry for [LR](#). See also [COLΣ](#), [YCOL](#).

n →

The format of ΣPAR is: { indep depen y-int slope }.

Access: 96 MENU-ΣPAR

XGET

Retrieves a file from a remote Xmodem server.

'name' →

Access: CAT

XLAT

See the entry for [TRANSIO/XLAT](#).

XMIT

Transmits through selected I/O without using Kermit; returns error digit.

"string" → 0/1

Timeout is 10 seconds, or can be set by STIME:

n →

where n is a real from 0 to 24.5. If n=0, it waits indefinitely.

Access: 104 MENU-SERIAL

XOR

Logical exclusive OR of 2 arguments. The ones on the left side below can be any non-zero value. The result is 1 if either input is 1, but not both.

1	1	→	0
0	0	→	0
0	1	→	1
1	0	→	1

Reals or integers are taken as zero or non-zero:

5	0	→	1.
5	3	→	0.
0	0	→	0.

In bitwise calculations with user binary integers, processing is done bit by corresponding bit, so that

110001b # 101001b → # 11000b

is the same as

110001
101001

011000

BASE should be in the BIN mode. Leading zeros are not displayed on the stack.

It can be used in an algebraic:

'X>2 XOR Y<4'

This will return 1 or 0 when evaluated, depending on the values stored in X and Y.

It can be used in programming to accept one or the other of two conditions, but not both:

```
IF <test1> <test2> XOR THEN...
```

The THEN command proceeds only if one of the tests is non-zero.

Access: PRG-TEST-NXT,
BASE-NXT-LOGIC

XPON

Returns exponent part of argument using E notation.

```
2.E15 → 15.
```

For other exponent notation, returns the E equivalent exponent:

```
'13^16' → 17.
```

since $13^{16} = 6.65416609183E17$. See also [MANT](#).

Access: MTH-REAL-NXT

XPUT

Sends a file by Xmodem to a remote Xmodem server.

```
'name' →
```

Access: CAT

XRECV

Receives a file from a remote server using Xmodem.

```
'name' →
```

Access: APPS-I/O-Transfer,
104 MENU

XRNG

Sets X-axis display range for plots. Default is -6.5 to 6.5. See also [PPAR](#), [YRNG](#).

```
Xmin Xmax →  
-5 5 →
```

This is the programmable version of H-View in LS+WIN. To set the plot range separately, see [INDEP](#).

Access: 81 MENU-PPAR

XROOT

X root of Y. Program/algebraic version:

'XROOT (X, Y) '

$y^x \rightarrow \sqrt[x]{y}$

Access: RS-√X

XSEND

Sends file via Xmodem.

'name' →

Access: APPS-I/O-Transfer,
Filer,
104 MENU

XSERV

Starts Xmodem server mode. The command can be entered with the shortcut below, typed in, or selected from CAT.

For the Kermit server, press RS+right cursor.

Access: RS-right cursor

XVOL

Sets width of view volume in 3D - called X-Left, X-Right in LS+WIN 3D plots.

See also the entry for [VPAR](#), where it appears as variables 1 and 2.

X-Left X-Right →

Access: LS+WIN
81 MENU-3D-VPAR

XXRNG

Sets X range of input plane (domain) for [GRIDMAP](#) and [PARSURFACE](#) (**Pr-Surface**). Called XLeft, XRight in LS+WIN.

Xmin Xmax →

See also the entry for **VPAR**.

Access: LS+WIN
81 MENU-3D-VPAR

YCOL

Sets the dependent-var column for the matrix in **ΣPAR** (the default for Y is column 2). The values are used by the plotter - see the entry for LR. See also **COLΣ**, **XCOL**.

$n \rightarrow$

The format of **ΣPAR** is: { indep depen y-int slope }.

Access: 96 MENU-ΣPAR

Y=

Allows creation, selection, editing of the function(s) in EQ and Y(n) variables, used for plotting. Creates **EQ**, **PPAR**, and the Y(n) vars as needed.

Only functions can be plotted. For other plot types, use LS+2D/3D. See also the entry for **Plots** for more about functions.



If EQ already exists and contains functions, they will also display and plot, but are not stored using the Y system unless they were previously made with Y=.

If the equation list EQ doesn't exist, it will say No Equation. Press ADD. Pressing ADD will start the Equation Writer with the left side Y1(X)=. Enter the first one, for example SIN(X), and press ENTER to return to Y=.

Plotting this will create EQ, which will contain {Y1(X)}, and var Y1, which will contain the program << → X 'SIN(X)' >>.

Repeat as required to create Y2, Y3, and so on. Each Yn(X) is added to the list in EQ.

Note that DRAW plots all the equations showing in Y=. To plot only certain ones, delete the unwanted ones with DEL (see below). Restore with CHOOSE.

Note also that Y= has no control over plot parameters - use LS+2D/3D or the plot screen for this.

DEL removes the highlighted equation from the list and from EQ. It does not delete the Y var from the dir.

