BASIC Statements

AUTO *start, increment* Numbers lines automatically.
AUTO AUTO 150, 20 AUTO ,5

CLEAR *n* Reserves *n* bytes of string storage space; initializes all variables.
CLEAR CLEAR 75 CLEAR 0

CLOAD Loads BASIC program file from cassette. Only the first character of the file name is used.
CLOAD CLOAD "MIXIT" CLOAD # −2, "PROG"

CLOAD? Compares program on tape byte-for-byte with resident program.
CLOAD? CLOAD? "MIXIT"
CLOAD # −2, ? "PROG"

CLOSE *b* Closes all open file-buffers or specified buffer(s) *b*.
CLOSE CLOSE 1, 2, 8 CLOSE N

CLS Clears the display.
CLS

CMD"D" Loads and executes DEBUG (type G ENTER to return to BASIC, BREAK anytime thereafter to execute DEBUG).
CMD "D"

CMD "I" Returns to TRSDOS and executes specified command or loads specified file (parameters not allowed).
CMD "I", "DIR" CMD "I", "MIXIT"

CMD "R" Starts clock and enables interrupts; use after cassette operations.
CMD "R"

CMD "S" Returns control to TRSDOS.
CMD "S"

CMD "T" Stops clock and disables interrupts; use before cassette operations.
CMD "T"

CONT Continues execution of program after BREAK or STOP.
CONT

CSAVE Stores resident program on cassette tape. A file name is required. Only the first character of the file name is used.
CSAVE "MIXIT" CSAVE # −2, "PROG"

DATA Stores data to be accessed by a READ statement.
DATA "LINCOLN, A.", 1861, ILLINOIS

DEFDBL Defines variables as double-precision.
DEFDBL V, X-Z

DEF FN Defines a user-created function.
DEF FNL$ (X) = STRING$ (X, 45)

DEFINT Defines variables as integer type.
DEFINT A, I-N

DEFSNG Defines variables as single-precision.
DEFSNG I, W-Z

DEFSTR Defines variables as string type.
DEFSTR C, L-Z

DEFUSR*n* Defines entry point for machine-language subroutine called by USR*n*. If *n* is omitted, 0 is assumed.
DEFUSR=32000 DEFUSR4 = &H7D7E

DELETE Erases program lines from memory.
DELETE 1205 DELETE −80 DELETE.

DIM Dimensions one or more arrays.
DIM R(75), W(40) DIM AR$(8, 25)
DIM L%(3, 18, 5)

EDIT Puts computer into edit mode for specified line. See **Edit Commands.**
EDIT 100 EDIT.

END Ends program execution.
END

ERROR(*n*) Simulates the specified error, *n* = 1−23.
ERROR(1)

FIELD Organizes a random file buffer into fields.
FIELD 3, 16 AS NM$, 25 AS AD$

FOR . . . TO . . . STEP/NEXT Opens program loop.
FOR I = 1 TO 8 (. . .) NEXT I
FOR C!=0 TO 5 STEP .2 (. . .) NEXT C!

GET *b, record number* Gets specified or next record from a disk file (random access); stores it in buffer *b*
GET 1 GET 1, 25

GOSUB Transfers program control to the specified subroutine.
GOSUB 750

GOTO Transfers program control to the specified line.
GOTO 180

IF . . . THEN . . . ELSE Tests conditional expression.
IF P = Q THEN 200
IF N% < 0 THEN 150 ELSE N% = N% + 1

INPUT Inputs data from keyboard.
INPUT X# INPUT L, M, N INPUT "NEXT";N

INPUT # −*b* Inputs data from specified cassette unit.
INPUT # −1, A

INPUT #*b* Inputs data from buffer *b* (sequential access)
INPUT #1, A, B

KILL Deletes a disk file.
KILL "PRG/BAS" KILL "FILE1:1"

LET Assigns value to variable (optional).
LET X = 7.05 LET R2 = R1 LET C$ = "RED"

LINE INPUT Line inputs from keyboard; ENTER ends input.
LINE INPUT A$ LINE INPUT "ENTER YOUR NAME?"; N$

LINE INPUT # *b* Line inputs from disk into buffer *b*; carriage return, end of file, or 255th character ends input.
LINE INPUT #1, A$

LIST Lists program lines to the video display.
LIST LIST 50-85

LLIST Lists program lines to the line printer.
LLIST LLIST 50−

LOAD Loads program file from disk. R option causes program to run, leaving open files open.
LOAD "PRG/BAS" LOAD "PRG:2",R

LPRINT Prints an item or list of items on the printer.
LPRINT CAP$; "IS THE CAPITAL OF"; ST$

