DR LOGO

The Guide at your Side

How to use this chart

As the concluding part to the 8000 Plus Logo series, here is a list of all the various commands (or 'primitives') that Dr. Logo on the Amstrad PCW recognises. There is not room to explain in full what each does, but there is enough information to jog your memory as you are programming.

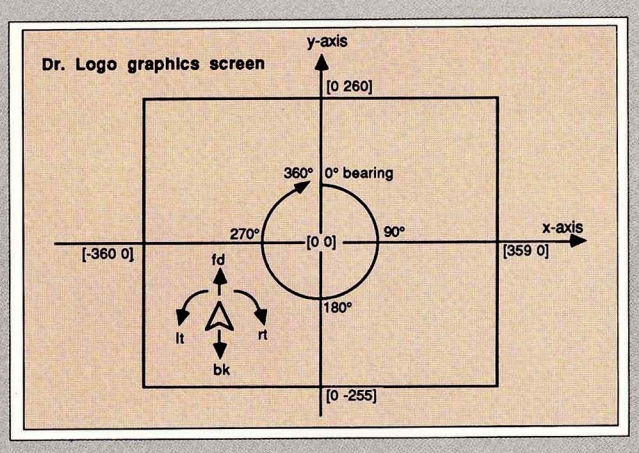
Each entry has the keyword in bold. If the command expects extra information, some example inputs (not in bold) are listed with it.

Where a primitive is described as 'returning' a result, that result must be used in some way – eg. the result of `rc` could be assigned to a variable such as in `make "fred rc` – or an error will occur.

In general, wherever a number is given literally you could use a variable instead. For example, if you had a variable `fred` set to a value of 50 then `fd 50` and `fd :fred` are identical in effect. The exception to this is inside list brackets, [ ], where everything is treated literally.

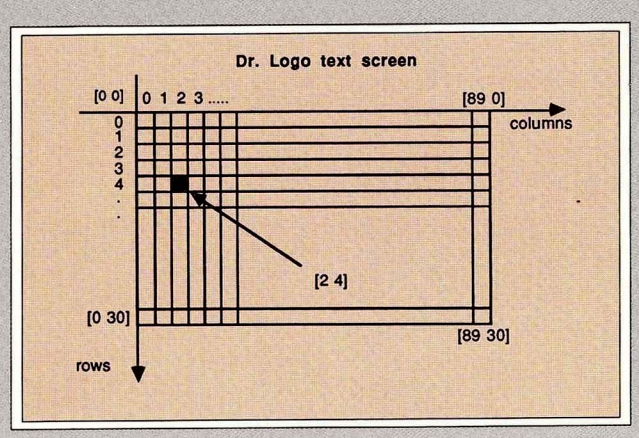
Graphics screen/turtle

fd 50	move turtle forward by specified amount
bk 50	move turtle backward by specified amount
ht	hide turtle
st	show turtle
lt 90	rotate turtle left by angle specified (degrees)
rt 90	rotate turtle right by angle specified (degrees)
seth 135	set turtle's heading to the specified bearing
towards [50 50]	returns bearing of given dot. Use `seth towards` [*x y*] to point turtle at dot [*x y*]
pu	lift pen up (turtle movement doesn't draw anything)
pd	pen down
pe	pen erase (turtle erases previously drawn lines it crosses)
px	makes turtle erase where there is a line, draw where there isn't
setpc 1	set pen colour – 1 means green, 0 black
home	move turtle to centre screen ([0 0]) & bearing 0
setpos [50 50]	move turtle to specified co-ordinate
setx 50	changes turtle's x-axis position to that specified
sety 50	changes turtle's y-axis position
tf	turtle facts – prints out turtle's position etc
fs	full screen – screen devoted to graphics, no text window
clean	clears the graphics, leave turtle where it is
cs	clear screen, return turtle to [0 0] bearing 0
dot [50 50]	draw a dot at the given co-ordinate (doesn't move turtle)
fence	confines turtle to screen limits
window	allows turtle outside screen limits
wrap	turtle going off screen reappears on opposite side
setscrunch 0.5	sets screen aspect ratio to from 0.1 to 10 (set it to 0.5 to get true circles on screen dumps)



Text screen

setsplit 10	make the lower 10 (or whatever) lines of the screen text, the rest graphics
ct	clear text screen
cursor	display current cursor position
setcursor [5 8]	move cursor to specified column and row
ss	select standard screen split
ts	devote whole screen to text only
sf	screen facts – print out cursor position etc.



Output and input

pr :fred ...	prints out the item(s) then a carriage return
type :fred ...	prints out the item(s) without a carriage return
show :fred	prints out the item then a carriage return. Lists are printed with their brackets
po "fred	prints out the value of the named variable as "fred is ..."
pons	prints out all global variables in `po` style
pops	prints out all procedure definitions
pots	prints out titles of all procedures
poall	prints out all variables and procedure definitions
copyon	echo all screen text to the printer
copyoff	stop text echoing to printer
rc	returns next character typed
rq	returns next word(s) typed until [RETURN] pressed
rl	as `rq`, but returns words read as a list
noformat :fred ...	uncertain use (!) – seems the same as `pr`

