





Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Bad Packet Block check	Computed packet checksum doesn't match checksum in packet.	C01
Can't Edit Null Char.	Attempted to edit a string containing character # (character code 0).	102
Circular Reference	Attempted to store a variable name into itself.	129
Connecting	Indicates verifying IR or serial connection.	C0A
Constant?	HP Solve application or ROOT returned same value at every sample point of current equation.	A02
Copied to stack	 copied selected equation to stack.	623
Current equation:	Identifies current equation.	608
Deleting Column	MatrixWriter application is deleting a column.	504
Deleting Row	MatrixWriter application is deleting a row.	503
Directory Not Allowed	Name of existing directory variable used as argument.	12A
Directory Recursion	Attempted to store a directory into itself.	002
Empty catalog	No data in current catalog (Equation, Statistics, Alarm).	60D
Empty stack	The stack contains no data.	C15

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Enter alarm, press SET	Alarm entry prompt.	61A
Enter eqn, press NEW	Store new equation in EQ.	60A
Enter value (zoom out if >1), press ENTER	Zoom operations prompt.	622
EQ Invalid for MINIT	EQ must contain at least two equations (or programs) and two variables.	E403
Extremum	Result returned by HP Solve application or ROOT is an extremum rather than a root.	A06
HALT Not Allowed	A program containing HALT executed while MatrixWriter application, DRAW, or HP Solve application active.	126
I/O setup menu	Identifies I/O setup menu.	61C
Illegal During MROOT	Multiple-Equation Solver command attempted during MROOT execution.	E406
Implicit () off	Implicit parentheses off.	207
Implicit () on	Implicit parentheses on.	208
Incomplete Subexpression	 ,  , or  pressed before all function arguments supplied.	206

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Inconsistent Units	Attempted unit conversion with incompatible units.	B02
Infinite Result	Math exception: Calculation such as 1/0 infinite result.	305
Inserting Column	MatrixWriter application is inserting a column.	506
Inserting Row	MatrixWriter application is inserting a row.	505
Insufficient Memory	Not enough free memory to execute operation.	001
Insufficient Z Data	A Statistics command was executed when $\Sigma$ DATA did not contain enough data points for calculation.	603
Interrupted	The HP Solve application or ROOT was interrupted by <b>CANCEL</b> .	A03
Invalid Array Element	<b>ENTER</b> returned object of wrong type for current matrix.	502
Invalid Card Data	HP 48 does not recognize data on plug-in card.	008
Invalid Date	Date argument not real number in correct format, or was out of range.	D01
Invalid Definition	Incorrect structure of equation argument for DEFINE.	12C
Invalid Dimension	Array argument had wrong dimensions.	501

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Invalid EQ	Attempted operation from PICTURE FCN menu when EQ did not contain algebraic, or, attempted DRAW with CONIC plot type when EQ did not contain algebraic.	607
Invalid IOPAR	IOPAR not a list, or one or more objects in list missing or invalid.	C12
Invalid Mpar	Mpar variable not created by MINIT.	E401
Invalid Name	Received illegal filename, or server asked to send illegal filename.	C17
Invalid PPAR	PPAR not a list, or one or more objects in list missing or invalid.	12E
Invalid PREPAR	PREPAR not a list, or one or more objects in list missing or invalid.	C13
Invalid PTYPE	Plot type invalid for current equation.	620
Invalid Repeat	Alarm repeat interval out of range.	D03
Invalid Server Cmd.	Invalid command received while in Server mode.	C08
Invalid Syntax	HP 48 unable execute <b>ENTER</b> , OBJ →, or STR → due to invalid object syntax.	106

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Invalid Time	Time argument not real number in correct format, or out of range.	D02
Invalid Unit	Unit operation attempted with invalid or undefined user unit.	B01
Invalid User Function	Type or structure of object executed as user-defined function was incorrect.	103
Invalid $\Sigma$ Data	Statistics command executed with invalid object stored in $\Sigma$ DATA.	601
Invalid $\Sigma$ Data LN(CM)	Non-linear curve fit attempted when $\Sigma$ DATA matrix contained a negative element.	605
Invalid $\Sigma$ Data LN(C)	Non-linear curve fit attempted when $\Sigma$ DATA matrix contained a 0 element.	606
Invalid $\Sigma$ PAR	$\Sigma$ PAR not list, or one or more objects in list missing or invalid.	604
Keyword Conflict	A plug-in card conflicts with an equation library variable. Remove the card to continue.	E303
LAST CMD Disabled	$\rightarrow$ [CMD] pressed while that recovery feature disabled.	125
LAST STACK Disabled	$\rightarrow$ [UNDO] pressed while that recovery feature disabled.	124
LASTARG Disabled	$\rightarrow$ [ARG] executed while that recovery feature disabled.	205

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Low Battery	System batteries too low to safely print or perform I/O.	C14
Memory Clear	HP 48 memory was cleared.	005
Name Conflict	Execution of   (where) attempted to assign value to variable of integration or summation index.	13C
Name the equation; press ENTER	Name equation and store it in EQ.	60B
Name the stat data; press ENTER	Name statistics data and store it in $\Sigma$ DATA.	621
Negative Underflow	Math exception: Calculation returned negative, non-zero result greater than -MINR.	302
No Current Equation	<del>SOLVE</del> , DRAW, or RCEQ executed with nonexistent EQ.	104
No Current equation.	Plot and HP Solve application status message.	609
No Picture Available	No picture is included for the selected equation.	E304
No Room in Port	Insufficient free memory in specified RAM port.	00B
No Room to Save Stack	Not enough free memory to save copy of the stack. LAST STACK is automatically disabled.	101

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
No Room to Show Stack	Stack objects displayed by type only due to low memory condition.	131
No Stat data to Plot	No data stored in $\Sigma DAT$ .	60F
Non-Empty Directory	Attempted to purge non-empty directory.	12B
Non-Real Result	Execution of HP Solve application, ROOT, DRAW, or $\int$ returned result other than real number or unit.	12F
Nonexistent Alarm	Alarm list did not contain alarm specified by alarm command.	D04
Nonexistent $\Sigma DAT$	Statistics command executed when $\Sigma DAT$ did not exist.	602
Object Discarded	Sender sent an EOF (Z) packet with a "D" in the data field.	C0F
Object In Use	Attempted PURGE or STO into a backup object when its stored object was in use.	009
Object Not in Port	Attempted to access a nonexistent backup object or library.	00C
(OFF SCREEN)	Function value, root, extremum, or intersection was not visible in current display.	61F

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Out of Memory	One or more objects must be purged to continue calculator operation.	135
Overflow	Math exception: Calculation returned result greater in absolute value than MAXR.	303
Packet #	Indicates packet number during send or receive.	C10
Parity Error	Received bytes' parity bit doesn't match current parity setting.	C05
Plot Type:	Label introducing current plot type.	61D
Port Closed	Possible I/R or serial hardware failure. Run self-test.	C09
Port Not Available	Used a port command on an empty port, or one containing ROM instead of RAM.	00A
Positive Underflow	Attempted to execute a server command that itself uses the I/O port.	301
Power Lost	Math exception: Calculation returned positive, non-zero result less than MINR. Calculator turned on following a power loss. Memory may have been corrupted.	006


Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Processing Command	Indicates processing of host command packet.	C11
Protocol Error	Received a packet whose length was shorter than a null packet. Maximum packet length parameter from other machine is illegal.	C07
Receive Buffer Overrun	Kermit: More than 255 bytes of retries sent before HP 48 received another packet. SRECV: Incoming data overflowed the buffer.	C04
Receive Error	UART overrun or framing error.	C03
Receiving	Identifies object name while receiving.	C0E
Retry #	Indicates number of retries while retrying packet exchange.	C0B
Select a model	Select statistics curve fitting model.	614
Select plot type	Select plot type.	60C
Select repeat interval	Select alarm repeat interval.	61B
Sending	Identifies object name while sending.	C0D

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Sign Reversal	HP Solve application or ROOT unable to find point at which current equation evaluates to zero, but did find two neighboring points at which equation changed sign.	A05
Single Equation	Only one equation supplied to Multiple-Equation Solver. Use HP Solve application.	E402
Timeout	Printing to serial port: Received XOFF and timed out waiting for XON. Kermit: Timed out waiting for packet to arrive.	C02
Too Few Arguments	Command required more arguments than were available on stack.	201
Too Many Unknowns	Multiple Equation Solver can't calculate a value given the current knowns. Supply another value or add an equation.	E404
Transfer Failed	Ten successive attempts to receive a good packet were unsuccessful.	C06
Unable to find root	PROOT is unable to determine all roots of the polynomial.	C001
Unable to Isolate	ISOL failed because specified name absent or contained in argument a function with no inverse.	130

Messages Listed Alphabetically (continued)

Message	Meaning	# (hex)
Undefined Local Name	Executed or recalled local name for which corresponding local variable did not exist.	003
Undefined Name	Executed or recalled global name for which corresponding variable does not exist.	204
Undefined Result	Calculation such as 0/0 generated mathematically undefined result.	304
Undefined XLIB Name	Executed an XLIB name when specified library absent.	004
Warning:	Label introducing current status message.	007
Wrong Argument Count	User-defined function evaluated with an incorrect number of parenthetical arguments.	128
X and Y-axis zoom.	Identifies zoom option.	627
X axis zoom.	Identifies zoom option.	625
X axis zoom w/AUTO.	Identifies zoom option.	624
Y axis zoom.	Identifies zoom option.	626
ZERO	Result returned by the HP Solve application or ROOT is a root (a point at which current equation evaluates to zero).	A04
""	Identifies no execution action when  pressed.	61E

Messages Listed Numerically

# (hex)	Message
General Messages	
001	Insufficient Memory
002	Directory Recursion
003	Undefined Local Name
004	Undefined XLIB Name
005	Memory Clear
006	Power Lost
008	Invalid Card Data
009	Object In use
00A	Port Not Available
00B	No Room in Port
00C	Object Not in Port
101	No Room to Save Stack
102	Can't Edit Null Char.
103	Invalid User Function
104	No Current Equation
106	Invalid Syntax
124	LAST STACK Disabled
125	LAST CMD Disabled
126	HALT Not Allowed
128	Wrong Argument Count
129	Circular Reference
12A	Directory Not Allowed
12B	Non-Empty Directory
12C	Invalid Definition
12E	Invalid PPAR
12F	Non-Real Result
130	Unable to Isolate
131	No Room to Show Stack
Out-of-Memory Prompts	
135	Out of Memory
13C	Name Conflict

Messages Listed Numerically (continued)

# (hex)	Message
<b>Stack Errors</b>	
201	Too Few Arguments
202	Bad Argument Type
203	Bad Argument Value
204	Undefined Name
205	LASTARG Disabled
<b>Equation Writer Application Messages</b>	
206	Incomplete Subexpression
207	Implicit () off
208	Implicit () on
<b>Floating-Point Errors</b>	
301	Positive Underflow
302	Negative Underflow
303	Overflow
304	Undefined Result
305	Infinite Result
<b>Array Messages</b>	
501	Invalid Dimension
502	Invalid Array Element
503	Deleting Row
504	Deleting Column
505	Inserting Row
506	Inserting Column
<b>Statistics Messages</b>	
601	Invalid Z Data
602	Nonexistent ZDAT
603	Insufficient Z Data
604	Invalid ZPAR
605	Invalid Z Data LN(Neg)
606	Invalid Z Data LN(0)

Messages Listed Numerically (continued)

# (hex)	Message
<b>Plot, I/O, Time and HP Solve Application Messages</b>	
607	Invalid EQ
608	Current equation:
609	No current equation.
60A	Enter eqn, press NEW
60B	Name the equation, press ENTER
60C	Select plot type
60D	Empty catalog
60F	No stat data to plot
610	Autoscaling
614	Select a model
619	Acknowledged
61A	Enter alarm, press SET
61B	Select repeat interval
61C	I/O setup menu
61D	Plot type:
61E	" "
61F	(OFF SCREEN)
620	Invalid PTYPE
621	Name the stat data, press ENTER
622	Enter value (zoom out if >.), press ENTER
623	Copied to stack
624	x axis zoom w/AUTO.
625	x axis zoom.
626	y axis zoom.
627	x and y-axis zoom.
A01	Bad Guess(es)
A02	Constant?
A03	Interrupted
A04	Zero
A05	Sign Reversal
A06	Extremum

Messages Listed Numerically (continued)

# (hex)	Message
<b>Unit Management</b>	
B01	Invalid Unit
B02	Inconsistent Units
<b>I/O and Printing</b>	
C01	Bad Packet Block check
C02	Timeout
C03	Receive Error
C04	Receive Buffer Overrun
C05	Parity Error
C06	Transfer Failed
C07	Protocol Error
C08	Invalid Server Cmd.
C09	Port Closed
C0A	Connecting
C0B	Retry #
C0C	Awaiting Server Cmd.
C0D	Sending
C0E	Receiving
C0F	Object Discarded
C10	Packet #
C11	Processing Command
C12	Invalid IOPAR
C13	Invalid PRTPAR
C14	Low Battery
C15	Empty Stack
C17	Invalid Name

Messages Listed Numerically (continued)

# (hex)	Message
<b>Time Messages</b>	
D01	Invalid Date
D02	Invalid Time
D03	Invalid Repeat
D04	Nonexistent Alarm
<b>Equation Library Messages</b>	
E303	Keyword Conflict
E304	No Picture Available
<b>Multiple-Equation Solver Messages</b>	
E401	Invalid Mpar
E402	Single Equation
E403	EQ Invalid for MINIT
E404	Too Many Unknowns
E405	All Variables Known
E406	Illegal During MROOT
<b>Miscellaneous Messages</b>	
70000	(user-defined message created with DOERR)



**Table of Units**

**HP 48 Units**

Unit (Full Name)	Value in SI Units
a (arc)	100 m <sup>2</sup>
A (ampere)	1 A
acre (acre)	4046.87260987 m <sup>2</sup>
arcmin (minute of arc)	2.90888208666 × 10 <sup>-4</sup> r
arcs (second of arc)	4.8481368111 × 10 <sup>-6</sup> r
atm (atmosphere)	101325 kg/m·s <sup>2</sup>
au (astronomical unit)	1.495979 × 10 <sup>11</sup> m
Å (Angstrom)	1 × 10 <sup>-10</sup> m
b (barn)	1 × 10 <sup>-28</sup> m <sup>2</sup>
bar (bar)	100000 kg/m·s <sup>2</sup>
bb1 (barrel)	.158987294928 m <sup>3</sup>
Bq (becquerel)	1 1/s
Btu (international table Btu)	1055.05585262 kg·m <sup>2</sup> /s <sup>2</sup>
bu (bushel)	.03523907 m <sup>3</sup>
°C (degree Celsius)	1 K or 274.15 K
c (speed of light)	299792458 m/s
C (coulomb)	1 A·s
cal (calorie)	4.1868 kg·m <sup>2</sup> /s <sup>2</sup>
cd (candela)	1 cd
chain (chain)	20.1168402337 m
Ci (curie)	3.7 × 10 <sup>10</sup> 1/s
ct (carat)	.0002 kg
cu (US cup)	2.365882365 × 10 <sup>-4</sup> m <sup>3</sup>
° (degree)	1.74532925199 × 10 <sup>-2</sup> r
d (day)	86400 s

HP 48 Units (continued)

