TANDY®

Color Computer 3

BASIC

Quick Reference Guide

RADIO SHACK, A Division of Tandy Corporation
FORT WORTH, TEXAS 76102

7A6

Printed in Korea
811019430A

Tandy Color Computer 3
BASIC

Quick Reference Guide

Contents

Tandy Color Computer 3 BASIC Quick Reference Guide

10 9 8 7 6 5 4 3 2 1

Start-Up

1. Turn on the display.
2. If you are using a television, select channel 3 or 4, and set the antenna switch to **COMPUTER**.
3. If you are using a Program Pak™, insert it now before turning on the computer.
4. Turn on any accessory equipment (printer, Multi-Pak interface, etc.).
5. Turn on the computer.
6. If you are not using a Program Pak, the BASIC start-up message appears on the display, followed by:

   ```
   OK
   ```

7. Some Program Pak documentation may tell you to type in EXEC &HC000. To run these Program Paks type in:

   ```
   EXEC &HEØ1Ø
   ```

The computer is now ready for use.

Commands

ATTR *c1,c2*,B,U

Sets display attributes of a high-resolution text screen.

c1 Foreground color

c2 Background color

B Character blink on

U Underline on

```
ATTR 3,2,U
```

AUDIO *switch*

Connects or disconnects cassette output to the display speaker

ON Switches ON sound from cassette player to display speaker.

OFF Switches OFF sound from cassette player to display speaker.

```
AUDIO OFF
```

CIRCLE *(x,y),r,c,h,s,e*

Draws a circle on the current low-resolution graphics screen.

x,y Center point

r Radius

c Color

h Height/width ratio

s Starting point

e Ending point

```
CIRCLE (65,43),20,1,,,.5,.8
```

CLEAR *n,h*

Erases variables, reserves string workspace, and reserves high memory for machine language programs

n String workspace size

h Highest BASIC memory address

```
CLEAR 200,20000
```

CLOAD "*filename*"

Loads program *filename* from cassette. If *filename* is not specified, BASIC loads the first program file found.

filename Name of desired program. Name can have as many as 8 characters.

```
CLOAD "PUPPIES"
```


CLOADM "*filename*",*o*

Loads machine-language program *filename* from cassette. If *filename* is not specified, BASIC loads the first machine-language program found.

filename Name of desired machine-language program. Name can have as many as 8 characters.

o Memory address offset. If specified, BASIC loads the machine-language program *o* bytes higher in memory than normal.

```
CLOADM "GRAPHICS",2730
```

CLOSE# *d*

Closes access to specified device or file. If *d* is not specified, BASIC closes all open devices and files.

d Number of device or file

```
CLOSE #-1
```

CLS *c*

Clears the text screen to a specified color. When in high-resolution text mode, BASIC also sets the background color. If *c* is not specified, BASIC uses the current background color.

c Color code

```
CLS 2
```

COLOR *c1, c2*

Sets foreground and background colors of the current low-resolution graphics screen.

c1 Foreground color code (0-8)

c2 Background color code (0-8)

```
COLOR 2,3
```

CONT

Continues program execution after a program halt from the BREAK key or a STOP instruction.

```
CONT
```

CSAVE "*filename*",*A*

Saves program *filename* on cassette.

filename Name of program to save. Name can have as many as 8 characters.

A Selects ASCII format

```
CSAVE "NEWFILE",A
```

CSAVEM *"filename",l,h,e*

Saves machine-language program *filename* on cassette.

filename	Name of machine-language program being saved. Name can have as many as 8 characters
l	Lowest address of machine-language program.
h	Highest address of machine-language program.
e	Exec address of machine-language program.

```
CSAVEM "GRAPHICS",28000,29000,28032
```

DATA *constant,constant,...*

Stores numeric and string constants for use with READ statement.

constant	String or numeric constant(s). such as: 127.2985 or Beagle.

```
DATA 45,CAT,98,DOG,24.3,1000
```

DEF FN *name (variables) = formula*

Defines a numeric function.