Words and lists

ascii "a	returns the ASCII value of first letter of word/list
bf [1 2 3]	returns all but first item of list/letter of word
bl [1 2 3]	returns all but last item of list/letter of word
char 27	returns character whose ASCII value is given
count [1 2 3]	returns number of items in list/letters in word
first [1 2 3]	returns first item of list/letter of word
last [1 2 3]	returns last item of list/letter of word
fput 1 [2 3]	joins first item to beginning of word/list and returns result
lput 3 [1 2]	joins first item to end of word/list and returns result
item 2 [1 2 3]	returns requested numbered item of list/letter of word (eg. 2nd item of list)
lc "FRED	returns lower case version of word
uc "fred	returns upper case version of word
(**list** 1 2 3)	returns a list of the following inputs
(**se** 1 2 3)	as `list` (Logo calls lists 'sentences' too)
piece 2 4 [a b c d e]	returns (in this case) items 2 to 4 of the list or word
shuffle [1 2 3]	returns a randomly shuffled version of list
word ("a "n "d)	joins together all the inputs and returns them as a word
where	returns location of item in list (use after `memberp`, eg. `memberp "x [a x b]` then `where` gives 2)

Arithmetic

arctan 0.707	returns inverse tangent (in degrees) of value
cos 60	returns cosine of angle in degrees
sin 30	returns sine of angle in degrees
int 1.7	returns integer part of number
round 1.7	returns number rounded to nearest integer
quotient 9 2	returns results of integer division (eg. 4 in this case)
remainder 9 2	returns remainder from integer division (eg. 1 in this case)
random 100	returns random integer from 0 to number
rerandom	resets random number generator sequence
1 + 2	returns result of addition
2 - 1	returns result of subtraction
2 * 2	returns result of multiplication
9 / 2	returns result of floating point division

Procedures and variables

to fred :var ...	Starts definition of named procedure. List of its input variables follows.
end	ends procedure definition
local "fred ...	Restricts named variables to current procedure & sub-procedures – use in recursive procedures
make "fred 1	Set the named variable to the value given
thing "fred	returns the value of the named variable
pause	wait until the user types `co`
co	resume program after a `pause` command
label "fred	sets up a label for `go` statements to jump to
go "fred	go to the instruction following the named label
if *test* [*c1*] [*c2*]	if the condition *test* is TRUE then do command list *c1*, otherwise do command list *c2*.
op 10	exit the current procedure making the specified value the procedure's output
repeat *n* [*cmnds*]	repeat the command list in the brackets *n* times
run [*cmnds*]	run the commands in the brackets
stop	return to the previous procedure (or stop program if at top level)
throw "fred	passes control to correspondingly labelled `catch`. Use `throw TOPLEVEL` to jump back to command mode.
catch "fred	receives control from corresponding `throw`.

Conditional tests

(use with the `if` command to determine a course of action)

dotc [50 50]	returns TRUE if there is a dot at the specified graphics screen co-ordinate
emptyp [ ]	returns TRUE if the input is an empty list
equalp :fred 1	returns TRUE if the two inputs are equivalent
memberp 1 [1 2 3]	returns TRUE if the first item is a member of the list
listp :fred	returns TRUE if the input is a valid list
numberp 1	returns TRUE if the input is a valid number
wordp "fred	returns TRUE if the input is a valid word
namep "fred	returns TRUE if the input is the name of a valid variable
keyp	returns TRUE if a key is currently being pressed
:x = 50	returns TRUE if the two inputs are equal
:x > 50	returns TRUE if the first input is greater than the second
:x < 50	returns TRUE if the first input is less than the second
(:x=50) **and** (:y=50)	returns TRUE if both input conditions are TRUE
(:x=50) **or** (:y=50)	returns TRUE if either of the input conditions are TRUE
not (:x=50)	returns TRUE if the input condition is FALSE

Property lists

pprop "fred "age 45	'put property' – sets up 'fred' to have a property 'age' with value '45'
plist "fred	returns the current property list of 'fred'
gprop "fred "age	'get property' – returns the value of the named property (eg. 'fred's age' here)
glist "age	returns a list of all people/objects with a property 'age' defined
remprop "fred "age	removes the property 'age' from fred's property list
pps	prints out all property list pairs

Housekeeping

ed "procedure	calls up the Logo editor on the named procedure
edall	edits all known procedures/variables
edf "filename	edits the named file and then loads it into Logo
er "procedure	erase the named procedure definition
erall	erase everything from working memory
ern "fred	erase the named procedure
nodes	gives a measure of free space left
recycle	cleans up the internal workspace
changef "new "old	change file name on disc from 'old' to 'new'
defaultd	display current logged disc drive
dir "a:	list all Logo program files on specified drive
dirpic "b:	list all Logo picture files on specified drive
erasefile "filename	erase the named Logo program file
erasepic "filename	erase the named Logo picture file
load "filename	add the procedures in the named file to the workspace
save "filename	save the currently defined procedure to the named file
loadpic "filename	load the names picture file
savepic "filename	save the current screen as a picture file
setd "m:	sets the default disc drive as specified
error	prints out list of recent errors
bye	exit back to CP/M

Debugging

trace	display procedure calls as they are executed
notrace	turn `trace` off
watch	as `trace`, pauses at each procedure call
nowatch	turn `watch` off

Advanced miscellany

.contents	display all names Logo knows about
define "procedure [[*argument_list*][*instruction_list*]]	equivalent to `to...end`. Allows procedures to define other procedures.
text "procedure	prints out procedure definitions as expected by `define`
.deposit 65300 255	equivalent of BASIC's POKE
.examine 65300	equivalent of BASIC's PEEK
.out 245 1	equivalent of BASIC's OUT
.in 245	equivalent of BASIC's INP

Unimplemented

The following names appear in the list of known Logo commands listed out by `.contents`, but do not do anything(!): **paddle**, **pal**, **wait**, **tones**, **buttonp**, **setpal.**

Idea and material by John Connell