Unit (Full Name)	Value in SI Units
dyn (dyne)	.00001 kg·m/s <sup>2</sup>
erg (erg)	.0000001 kg·m <sup>2</sup> /s <sup>2</sup>
eV (electron volt)	1.60217733 × 10 <sup>-19</sup> kg·m <sup>2</sup> /s <sup>2</sup>
F (farad)	1 A <sup>2</sup> ·s <sup>4</sup> /kg·m <sup>2</sup>
°F (degrees Fahrenheit)	0.55555555555556 K or 255.927777778 K
fath (fathom)	1.82880365761 m
fbm (board foot)	.002359737216 m <sup>3</sup>
fc (footcandle)	10.7639104167 cd·sr/m <sup>2</sup>
Fdy (faraday)	96487 A·s
fermi (fermi)	1 × 10 <sup>-15</sup> m
flam (footlambert)	3.42625909964 cd/m <sup>2</sup>
ft (international foot)	.3048 m
ftUS (survey foot)	.304800609601 m
g (gram)	.001 kg
ga (standard freefall)	9.80665 m/s <sup>2</sup>
gal (US gallon)	.003785411784 m <sup>3</sup>
galC (Canadian gallon)	.00454609m <sup>3</sup>
galUK (UK gallon)	.004546092 m <sup>3</sup>
gf (gram-force)	.00980665 kg·m/s <sup>2</sup>
grad (gradient)	1.57079632679 × 10 <sup>-2</sup> r
grain (grain)	.00006479891 kg
gy (gray)	1 m <sup>2</sup> /s <sup>2</sup>
H (henry)	1 kg·m <sup>2</sup> /A <sup>2</sup> ·s <sup>2</sup>
h (Hour)	3600 s
hp (horsepower)	745.699871582 kg·m <sup>2</sup> /s <sup>3</sup>
Hz (hertz)	1/s
in (inch)	.0254 m
inHg (inches of mercury, 0°C)	3386.38815789 kg/m·s <sup>2</sup>
inH <sub>2</sub> O (inches of water, 60°F)	248.84 kg/m·s <sup>2</sup>
J (joule)	1 kg·m <sup>2</sup> /s <sup>2</sup>
K (kelvins)	1 K
kg (kilogram)	1 kg

HP 48 Units (continued)

Unit (Full Name)	Value in SI Units
kip (kilopound-force)	4448.22161526 kg·m/s <sup>2</sup>
knot (nautical miles per hour)	.514444444444 m/s
kpm (kilometers per hour)	.277777777778 m/s
l (liter)	.001 m <sup>3</sup>
lam (lambert)	3183.09886184 cd/m <sup>2</sup>
lb (avoirdupois pound)	.45359237 kg
lbf (pound-force)	4.44822161526 kg·m/s <sup>2</sup>
lbt (troy pound)	.3732417216 kg
lm (lumen)	1 cd·sr
Lx (lux)	1 cd·sr/m <sup>2</sup>
lyr (light year)	9.46052840488 × 10 <sup>15</sup> m
m (meter)	1 m
μ (micron)	1 × 10 <sup>-6</sup> m
mho (mho)	1 A <sup>2</sup> ·s <sup>3</sup> /kg·m <sup>2</sup>
mi (international mile)	1609.344 m
mil (mil)	.0000254 m
min (minute)	60 s
miUS (US statute mile)	1609.34721869 m
mmHg (millimeter of mercury (torr), 0°C)	133.322368421 kg/m·s <sup>2</sup>
mol (mole)	1 mol
mph (miles per hour)	.44704 m/s
N (newton)	1 kg·m/s <sup>2</sup>
nmi (nautical mile)	1852 m
Ω (ohm)	1 kg·m <sup>2</sup> /A <sup>2</sup> ·s <sup>2</sup>
oz (ounce)	.028349523125 kg
ozFL (US fluid ounce)	2.95735295625 × 10 <sup>-6</sup> m <sup>3</sup>
ozt (troy ounce)	.0311034768 kg
ozUK (UK fluid ounce)	2.8413075 × 10 <sup>-6</sup> m <sup>3</sup>
P (poise)	.1 kg/m·s
Pa (pascal)	1 kg/m·s <sup>2</sup>
pc (parsec)	3.08567818585 × 10 <sup>16</sup> m
pdL (poundal)	.138254954376 kg·m/s <sup>2</sup>
ph (phot)	10000 cd·sr/m <sup>2</sup>
pk (peck)	.0088097675 m <sup>3</sup>

HP 48 Units (continued)

Unit (Full Name)	Value in SI Units
psi (pounds per square inch)	6894.75729317 kg/m.s <sup>2</sup>
pt (pint)	.000473176473 m <sup>3</sup>
qt (quart)	.000946352946 m <sup>3</sup>
r (radian)	1 r
R (roentgen)	.000258 A.s/kg
°R (degrees Rankine)	0.555555555556 K
rad (rad)	.01 m <sup>2</sup> /s <sup>2</sup>
rd (rod)	5.02921005842 m
rem (rem)	.01 m <sup>2</sup> /s <sup>2</sup>
s (second)	1 s
S (siemens)	1 A <sup>2</sup> .s <sup>3</sup> /kg.m <sup>2</sup>
sb (scilb)	10000 cd/m <sup>2</sup>
slug (slug)	14.5939029372 kg
sr (steradian)	1 sr
st (stere)	1 m <sup>3</sup>
St (stokes)	.0001 m <sup>2</sup> /s
Sv (sievert)	1 m <sup>2</sup> /s <sup>2</sup>
t (metric ton)	1000 kg
T (tesla)	1 kg/A.s <sup>2</sup>
tbsp (tablespoon)	1.47867647813 x 10 <sup>-5</sup> m <sup>3</sup>
therm (EEC therm)	105506000 kg.m <sup>2</sup> /s <sup>2</sup>
ton (short ton)	907.18474 kg
tonUK (long ton (UK))	1016.0469088 kg
torr (torr (mmHg))	133.322368421 kg/m.s <sup>2</sup>
tsp (teaspoon)	4.92892159375 x 10 <sup>-6</sup> m <sup>3</sup>
u (unified atomic mass)	1.6605402 x 10 <sup>-27</sup> kg
V (volt)	1 kg.m <sup>2</sup> /A.s <sup>3</sup>
W (watt)	1 kg.m <sup>2</sup> /s <sup>3</sup>
Wb (weber)	1 kg.m <sup>2</sup> /A.s <sup>2</sup>
yd (international yard)	.9144 m
yr (year)	31556925.9747 s

C

System Flags

This appendix lists the HP 48 system flags. You can set, clear, and test all flags. The default state of the flags is *clear*—except for the Binary Integer Wordsize flags (flags -5 through -10).

System Flags

Flag	Description
-1	Principal Solution. <i>Clear</i> : QUAD and ISOL return a result representing all possible solutions. <i>Set</i> : QUAD and ISOL return only the principal solution.
-2	Symbolic Constants. <i>Clear</i> : Symbolic constants (e, i, π, MAXR, and MINR) retain their symbolic form when evaluated, unless the Numerical Results flag -3 is set. <i>Set</i> : Symbolic constants evaluate to numbers, regardless of the state of the Numerical Results flag -3.
-3	Numerical Results. <i>Clear</i> : Functions with symbolic arguments, including symbolic constants, evaluate to symbolic results. <i>Set</i> : Functions with symbolic arguments, including symbolic constants, evaluate to numbers.
-4	Not used.
-5 thru -10	Binary Integer Wordsize. Combined states of flags -5 through -10 set the wordsize from 1 to 64 bits.

System Flags (continued)

Flag	Description
-11 and -12	Binary Integer Base. HEX: -11 set, -12 set. DEC: -11 clear, -12 clear. OCT: -11 set, -12 clear. BIN: -11 clear, -12 set.
-13	Not used.
-14	Financial Payment Mode. Clear: TVM calculations assume end-of-period payments. Set: TVM calculations assume beginning-of-period payments.
-15 and -16	Rectangular: -16 clear. Polar/Cylindrical: -15 clear, -16 set. Polar/Spherical: -15 set, -16 set.
-17 and -18	Degrees: -17 clear, -18 clear. Radians: -17 set. Grads: -17 clear, -18 set.
-19	Clear: →V2 and creates a 2-dimensional vector from 2 real numbers. Set: →V2 and creates a complex number from 2 real numbers.
-20	Underflow Exception. Clear: Underflow exception returns 0, sets flag -23 or -24. Set: Underflow exception treated as an error.
-21	Overflow Exception. Clear: Overflow exception returns ±9.999999999999E499 and sets flag -25. Set: Overflow exception treated as an error.
-22	Infinite Result Exception. Clear: Infinite result exception treated as an error. Set: Infinite result exception returns ±9.999999999999E499 and sets flag -26.
-23	Negative Underflow Indicator.
-24	Positive Underflow Indicator.
-25	Overflow Indicator.
-26	Infinite Result Indicator. When an exception occurs, corresponding flag (-23 through -26) is set only if the exception is not treated as an error.

System Flags (continued)

Flag	Description
-27	Display of symbolic complex numbers. Clear: Displays symbolic complex numbers in coordinate form (i.e. '(x,y)') Set: Displays symbolic complex numbers using 'i' (i.e. 'x+y*i').
-28	Simultaneous Plotting of Multiple Functions. Clear: Multiple equations are plotted serially. Set: Multiple equations are plotted simultaneously.
-29	Draw Axes. Clear: Axes are drawn for two-dimensional and statistical plots. Set: Axes are not drawn for two-dimensional and statistical plots.
-30	Not used.
-31	Curve Filling. Clear: Curve filling between plotted points enabled. Set: Curve filling between plotted points suppressed.
-32	Graphics Cursor. Clear: Graphics cursor always dark. Set: Graphics cursor dark on light background and light on dark background.
-33	I/O Device. Clear: I/O directed to serial port. Set: I/O directed to IR port.
-34	Printing Device. Clear: Printer output directed to IR printer. Set: Printer output directed to serial port if flag -33 is clear.
-35	I/O Data Format. Clear: Objects transmitted in ASCII form. Set: Objects transmitted in binary (memory image) form.

System Flags (continued)

Flag	Description
-36	I/O Receive Overwrite. <i>Clear:</i> If file name received by HP 48 matches existing HP 48 variable name, new variable name with number extension is created to prevent overwrite. <i>Set:</i> If file name received by HP 48 matches existing HP 48 variable name, existing variable is overwritten.
-37	Double-Spaced Printing. <i>Clear:</i> Single-spaced printing. <i>Set:</i> Double-spaced printing.
-38	Line Feed. <i>Clear:</i> Linefeed added at end of each print line. <i>Set:</i> No linefeed added at end of each print line.
-39	I/O Messages. <i>Clear:</i> I/O messages displayed. <i>Set:</i> I/O messages suppressed.
-40	Clock Display. <i>Clear:</i> Clock displayed only when TIME menu selected. <i>Set:</i> Ticking clock displayed at all times.
-41	Clock Format. <i>Clear:</i> 12-hour clock. <i>Set:</i> 24-hour clock.
-42	Date Format. <i>Clear:</i> MM/DD/YY (month/day/year) format. <i>Set:</i> DD.MM.YY (day.month.year) format.
-43	Repeat Alarms Not Rescheduled. <i>Clear:</i> Unacknowledged repeat appointment alarms automatically rescheduled. <i>Set:</i> Unacknowledged repeat appointment alarms not rescheduled.
-44	Acknowledged Alarms Saved. <i>Clear:</i> Acknowledged appointment alarms deleted from alarm list. <i>Set:</i> Acknowledged appointment alarms saved in alarm list.

System Flags (continued)

Flag	Description
-45 thru -48	Number of Decimal Digits. Combined states of flags -45 through -48 sets number of decimal digits in Fix, Scientific, and Engineering modes.
-49 and -50	Number Display Format. <i>Standard:</i> -49 <i>clear</i> , -50 <i>clear</i> . <i>Fix:</i> -49 <i>set</i> , -50 <i>clear</i> . <i>Scientific:</i> -49 <i>clear</i> , -50 <i>set</i> . <i>Engineering:</i> -49 <i>set</i> , -50 <i>set</i> .
-51	Fraction Mark. <i>Clear:</i> Fraction mark is . (period). <i>Set:</i> Fraction mark is , (comma).
-52	Single-Line Display. <i>Clear:</i> Display gives preference to object in level 1, using up to four lines of stack display. <i>Set:</i> Display of object in level 1 restricted to one line.
-53	Precedence. <i>Clear:</i> Certain parentheses in algebraic expressions suppressed to improve legibility. <i>Set:</i> All parentheses in algebraic expressions displayed.
-54	Tiny Array Elements. <i>Clear:</i> Singular values computed by RANK (and other commands that compute the rank of a matrix) that are more than $1 \times 10^{-14}$ times smaller than the largest computed singular value in the matrix are converted to zero. <i>Automatic rounding for DET is enabled.</i> <i>Set:</i> Small computed singular values (see above) not converted. Automatic rounding for DET is disabled.
-55	Last Arguments. <i>Clear:</i> Command arguments saved. <i>Set:</i> Command arguments not saved.
-56	Error Beep. <i>Clear:</i> Error and BEEP-command beeps enabled. <i>Set:</i> Error and BEEP-command beeps suppressed.

### System Flags (continued)

Flag	Description
-57	Alarm Beep. <i>Clear:</i> Alarm beep enabled. <i>Set:</i> Alarm beep suppressed.
-58	Verbose Messages. <i>Clear:</i> Parameter variable data automatically displayed. <i>Set:</i> Automatic display of parameter variable data is suppressed.
-59	Fast Browser Display. <i>Clear:</i> Variable Browser shows variable names and contents. <i>Set:</i> Variable Browser shows variable names only.
-60	Alpha Lock. <i>Clear:</i> Single-Alpha activated by pressing $\alpha$ once. Alpha lock activated by pressing $\alpha$ twice. <i>Set:</i> Alpha lock activated by pressing $\alpha$ once. (Single-Alpha not available.)
-61	User-Mode Lock. <i>Clear:</i> 1-User mode activated by pressing $\leftarrow$ (USER) once. User mode activated by pressing $\leftarrow$ (USER) twice. <i>Set:</i> User mode activated by pressing $\leftarrow$ (USER) once. (1-User mode not available.)
-62	User Mode. <i>Clear:</i> User mode not active. <i>Set:</i> User mode active.
-63	Vectored (ENTER). <i>Clear:</i> (ENTER) evaluates command line. <i>Set:</i> User-defined (ENTER) activated.
-64	Index Wrap Indicator. <i>Clear:</i> Last execution of GETI or PUTI did not increment index to first element. <i>Set:</i> Last execution of GETI or PUTI did increment index to first element.

# D

## Reserved Variables

The HP 48 uses the following *reserved variables*. These have specific purposes, and their names are used as implicit arguments for certain commands. Avoid using these variables' names for other purposes, or you may interfere with the execution of the commands that use these variables.

You can change some of the values in these variables with programmable commands, while others require you to store new values into the appropriate place.