CHOOSE displays all the dir's Y equations that are not in EQ. Use it to restore equations removed by DEL.

ERASE clears the plot screen.

DRAW plots all the listed eqs.

MOVE↓, MOVE↑ let you set the order of the eqs.

CLEAR removes all equations. You can restore any or all with **CHOOSE**.

Access: LS+F1

YRNG

Sets Y-axis display range for plots. Default is 4 to -3.9. See also **PPAR**, **XRNG**. This is the programmable version of V-view in LS+WIN. See also **DEPND**.

```
Ymin Ymax  →
      2  -2  →
```

Access: 81 MENU-PPAR
 LS+WIN-V-View

YSLICE

Sets the plot type to Yslice and modifies or creates **VPAR**, **PPAR**.

LS+2D/3D is normally used to store an expression in EQ and LS+WIN is used to set the view. YSLICE is mainly for programming.

When run with DRAW, Yslice creates eight different samples (the default - change this with LS+WIN-Depnd, or **NUMY**) and then animates them.

The Save Animation feature in LS+2D/3D doesn't seem to do anything - if you use the program version, the 8 grobs will be on the stack, or you can exit 2D/3D and run **DRAW** (CAT, 81 MENU) instead of 2D/3D's DRAW.

The program syntax is:

```
YSLICE 'expr' STEQ DRAW
```

STEQ can be followed by -n n **ZVOL** to change the plot size (these are Z-Low and Z-High in LS+WIN) - small numbers give larger plots. See the example below.

The demo program below runs YSLICE without 2D/3D - just paste it to the stack and press EVAL. Exit with ON - it often requires multiple presses. The grobs will be on the stack with the number of grobs on Lev1.

```
%HP: T(3)A(D)F(.);
\<<RAD ERASE YSLICE 'X^3*Y-X*Y^3'
STEQ -.5 .5 ZVOL DRAW \>>
```


To see graphically what Yslice is doing, run the above and then LS+2D/3D. Change the plot type to **FAST3D** and press DRAW. Yslice is drawing sample slices as it moves through this figure, which you can rotate with the cursor keys.

See also **Plots** for more on general plotting.

Access: LS+2D/3D-Type
85 MENU

YVOL

Sets depth of view volume in **VPAR**, variables 3 and 4. Called Y-Near, Y-Far in LS+WIN 3D plots.

Y-Near Y-Far →

Access: LS+WIN
81 MENU-3D-VPAR

YYRNG

Sets X range of input plane (domain) for **GRIDMAP** and **PARSURFACE (Pr-Surface)**. Called YYNear, YYFar in LS+WIN. See also the entry for **VPAR**.

YYNear YYFar →

Access: LS+WIN,
81 MENU-3D-VPAR

ZFACT

1. Sets the zoom factor in the Function plot ZOOM menu and other plot types that support zooms. The default is 2. See the entry for **Graphics menu** for more detail.

Access: LeftCursor-ZOOM-ZFACT

2. Returns gas compressibility correction factor for non-ideal hydrocarbon gas.

Xtr YPr → Zfactor

Access: 117 MENU,
APPS-EqnLib-UTILS

ZPAR

Variable containing zoom factors and a copy of the **PPAR** before the zoom. Created in the current dir by any zoom. Zoomfactors are set with ZOOM-ZFACT.

Numbers below are default values.

```
{
  h-zoomfactor 4
  v-zoomfactor 4
  recenter      0
  {} empty or copy of last PPAR
}
```

ZVOL

Sets height of 3D view volume in **VPAR** (elements 5 and 6). Called Z-Low, Z-High in LS+WIN 3D plots.

Z-Low Z-High →

Access: LS+WIN,
 81 MENU-3D-VPAR

\geq

Equal to or greater than; compares y to x and returns 1 (yes) or 0 (no).

30 50 → 0

Inserts \geq into program.

Access: LS-1/X,
 PRG-TEST

\leq

Equal to or less than; compares y to x and returns 1 (yes) or 0 (no).

30 50 → 1

Inserts \leq into program.

Access: LS-X
 PRG-TEST

>

Greater than; compares y to x and returns 1 (yes) or 0 (no).

30 50 → 0

Inserts > into program.

Access: RS-1/X
PRG-TEST

\

Used in text to replace a symbol with an ASCII equivalent; *eg*, the right arrow becomes \->.

During transmission of the file in ASCII (or Conn4x Xmodem), the ASCII is replaced with the proper HP symbol. See the entries for **TRANSFERRING** and **Translations**.

To embed double quotes in a string (*ie*, one string inside another), precede each double quote of the inside string with a backslash.

Access: Alpha-RS-5
CHARS, CHR 92

<

Less than; compares y to x and returns 1 (yes) or 0 (no).

30 50 → 1

Inserts < into program.

Access: RS-X,
PRG-TEST

≠ (not eq.)

Not equal to; compares y to x and returns 1 (yes) or 0 (no).