LPRINT TAB Moves printer carriage to specified position.
LPRINT TAB(25) "NAME"

LPRINT USING Prints formatted numbers and strings on the printer. See PRINT USING for list of field specifiers.
LPRINT USING "####,"; 1234

LSET Left-justifies data into a random access field.
LSET CITY$ "DULUTH"

MERGE Merges disk program in ASCII format with resident program
MERGE "PR/BAS"

MID$ (*old, pos, len*) = *repl* Replaces one portion of a string with another. If length option is omitted, same number of characters in the old string will be changed as the number of characters in the replacement string
MID$ (A$, 3, 4) "USAFX" MID$ (A$, 5) = "01"

NAME *new, start, increment* Renumbers resident program and all internal references (BASICR only).
NAME NAME 6000, 5000, 100 NAME,,5

NEW Erases program from memory; initializes all variables.
NEW

ON ERROR GOTO Sets up an error-handling routine.
ON ERROR GOTO 2100

ON ERROR GOTO 0 Disables an error-handling routine.
ON ERROR GOTO 0

ON . . . GOSUB Multi-way branch to specified subroutines.
ON Y GOSUB 50, 100, 150, 200

ON . . . GOTO Multi-way branch to specified lines.
ON X GOTO 190, 200, 210

OPEN *mode, b, file* Opens file; assigns mode (I = input, O = output, R = random); assigns buffer number *b*.
OPEN "O", 1 "CLIENTS/TXT"

OUT *p, v* Sends value to specified port. *p* and *v* = 0−255.
OUT 255, 0

POKE *I, v* Puts value *v* (0−255) into location *I* (15360 to end of memory).
POKE 21547, 81

PRINT Prints an item or list of items on the display at current cursor position.
PRINT X! + Y! PRINT "U.S.A."

PRINT @ *n* Prints beginning at *n*, *n* = 0−1023.
PRINT @ 477, "CENTER"

PRINT # −*b* Writes data to specified cassette deck.
PRINT # −1, A

PRINT #*b* Writes data to file-buffer *b* (sequential access).
PRINT #1, A

PRINT TAB Moves cursor right to specified tab position.
PRINT TAB(20) "NAME"

PRINT USING Formats strings and numbers:
Formats numbers.
PRINT USING "#####"; 66.2
. Decimal point.
PRINT USING "##.#"; 58.76
, Displays comma to left of every third digit.
PRINT USING "###,"; 1234
****** Fills leading spaces with asterisks.
*PRINT USING "**####"; 44.0*
$$ Floating dollar sign.
PRINT USING "$$##.#"; 118.6735
****$** Floating dollar sign; fills leading spaces with asterisks.
*PRINT USING "**$#.##"; 8.333*
↑↑↑↑ Exponential format.
PRINT USING "###.#↑↑↑↑"; 8527100
+ In first position, causes sign to be printed; in last position, causes sign to be printed after the number.
PRINT USING "+###"; −216 "###+"; 216
− Minus sign after negative numbers, space after positive.
PRINT USING "####.#−"; −8124.420
! Returns first string character.
PRINT USING "!"; "YELLOW"
%spaces% String field; length of field is number of spaces + 2.
PRINT USING "% %"; "BLUE"

PUT *b, record number* Moves data from file-buffer *b* into the specified record (random access). If *record number* is omitted, current record number is used.
PUT 1, 25 PUT 1 PUT C, N

RANDOM Reseeds random number generator.
RANDOM

READ Reads value(s) from a DATA statement.
READ T READ S$ READ NM$, AGE

REM Remark; instructs computer to ignore rest of line. ' is an abbreviation for REM.
REM PLACE COMMENTS HERE ' HERE TOO

RESET (*x, y*) Turns off graphics block at specified location. *x* (horizontal) = 0−127; *y* (vertical) = 0−47.
RESET (21, 40) RESET (L1, L2)

RESTORE Resets data pointer to first item in first data line.
RESTORE

RESUME Ends an error-handling routine by specifying where normal execution is to resume.
RESUME RESUME 40 RESUME NEXT

RETURN Returns from subroutine to next statement after GOSUB.
RETURN

RSET Right-justifies data into a random access field.
RSET CITY$ = "SPOKANE"

RUN Executes resident program or portion of it.
RUN RUN 150

RUN prog Loads and executes disk program. R option leaves open files open.
RUN "PROG/BAS" RUN "PRG:1", R

SAVE Saves BASIC program on disk. A option causes file to be stored in ASCII format.
SAVE "FL1/BAS:3" SAVE "PR/TXT", A