Reserved Variable	What It Contains	Used By
ALRMDAT	Alarm parameters.	TIME ALRM operations
CST	List defining the CST (custom) menu.	MENU, (CST)
"der"-names	User-defined derivative.	$\partial$
EQ	Current equation.	ROOT, DRAW
EXPR	Current expression.	SYMBOLIC
IOPAR	I/O parameters.	I/O commands
MHpar	Minehunt game status.	MINEHUNT
Mpar	Multiple-Equation Solver equations.	EQ LIB
$n1, n2, \dots$	Arbitrary integers.	ISOL, QUAD
Nmines	Minehunt game data.	MINEHUNT

Reserved Variable	What It Contains	Used By
PPAR	Plotting parameters.	DRAW
PRTPAR	Printing parameters.	PRINT commands
s1, s2, ...	Arbitrary signs.	ISOL, QUAD
VPAR	Viewing parameters.	DRAW
ZPAR	Plot zoom factors.	DRAW
ΣDAT	Statistical data.	Statistics application, DRAW
EPAR	Statistical parameters.	Statistics application, DRAW

## Contents of the Reserved Variables

Most reserved variables (except *ALRMDAT*, *IOPAR* and *PRTPAR*) can be stored with different contents in different directories. This allows you, for example, to save several sets of statistical data in different directories.

### ALRMDAT

*ALRMDAT* does not reside in a particular directory. You cannot access the variable itself, but you can access its data from any directory using the *RCLALARM* and *STOALARM* commands, or through the Alarm Catalog.

*ALRMDAT* contains a list of these alarm parameters:

Parameter (Command)	Description	Default Value
<i>date</i> (→DATE)	A real number specifying the date of the alarm: <i>MM.DDYYYY</i> (or <i>DD.MYYYY</i> if flag -42 is set). If <i>YYYY</i> is not included, the current year is used.	Current date.
<i>time</i> (→TIME)	A real number specifying the time of the alarm: <i>HH.MMSS</i> .	00.0000
<i>action</i>	A string or object: <ul style="list-style-type: none"> <li>■ a string creates an <i>appointment alarm</i>, which beeps and displays the string</li> <li>■ any other object creates a <i>control alarm</i>, which executes the object</li> </ul>	Empty string (appointment alarm).
<i>repeat</i>	A real number specifying the interval between automatic recurrences of the alarm, given in ticks (a tick is $\frac{1}{8192}$ of a second).	0

Parameters without commands can be modified with a program by storing new values in the list contained in *ALRMDAT* (use the *PUT* command).

### CST

*CST* contains a list (or a name specifying a list) of the objects that define the *CST* (*custom*) menu. Objects in the custom menu behave as do objects in built-in menus. For example:

- Names behave like the VAR menu keys. Thus, if *ABC* is a variable name, **FEV** evaluates *ABC*, **CR** recalls its contents, and **GV** stores new contents in *ABC*.
- The menu label for the name of a directory has a bar over the left side of the label; pressing the menu key switches to that directory.
- Unit objects act like unit catalog entries (and have left-shifted conversion capabilities, for example).

- String keys echo the string.
- You can include backup objects in the list defining a custom menu by tagging the name of the backup object with its port location (0 through 33).

You can specify menu labels and key actions independently by replacing a single object within the custom-menu list with a list of the form { "label-object", action-object *i*. (See "Customizing Menus" and "Enhancing Custom Menus" in chapter 30 of the *HP 48 User's Guide* for more information.)

To provide different shifted actions for custom menu keys, *action-object* can be a list containing three action objects in this order:

- The unshifted action (required if you want to specify the shifted actions).
- The left-shifted action.
- The right-shifted action.

See "Enhancing Custom Menus" in chapter 30 of the *HP 48 User's Guide*.

### "der-" Names

If  $\partial$  is applied to a function for which there is no built-in derivative, it returns a new function whose name is "der" followed by the original function name. These "der"-function names are reserved variable names.

For an example, refer to "Creating User-Defined Derivatives" in chapter 20 of the *HP 48 User's Guide*.

### EQ

*EQ* contains the current equation or the name of the variable containing the current equation.

*EQ* supplies the equation for *ROOT*, as well as for the plotting command *DRAW* when the plot type is *FUNCTION*, *CONIC*, *POLAR*, *PARAMETER*, *TRUTH*, or *DIFFEQ*. (*ΣDAT* supplies the information when the plot type is *HISTOGRAM*, *BAR*, or *SCATTER*.)

The object in *EQ* can be an algebraic object, a number, a name, or a program. How *DRAW* interprets *EQ* depends on the plot type.

For graphics use, *EQ* can also be a list of equations or other objects. If *EQ* contains a list, then *DRAW* treats each object in turn as the current equation, and plots them successively. However, *ROOT* in the *HP Solve* application *cannot* solve an *EQ* containing a list.

To alter the contents of *EQ*, use the command *STEQ*.

### EXPR

*EXPR* contains the current algebraic expression (or the name of the variable containing the current expression) used by the *SYMBOLIC* application and its associated commands. The object in *EQ* must be an algebraic or a name.

### IOPAR

*IOPAR* is a variable in the *HOME* directory that contains a list of the I/O parameters needed for a communications link with a computer. It is created the first time you transfer data or open the serial port (*OPENIO*), and is automatically updated whenever you change the I/O settings. All *IOPAR* parameters are integers.

Parameter (Command)	Description	Default Value
<i>baud</i> (BAUD)	The baud rate: 1200, 2400, 4800, or 9600.	9600
<i>parity</i> (PARITY)	The parity used: 0=none, 1=odd, 2=even, 3=mark, 4=space. The value can be positive or negative: a positive parity is used upon both transmit and receive; a negative parity is used only upon transmit.	0



Parameter (Command)	Description	Default Value
<i>receive pacing</i>	Controls whether receive pacing is used: a nonzero real value enables pacing, while zero disables it. Receive pacing sends an XOFF signal when the receive buffer is almost full, and sends an XON signal when it can take more data again. Pacing is not used for Kermit I/O, but is used for other serial I/O transfers.	0 (no pacing)
<i>transmit pacing</i>	Controls whether transmit pacing is used: a nonzero real value enables pacing, while zero disables it. Transmit pacing stops transmission upon receipt of XOFF, and resumes transmission upon receipt of XON. Pacing is not used for Kermit I/O, but is used for other serial I/O transfers.	0 (no pacing)
<i>checksum (CKSM)</i>	Error-detection scheme requested when initiating SEND: <ul style="list-style-type: none"> <li>■ 1=1-digit arithmetic checksum</li> <li>■ 2=2-digit arithmetic checksum</li> <li>■ 3=3-digit cyclic redundancy check.</li> </ul>	3
<i>translation code (TRANSIO)</i>	Controls which characters are translated: <ul style="list-style-type: none"> <li>■ 0=none</li> <li>■ 1=translate character 10 (line feed only) to/from characters 10 and 13 (line feed and carriage return)</li> <li>■ 2=translate characters with numbers 128 through 159 (80-9F hex)</li> <li>■ 3=translate characters with numbers 128 through 255.</li> </ul>	1

Parameters without commands can be modified with a program by storing new values in the list contained in *IOPAR* (use the *PUT* command), or by editing *IOPAR* directly.

## MHpar

*MHpar* stores the status of an interrupted Minehunt game. *MHpar* is created when you exit Minehunt by pressing (SIO). If *MHpar* still exists when you restart Minehunt, the interrupted game resumes and *MHpar* is purged.

## Mpar

*Mpar* is created when you use the Equation Library's Multiple-Equation Solver, and it stores the set of equations you're using.

When the Equation Library starts the Multiple-Equation Solver, it first stores a list of the equation set in *EQ*, and stores the equation set, a list of variables, and additional information in *Mpar*. *Mpar* is then used to set up the Solver menu for the current equation set.

*Mpar* is structured as library data dedicated to the Multiple Equation Solver application. This means that although you can view and edit *Mpar* directly, you can edit it only indirectly by executing commands that modify it.

You can also use the MINIT command (EQ LIB) to create *MHpar* from a set of equations on the stack. See "Defining a Set of Equations" in chapter 25 of the *HP 48 User's Guide*.

## n1, n2, ...

The ISOL and QUAD commands return *general* solutions (as opposed to *principal* solutions) for operations. A general solution contains variables for arbitrary integers or arbitrary signs, or both.

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.

## Nmines

*Nmines* is a variable you create in the current directory to control the number of mines used in the Minehunt game. *Nmines* contains an integer in the range 1 to 64; if *Nmines* is negative, the mines are visible during the game.

## PPAR

*PPAR* is a variable in the current directory. It contains a list of plotting parameters used by the DRAW command for all mathematical and statistical plots, by AUTO for autoscaling, and by the interactive (nonprogrammable) graphics operations.

Parameter (Command)	Description	Default Value
$(x_{min}, y_{min})$ (XRNG, YRNG)	A complex number specifying the lower left corner of <i>PICT</i> (the lower left corner of the display range).	(-6.50, -3.1)
$(x_{max}, y_{max})$ (XRNG, YRNG)	A complex number specifying the upper right corner of <i>PICT</i> (the upper right corner of the display range).	(6.5, 3.2)
<i>indep</i> (INDEP)	A name specifying the independent variable, or a list containing that name and two numbers that specify the minimum and maximum values for the independent variable (the plotting range).	X

Parameter (Command)	Description	Default Value
<i>res</i> (RES)	Resolution. A real number specifying the interval between values of the independent variable. For plots of equations, this determines the plotting interval along the <i>x</i> -axis. A binary number specifies the <i>pixel</i> resolution (how many columns of pixels between points). An integer specifies the resolution in <i>user</i> units (how many user units between points). Resolution for statistical plots is different (see below).	0
<i>axes</i> (AXES)	A complex number specifying the user-unit coordinates of the plot origin, or a list containing the following: <ul style="list-style-type: none"> <li>■ the complex number specifying the origin</li> <li>■ a real number, binary integer, or list containing two real numbers or binary integers specifying the tick-mark annotation (see <i>ATICK</i>)</li> <li>■ two strings specifying labels for the horizontal and vertical axes</li> </ul>	(0, 0)
<i>pctype</i> (BAR, etc.)	A command name specifying the plot type (BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, or YSLICE).	FUNCTION

Parameter (Command)	Description	Default Value
<i>depend</i> (DEPND)	A name specifying the dependent variable, or a list containing the name and two numbers that specify vertical plotting range. For DIFFEQ, the second element of the list may also be a real vector that represents the initial value.	Y

Parameters without commands can be modified with a program by storing new values in the list contained in *PPAR* (use the *PUT* command).

The **RESET** operation (**↵** **PLOT** **RESET**) resets the *PPAR* parameters (except *ptype*) to their default values, and erases *PICT*.

Note that *res* behaves differently for the statistical plot types *BAR*, *HISTOGRAM*, and *SCATTER* than for other plot types. For *BAR*, *res* specifies bar width; for *HISTOGRAM*, *res* specifies bin width; *res* does not affect *SCATTER*.

## PRTPAR

*PRTPAR* is a variable in the *HOME* directory that contains a list of printing parameters. It is created automatically the first time you use a printing command.

Parameter (Command)	Description	Default Value
<i>delay time</i> (DELAY)	A real number, in the range 0 to 6.9, specifying the number of seconds the HP 48 waits between sending lines. This should be at least as long as the time required to print the longest line. If the delay is too short for the printer, you will lose data. The delay setting also affects serial printing if transmit-pacing (in <i>IOPAR</i> ) is not being used.	1.8
<i>remap</i> (OLDPRT stores the character-remapping string for the HP 82240A Infrared Printer)	A string defining the current remapping of the extended character set for printing. The string can contain as many characters as you want to remap, with the first character being the new character 128, the second being the new character 129, etc. (Any character number that exceeds the string length will not be remapped.) See the example below.	Empty string.
<i>line length</i>	A real number specifying the number of characters in a line for serial printing. This does <i>not</i> affect infrared printing.	80
<i>line termination</i>	A string specifying the line-termination method for serial printing. This does <i>not</i> affect infrared printing.	Control characters 13 (carriage return) and 10 (line feed).

Parameters without commands can be modified with a program by storing new values in the list contained in *PRTPAR* (use the *PUT* command).

A change in a parameter is effective immediately, *except* when printing the display using the simultaneous keystrokes **ON** **I/O** (because this does not use *PRTPAR*). This printing method is affected only by the delay parameter, a change in which will not affect **ON** **I/O** until after the next printing command has been executed. To use a new delay time with **ON** **I/O** immediately, use the *DELAY* command.

**Example:** If the remapping string were "ABCDEF~~GH~~" and the character to be printed had value 131, then the character actually printed would be "D", since 131-128=3 and "A" has the value zero. A character code of 136 or greater would not be remapped since 136-128=8, which exceeds the length of the string.

### **s1, s2, ...**

The *ISOL* and *QUAD* commands return *general* solutions (as opposed to *principal* solutions) for operations. A general solution contains variables for arbitrary integers or arbitrary signs or both.

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 +1.

## **VPAR**

*VPAR* is a variable in the current directory. It contains a list of parameters used by the 3D plot types. The main data structure stored in *VPAR* describes the "view volume," the abstract three-dimensional region in which the function is plotted.

Parameter (Command)	Description	Default Value
( <i>x</i> <sub>left</sub> , <i>x</i> <sub>right</sub> ) (XVOL)	Real numbers that specify the width of the view volume.	(-1, 1)
( <i>y</i> <sub>far</sub> , <i>y</i> <sub>near</sub> ) (YVOL)	Real numbers that specify the depth of the view volume.	(-1, 1)
( <i>z</i> <sub>low</sub> , <i>z</i> <sub>high</sub> ) (ZVOL)	Real numbers that specify the height of the view volume.	(-1, 1)
( <i>x</i> <sub>eye</sub> , <i>y</i> <sub>eye</sub> , <i>z</i> <sub>eye</sub> ) (EYEPT)	Real numbers that specify the point in space from which the plot is viewed.	(0, -3, 0)
( <i>x</i> <sub>step</sub> , <i>y</i> <sub>step</sub> ) (NUMX, NUMY)	Real numbers that specify the increments between of x-coordinates and y-coordinates plotted. The increments are equal to the range for the axes divided by the number of steps. Used instead of (or in combination with) <i>res</i> .	(10, 8)
( <i>xx</i> <sub>left</sub> , <i>xx</i> <sub>right</sub> ) (XXRNG)	Real numbers that specify the width of the input plane (domain). Used by <i>GRIDMAP</i> and <i>PARSURFACE</i> .	(-1, 1)
( <i>yy</i> <sub>far</sub> , <i>yy</i> <sub>near</sub> ) (YYRNG)	Real numbers that specify the depth of the input plane (domain). Used by <i>GRIDMAP</i> and <i>PARSURFACE</i> .	(-1, 1)


Parameters without commands can be modified programmatically by storing new values in the list contained in *VPAR* (use the *PUT* command).

The **RESET** operation (**←** **PLOT** **NXT** **VIEW** **NXT**) (**RESET**) resets the *VPAR* parameters to their default values.

## ZPAR

*ZPAR* is a variable in the current directory. It contains a list of zooming parameters used by the DRAW command for all 2-D mathematical and statistical plots.

Parameter (Command)	Description	Default Value
<i>h-factor</i>	Real number that specifies the horizontal zoom factor.	4
<i>v-factor</i>	Real number that specifies the vertical zoom factor.	4
<i>recenter flag</i>	0 or 1 depending on whether the recenter at crosshairs option was selected in the set zoom factors input form.	0
{ list }	An empty list, or a copy of the last <i>PPAR</i> .	

Use the set zoom factors input form (  ) to modify *ZPAR*.

## ΣDAT

*ΣDAT* is a variable in the current directory that contains either the current statistical matrix or the name of the variable containing this matrix. This matrix contains the data used by the Statistics applications.