name	Name of function. Must be a valid variable name.
variables	List of dummy variables used in *formula*.
formula	Defines the operation.

```
DEF FNA(B)=B*(B+(1/B))
```

DEFUSR *n = addr*

Defines the starting address of a machine-language subroutine

n	Number of machine-language routine. (0-9)
addr	Starting address of machine-language routine. (0-65535)

```
DEFUSR0=28032
```

DEL *l1-l2*

Deletes program lines.

l1 Lowest line number to delete.

l2 Highest line number to delete

l1	Deletes 1 line.
-l2	Deletes from beginning of program up to and including *l2*
l1-	Deletes from and including *l1* to the end of the program.
l1-l2	Deletes from and including *l1* to and including *l2*.

```
DEL 40-75
```


DIM *array(size),array(size),...*

Dimensions one or more arrays.

```
DIM A$(3,10),R4(22)
```

DRAW *string*

Draws a line on the current low-resolution graphics screen as specified by *string*. The *string* commands are:

A	Angle
BM	Blank move
C	Color
D	Down
E	45 degree angle
F	135 degree angle
G	225 degree angle
H	315 degree angle
L	Left
M	Move draw position
N	No update
R	Right
S	Scale
U	Up
X	Execute substring

```
DRAW "BM128,96;U25;R25;D25;L25"
```

EDIT *line number*

Edits a program line. After fetching specified *line number*. EDIT recognizes several commands:

C	Changes characters
D	Deletes characters
H	Hacks off rest of line and permits insertion
I	Inserts characters
K	Kills rest of line
L	Lists line being edited
S	Searches for a specified character
X	Extends line
SHIFT ←	Returns to line mode

```
EDIT 40
```

END

Marks the end of a BASIC program.

```
END
```

EXEC (*address*)

Transfers control to a machine-language program at *address*. If *address* is omitted, control is transferred to address set in the last CLOADM.

```
EXEC 28032
```

FOR *variable* = *n1* TO *n2* STEP *n3*

Defines the beginning of a loop. The end is specified by NEXT.

variable	Loop counter variable
n1	Starting value of counter
n2	Ending value of counter
n3	Increment or decrement value of counter

```
FOR Z=35 TO 125 STEP 5
```

GET (*sx*,*sy*)-(*ex*,*ey*),*array*,G

Stores a rectangle that is on the low-resolution graphics screen in an *array*, for future use by the PUT command.

sx,sy	First corner of rectangle
ex,ey	Opposite corner of rectangle
array	Two dimensional array
G	Selects full graphic detail storage. Requires use of PSET, PRESET, AND, OR, or NOT when using PUT.

```
GET (22,34)-(47,38),M,G
```

GOSUB *line number*

Calls a subroutine beginning at the specified line number.

```
GOSUB 330
```

GOTO *line number*

Jumps to the specified line number.

```
GOTO 125
```

HBUFF *buffer*,*size*

Reserves an area in memory for high-resolution graphics.

buffer	Number of buffer selected
size	Defines buffer size. BASIC allows a *buffer* to have a maximum size of 7931.

```
HBUFF 1,65
```


HCIRCLE (*x*,*y*),*r*,*c*,*h*,*s*,*e*

Draws a circle on the high-resolution graphics screen.

x,y	Center point
r	Radius
c	Color
h	Height/width ratio
s	Starting point
e	Ending point

```
HCIRCLE (55,64),20,2,3,.4,.7
```

HCLS *c*

Clears the high-resolution graphics screen to a specified color.

c	Color If unspecified, BASIC uses current background color.

```
HCLS 2
```

HCOLOR *c1*,*c2*

Sets foreground and background color on the high-resolution graphics screen.

c1	Foreground color (0-15)
c2	Background color (0-15)

```
HCOLOR 2,3
```

HDRAW *string*

Draws a line on the high-resolution graphics screen as specified by *string*. The *string* commands are:

A	Angle
BM	Blank move
C	Color
D	Down
E	45 degree angle
F	135 degree angle
G	225 degree angle
H	315 degree angle
L	Left
M	Move draw position
N	No update
R	Right
S	Scale
U	Up
X	Execute substring

```
HDRAW "BM128,96;U25;R25;D25;L25"
```