30 50 → 1

Inserts ≠ into program.

Access: LS-+/-
PRG-TEST

π (pi)

Places π or numeric value on Lev1:

Flags -2 and -3 clear:

→ π

Flags -2 and -3 set:

→ 3.14159265359

Note that it performs an ENTER - it doesn't appear on the command line as numbers do. It does not do an ENTER during algebraic entry: '2* π +X'.

Access: LS-SPC

::

Inserts a double colon to indicate a user-specified tag or a port number. Only the right-hand colon shows on the stack. See →[TAG](#).

In alpha mode (alpha-LS::), inserts a single colon.

→**Tag example:**

"result" 3.141 → result:3.141

An empty tag can be used to put a cmd or programname on the stack without evaluating it:

::COS ::HLP49

See also LIST series for another way of doing this.

Port example (var in flash ROM):

:2:MATHDOC.TXT

Port example (var in SYSRPL dir):

:2:{ SYSRPL RPLMAN.DOC }

Access: LS-period

+/-

Plus or minus. Changes the sign of the value on Lev1 from plus to minus and back again.

Inserts NEG into program.

Toggles highlighted values in menus.

Access: +/- key
 MTH-NXT-CMPLX-NXT (NEG)
 CMPLX (NEG)

—

The underscore links a value to its unit:

237_mm

The calculator accepts any unit you want to enter:

237_foo

It will handle these in the usual algebraic way:

$(237_foo)^2 \rightarrow 56169_foo^2$

To remove the underscore and its unit, use **UVAL** (in RS-UNITS-TOOLS).

► Currently (ROM v2.09), units cannot be used in the Equation Writer.

Access: RS-minus key

► End of main entries - see below for the Appendix ◀

APPENDIX

Cross-references to programs and utilities that are in other entries. Pressing the calculator SPC key will make the index jump to it.

<u>SUBJECT</u>	<u>ENTRY</u>
→ (in program)	Assign
Algebraics→complex	i
All-purpose sine	Plots
Auto-binaries	Custom Enter
Backups	ARCHIVE, Conn4x basics
BASIC arrays	ARRAY-LIST
Better beep	BEEP
Black square removal	NEWLINE, TRANSFERRING
Clean exits	IFERR
Clean Solver	SOLVING basics
Clock set program	CLKADJ
Clock adjust alarm	Alarms
CNFIG program	A Few Basics
Coefficients→algebraic	DOSUBS, PEVAL
Connectivity	Conn4x basics
Create library	CRLIB
CST	CUSTOM
Custom EQW	STARTEQW
Degrees, using	DMS (degrees, minutes, seconds)
Delay, adding	START
ΔLIST	LIST series
DESOLVE	Derivative (∂)
Diffeq numerical	DIFFEQ - part 2
Diffeq plots	DIFFEQ - part 1
Doubleclick	Helptext (in pgms)
Editor start	NOT, STARTED
Ellipse plot	CONIC
Error messages	DOERR, ERRM
Error message, header	STARTERR
EXCO	SOLVING basics
Fast truth plot	TRUTH
Flag toggle	FLAGS - list of flags
GC, forced	Garbage Collection
Global vars	VAR, LOCAL
Help in programs	Helptext (in pgms)
HP48 program slow	TRANSFERRING
IN/OUT programs	Translations ASCII - Symbols
Left quotes (``)	Backquotes (left quotes)
Lissajous plot	PARAMETRIC
Local vars	VAR, Assign, LOCAL
Mode toggle	Numerical/Symbolic modes
Mode switching	REAL Mode, APPROX. mode

SUBJECT**ENTRY**

MODIF font editor	CHARS
Names, naming	STO
ODEs (DESOLVE)	Derivative (∂)
ODEs (plots)	DIFFEQ (Plots, Numerical)
ODETYPE	Derivative (∂)
π LIST	LIST series
PICT with menu	PICT, PVIEW, TMENU
Plot program	DRAW
Polar notation, easy	CYLIN
Polar/rect toggle	CYLIN, RECT
Power program	Programming basics
PUSH/POP	Programming basics
Reboot	Recovery, Reset
Repeating alarm	Alarms, Recovery
Root finder program	INPUT
Rose-like plot	POLAR
Σ LIST	LIST series
S~N str \leftrightarrow name	256 MENU
STIME	XMIT
Sum of sqrs program	Sigma (Σ)
Symbols \leftrightarrow ASCII	Translations ASCII ? Symbols
Symbolic sequence solve	SEQ
Terminating FOR	FOR-NEXT
Time to auto-off	TOFF
30 MENU solver	SOLVING basics
TYPE test program	CASE
"Undefined variable"	Numerical/Symbolic modes
UNINSTALL program	A Few Basics
User Flags	FLAGS - list of flags
Vector plot program	Arrays/vectors
Vector rotate	Arrays/vectors
VX	SOLVING basics, STOVX
Warmstart	Reset
Wildcard (&)	A Few Basics, MATCH, RCL