### Statistical Matrix for Variables 1 to m

<i>var1</i>	<i>var2</i>	...	<i>varm</i>
$x_{11}$	$x_{21}$	...	$x_{m1}$
$x_{12}$	$x_{22}$	...	$x_{m2}$
⋮			
$x_{1n}$	$x_{2n}$	...	$x_{mn}$

You can designate a new current statistical matrix by entering new data, editing the current data, or selecting another matrix.

The command CLEΣ clears the current statistical matrix.

## ΣPAR

ΣPAR is a variable in the current directory that contains either the current statistical parameter list or the name of the variable containing this list.

Parameter (Command)	Description	Default Value
<i>column<sub>indep</sub></i> (XCOL)	A real number specifying the independent-variable's column number.	1
<i>column<sub>dep</sub></i> (YCOL)	A real number specifying the dependent-variable's column number.	2
<i>intercept</i> (LR)	A real number specifying the coefficient of intercept as determined by the current regression.	0
<i>slope</i> (LR)	A real number specifying the coefficient of slope as determined by the current regression.	0
<i>model</i> (LINFIT, EXPFIT, PWRFIT, or LOGFIT)	A command specifying the regression model (LINFIT, EXPFIT, PWRFIT, or LOGFIT).	LINFIT

## New Commands

In the following tables, new commands (commands that were not available on a standard HP 48S series calculator) are arranged alphabetically and followed by brief descriptions. All of these commands are described in chapter 3.

### New Commands Listed Alphabetically

Command	Brief Description
ADD	Adds list elements.
AMORT	Amortizes a loan or investment based upon the current amortization settings.
ANIMATE	Displays graphic objects in sequence.
ATTCK	Sets the axes tick-mark annotation in the reserved variable <i>PPAR</i> .
CHOOSE	Creates a user-defined choose box.
CLTEACH	Removes the <i>EXAMPLES</i> subdirectory and its contents from the <i>HOME</i> directory.
COL+	Inserts an array (vector or matrix) into a matrix.
COL-	Deletes a column from a matrix.
COL→	Transforms a series of column vectors and a column count into a matrix, or transforms a sequence of numbers and an element count into a vector.
→COL	Transforms a matrix into a series of column vectors, or transforms a vector into its elements.

New Commands Listed Alphabetically (continued)

Command	Brief Description
COND	Returns the 1-norm (column norm) condition number of a square matrix.
CONLIB	Opens the Constants Library catalog.
CONST	Returns the value of a constant.
CSWP	Swaps columns in a matrix.
CYLIN	Sets Cylindrical coordinate mode.
DARCY	Calculates the Darcy friction factor of certain fluid flows.
DIAG →	Takes an array and a specified dimension and returns a matrix whose main diagonal elements are the elements of the array.
→DIAG	Returns a vector that contains the major diagonal elements of a matrix.
DIFFEQ	Specifies differential equations as the plot type.
DOLIST	Applies commands, programs, or user-defined functions to lists.
DOSUBS	Applies a program or command to groups of elements in a list.
EGV	Computes the eigenvalues and right eigenvectors for a square matrix.
EGVL	Computes the eigenvalues of a square matrix.
ENDSUB	Provides a way to access the total number of sublists used while executing a program or command using DOSUBS.
EQNLIB	Starts the Equation Library application.
EYEPT	Specifies the coordinates of the eye point in a perspective plot.

New Commands Listed Alphabetically (continued)

Command	Brief Description
FOA	Returns the fraction of total black-body emissive power.
FANNING	Calculates the Fanning friction factor of certain fluid flows.
FFT	Computes the one- or two-dimensional discrete Fourier transform of an array.
FREE1	Frees the previously merged RAM in port 1.
GRIDMAP	Specifies grid mapping as the plot type.
HEAD	Returns the first element of a list or string.
IFFT	Computes the one- or two-dimensional inverse discrete Fourier transform of a vector or matrix.
INFORM	Creates a user-defined dialog box (Input Form).
LIBEVAL	Evaluates unnamed library objects by their memory addresses.
LININ	Tests whether an algebraic is structurally linear for a given variable.
ΣLIST	Returns the sum of the elements in a list.
ΠLIST	Returns the product of the elements in a list.
ΔLIST	Returns the set of first differences.
LQ	Returns the LQ factorization of an $n \times m$ matrix.
LSQ	Returns the minimum norm least squares solution to any system of linear equations.
LU	Returns the LU decomposition of a square matrix.
MCALC	Designates a variable as a calculated value (not user-defined).
MERGE1	Merges the RAM from the card in port 1 with the rest of main user memory.

New Commands Listed Alphabetically (continued)

Command	Brief Description
MINEHUNT	Starts the MINEHUNT game.
MINIT	Creates the reserved variable <i>Mpar</i> .
MITM	Changes multiple equation menu titles and order.
MROOT	Solves for one or more variables.
MSGBOX	Creates a user-defined message box.
MSOLVR	Gets the Multiple-Equation Solver variable menu for the set of equations defined by <i>Mpar</i> .
MUSER	Designates a variable as user-defined.
NDIST	Returns the normal probability distribution.
NOVAL	Place holder for reset and initial values in user-defined dialog boxes.
NSUB	Provides a way to access the current sublist number during an iteration of a program or command applied using DOSUBS.
NUMX	Sets the number of x-steps for each y-step in 3D perspective plots.
NUMY	Sets the number of y-steps across the view volume in 3D perspective plots.
PARSURFACE	Specifies 3D parameterized surface grip mapping as the plot type.
PCOEF	Returns the coefficients of a monic polynomial.
PCOV	Calculates population covariance.
PCONTOUR	Specifies pseudo-contour as the plot type.
PEVAL	Evaluates an <i>n</i> -degree polynomial at <i>x</i> .
PINIT	Initializes the plug-in card ports.
PROOT	Returns all roots of an <i>n</i> -degree polynomial having real or complex coefficients.

New Commands Listed Alphabetically (continued)

Command	Brief Description
PSDEV	Calculates population standard deviation.
PVAR	Calculates population variance.
QR	Returns the QR factorization of an $n \times m$ matrix.
RANK	Returns the rank of a rectangular matrix.
RANM	Returns a matrix of random integers.
RCI	Multiplies a row of a matrix by a constant.
RCIJ	Multiplies a row of a matrix by a constant, and then adds the product to another row of the matrix.
RECT	Sets Rectangular coordinate mode.
REVLIST	Reverses the order of the elements in a list.
RKF	Computes the solution to an initial value problem for a differential equation, using the Runge-Kutta-Fehlberg method.
RKFERR	Returns the absolute error estimate for a given step when solving an initial value problem for a differential equation (using RKF method).
RKFSTEP	Computes the next solution step to an initial value problem for a differential equation.
ROW+	Inserts an array into a matrix.
ROW-	Deletes a row from a matrix.
RREF	Converts a rectangular matrix to reduced row echelon form.
RRK	Computes the solution to an initial value problem for a differential equation with known partial derivatives.
RRKSTEP	Computes the next solution step to an initial value problem for a differential equation, and displays the method used to arrive at that result.



New Commands Listed Alphabetically (continued)

Command	Brief Description
RSBERR	Returns an error estimate for a given step when solving an initial values problem for a differential equation (using the Rosenbrock method).
RSWP	Swaps rows in a matrix.
SCHUR	Returns the Schur decomposition of a square matrix.
SEQ	Returns a list of results generated by repeatedly executing an object on a specified range of elements.
SIDENS	Calculates the intrinsic density of silicon as a function of temperature.
SLOPEFIELD	Specifies slopefield as the plot type.
SNRM	Returns the spectral norm of an array.
SOLVEQN	Starts the solver for a specified set of equations.
SORT	Sorts the elements in a list in ascending order.
SPHERE	Sets Spherical coordinate mode.
SRAD	Returns the spectral radius of a square matrix.
STREAM	Applies an object to every element in a list.
SVD	Returns the singular value decomposition of an $n \times m$ matrix.
SVL	Returns the singular values of an $m \times n$ matrix.
TAIL	Returns all but the first element of a list or string.
TDELTA	Calculates a temperature change.
TEACH	Creates an EXAMPLES subdirectory in the HOME directory and loads HP 48 programming, graphing, and solver examples from ROM into it.
TINC	Calculates a temperature increment.
TRACE	Returns the trace of a square matrix.

New Commands Listed Alphabetically (continued)

Command	Brief Description
TVM	Start the TVM solver.
TVMBEG	Specifies that payments are made at the beginning of compounding periods.
TVMEND	Specifies that payments are made at the end of compounding periods.
TVMROOT	Solves for the specified TVM variable using values from the remaining TVM variables.
VERSION	Returns the software version and copyright message.
WIREFRAME	Specifies wireframe as the plot type.
XRECV	Receives an object via XModem.
XSEND	Sends an object via XModem.
XVOL	Sets the width of the view volume in the reserved variable <i>VPAR</i> .
XXRNG	Specifies the x range of an input plane (domain) for GRIDMAP and PARSURFACE plots.
YSLICE	Specifies y-slice cross sections as the plot type.
YVOL	Sets the depth of the view volume in the reserved variable <i>VPAR</i> .
YYRNG	Specifies the y range of an input plane (domain) for GRIDMAP and PARSURFACE plots.
ZFACTOR	Calculates the the gas compressibility correction factor for nonideal behavior of a hydrocarbon gas.
ZVOL	Sets the height of the view volume in the reserved variable <i>VPAR</i> .

## Technical Reference

---

This appendix contains the following information:

- Object sizes.
- Mathematical simplification rules used by the HP 48.
- Symbolic differentiation patterns used by the HP 48.
- The EquationWriter's expansion rules.
- References used as sources for constants and equations in the HP 48 (other than those in the Equation Library).

## Object Sizes

The following table lists object types and their size in bytes. (Note that characters in names, strings, and tags use 1 byte each.)

Object Type	Object Size	Size (bytes)
Algebraic		5 + size of included objects
Backup Object		12 + number of name characters + size of included object
Binary Integer		13
Command		2.5
Complex matrix		15 + 16 × number of elements
Complex number		18.5
Complex vector		12.5 + 16 × number of elements
Directory		6.5 + size of included variables
Graphics Object		10 + number of rows × CELL(columns/8)
List		5 + size of included objects
Matrix		15 + 8 × number of elements
Program		12.5 + size of included objects
Quoted global or local name		8.5 + number of characters
Real number		10.5
String		5 + number of characters
Tagged Object		3.5 + number of tag characters + size of untagged object
Unit Object		7.5 +
real magnitude		2.5 or 10.5
each prefix		6
each unit name		5 + number of characters
each x, y, or z		2.5
each exponent		2.5 or 10.5
Unquoted global or local name		3.5 + number of characters
Vector		12.5 + 8 × number of elements
XLIB name		5.5

## Automatic Simplification Rules

The following tables list the automatic simplification rules for the HP 48.

### Addition and Subtraction

Object	Simplified	Object	Simplified
$x-x$	0	$x+(0,0)$	$x$
$0+x$	$x$	$x+-p$	$x-p$
$(0,0)+x$	$x$	$x-0$	$x$
$0-x$	NEG( $x$ )	$x-(0,0)$	$x$
$(0,0)-x$	NEG( $x$ )	$x--p$	$x+p$
$x+0$	$x$		

### Multiplication and Division

Object	Simplified	Object	Simplified
INV( $i$ )	$-i$	$x \times (1,0)$	$x$
$y \times \text{INV}(x)$	$y/x$	$x \times (-1)$	NEG( $x$ )
$y/\text{INV}(x)$	$y \times x$	$x \times (-1,0)$	NEG( $x$ )
$0 \times x$	0	$x/1$	$x$
$(0,0) \times x$	(0,0)	$x/(1,0)$	$x$
$i \times i$	-1	$x/(-1)$	NEG( $x$ )
$1 \times x$	$x$	$x/(-1,0)$	NEG( $x$ )
$(1,0) \times x$	$x$	$0/x$	0
$(-1) \times x$	NEG( $x$ )	$(0,0)/x$	(0,0)
$(-1,0) \times x$	NEG( $x$ )	$1/x$	INV( $x$ )
$x \times 0$	0	$(1,0)x$	INV( $x$ )
$x \times (0,0)$	(0,0)	$(-1)/x$	-INV( $x$ )
$x \times 1$	$x$	$(-1,0)/x$	-INV( $x$ )

**Powers**

Object	Simplified	Object	Simplified
$I^x$	1	$x^{(1,0)}$	$x$
$(1,0)^x$	(1,0)	$x^{(-1)}$	INV( $x$ )
SQ( $\sqrt{x}$ )	$x$	$x^{(-1,0)}$	INV( $x$ )
SQ( $y^x$ )	$y^{(2 \times x)}$	$(\sqrt{x})^2$	$x$
SQ( $i$ )	-1	$(\sqrt{x})^{(2,0)}$	$x$
$x^0$	1	$i^2$	-1
$x^{(0,0)}$	(1,0)	$i^{(2,0)}$	(-1,0)
$x^{-1}$	$x$		

**Parts**

Object	Simplified	Object	Simplified
ABS(ABS( $x$ ))	ABS( $x$ )	MIN( $x, x$ )	$x$
ABS(NEG( $x$ ))	ABS( $x$ )	MOD(0, $x$ )	0
CONJ(CONJ( $x$ ))	$x$	MOD( $x, x$ )	0
CONJ(IM( $x$ ))	IM( $x$ )	MOD( $x, 0$ )	$x$
CONJ(RE( $x$ ))	RE( $x$ )	$x$ MOD $y$ MOD $y$	$x$ MOD $y$
CONJ( $i$ )	- $i$	RE(CONJ( $x$ ))	RE( $x$ )
IM(CONJ( $x$ ))	-IM( $x$ )	RE(IM( $x$ ))	IM( $x$ )
IM(IM( $x$ ))	0	RE(RE( $x$ ))	RE( $x$ )
IM(RE( $x$ ))	0	RE( $\pi$ )	$\pi$
IM( $p$ )	0	RE( $i$ )	0
IM( $i$ )	1	SIGN(SIGN( $x$ ))	SIGN( $x$ )
MAX( $x, x$ )	$x$		

**Symbolic Integration Patterns**

This table lists the symbolic integration patterns used by the HP 48. These are the integrands that the HP 48 can integrate symbolically.  $\phi$  is a linear function of the variable of integration. The antiderivatives should be divided by the first-order coefficient in  $\phi$  to reduce the expression to its simplest form. Also, patterns beginning with 1/ match INV; for example, 1/ $\phi$  is the same as INV( $\phi$ ).