HGET (*sx,sy*)-(*ex,ey*),*buffer*

Stores a rectangle that is on the high-resolution graphics screen into a *buffer* previously set up by the HBUFF command for future use by the HPUT command.

sx,sy	First corner of rectangle
ex,ey	Opposite corner of rectangle
buffer	Number of *buffer*

```
HGET (21,32)-(28,37),1
```

HLINE (*x1,y1*)-(*x2,y2*),*c*,*a*

Draws a line on the high-resolution graphics screen.

(*x1,y1*)	Starting point of line. If omitted the line starts at the last ending point, or the center of the screen.
-(*x2,y2*)	Ending point of HLINE.
c	Defines color (Required). PSET selects current foreground color. PRESET selects current background color.
a	Box action (Optional). If omitted, BASIC draws a line. If **B** is used, BASIC draws a box, using the starting and ending points as opposite corners of the box. If **BF** is used, BASIC draws a solid box.

```
HLINE (22,33)-(100,90),3,BF
```

HPAINT (*x,y*),*c1*,*c2*

Paints an area on the high-resolution graphics screen.

x,y	Starting point
c1	Paint color
c2	Border color

```
HPAINT (55,66),2,3
```

HPRINT (*x,y*),*message*

Prints *message* on high-resolution graphics screen.

x,y	Starting character position
message	String to print

```
HPRINT (20,12),"HELLO!"
```


HPUT (*sx,sy*)-(*ex,ey*),*b*,*a*

Copies graphics from a buffer to a rectangle on the high-resolution graphics screen.

sx,sy	First corner of rectangle
ex,ey	Opposite corner of rectangle
b	Buffer number
a	Action used. Actions include: PSET, PRESET, AND, OR, NOT

```
HPUT (22,33)-(28,37),1,PSET
```

HRESET (*x,y*)

Resets a point on the high-resolution graphics screen to the background color.

```
HRESET (22,33)
```

HSCREEN *mode*

Selects a high-resolution graphics screen *mode*. Modes 1-4 also clear the high-resolution graphics screen.

mode	Mode number. Mode numbers are: 0 — Low resolution 1 — 320 X 192, 4-color 2 — 320 X 192, 16-color 3 — 640 X 192, 2-color 4 — 640 X 192, 4-color

```
HSCREEN 4
```

HSET (*x,y,c*)

Sets point *x,y* on the high-resolution graphics screen to Color *c*. If you omit *c*, BASIC uses the foreground color.

```
HSET (22,33,2)
```

HSTAT *v1,v2,v3,v4*

Returns information regarding the high-resolution text screen cursor to variables *v1,v2,v3*, and *v4*.

v1	Character code
v2	Character attribute
v3	Cursor X coordinate
v4	Cursor Y coordinate

```
HSTAT C,A,X,Y
```

IF *test* THEN *#1* ELSE *#2*

Performs a test. If the results are true, the computer executes the first instruction (*#1*). If the results are false, the computer executes the second instruction (*#2*).

```
IF A<N THEN PRINT "A<N" ELSE PRINT "A>=N"
```

INPUT *var1,var2,...*

Reads data from the keyboard, and saves it in one or more variables.

```
INPUT K3
```

INPUT #-1 *var1,var2,...*

Reads data from a cassette, and saves it in one or more variables.

```
INPUT #-1,C$
```

LET

Assigns a value to a variable (optional).

```
LET A3=27
```

LINE *(x1,y1)-(x2,y2),c,a*

Draws a line on the current low-resolution graphics screen.

- *(x1,y1)* Starting point of line. If omitted the line starts at the last ending point, or the center of the screen.
- *-(x2,y2)* Ending point of line.
- *c* Defines color (Required). PSET selects current foreground color. PRESET selects current background color.
- *a* Box action (Optional). If omitted, BASIC draws a line. If **B** is used, BASIC draws a box using the starting and ending points as opposite corners of the box. If **BF** is used, BASIC draws a solid box.

```
LINE (22,33)-(27,39),PSET,BF
```

LINE INPUT

Reads data from the keyboard, and saves it in a variable. Commas are characters, and not delimiters.

```
LINEINPUT A$
```


LIST *l1-l2*

Lists specified program line(s) or the entire program on the screen.

l1 Lowest line number to list.

l2 Highest line number to list.