**Symbolic Integration**

Pattern	Antiderivative
ACOS( $\phi$ )	$\phi \times \text{ACOS}(\phi) - \sqrt{1 - \phi^2}$
ALOG( $\phi$ )	.434294481904 $\times$ ALOG( $\phi$ )
ASIN( $\phi$ )	$\phi \times \text{ASIN}(\phi) + \sqrt{1 - \phi^2}$
ATAN( $\phi$ )	$\phi \times \text{ATAN}(\phi) - \text{LN}(1 + \phi^2) / 2$
COS( $\phi$ )	SIN( $\phi$ )
1/(COS( $\phi$ ) $\times$ SIN( $\phi$ ))	LN(TAN( $\phi$ ))
COSH( $\phi$ )	SINH( $\phi$ )
1/(COSH( $\phi$ ) $\times$ SINH( $\phi$ ))	LN(TANH( $\phi$ ))
1/(COSH( $\phi$ ) <sup>2</sup> )	TANH( $\phi$ )
EXP( $\phi$ )	EXP( $\phi$ )
EXPM( $\phi$ )	EXP( $\phi$ ) - $\phi$
LN( $\phi$ )	$\phi \times \text{LN}(\phi) - \phi$
LOG( $\phi$ )	.434294481904 $\times$ $\phi \times \text{LN}(\phi) - \phi$
SIGN( $\phi$ )	ABS( $\phi$ )
SIN( $\phi$ )	-COS( $\phi$ )
1/(SIN( $\phi$ ) $\times$ COS( $\phi$ ))	LN(TAN( $\phi$ ))
1/(SIN( $\phi$ ) $\times$ TAN( $\phi$ ))	-INV(SIN( $\phi$ ))
1/(SIN( $\phi$ ) $\times$ TAN( $\phi$ ))	-INV(SIN( $\phi$ ))
1/(SIN( $\phi$ ) <sup>2</sup> )	-INV(TAN( $\phi$ ))
SINH( $\phi$ )	COSH( $\phi$ )
1/(SINH( $\phi$ ) $\times$ $x^2$ )	-INV(SIN( $\phi$ ))

### Symbolic Integration (continued)

Pattern	Antiderivative
$1/(\text{SINH}(\phi) \times \text{COSH}(\phi))$	$\text{LN}(\text{TANH}(\phi))$
$1/(\text{SINH}(\phi) \times \text{TANH}(\phi))$	$-\text{INV}(\text{SINH}(\phi))$
$\text{SQ}(\phi)$	$\phi^3/3$
$\text{TAN}(\phi)^2$	$\text{TAN}(\phi) - \phi$
$\text{TAN}(\phi)$	$-\text{LN}(\text{COS}(\phi))$
$\text{TAN}(\phi)/\text{COS}(\phi)$	$\text{INV}(\text{COS}(\phi))$
$1/\text{TAN}(\phi)$	$\text{LN}(\text{SIN}(\phi))$
$1/\text{TAN}(\phi) \times \text{SIN}(\phi)$	$-\text{INV}(\text{SIN}(\phi))$
$\text{TANH}(\phi)$	$\text{LN}(\text{COSH}(\phi))$
$\text{TANH}(\phi)/\text{COSH}(\phi)$	$\text{INV}(\text{COSH}(\phi))$
$1/\text{TANH}(\phi)$	$\text{LN}(\text{SINH}(\phi))$
$1/\text{TANH}(\phi) \times \text{SINH}(\phi)$	$-\text{INV}(\text{SINH}(\phi))$
$\sqrt{\phi}$	$2 \times \phi^{1.5}/3$
$1/\sqrt{\phi}$	$2 \times \sqrt{\phi}$
$1/(2 \times \sqrt{\phi})$	$2 \times \sqrt{(\phi) \times .5}$
$\phi^z$ (z symbolic)	$\text{IFTE}(z == -1, \text{LN}(\phi), \phi^{(z+1)})/(z+1)$
$\phi^z$ (z real, $\neq 0, -1$ )	$\phi^{(z+1)}/(z+1)$
$\phi^0$	$\phi$
$\phi^{-1}$	$\text{LN}(\phi)$
$1/\phi$	$\text{LN}(\phi)$
$1/(1-\phi^2)$	$\text{ATANH}(\phi)$
$1/(1+\phi^2)$	$\text{ATAN}(\phi)$
$1/(\phi^2+1)$	$\text{ATAN}(\phi)$
$1/(\sqrt{(\phi-1)} \times \sqrt{(\phi+1)})$	$\text{ACOSH}(\phi)$
$1/\sqrt{1-\phi^2}$	$\text{ASIN}(\phi)$
$1/\sqrt{1+\phi^2}$	$\text{ASINH}(\phi)$
$1/\sqrt{\phi^2+1}$	$\text{ASINH}(\phi)$

### Trigonometric Expansions

The following tables list expansions for trigonometric functions in Radians mode when using the  $\rightarrow$ DEF, TRG\*, and  $\rightarrow$ TRG operations. These operations appear in the EquationWriter RULES menu.

#### $\rightarrow$ DEF Expansions

Function	Expansion
$\text{SIN}(x)$	$\frac{\text{EXP}(x \times i) - \text{EXP}(-x \times i)}{2 \times i}$
$\text{COS}(x)$	$\frac{\text{EXP}(x \times i) + \text{EXP}(-x \times i)}{2}$
$\text{TAN}(x)$	$\frac{\text{EXP}(x \times i \times 2) - 1}{(\text{EXP}(x \times i \times 2) + 1) \times i}$
$\text{SINH}(x)$	$-(\text{SIN}(x \times i) \times i)$
$\text{COSH}(x)$	$\text{COS}(x \times i)$
$\text{TANH}(x)$	$\text{TAN}(x \times i) \times -i$
$\text{ASIN}(x)$	$-i \times \text{LN}(\sqrt{1-x^2} + i \times x)$
$\text{ACOS}(x)$	$\frac{\pi}{2} + i \times \text{LN}(\sqrt{1-x^2} + i \times x)$
$\text{ATAN}(x)$	$-i \times \text{LN}\left(\frac{1+i \times x}{\sqrt{1+x^2}}\right)$
$\text{ASINH}(x)$	$-\text{LN}(\sqrt{1+x^2} - x)$
$\text{ACOSH}(x)$	$\sqrt{-\left(\frac{\pi}{2} + i \times \text{LN}(\sqrt{1-x^2} + i \times x)\right)^2}$
$\text{ATANH}(x)$	$-\text{LN}\left(\frac{1-x}{\sqrt{1-x^2}}\right)$

### TRG\* Expansions

Function	Expansion
$SIN(x + y)$	$SIN(x) \times COS(y) + COS(x) \times SIN(y)$
$COS(x + y)$	$COS(x) \times COS(y) + SIN(x) \times SIN(y)$
$TAN(x + y)$	$\frac{TAN(x) + TAN(y)}{1 - TAN(x) \times TAN(y)}$
$SINH(x + y)$	$SINH(x) \times COSH(y) + COSH(x) \times SINH(y)$
$COSH(x + y)$	$COSH(x) \times COSH(y) + SINH(x) \times SINH(y)$
$TANH(x + y)$	$\frac{TANH(x) + TANH(y)}{1 + TANH(x) \times TANH(y)}$

### → TRG Expansion

Function	Expansion
$EXP(x)$	$COS(\frac{x}{i}) + SIN(\frac{x}{i}) \times i$

### Source References

The following references were used as sources for many of the constants and equations used in the HP 48. (See "References" in chapter 4, "Equation Reference," for the references used as sources for the Equation Library.)

1. E.A. Mechtly. *The International System of Units, Physical Constants and Conversion Factors*, Second Revision. National Aeronautics and Space Administration, Washington DC, 1973.
2. *The American Heritage Dictionary*. Houghton Mifflin Company, Boston, MA, 1979.
3. *American National Standard Metric Practice ANSI/IEEE Std 268-1982*. The Institute of Electrical and Electronics Engineers, Inc., New York, 1982.
4. *ASTM Standard Practice for Use of the International System of Units (SI) E380-89a*. American Society for Testing and Materials, Philadelphia, 1989.
5. *Handbook of Chemistry and Physics*, 64th Edition, 1983-1984. CRC Press, Inc, Boca Raton, FL, 1983.
6. *International Standard publication No. ISO 31/1-1978 (E)*.
7. *The International System of Units (SI)*, Fourth Edition. The National Bureau of Standards Special Publication 330, Washington D.C., 1981.
8. *National Aerospace Standard*. Aerospace Industries Association of America, Inc., Washington D.C., 1977.
9. *Physics Letters B*, vol 204, 14 April 1988 (ISSN 0370-2693).

## Parallel Processing with Lists

Parallel processing is the idea that, generally, if a command can be applied to one or more individual arguments, then it can also be extended to be applied to one or more *sets* of arguments.

Some examples:

- `5 INV returns .2, so { 4 5 8 } INV returns { .25 .2 .125 }.`
- `4 5 * returns 20, so { 4 5 6 } { 5 6 7 } * returns { 20 30 42 }.`  
and `{ 4 5 6 } 5 * returns { 20 25 30 }.`

### General Rules for Parallel Processing

As a rule-of-thumb, a given command can use parallel list processing if all the following are true:

- The command checks for valid argument types. Commands that apply to all object types, such as DUP, SWAP, ROT, and so forth, do not use parallel list processing.
- The command takes exactly one, two, three, four, or five arguments, none of which may itself be a list. Commands that use an indefinite number of arguments (such as `→LIST`) do not use parallel list processing.
- The command isn't a programming branch command (IF, FOR, CASE, NEXT, and so forth).

The remainder of this appendix describes how the many and various commands available on the HP 48 are grouped with respect to parallel processing.

### Group 1: Commands that cannot parallel process

A command must take arguments before it can parallel process, since a zero-argument command (such as RAND, VARS, or REC) has no arguments with which to form a group.

## Group 2: Commands that must use DOLIST to parallel process

This group of commands cannot use parallel processing directly, but can be "coerced" into it using the DOLIST command (see "Using DOLIST for Parallel Processing" later in this appendix). This group consists of several subgroups:

- **Stack manipulation commands.** A stack manipulation command cannot parallel process because the stack is manipulated as a whole and list objects are treated the same as any other object. Stack commands (such as DROPN) that take level 1 arguments will not accept level 1 list arguments.
- **Commands that operate on lists as wholes.** Certain commands accept lists as arguments but treat them no differently than any other data object. They perform their function on the object as a whole without respect to its elements. For example,  $\rightarrow$ STR converts the entire list object to a string rather than converting each individual element, and the  $\equiv$  command tests the level 1 object against the level 2 object regardless of the objects' types.
- **List manipulation commands.** List manipulation commands will not parallel process since they operate on list arguments as lists rather than as sets of parallel data. However, a list manipulation command can be forced to parallel process lists of lists by using the DOLIST command. For example,  $\{\{1\ 2\ 3\}\ \{4\ 5\ 6\}\}$   $\ast$  FLIST  $\ast$  DOLIST returns  $\{5\ 120\}$ .
- **Other commands that have list arguments.** Because a list can hold any number of objects of any type, it is commonly used to hold a variable number of parameters of various types. Some commands accept such lists, and because of this are insensitive to parallel processing, except by using DOLIST.
- **Index-oriented commands.** Many array commands either establish the size of an array in rows and columns or manipulate individual elements by their row and column indices. These commands expect these row and column indices to be real number pairs collected in lists. For example,  $\{3\ 4\}$   $\rightarrow$  RPNM will generate a random integer matrix having 3 rows and 4 columns. Since these commands can normally use lists as arguments, they cannot perform parallel processing, except by using DOLIST.
- **Program control commands.** Program control structures and commands do not perform parallel processing and cannot be forced

to do so. However, programs containing these structures can be made to parallel process by using DOLIST. For example,  $\{1\ 2\ 3\ 4\ 5\}$   $\rightarrow$  I  $\ast$  IF DUF 3  $\equiv$  THEN DROP END  $\ast$  DOLIST returns  $\{3\ 4\ 5\}$ .

## Group 3: Commands that sometimes work with parallel processing

Graphics commands that can take pixel coordinates as arguments expect those coordinates to be presented as two-element lists of binary integers. Since these commands can normally use lists as arguments, they cannot parallel process, except by using DOLIST.

For the two-argument graphics commands (BOX, LINE, TLINE), if either argument is not a list (a complex number, for example), then the commands will parallel process, taking the list argument to be multiple complex number coordinates. For example,  $\{\{0,0\}\ \{1,1\}\ \{3,2\}\}$  LINE will draw two lines—between (0,0) and (1,1) and between (0,0) and (3,2).

## Group 4: ADD and +

On HP 48S and HP 48SX calculators, the + command has been used to append lists or to append elements to lists. Thus  $\{1\ 2\ 3\}$   $\rightarrow$  4  $\rightarrow$  returns  $\{1\ 2\ 3\ 4\}$ . With the advent of parallel processing in the HP 48G series, the ADD command was created to perform parallel addition instead of +.

This has several ramifications:

- To add two lists in parallel, you must do one of the following:
  - Use ADD from the **(MTH)** menu.
  - Create a custom menu containing the ADD command.
  - Assign the ADD command to a user-defined key.
- User programs must be written using ADD instead of + if the program is to be able to perform direct parallel processing, or written with + and applied to their arguments by using DOLIST. For example, programs such as  $\ast$   $\rightarrow$   $\ast$   $\ast$   $\rightarrow$   $\ast$  will produce list concatenation when  $x$  is a list rather than parallel addition, unless rewritten as  $\ast$   $\rightarrow$   $\ast$   $\rightarrow$   $\ast$   $\rightarrow$   $\ast$ .
- Algebraic expressions capable of calculating with variables containing lists (including those intended to become user-defined functions) cannot be created in RPN syntax since using



ADD to add two symbolic arguments concatenates the arguments with + rather than with ADD. For example, 'X' DUP 2 ^ SWAP 4 \* ADD 'F(X)' SWAP = produces 'F(X)=X^2+4\*X' rather than 'F(X)=X^2 ADD 4\*X'.

### Group 5: Commands that set modes/states

Commands that store values in system-specific locations so as to control certain modes and machine states can generally be used to parallel process data. The problem is that each successive parameter in the list cancels the setting established by the previous parameter. For example, { 1 2 3 4 5 } FIX is effectively the same as 5 FIX.

### Group 6: One-argument, one-result commands

These commands are the easiest to use with parallel processing. Simply provide the command with a list of arguments instead of the expected single argument. Some examples:

```
{ 1 -2 3 -4 } ABS returns { 1 2 3 4 }
NEG { 0 30 60 90 } SIN returns { 0 .5 .866025403784 1 }
{ 1 A 'SIN(Z)' } INV returns { 1 'INV(A)' 'THW(SIN(Z))' }
```

### Group 7: Two-argument, one-result commands

Two-argument commands can operate in parallel in any of three different ways:

- { list } { list }
- { list } object
- object { list }

In the first form, parallel elements are combined by the command:

```
{ 1 2 3 } { 4 5 6 } % returns { 04 . 1 . 18 }.
```

In the second form, the level 1 object is combined with each element in the level 2 list in succession:

```
{ 1 2 3 } 30 XCH returns { 2900 1400 900 }.
```

In the third form, the level 2 object is combined with each element of the level 1 list in succession:

```
50 { 1 2 3 } %T returns { 2 4 6 }.
```

### Group 8: Multiple-argument, one-result commands

Commands that take multiple (3, 4, or 5) arguments can perform parallel processing only if all arguments are lists. For example, { 'SIN(X)' 'COS(X)' 'TAN(X)' } { 'XXXX' } { 0 0 0 } ROOT returns { 0 90 0 }. Notice that lists must be used even though the level 1 and level 2 lists each contain multiples of the same element.

### Group 9: Multiple-result commands

Any command that allows parallel processing, but produces multiple results from its input data, will return its results as a single list. For example, { 1 2 3 } { 4 5 6 } R+C C+R produces { 1 4 2 5 3 6 } rather than the more expected { 1 2 3 } { 4 5 6 }.

The following UNMIX program will unmix the data given the number of expected result lists:

```
* OVER SIZE + 1 n %
* i n
FOR j j %
FOR i 1 i GET n
STEP % n / +LIST
NEXT
*
```

Taking { 1 4 2 5 3 6 } from above as the result of C→R (a command which should return two results), 2 UNMIX gives { 1 2 3 } { 4 5 6 }.

### Group 10: Quirky commands

A few commands behave uniquely with respect to parallel processing:

- **DELALARM.** This command can take a list of arguments. Note, however, that deletions from early in the alarm list will change the alarm indices of the later alarm entries. Thus, if there are only three alarms, { 1 3 } DELALARM will cause an error, whereas { 3 1 } DELALARM will not.
- **DOERR.** This command forces an error state that causes all running programs and commands to halt. Thus, even though providing the command with a list argument will cause the command to perform parallel processing, the first error state

will cause the command to abort and none of the rest of the list arguments will be used.

- **FREE, MERGE.** Only port 1 can be freed or merged on the HP 48GX. Thus, even though a list argument is acceptable, an error will occur for any list except { 1 }.
- **RESTORE.** This command performs a system warmstart after installing the backup object into memory. All functions are terminated at that time. Thus, only the first backup object in a list will be restored.
- **-(Attach Unit).** This command will create unit objects in parallel only if level 1 contains a list. Thus `i { ft 1 r m } ...` produces `{ i_ft 1_1 r 1_m } while { i 2 } 'm' ...` produces an error.
- **STO+, STO-** performs parallel list addition only if both arguments are lists. If one argument is a list and the other is not, STO+ appends the non-list argument to each element in the list.
- **STO-, STO\*, STO/.** These commands perform parallel processing if both arguments are lists, but fail otherwise.

### Using DOLIST for Parallel Processing

Almost any command or user program can be made to work in parallel over a list or lists of data by using the DOLIST command. Use DOLIST as follows.

- Level 1 must contain a command, a program object, or the name of a variable that contains a command or program object.
- Level 2 must contain an argument count unless the level 1 object is a command that accepts parallel processing, a program containing only one command that accepts parallel processing, or a user-defined function. In these special cases, Level 2 contains the first of the list arguments.
- If level 2 was the argument count, then level 3 is the first of the argument lists. Otherwise, levels 2 through *n* are the argument lists.

As an example, the following program takes three objects from the stack, tags them with the names a, b, and c, and displays them one after the other in line 1 of the display.

```

* → a b c
* { a b c } DUP * EVAL * DOLIST
SWFF * →TAG * DOLIST
CLLCD 1 * 1 DISP 1 WAIT * DOLIST
*
*

```

# Index

## Special characters

⌘ character, 1-10

## A

absolute value, 3-5

alarms

acknowledging, 3-6

deleting, 3-78

finding, 3-116

index number, 3-116

recalling, 3-262

storing, 3-326

ALG annunciator, 1-9

algebraic

linear structure, 1-20

Algebraic/Program-entry mode,

1-9, 1-63

algebraics

action in programs, 1-2

collecting terms, 3-56

comparing, 1-19

conditional testing, 1-22

editing in programs, 1-9

expanding, 3-109

in local variable structure,

i-3, 1-11

isolating variables, 3-158

rearranging, 3-104, 3-158

rearranging programmatically,

2-19

simplifying, 3-56

testing for linearity, 3-168

tests in, 1-19

algebraic syntax

conditional testing, 1-22

in local variable structures,

1-4

test commands, 1-17, 1-19

alpha keyboard

automatically locking, 1-63

amortization (TVM), 3-12

angle mode

setting, 3-78, 3-131, 3-256

angles

converting units, 3-99, 3-294

angular mechanics, 4-29

angular motion, 4-48

animation, 2-45, 2-56, 3-14

annunciators

user flags, 1-42

applications, 1-79

archives

creating, 3-18

arcs, 3-17

arguments

comparing, 3-183, 3-191

recalling last, 3-162, 3-163

verifying, 2-36

arrays

applying a program to, 2-29

column norm, 3-53

complex conjugates, 3-62

constant, 3-59

creating from stack, 3-20, 3-369  
 deleting columns, 3-55  
 disassembling, 3-19  
 disassembling complex, 3-71  
 extracting elements, 3-127, 3-129  
 Fourier transforms, 3-115  
 inserting columns, 3-54  
 inserting diagonals, 3-84  
 inverse Fourier transform, 3-147  
 manipulating, 2-16, 2-49  
 maximum and minimum elements, 2-22  
 redimensioning, 3-266  
 replacing elements in, 3-244, 3-246  
 residual, 3-291  
 row norm, 3-280  
 sorting elements, 2-23  
 spectral norm of, 3-314  
 swapping columns, 3-70  
 symbolic, 2-29  
 axes (plots)  
 controlling, 3-33  
 including, 3-95  
 labeling, 3-162

**B**

backup objects, 3-248  
 creating, 3-18  
 restoring, 3-273  
 bar graphs, 3-34  
 base  
 setting, 3-38, 3-75, 3-137, 3-210  
 baud rate  
 specifying, 3-36  
 beams, 4-1

beeper  
 in programs, 1-71  
 sounding, 3-37  
 specifying tone and duration, 3-37  
 Bernoulli equation, 4-23  
 Bessel functions, 2-43  
 binary integers  
 comparing, 1-19  
 converting to floating-point, 3-42  
 custom display, 2-7  
 representing flags, 1-44  
 shifting one bit right, 3-25  
 wordsize, 1-19, 3-335  
 binary wordsize  
 recalling, 3-265  
 bipolar transistors, 4-73  
 bit  
 rotate left, 3-278  
 rotate right, 3-285  
 shift left, 3-310  
 shift right, 3-317  
 black-body emissive power, 3-112  
 black body radiation, 4-42  
 boxes, 3-40  
 branch cuts, 3-7, 3-9, 3-21, 3-23, 3-26, 3-29  
 branching structures  
 conditional structures, 1-20, 1-53, 3-101  
 ending, 3-102  
 loop structures, 1-27  
 program element, 1-3  
 BRCH menu, 1-20, 1-27  
 Brewster angle, 4-52  
 buckling, 4-3  
 buffer (serial)  
 clearing, 3-51  
 sizing data in, 3-40

byte  
 rotate left, 3-278  
 rotate right, 3-285  
 shift left, 3-311  
 shift right, 3-318  
**C**  
 calculator  
 turning off, 1-82  
 calculator clock, 2-5  
 cantilevers, 4-1  
 capacitor, 4-16, 4-17, 4-19, 4-20  
 "case" branching, 1-22, 2-38, 2-49, 3-42  
 case structures, 2-49  
 centripetal force, 4-29  
 character codes  
 remapping to match HP 82240A, 3-211  
 to characters, 3-48  
 characters  
 codes, 3-204  
 character translation, 3-352  
 checksums, 3-41  
 specifying type of, 3-49  
 verify programs, 2-1  
 chi-square distribution, 3-363  
 choose boxes  
 custom, 1-68, 3-46  
 in programs, 1-68  
 circle, 4-59, 4-79  
 circular motion, 4-49  
 clearing  
 command line, 3-50  
 directories, 3-53  
 display, 1-74  
 flags, 1-42, 3-45  
 stack, 3-50  
 stack display, 3-51  
 subdirectories, 3-52  
 variables, 3-52, 3-53

clock  
 adjusting, 3-50  
 coefficients  
 of monic polynomials, 3-220  
 regression, 3-177  
 collisions, 4-30  
 column operations  
 converting matrices into columns, 3-54  
 converting vectors into elements, 3-54  
 creating matrices from columns, 3-56  
 creating vectors from elements, 3-56  
 deleting, 3-55  
 inserting, 3-54  
 columns, 4-1  
 combinations, 3-58  
 command line  
 clearing, 3-50  
 during program input, 1-62  
 commands  
 applying to list elements, 3-92, 3-102, 3-203  
 applying to lists, 3-91  
 in programs, 1-2  
 comments, 1-10  
 comparing objects, 3-294  
 comparison functions, 1-17, 1-19  
 compiled local variable structures  
 defining procedure, 1-15  
 complex numbers  
 conjugates, 3-62  
 disassembling, 3-71  
 imaginary parts, 3-150  
 polar angle  $\theta$ , 3-19  
 real parts, 3-267  
 computer  
 creating programs on, 1-10

conditional commands, 1-20,  
1-21, 1-22  
conditionals  
  nested, 2-23, 2-26, 2-36  
conditional structures  
  "case" branching, 1-22, 3-42  
  conditional commands, 1-20  
  ending, 3-102  
  error branching, 1-53, 1-54,  
  3-146  
  examples, 1-23  
  "if" branching, 1-20, 1-21,  
  1-22, 1-53, 1-54, 3-101,  
  3-144, 3-148, 3-149  
  program element, 1-3  
  test commands in, 1-17, 1-20  
conduction, 4-39, 4-41  
cone, 4-64  
conjugates, 3-62  
of matrices, 3-353  
of objects, 3-300  
constants  
  symbolic, 3-99, 3-143, 3-183,  
  3-193  
Constants Library  
  opening, 3-63  
continuing execution, 1-47,  
  1-48, 1-57, 1-59, 3-64  
convection, 4-40, 4-41  
converting base units, 3-360  
coordinate modes  
  specifying, 3-70, 3-269, 3-316  
coordinates  
  pixels to user units, 3-250  
  specifying for *PICT*, 3-231  
  user units to pixels, 3-71  
coordinates of *PICT*  
  specifying, 3-231  
correlation (statistical), 3-65  
Coulomb's law, 4-11

counters  
  loop structures, 1-29, 1-31,  
  1-33, 1-35  
  negative steps, 1-31, 1-35,  
  1-39  
  stepping, 1-39  
  covariance, 3-222  
  creating 2D vectors, 3-369  
  creating 3D vectors, 3-370  
  critical angle, 4-51  
  current, 4-9, 4-43, 4-67  
  cursor (command line), 1-63  
  curve fitting, 3-37, 3-110, 3-166,  
  3-167, 3-176, 3-250  
  custom menus  
  creating, 3-187  
  displaying, 3-187  
  in programs, 1-78, 1-79  
  menu-based applications,  
  1-79  
  cylinder, 4-64

## D

Darcy friction factor, 3-72  
data input, 3-154  
data transmission  
  closing ports, 3-51  
  detecting errors, 3-49  
  error testing, 3-40  
  parity, 3-217  
  receiving, 3-268, 3-269  
  serial, 3-381  
  size of, 3-40  
  specifying baud rate, 3-36  
  terminating Server mode,  
  3-117  
  time-out, 3-324  
  via Kermit, 3-160, 3-301  
  via Kermit server, 3-160,  
  3-303

data transmissions  
  opening ports, 3-212  
dates  
  calculating days between,  
  3-75  
  calculating past or future,  
  3-74  
  displaying, 3-73, 3-356  
  setting, 3-73  
debugging, 1-47, 1-49, 3-74,  
  3-201  
decomposing vectors, 3-371  
defining procedures  
  compiled local variables in,  
  1-15  
  local variables in, 1-14  
  local variable structures, 1-11  
definite loops, 2-3, 2-26, 2-46,  
  2-47, 2-49, 2-56  
  with counters, 2-10, 2-16  
delimiters  
  \* \* for programs, 1-1  
dependent variables  
  specifying in matrices, 3-389  
  specifying in plots, 3-81  
  specifying in statistical data,  
  3-57  
  summation of squares, 3-389  
  summations of, 3-388  
dialog boxes (input forms),  
  3-152  
dimensions  
  converting, 3-65  
  of *PICT*, 3-223  
  diodes, 4-69  
directories  
  changing, 3-142, 3-363  
  clearing, 3-53  
  creating, 3-69  
  current, 3-142  
  HOME, 3-142

## E

editing  
  programs, 1-9  
eigenvalues  
  of matrices, 3-100, 3-101  
eigenvectors  
  of matrices, 3-100  
elastic buckling, 4-3  
elastic collisions, 4-30  
electricity, 4-9  
electrostatic force, 4-11  
ellipse, 4-59  
energy, 4-15, 4-31  
Equation Library  
  references, 4-1, 4-82  
  starting, 3-104  
  subjects, 4-1  
  titles, 4-1  
equations  
  defining sets, 3-193  
  disassembling, 3-104  
  expanding, 3-109  
  least squares solution, 3-178  
  rearranging, 3-104

recalling, 3-259  
 reordering sets, 3-194  
 retitling sets, 3-194  
 sets, 3-194  
 solving linear systems, 3-178  
 solving quadratic, 3-253  
 solving sets, 3-197  
 testing for linearity, 3-168  
 equation sets  
 changing titles, 3-194  
 defining, 3-193  
 reordering, 3-194  
 solving, 3-197  
 errors  
 actions in programs, 1-51  
 analyzing, 1-51  
 causes, 1-51  
 causing, 1-51, 3-90  
 clearing last, 1-51, 3-106  
 conditional structures, 1-53,  
 1-54, 3-146  
 detecting in transmission,  
 3-49  
 display messages, 1-51  
 Kermit, 3-159  
 numbers for, 1-51, 3-106  
 recalling messages, 1-51,  
 3-105, 3-159  
 trapping, 1-53, 1-54, 3-146  
 user-defined, 1-51  
 error-trapping, 2-29  
 error trapping, 2-9, 2-10  
 escape velocity, 4-49  
 evaluation  
 of local variables, 1-13  
 of test clauses, 1-21, 1-22,  
 1-36, 1-38  
 example program  
 UNMIX, G-5

example programs  
 animating graphics, 2-45,  
 2-47, 2-56  
 applying programs repeatedly,  
 2-19  
 Bessel functions, 2-43  
 calculating median, 2-14  
 converting from algebraic to  
 RPN, 2-40  
 converting plots to grobs,  
 2-45  
 custom menus, 1-80  
 displaying binary integers,  
 2-10  
 execution times, 2-5  
 Fibonacci numbers, 2-2  
 input forms, 1-67  
 input routines, 1-54, 1-57,  
 1-60, 1-64, 1-66  
 inverse functions, 2-54  
 manipulating math curves,  
 2-46  
 mass of an object, 1-79  
 maximum and minimum  
 elements, 2-22  
 percentile of a list, 2-14  
 phone list, 1-67  
 plotting pie charts, 2-49  
 preserving calculator status,  
 2-8  
 rearranging algebraics, 2-19  
 right-justifying strings, 2-7  
 summations, 1-41  
 surface area of a torus, 1-45  
 system flags, 1-43  
 Taylor's polynomials, 2-45  
 trace mode, 2-53  
 using conditionals, 1-23, 1-24,  
 1-25, 1-26  
 using loops, 1-29, 1-31, 1-33,  
 1-35, 1-37, 1-40

verifying arguments, 2-36  
 volume of a sphere, 1-5, 1-6  
 volume of a spherical cap,  
 1-6, 1-9, 1-12  
 Examples subdirectory  
 creating, 3-345  
 removing, 3-52  
 exponents, 3-383  
 expressions  
 creating from arguments,  
 3-15  
 pattern replacement, 3-180,  
 3-181  
 rewriting, 3-180, 3-181  
 extrapolation, 3-234, 3-235,  
 3-236