- *l1* Lists 1 line.
- *-l2* Lists from beginning of program up to and including *l2*.
- *l1-* Lists from and including *l1* to the end of the program.
- *l1-l2* Lists from and including *l1* to and including *l2*.

```
LIST 20-45
```

LLIST *l1-l2*

Lists specified program line(s) or the entire program on the printer.

l1 Lowest line number to list.

l2 Highest line number to list.

- *l1* Lists 1 line.
- *-l2* Lists from beginning of program up to and including *l2*.
- *l1-* Lists from and including *l1* to the end of the program.
- *l1-l2* Lists from and including *l1* to and including *l2*.

```
LLIST -90
```

LOCATE *x,y*

Moves the high-resolution text screen cursor to position *x,y*.

```
LOCATE 20,12
```

LPOKE *location,value*

Stores a value (0-255) in a virtual memory location (0-524287 decimal or 0-$7FFFF hexadecimal).

```
LPOKE 480126,241
```

MID$ *(s,p,l)*

Replaces a portion of the contents of string variable *s* with another string.

- *s* String being modified
- *p* Starting position in string
- *l* Length of section being modified

```
MID$ (A$,4,3)="CAT"
```

MOTOR

Turns the cassette ON or OFF.

```
MOTOR ON
```

NEW

Erases everything in memory.

```
NEW
```

NEXT *v1,v2,...*

Defines the end of a FOR loop.

v1,v2 Optional variable names used for nested loops. If used, list in reverse order of FOR variables. If omitted, only defines the end of the last loop declared.

```
NEXT X,Y,Z
```

ON BRK GOTO *line number*

Jumps to *line number* when the BREAK key is pressed.

```
ON BRK GOTO 120
```

ON ERR GOTO *line number*

Jumps to *line number* when an error occurs.

```
ON ERR GOTO 120
```

ON...GOSUB

Multiway call to specified subroutines.

```
ON A GOSUB 100,230,500,1125
```

ON...GOTO

Multiway branch to specified lines.

```
ON A GOTO 100,230,500,1125
```

OPEN *m,#dev,f*

Opens specified file for data transmission.

m Transmission mode
I — Input
O — Output

#dev #-2 — Printer
#-1 — Cassette
#0 — Keyboard or screen

f Filename

```
OPEN "O",#-1,"DATA"
```


PAINT *(x,y),c1,c2*

Paints an area on the current low-resolution graphics screen.

x,y Starting point

c1 Paint color

c2 Border color

```
PAINT (44,55),2,3
```

PALETTE CMP or **RGB**

Resets the palette registers to the standard colors for a composite monitor or a television set (PALETTE CMP). or for an RGB monitor (PALETTE RGB).

```
PALETTE CMP
```

PALETTE *pr, cc*

Stores Color Code *cc* (0-63) into Palette Register *pr* (0-15).

```
PALETTE 1,13
```

PCLEAR *n*

Reserves *n* number of 1.5 K graphics memory pages.

```
PCLEAR 4
```

PCLS *c*

Clears current low-resolution graphics screen with Color *c*. If you omit *c*, BASIC uses the background color.

```
PCLS 0
```

PCOPY *s* TO *d*

Copies low-resolution graphics from source page to destination page.

s Source page number

d Destination page number

```
PCOPY 1 TO 2
```

PLAY *string*

Plays music as specified by *string*. The string commands are:

A-G Notes

L Length

O Octave

P Pause

T Tempo

#or+ Sharp

\- Flat

```
PLAY "L1;A;A#;A-"
```

PMODE *mode,page*

Selects resolution and first memory page of a low-resolution graphics screen.