## F

factorials, 3-112  
 factor units, 3-361  
 false (test result), 1-17, 1-19  
 Fanning friction factor, 3-113  
 F distribution, 3-364  
 Fibonacci numbers, 2-2  
 flags  
 annunciators, 1-42  
 binary integer form, 1-44  
 clearing, 1-42, 3-45, 3-114,  
 3-125  
 control behavior, 1-42  
 controlling logic with, 2-23,  
 2-26  
 default states, C-1  
 preserving and restoring  
 status, 2-8, 2-9, 2-49  
 program control, 1-42  
 recalling, 3-262  
 recalling states, 1-44  
 restoring states, 1-44  
 setting, 1-42, 2-10, 2-23, 2-26,  
 2-53, 3-304, 3-327

storing states, 1-44  
 system, 1-42, 1-44, C-1  
 testing, 1-42, 2-23, 2-26,  
 3-114, 3-124, 3-125  
 testing and clearing, 3-114,  
 3-125  
 types, 1-42  
 user, 1-42, 1-44  
 fluid flows  
 Darcy friction factor, 3-72  
 Fanning friction factor, 3-113  
 fluids, 4-22  
 focal length, 4-50  
 force, 4-11, 4-27, 4-44  
 "for" looping, 1-32, 1-34, 2-10,  
 2-16, 2-26, 2-46, 2-47, 2-49,  
 2-56, 3-119, 3-201  
 formatting  
 ports, 3-228  
 Fourier transforms  
 inverse, 3-147  
 of arrays, 3-115, 3-147  
 free fall motion, 4-47  
 freeing merged memory, 3-121,  
 3-122  
 frequency  
 resonant, 4-19  
 friction losses, 4-26  
 functions  
 applying to list elements,  
 3-92, 3-102, 3-203  
 applying to lists, 3-91  
 defining, 3-77

## G

gamma function, 3-112  
 gas compressibility correction  
 factor, 3-393  
 gas-compressibility factor, 4-36  
 gases, 4-32  
 geometry, 4-58, 4-63

global variables  
 action in programs, 1-2  
 disadvantages in programs, 1-11  
 list of, 3-357  
 graphics  
 creating, 3-17, 3-40, 3-131, 3-166, 3-227  
 custom, 2-56  
 displaying, 3-249  
 environment, 3-131  
 Picture environment, 3-227  
 graphics commands  
 parallel processing with, G-3  
 graphics objects  
 animating, 3-14  
 creating blank, 3-39  
 creating from display, 3-163  
 creating from stack, 3-133  
 displaying, 3-164  
 manipulating, 2-23, 2-46, 2-47, 2-49, 2-56  
 superimposing, 3-130, 3-134  
 gravitation, 4-31, 4-47, 4-48, 4-49

**H**  
 HFL T annunciator, 1-47, 1-59  
 halting programs, 1-48, 3-136  
 harmonic motion, 4-54  
 head loss, 4-24  
 heat, 4-37  
 heat capacity, 4-38  
 heat transfer, 4-37  
 histograms, 3-137, 3-139  
 HMS format  
 adding in, 3-139  
 converting from, 3-141  
 converting to, 3-142  
 subtracting in, 3-140  
 Hooke's law, 4-30

**I**  
 ideal gases, 4-32  
 "if" branching, 1-20, 1-21, 1-22, 1-53, 1-54, 2-2, 2-23, 2-26, 2-36, 3-101, 3-144, 3-148, 3-149, 3-346  
 "iferror" branching, 2-29  
 implicit variable references, 3-305  
 indefinite loops, 2-7, 2-19, 2-23  
 ending, 3-89, 3-102, 3-374  
 with counters, 2-43  
 independent variables  
 specifying for plotting, 3-151  
 specifying in matrices, 3-380  
 specifying in statistical data, 3-57  
 summation of squares, 3-379  
 summations of, 3-379, 3-388  
 index of refraction, 4-50  
 inductor, 4-17, 4-20, 4-21  
 initial value problems, 3-274  
 error estimate, 3-276, 3-290  
 next step, 3-288  
 solution step size, 3-277  
 with known partials, 3-286  
 input  
 prompting for, 3-238  
 input forms  
 creating, 3-152  
 custom, 1-67, 3-152  
 for program input, 1-67  
 in programs, 1-67  
 resetting, 3-203  
 saving initial values, 3-203  
 input plane  
 setting the x range, 3-387  
 setting the y range, 3-393  
 intrinsic density of silicon, 3-305

inverses  
 calculating, 3-156  
 logical, 3-201  
 of matrices, 3-156  
 storing, 3-309  
 I/O  
 closing ports, 3-51  
 IR port, 3-157  
 Kermit errors, 3-159  
 Kermit transmission, 3-160  
 selecting port, 3-157  
 serial port, 3-157  
 isothermal expansion, 4-34

**J**  
 junction field-effect transistors, 4-74

**K**  
 Kermit, 3-212, 3-217, 3-230, 3-268, 3-269, 3-301, 3-303  
 error messages, 3-159  
 server, 3-160  
 transmission, 3-160  
 keyboard  
 defining user, 3-24, 3-328  
 in programs, 1-72, 1-73  
 recalling definitions, 3-263  
 unassigning user keys, 3-80  
 key location numbers, 1-73  
 keys  
 defining in user keyboard, 3-24  
 testing, 3-159  
 keystrokes  
 as program input, 1-72, 1-73  
 waiting for, 3-373  
 killing programs, 1-47, 1-48, 3-161

**L**  
 last argument  
 recalling, 2-10, 3-162, 3-163  
 length factor, 4-3  
 libraries  
 attaching, 3-31  
 detaching, 3-84  
 displaying menus, 3-187  
 listing, 3-165  
 light, 4-50  
 linear mechanics, 4-28  
 linear motion, 4-47  
 linear structure, 1-20  
 lines, 3-166  
 list concatenation, G-3  
 list processing  
 programming example, 2-29  
 lists  
 action in programs, 1-2  
 adding elements of two, 3-11  
 applying commands, functions, or programs to, 3-91, 3-92, 3-102, 3-203, 3-302, 3-333, 3-336  
 applying executable object repeatedly, 3-302  
 assembling, 3-169  
 creating from stack, 3-169  
 differences between elements, 3-170  
 disassembling, 3-169  
 extracting elements, 3-127, 3-129, 3-136, 3-341  
 first element, 3-136  
 last elements, 3-341  
 locating objects in, 3-234  
 multiplying elements of, 3-171  
 parallel processing, G-1  
 product of, 3-171  
 replacing elements in, 3-244, 3-246

reversing element order, 3-274  
 sorting, 2-14, 3-315  
 sublist position in, 3-203  
 summing elements of, 3-170  
 local variables, 2-9  
 action in programs, 1-2  
 compiled, 1-15  
 creating, 1-3, 1-11  
 evaluating, 1-13, 2-19  
 exist temporarily, 1-11, 1-12, 1-14  
 naming, 1-11  
 nested, 2-43, 2-49  
 passing between programs, 2-47  
 storing objects in, 2-19, 2-38  
 local variable structures  
 advantages, 1-12  
 as user-defined functions, 1-16  
 calculations with, 1-3  
 create local variables, 1-11  
 defining procedure, 1-11, 1-14  
 entering, 1-11  
 operation, 1-3, 1-11  
 program element, 1-3  
 syntax, 1-3, 1-11  
 logic  
 controlling, 2-26  
 controlling with flags, 2-23, 2-26  
 functions, 2-23, 2-26, 2-36, 2-38  
 logical functions, 1-17, 1-19, 3-13, 3-201, 3-213, 3-382  
 longitudinal waves, 4-81  
 loop structures  
 counters, 1-29, 1-31, 1-33, 1-35, 1-39, 3-323

definite, 1-27, 1-28, 2-3, 2-26, 2-46, 2-47, 2-49, 2-56, 3-119, 3-201, 3-321  
 "do" looping, 1-36, 3-89  
 "for" looping, 1-32, 1-34, 3-119, 3-201, 3-323  
 indefinite, 1-27, 1-36, 2-7, 2-19, 2-23, 3-374  
 keystroke input, 1-73  
 negative steps, 1-31, 1-35  
 program element, 1-3  
 "start" looping, 1-28, 1-30, 3-201, 3-321  
 summation alternative, 1-40  
 test commands in, 1-36, 1-38  
 "while" looping, 1-38  
 lowercase letters  
 in names, 1-11  
**M**  
 Mach number, 4-35  
 magnetic field, 4-15, 4-43, 4-44, 4-45  
 magnetism, 4-43  
 magnification, 4-50  
 mantissas, 3-179  
 mass  
 related to energy, 4-31  
 mathematical data  
 plotting, 3-94  
 matrices  
 adding rows, 3-283  
 condition number, 3-60  
 conjugates of, 3-353  
 converting to columns, 3-54  
 converting to rows, 3-282  
 creating from columns, 3-56  
 creating from rows, 3-284  
 deleting rows, 3-284  
 determinants, 3-82  
 eigenvalues, 3-101  
 eigenvalues and eigenvectors,  
 3-100  
 extracting diagonals, 3-85  
 identity, 3-143  
 inverting, 3-156  
 least squares solution, 3-178  
 LQ factorization, 3-176  
 LU decomposition, 3-179  
 multiplying rows by a  
 constant, 3-260  
 QR factorization, 3-253  
 random element, 3-257  
 rank of, 3-257  
 reduced row echelon form,  
 3-286  
 Schur decomposition, 3-298  
 singular value decomposition,  
 3-337  
 singular values of, 3-338  
 spectral radius, 3-317  
 squaring, 3-316  
 sum of diagonal elements,  
 3-351  
 swapping rows, 3-292  
 transposing, 3-353  
 mechanics, 4-28, 4-29  
 memory  
 checking available, 3-186  
 freeing merged, 3-121  
 merging RAM card, 3-190  
 menu-based applications, 1-79  
 menu descriptions  
 PRG BRCH, 1-20, 1-27  
 PRG RUN, 1-49  
 PRG TEST, 1-18  
 menus  
 custom, 1-78, 1-79, 2-23, 3-187  
 defining, 3-187  
 delayed display, 1-72, 1-78  
 displaying, 3-187

displaying in programs, 1-72, 1-77, 1-79  
 for libraries, 1-77  
 for program input, 1-79  
 last menu, 1-78  
 numbers for, 1-77, 1-78, 3-264  
 pages in, 1-78  
 programmatic uses, 1-77  
 recalling, 3-264  
 recalling numbers, 1-78  
 resuming programs, 1-79  
 running programs, 1-79  
 temporary, 2-49, 3-350  
 message boxes  
 creating, 3-196  
 custom, 1-77, 3-196  
 in programs, 1-77  
 messages, A-1-17  
 displaying, 3-88  
 prompting, 1-56  
 meta-objects, 2-29  
 Minehunt game, 3-192  
 cheating, 3-192  
 storing, 3-192  
 mode names  
 Algebraic-entry, 1-63  
 Algebraic/Program-entry,  
 1-9, 1-63  
 Program-entry, 1-4, 1-9  
 modes  
 program entry, 1-4, 1-9  
 setting, 1-9  
 Mohr's circle, 4-79  
 motion, 4-46  
**N**  
 names  
 action in programs, 1-2  
 negatives, 3-199  
 nested structures, 2-40, 2-43  
 new commands, E-1-7



newlines, 1-4, 1-59  
 NMOS transistors, 4-71  
 nonideal hydrocarbon gas, 3-393  
 normal distribution, 3-365  
 NPN bipolar transistors, 4-73  
 number bases  
   converting between, 2-32  
 numbers  
   action in programs, 1-2  
   complex, 3-19  
   complex conjugates, 3-62  
   disassembling complex, 3-71  
   fractional part, 3-121  
   imaginary parts, 3-150  
   integer part, 3-157  
   largest available, 3-183  
   rational form, 3-251  
   rational form with  $\pi$ , 3-251  
   real parts, 3-267  
   real to binary, 3-292  
   real to complex, 3-293  
   rounding, 3-279  
   rounding to integer, 3-44,  
     3-118  
   smallest available, 3-193  
   squaring, 3-316

**O**

objects  
 actions in programs, 1-2  
 backup, 3-248  
 comparing, 3-294  
 conjugates, 3-300  
 converting dimensions, 3-65  
 copying, 3-200, 3-226  
 decomposing, 3-209  
 displaying, 3-88  
 duplicating, 3-97, 3-98  
 entering in programs, 1-4  
 evaluating, 3-107  
 evaluating by addresses, 3-339

evaluating symbolic, 3-207  
 operating system, 3-339  
 printing, 3-241  
 recalling, 3-261  
 removing from stack, 3-95,  
   3-96  
 removing labels from, 3-97  
 removing pointers to, 3-200  
 replacing a portion of, 3-270  
 sign of, 3-306  
 size of, 3-41, 3-309  
 storing, 3-325  
 storing in reserved variables,  
   3-324  
 storing objects in, 3-325  
 testing types, 1-20  
 truncating, 3-353  
 type numbers, 1-20  
 type numbers of, 3-359  
 unevaluated, 3-254  
 object type numbers, 1-20  
 Ohm's law, 4-11  
 optics, 4-50  
 oscillations, 4-54  
 output  
   labeling, 2-5

**P**

packets  
 sending, 3-230  
 parallel addition, G-3  
 parallelepiped, 4-65  
 parallel processing, G-1  
 DOLIST, G-2, G-6  
 multiple-result commands,  
   G-5  
 parity  
   setting, 3-217  
 pendulum, 4-56, 4-57  
 percent functions, 3-47, 3-339  
 permutations, 3-224

phase delay, 4-16  
 PICT, 3-226  
   clearing, 3-105  
   editing, 3-17, 3-40, 3-166  
   specifying coordinates, 3-231  
   superimposing grobs onto,  
     3-130, 3-134  
 Picture environment  
   selecting, 3-131, 3-227  
 pie charts, 2-49  
 pixels  
   checking if on, 3-229  
   coordinates, 3-71, 3-250  
   toggling, 3-349  
   turning off, 3-228  
   turning on, 3-229  
 plane geometry, 4-58  
 plots  
   3D, 3-207, 3-208  
   3D perspective, 3-111  
   autoscaling, 3-32  
   center, 3-44  
   changing horizontal width,  
     3-372  
   controlling axes, 3-33  
   creating, 3-36  
   drawing, 3-94, 3-139, 3-296  
   drawing axes, 3-95  
   labeling axes, 3-162  
   mathematical data, 3-94  
   resolution, 3-207, 3-208, 3-271  
   scaling, 3-135, 3-299  
   setting axes tick-marks, 3-30  
   setting types, 3-34, 3-61, 3-86,  
     3-125, 3-132, 3-137, 3-215,  
     3-296, 3-297, 3-355, 3-375,  
     3-391  
   simultaneous, 3-307  
   specifying dependent variable,  
     3-81  
   specifying eye point, 3-111  
   specifying x-axis display range,  
     3-384  
   specifying y-axis display range,  
     3-390  
   statistical data, 3-36, 3-94,  
     3-137, 3-139  
 plot types  
   setting, 3-218, 3-221, 3-312  
   specifying, 3-232  
 plug-in cards  
   initializing, 3-228  
 PN step-junction devices, 4-69  
 polarization, 4-52  
 polygon, 4-61  
 polynomials  
   evaluating, 3-224  
   monic, 3-220  
   roots, 3-238  
   Taylor's, 3-343  
 polytropic processes, 4-34  
 population covariance, 3-222  
 population standard deviation,  
   3-242  
 population variance, 3-247  
 port  
   opening, 3-212  
   selecting, 3-157  
 ports  
   closing, 3-51  
   initializing, 3-228  
 pressure  
   hydrostatic, 4-23  
 PFE annunciator, 1-4, 1-9  
 PRG BRCH menu, 1-20, 1-27  
 PRG RUN menu, 1-49  
 PRG TEST menu, 1-18  
 print buffer  
   printing, 3-68  
   printing  
     HP 82240A, 3-211  
   print buffer, 3-68