mode 0 — 128 x 96, 2-color
1 — 128 x 96, 4-color
2 — 128 x 192, 2-color
3 — 128 x 192, 4-color
4 — 256 x 192, 2-color
If omitted, BASIC uses the last value set. At power on, BASIC uses 2.

page Start page. If omitted, BASIC uses the previously set page. At power on, BASIC uses 1.

```
PMODE 4,1
```

POKE *location,value*

Stores a value (0-255) in a memory location (0-65535 decimal or 0-$FFFF hexadecimal).

```
POKE 28000,241
```

PRESET (*x,y*)

Resets a point on the current low-resolution graphics screen to the background color.

```
PRESET (22,33)
```

PRINT *message*

Prints on the text screens.

```
PRINT "HELLO!"
```

PRINT #-1,*data*

Writes data to cassette.

```
PRINT #-1,A$
```

PRINT #-2,*data*

Prints on the printer.

```
PRINT #-2,"HELLO!"
```

PRINT TAB(*n*)

Moves the cursor to column *n* on the low and high-resolution text screens.

```
PRINT TAB(22);"HELLO!"
```


PRINT USING"*format*";*data*

Prints numbers in the specified format on the text screen. The *format* commands are:

#	Formats numbers.
.	Decimal point.
,	Prints comma to the left of every third character.
**	Fills leading spaces with asterisks.
$	Prints leading dollar sign.
$$	Floating dollar sign.
+	Leading or trailing sign.
↑↑↑↑	Exponential format.
-	Minus sign after negative numbers.
!	Prints first string character.
%spaces%	String field. Length of field is number of spaces plus 2.

```
PRINT USING "##.####";1/3
```

PRINT *@n,message*

Prints *message* on low-resolution text screen at position *n*.

```
PRINT @11,"HELLO!"
```

PSET (*x,y,c*)

Sets point *x,y* on the current low-resolution graphics screen to Color *c*. If *c* is omitted, BASIC uses the foreground color.

```
PSET (22,33,2)
```

PUT (*sx,sy*)-(*ex,ey*),*v,a*

Copies graphics from an array to a rectangle on the low-resolution graphics screen.

sx,sy	First corner of rectangle
ex,ey	Opposite corner of rectangle
v	Two dimensional array
a	Action used. Actions include: PSET, PRESET, AND, OR, NOT

```
PUT (22,33)-(27,39),A,PSET
```

READ *var1,var2,...*

Reads the next item(s) in a DATA line. Saves data in specified variable(s).

```
READ A1,B,C7
```

REM *comment*

Lets you insert comments in a program line. The computer ignores everything in the line, after the REM.

```
REM THIS IS A COMMENT LINE
```

RENUM *newline,startline,increment*

Renumbers program lines.

newline New starting line
startline Line where renumbering starts
increment Step value for lines

```
RENUM 1,1,10
```

RESET (*x,y*)

Resets a point on the low-resolution text screen to the background color.

```
RESET (22,33)
```

RESTORE

Sets the computer's pointer back to the first item on the first DATA line.

```
RESTORE
```

RETURN

Returns the computer from a subroutine to the BASIC word following GOSUB.

```
RETURN
```

RUN

Executes a program.

```
RUN
```

SCREEN *type,colors*

Selects low-resolution screen modes and color sets.

type 0 — Text
1 — Graphics

colors 0 — Color set 0
1 — Color set 1

```
SCREEN 0,1
```

SET (*x,y,c*)

Sets point *x,y* on the low-resolution text screen to Color *c*. If you omit *c*, BASIC uses the foreground color.

```
SET (11,11,3)
```


SKIPF *filename*

Skips to next program on cassette tape or to the end of a specified program.

filename Optional name of program to skip over.

```
SKIPF "DATA"
```

SOUND *tone,duration*

Sounds a specified tone for a specified duration.

tone 1-255 sets pitch
duration 1-255 sets duration

```
SOUND 33,22
```

STOP

Stops execution of a program.

```
STOP
```

TIMER = *n*

Sets timer to *n*.

```
TIMER=120
```

TROFF

Turns off program tracer.

```
TROFF
```

TRON

Turns on program tracer.

```
TRON
```

WIDTH *n*

Sets the text screen to resolution *n*:

32 — 32 X 16 (low-resolution text)
40 — 40 X 24 (high-resolution text)
80 — 80 X 24 (high-resolution text)

```
WIDTH 80
```

Functions

ABS (*n*)

Returns the absolute value of *n*.

```
A=ABS(B)
```

ASC (*string*)

Returns the code of the first character in *string*.

```
A=ASC(B$)
```

ATN (*n*)

Returns the arctangent of *n* in radians.

```
A=ATN(B/3)
```

BUTTON (*n*)

Returns 1 if Joystick Button *n* is being pressed; 0 if Joystick Button *n* is not being pressed. *n* can be:

0 — Right joystick, Button 1 (old joystick)
1 — Right joystick, Button 2
2 — Left joystick, Button 1 (old joystick)
3 — Left joystick, Button 2

```
A=BUTTON(Ø)
```

CHR$ (*n*)

Returns the character corresponding to character code *n*.

```
A$=CHR$(65)
```

COS (*angle*)

Returns the cosine of an *angle* using radians.

```
A=COS(B)
```

EOF (*d*)

Returns FALSE (0) if there is more data; TRUE (-1) if end of file has been read.

d Device number:
-1 Cassette

```
IF EOF(-1)=-1 THEN 22Ø
```

ERLIN

Returns the BASIC line number where an error has occurred.

```
IF ERLIN=11Ø THEN 2ØØ
```

ERNO

Returns the BASIC error number for the error that has occurred.

```
IF ERNO=2Ø THEN CLOSE
```


EXP (*n*)

Returns a natural exponential number (e^*n*).

```
A=EXP(B*1.15)
```

FIX (*n*)

Returns the truncated integer of *n*. Unlike INT, FIX does not return the next lower number for a negative *n*.

```
A=FIX(B-.2)
```

HEX$ (*n*)

Returns a string with the hexadecimal value of *n*.

```
PRINT HEX$(A);"=";A
```

HPOINT (*x,y*)

Returns information on point *x,y* from the high-resolution graphics screen:

0 Point is reset
Code Point is set

```
IF HPOINT(22,33)=Ø THEN 2ØØ
```

INKEY$

Checks the keyboard and returns the key being pressed or, if no key is being pressed, returns a null string (" ").

```
A$=INKEY$
```

INSTR (*p,s,t*)

Searches a string. Returns location of a target string in a search string.

p Start position of search
s String being searched
t Target string

```
A=INSTR (1,M5$,"BEETS")
```

INT (*n*)

Converts *n* to the largest integer that is less than or equal to *n*.

```
A=INT(B+.5)
```

JOYSTK (*j*)

Returns the horizontal or vertical coordinate (j) of the left or right joystick:

0 — Horizontal, right joystick
1 — Vertical, right joystick
2 — Horizontal, left joystick
3 — Vertical, left joystick

```
A=JOYSTK(Ø)
```

LEFT$ (*string,length*)

Returns the left portion of a string.

length Specifies number of characters returned.

```
A$=LEFT$(B$,3)
```

LEN (*string*)

Returns the length of *string*.

```
A=LEN(B$)
```

LOG (*n*)

Returns the natural logarithm of *n*.

```
A=LOG(B/2)
```

LPEEK (*memory location*)

Returns the contents of a virtual memory location (0-524287 decimal or 0-$7FFFF hexadecimal).

```
A=LPEEK(&H7FFFØ)
```

MEM

Returns the amount of free memory.

```
A=MEM
```

MID$ (*s,p,l*)

Returns a substring of string *s*.

s Source string

p Starting position of substring

l Length of substring

```
A$=MID$ (B$,Z,2)
```

PEEK (*memory location*)

Returns the contents of a memory location (0-65535 decimal or 0-&HFFFF hexadecimal).

```
A=PEEK (3ØØ2Ø)
```

POINT (*x,y*)

Returns information on point *x,y* from the low-resolution text screen:

-1 Point is part of an alphanumeric character

0 Point is reset

Code Point is set

```
A=POINT (22,33)
```

POS (*dev*)

Returns the current print position.

dev Print device number:

0 — Screen

-2 — Printer

```
A=POS (Ø)
```


PPOINT (*x,y*)

Returns information on point *x,y* from the low-resolution graphics screen:

0 Point is reset

Code Point is set

```
A=PPOINT (22,33)
```

RIGHT$ (*string,length*)

Returns the right portion of a string.

length Specifies number of characters returned.

```
A$=RIGHT$ (B$,4)
```

RND (*n*)

Generates a "random" number between 1 and *n* if *n*>1, or between 0 and 1 if *n*=0.

```
A=RND (Ø)
```

SGN (*n*)

Returns the sign of *n*:

-1 — Negative

0 — 0

1 — positive

```
A=SGN(A+.1)
```

SIN (*angle*)

Returns the sine of *angle* using radians.

```
A=SIN(B/3.14159)
```

STRING$ (*l,c*)

Returns a string of a repeated character.

l Length of string

c Character used. Can be a code, or a string.

```
A$=STRING$(22,"A")
```

STR$ (*n*)

Converts *n* to a string.

```
A$=STR$(1.234)
```

SQR (*n*)

Returns the square root of *n*.

```
A=SQR(B/2)
```

TAN (*angle*)

Returns the tangent of *angle* using radians.

```
A=TAN(B)
```

TIMER

Returns the contents of the timer (0-65535).

```
A=TIMER/18
```

USR*n* (*argument*)

Calls machine-language subroutine *n*, passes it an argument, and returns a value from the subroutine to the BASIC program.

```
A=USRØ(B)
```

VAL (*string*)

Converts a string to a number.

```
A=VAL("1.23")
```

VARPTR (*variable*)

Returns a pointer to where a variable is located in memory.

```
A=VARPTR(B)
```


Operators

↑	Exponentiation
–, +	Unary negative, positive
*, /	Multiplication, division
+, –	Addition and concatenation, subtraction
NOT	Logical operators
AND	
OR	
<, >, =,	Relational tests
<=, >=, <>	

Control Keys

←	Cancels last character typed; moves cursor back one space.
SHIFT ←	Erases current line.
BREAK	Interrupts program, and returns to command level.
CLEAR	Clears the screen.
ENTER	Marks the end of the current line
Space Bar	Enters a space (blank) character, and moves the cursor forward one space.
SHIFT @	Causes a running BASIC program to pause. Press the space bar to continue.
SHIFT 0	All-caps/upper-lowercase keyboard switch.
F1	Hold down during power up to select alternate color set.

Special Characters

'	Abbreviation for REM.
$	Makes variable string type.
&H	Makes numeric constants hexadecimal.
&O	Makes numeric constants octal.
:	Separates statements on same line.
?	Same as PRINT.
,	PRINT punctuation: spaces over to the next PRINT zone.
;	PRINT punctuation; separates items in a PRINT list, but does not add spaces when printed.

Video Control Codes

Dec	Hex	PRINT CHR$(code)
8	8	Backspaces and erases current character.
13	D	Line feed with carriage return.
32	20	Space

Error Codes

AO	18	Already Open
BS	8	Bad Subscript
CN	16	Cannot Continue
DD	9	Attempt to Redimension Array
DN	19	Device Number Error
DS	24	Direct Statement
/0	10	Division by Zero
FC	4	Illegal Function Call
FD	17	Bad File Data
FM	21	Bad File Mode
HP	39	Hires Print Error
HR	38	Hires Graphics Error
ID	11	Illegal Direct Statement
IE	23	Input Past End of File
IO	20	Input/Output Error
LS	14	String Too Long
NF	0	Next Without For
NO	22	File Not Open
OD	3	Out of Data
OM	6	Out of Memory
OS	13	Out of String Space
OV	5	Overflow
RG	2	Return Without Gosub
SN	1	Syntax Error
ST	15	String Formula Too Complex
TM	12	Type Mismatch
UL	7	Undefined Line