setting delay, 3-79  
 trace mode, 2-53  
 Program Development Link,  
   1-10, 2-1  
   programs included, 2-1  
 Program-entry mode, 1-4, 1-9  
 program examples  
   arbitrary number bases, 2-32  
   programming techniques  
   applied list processing, 2-29  
   "case" branches, 2-38, 2-49  
   case structures, 2-38, 2-49  
   controlling logic with flags,  
     2-23, 2-26  
   custom graphics, 2-56  
   custom menus, 2-23, 2-26  
   definite loops, 2-3, 2-26, 2-46,  
     2-47, 2-49, 2-56  
   definite loops with counters,  
     2-10, 2-16  
   "do" loops, 2-19, 2-23, 2-43  
   error-trapping, 2-29  
   error trapping, 2-9, 2-10  
   evaluating local variables,  
     2-19  
   "for" loops, 2-10, 2-16, 2-26,  
     2-46, 2-47, 2-49, 2-56  
   "if" branches, 2-23, 2-26  
   indefinite looping, 2-33  
   indefinite loops, 2-7, 2-19,  
     2-23  
   indefinite loops with counters,  
     2-43  
   interpolation, 2-14  
   labeling output, 2-5  
   list concatenation, 2-40  
   local variables, 2-9, 2-38,  
     2-43, 2-47, 2-49  
   logical functions, 2-23, 2-26,  
     2-36, 2-38  
   logic control, 2-26

manipulating grobs, 2-23,  
   2-46, 2-47, 2-49, 2-56  
 meta-object manipulation,  
   2-29  
   nested conditionals, 2-23,  
     2-26, 2-36  
   nested structures, 2-40, 2-43  
   object type-checking, 2-40  
   plot commands, 2-45, 2-46,  
     2-49  
   preserving flag status, 2-9,  
     2-49  
   programs as arguments, 2-10,  
     2-19, 2-54  
   recursion, 2-2, 2-40  
   restoring flag status, 2-9,  
     2-49  
   restoring last argument, 2-10  
   root-finder, 2-54  
   setting flags, 2-10, 2-23, 2-26,  
     2-53  
   simulating new object types,  
     2-29  
   sorting array elements, 2-23  
   sorting lists, 2-14  
   "start" loops, 2-3  
   string and character  
   manipulation, 2-33  
   string operations, 2-7  
   structures, 2-5  
   subroutines, 2-5, 2-10, 2-21,  
     2-38  
   tagged output, 2-33  
   temporary menus, 2-49  
   testing flags, 2-23, 2-26  
   using arrays, 2-16, 2-49  
   using calculator clock, 2-5  
   using flags, 2-29  
   using other programs, 2-5,  
     2-10, 2-21, 2-38

using statistics commands,  
   2-49  
 utility programs, 2-38  
 vectored enter, 2-53  
 "while" loops, 2-7  
 programs  
   actions for object types, 1-2  
   applying to elements of a  
     matrix, 2-29  
   applying to list elements,  
     3-92, 3-102, 3-203  
   applying to lists, 3-91  
   are sequences of objects, 1-1  
   beeping, 1-71  
   calculation styles, 1-3  
   cancelling halted, 3-161  
   causing errors, 1-51  
   checksums, 2-1  
   comments in, 1-10  
   conditional structures, 1-20,  
     1-53, 1-54  
   creating on computer, 1-10  
   cursor position during input,  
     1-63  
   debugging, 1-47, 3-74, 3-201  
   default input, 1-60  
   displaying input forms, 1-67  
   displaying menus, 1-72, 1-77,  
     1-79, 2-23, 2-26, 2-49  
   displaying output, 1-74, 1-75,  
     1-76  
   displaying string output, 1-75  
   editing, 1-9  
   elapsed time, 2-5  
   entering, 1-4  
   entry modes, 1-4, 1-9  
   entry modes during input,  
     1-63  
   error actions, 1-51  
   evaluating local variables,  
     1-13  
   executing, 1-5  
   finding roots in, 2-54  
   flags in, 1-42  
   getting input, 1-55, 1-56,  
     1-58, 1-60, 1-72, 1-73,  
     3-154  
   HPL T annunciator, 1-47  
   halting, 1-48, 3-136  
   in local variable structure,  
     1-3, 1-11  
   input as strings, 1-60  
   input forms, 3-152  
   introduction, 1-1  
   killing, 1-47, 1-48, 3-161  
   labeling output, 1-74, 1-75  
   local variable structures, 1-3,  
     1-11  
   loop structures, 1-27, 3-89,  
     3-374  
   naming, 1-5  
   newlines in, 1-4  
   not evaluating local variables,  
     1-13  
   not executing in programs,  
     1-2  
   objects in, 1-2  
   on the stack, 1-4  
   pausing, 3-373  
   pausing for output, 1-76  
   prompting, 1-56, 1-58, 1-60,  
     3-154  
   recursion, 2-2  
   resuming, 1-47, 1-48, 1-56,  
     1-59, 1-79, 1-82, 3-64  
   samples, 3-345  
   scope of local variables in,  
     1-14, 1-15  
   single-step execution, 1-47,  
     1-48, 1-49, 3-320  
   size of, 2-1  
   stepping through, 3-201

stopping, 1-5, 3-102, 3-161  
 storing, 1-5  
 structures in, 1-3  
 subroutines, 1-45  
 test commands, 1-17  
 trapping errors, 1-53, 1-54, 3-102  
 turning off calculator, 1-82  
 user-defined functions, 1-16  
 using as arguments, 2-10, 2-19, 2-54  
 using as subroutines, 2-5, 2-10, 2-21, 2-38  
 utility, 2-38  
 verifying, 2-1  
 verifying input, 1-63, 2-36  
 viewing, 1-9  
 waiting for keystrokes, 1-72, 1-73, 3-373  
 projectile motion, 4-48  
 prompting, 1-56, 1-58, 1-60  
 prompts, 3-154

**Q**

quadratic equations  
 solving, 3-253  
 quality factor, 4-19

**R**

RAM  
 checking available, 3-186  
 RAM cards  
 freeing, 3-121, 3-122  
 merging, 3-190  
 random numbers  
 generating, 3-256  
 in matrices, 3-257  
 seeding, 3-267  
 reactance, 4-16  
 real gases, 4-32

real numbers

converting to binary, 3-292  
 converting to complex, 3-293  
 manipulating, 2-33  
 recalling  
 flag states, 1-44  
 last arguments, 3-163  
 menu numbers, 1-78  
 rectangle, 4-60  
 recursion, 2-2, 2-40  
 reduced row echelon, 3-286  
 reflection, 4-52  
 refraction, 4-50  
 regression  
 calculating, 3-177  
 formula used, 3-166  
 power, 3-250  
 setting type, 3-37, 3-110, 3-167, 3-176  
 remainders, 3-195  
 reserved variables  
 storing objects in, 3-324  
 resistance  
 wire, 4-13  
 resonant frequency, 4-19  
 ring, 4-61  
 root-finder  
 in programs, 2-54  
 roots  
 finding, 3-281  
 in programs, 2-54, 3-385  
 of polynomials, 3-238  
 row operations  
 adding rows, 3-283  
 converting rows to a matrix, 3-284  
 deleting rows, 3-284  
 multiplying and adding to  
 another row, 3-260  
 multiplying by a constant, 3-260

norms, 3-280

swapping rows, 3-292  
 RPN syntax  
 converting to, 2-40  
 Runge-Kutta-Fehlberg, 3-274  
 RUN menu, 1-49

**S**

sample standard deviation, 3-300  
 sample variance, 3-367  
 Schur decomposition, 3-298  
 sequential calculations, 3-302, 3-336  
 serial input buffer, 3-318  
 serial interrupt, 3-295  
 serial transmissions, 3-381  
 input buffer, 3-318  
 interrupting, 3-295  
 Server mode  
 terminating, 3-117  
 silicon  
 intrinsic density of, 3-305  
 silicon devices, 4-67  
 single-step execution, 1-47, 1-48, 1-49  
 sinusoidal signal, 4-21  
 SI units, 3-360  
 size  
 binary wordsize, 3-265  
 initial value solution step, 3-277  
 of data transmissions, 3-40  
 of objects, 3-41, 3-309  
 of programs, 2-1  
 of stack, 3-82  
 software  
 version and date, 3-368  
 solenoid, 4-20, 4-44  
 solid geometry, 4-63  
 solid state devices, 4-67

solver

starting, 3-314  
 sound waves, 4-81  
 spectral norm, 3-314  
 spectral radius of a matrix, 3-317  
 sphere, 4-66  
 spring, 4-30, 4-55  
 squaring, 3-316  
 stack  
 calculations on, 1-3  
 clearing, 3-50  
 displaying, 3-345  
 duplicating objects in, 3-97, 3-98, 3-215  
 manipulating, 3-280, 3-281, 3-282, 3-338  
 printing, 3-239, 3-240, 3-241  
 removing objects from, 3-95, 3-96  
 selecting objects from, 3-226  
 size of, 3-82  
 stack display  
 clearing, 3-51  
 stack elements to vectors, 3-370  
 stack syntax  
 in local variable structures, 1-4  
 test commands, 1-17  
 standard deviation  
 population, 3-242  
 sample, 3-300  
 "start" looping, 1-28, 1-30, 2-3, 3-201  
 state change, 4-33, 4-36  
 statistical data  
 chi-square distribution, 3-363  
 clearing, 3-52  
 correlation, 3-65, 3-177  
 covariance, 3-222  
 extrapolating X, 3-235

extrapolating Y, 3-234, 3-236  
 F distribution, 3-364  
 maxima, 3-184  
 mean, 3-185, 3-198  
 minima, 3-194  
 normal distribution, 3-198, 3-365  
 plotting, 3-36, 3-94, 3-137, 3-139, 3-296  
 population covariance, 3-222  
 population standard deviation, 3-242  
 population variance, 3-247  
 recalling, 3-264  
 regression, 3-166, 3-177  
 sample covariance, 3-67  
 sample standard deviation, 3-300  
 sample variance, 3-367  
 sorting by frequency, 3-38  
 specifying dependent variable, 3-389  
 specifying independent and dependent variables, 3-57  
 specifying independent variable, 3-380  
 stored in  $\Sigma$ DATA, 3-208  
 storing, 3-332  
 summing, 3-351  
 summing dependent variables, 3-388  
 summing independent variables, 3-379  
 summing products of variables, 3-388  
 summing squared dependent variable, 3-389  
 summing squared independent variables, 3-379  
 t distribution, 3-365  
 upper chi-square distribution, 3-363  
 upper normal distribution, 3-365  
 upper Snedecor's F distribution, 3-364  
 upper students t distribution, 3-365  
 variance, 3-198  
 step junction, 4-69, 4-74  
 storing  
 flag states, 1-44  
 programs, 1-5  
 strain, 4-76  
 stress, 4-1, 4-76  
 strings  
 action in programs, 1-2  
 as program output, 1-75  
 concatenation, 2-33  
 evaluating, 3-333  
 first characters, 3-204  
 input converted to, 1-60, 3-334  
 locating elements in, 3-234  
 manipulating, 2-7  
 subdirectories  
 clearing, 3-52  
 sublists  
 number used with DOSUBS, 3-102  
 subroutines, 2-5, 2-10, 2-21, 2-38  
 debugging, 1-49  
 in programs, 1-45  
 operation, 1-45  
 single-step execution, 1-49, 3-320  
 summations  
 alternative to looping, 1-40  
 of dependent variables, 3-388

testing  
 algebraics, 1-19  
 binary integers, 1-19  
 flag states, 1-42  
 testing linear structure, 1-20  
 TEST menu, 1-18  
 thermal expansion, 4-39  
 tick-marks  
 setting in plots, 3-30  
 time, 3-347  
 and date, 3-356  
 setting, 3-347  
 time-out  
 during data transmission, 3-324  
 toroid, 4-21, 4-45  
 trace mode, 2-53  
 transistors, 4-67  
 transverse waves, 4-80  
 trapping errors, 1-51  
 triangle, 4-62  
 true (test result), 1-17, 1-19  
 turning off the calculator, 3-211  
 TVM, 3-357  
 begin mode, 3-358  
 end mode, 3-358  
 solving, 3-358  
**U**  
 units  
 converting angular, 3-99, 3-294  
 creating from stack, 3-362  
 factoring, 3-361  
 numeric portion, 3-366  
 SI, 3-360  
 upper chi-square distribution, 3-363  
 upper normal distribution, 3-365  
 of independent variables, 3-379  
 of products of statistical variables, 3-388  
 of squared dependent variables, 3-389  
 of squared independent variables, 3-379  
 of statistical data, 3-351  
 symbolic arrays, 2-29  
 symbolic constants, 3-99, 3-143, 3-183, 3-193  
 evaluating, 3-63  
 symbolic objects  
 evaluating, 3-207  
 system time, 3-346, 3-347  
 setting, 3-347  
**T**  
 tagged objects  
 as program output, 1-74  
 creating, 3-341  
 Taylor's polynomials  
 graphing, 2-45  
 t distribution, 3-365  
 temperature  
 change, 3-344  
 increment, 3-348  
 terminal velocity, 4-49  
 test commands  
 algebraic syntax, 1-17  
 combining results, 1-19  
 comparison functions, 1-17  
 flag tests, 1-42  
 in conditional structures, 1-17, 1-20  
 in loop structures, 1-36, 1-38  
 logical functions, 1-19  
 results of, 1-17, 1-18, 3-159  
 stack syntax, 1-17

upper Snedecor's F distribution, 3-364  
 upper students t distribution, 3-365  
 user-defined errors, 1-51  
 user-defined functions  
 internal structure, 1-16  
 user keyboard  
 defining keys, 3-24  
 user keys  
 assigning, 3-24  
 unassigning, 3-80  
 utility programs, 2-38

## V

variables  
 action in programs, 1-2  
 adding objects to, 3-329  
 clearing, 3-52, 3-53  
 decrementing, 1-39, 3-76  
 defining, 3-77  
 dependent, 3-57, 3-388, 3-389  
 designated as calculated, 3-185  
 designating user-defined, 3-197  
 dividing by, 3-331  
 incrementing, 1-39, 3-150  
 independent, 3-57, 3-151, 3-379, 3-380, 3-388  
 isolating, 3-158  
 list of currently used, 3-367  
 list of particular type, 3-357  
 multiplying, 3-330  
 negating, 3-313  
 ordering, 3-214  
 picture, 3-226  
 port, 3-248  
 printing, 3-240  
 purging, 3-243  
 showing, 3-305

solving multiple, 3-195  
 storing objects in, 3-325  
 subtracting objects from, 3-330  
 types, 3-368  
 variable types, 3-357  
 variance, 3-198  
 population, 3-247  
 sample, 3-367  
 vectored enter, 2-53  
 vectors  
 converting from matrices, 3-54

converting to elements, 3-54  
 creating from elements, 3-56  
 creating from stack, 3-369  
 from stack elements, 3-370  
 to stack elements, 3-371  
 vibrations, 4-80  
 view volume  
 setting the depth, 3-392  
 setting the height, 3-394  
 setting the width, 3-386  
 voltage, 4-9, 4-67  
 Vroom, Fruit of the, 2-52

## W

waiting  
 displaying output, 1-76  
 for keystrokes, 1-72, 1-73  
 warmstart log, 3-377  
 waves, 4-80  
 "while" looping, 1-38, 2-7, 3-270, 3-374  
 while loops, 3-374  
 wordsize (binary)  
 testing, 1-19  
 wordsize of binary integers, 3-335

## X

x-axis  
 specifying display range, 3-384  
 Xmodem  
 receiving objects, 3-384  
 sending objects, 3-386

## Y

y-axis  
 specifying display range, 3-390