SET (*x, y*) Turns on graphics block at specified location. *x* (horizontal) = 0−127; *y* (vertical) = 0−47.
SET (10, 0) SET (L1, L2)

STOP Stops execution of a program.
STOP

SYSTEM Puts computer in monitor mode, allows loading of object files.
SYSTEM

TROFF Turns off the trace.
TROFF

TRON Turns on the trace.
TRON

BASIC Error Messages

Code	Abbreviation	Explanation
1	NF	NEXT without FOR
2	SN	Syntax error
3	RG	RETURN without GOSUB
4	OD	Out of data
5	FC	Illegal function call
6	OV	Overflow
7	OM	Out of memory
8	UL	Undefined line
9	BS	Subscript out of range
10	DD	Redimensioned array
11	/0	Division by zero
12	ID	Illegal direct
13	TM	Type mismatch
14	OS	Out of string space
15	LS	String too long
16	ST	String formula too complex
17	CN	Can't continue
18	NR	No RESUME
19	RW	RESUME without error
20	UE	Undefined error
21	MO	Missing operand
22	FD	Bad file data
23	L3	Disk BASIC feature
50		Field overflow
51		Internal error
52		Bad file number
53		File not found
54		Bad file mode
57		Disk I/O error
61		Disk full
62		Input past end
63		Bad record number
64		Bad file name
66		Direct statement in file
67		Too many files
68		Disk write-protected
69		File access denied

TRS-80™ MODEL I MICRO-COMPUTER SYSTEM

Start-Up

Non-Disk Users

1. Turn on the expansion interface, if you have one, and then all other peripherals.
2. Turn on the Computer. If you have an expansion interface, press BREAK as you turn the Computer on. The question
   MEMORY SIZE?
   will appear on the display. Answer it with a number or press ENTER to enter BASIC.

Disk Users

1. Make sure the disk drives are empty.
2. Turn on the expansion interface, and then all other peripherals.
3. Turn on the Computer. Insert a system diskette into drive 0, close the drive door, and press the reset button. System will initialize; TRSDOS will load and start up.
4. Type
   BASIC ENTER
   to start BASIC. When BASIC asks
   HOW MANY FILES?
   type in the number of concurrent files you will want, or press ENTER for three concurrent files. Then answer the MEMORY SIZE? question with a number of press ENTER to enter BASIC.

Note: Shaded areas pertain to disk-based systems only.

TRS-80™ MODEL I LEVEL II/DISK SYSTEM

Radio Shack
The biggest name in little computers

© Copyright 1980 by Radio Shack. A Division of Tandy Corporation

TRS-80™ MODEL I LEVEL II/DISK SYSTEM

TRSDOS
Commands and Utilities

APPEND Adds one disk file onto the end of another.
APPEND FTW/TXT TO NORTX/TXT

ATTRIB Changes protection of specified file. (I, ACC, UPD, PROT)
ATTRIB OLD/DAT (ACC = JUL14, UPD = MOUSE, PROT = READ)

AUTO *command* Automatically executes the specified TRSDOS command each time TRSDOS starts up. (AUTO by itself erases the automatic command.)
AUTO VERIFY AUTO BASIC AUTO

BACKUP Calls utility program to copy all data from one diskette to another.
BACKUP BACKUP :0 TO :0

BASIC Loads Disk BASIC interpreter. An * resets BASIC and retains the program that was in memory before the return to TRSDOS.
*BASIC BASIC **

BASICR Loads BASIC renumber interpreter, for use with the BASIC statement NAME. An * resets BASICR and retains the program that was in memory before the return to TRSDOS.
*BASICR BASICR**

BASIC2 Transfers control to Level II BASIC.
BASIC2

CLOCK Switches on real-time clock display.
CLOCK (ON) CLOCK CLOCK (OFF)

COPY *oldfile* TO *newfile* Copies a file.
COPY FILE1/BAS TO UPDFL/BAS
COPY FILE/A TO FILE/A:1

DATE *newdate* Sets current date.
DATE 07/18/79

DEBUG Starts debugger utility.
DEBUG (ON) DEBUG DEBUG (OFF)

DEVICE Lists the three system I/O devices: KI = keyboard, DO = video display, PR = line printer.
DEVICE

DIR :*d* Lists the diskette directory on drive *d*. (S, I, A)
DIR DIR :3 DIR :1 (A) DIR (S, I, A)

DUMP Dumps contents of RAM into a machine-language program disk file. (START = *aaaa*, END = *bbbb*, TRA = *cccc*)
DUMP DATA/CIM: 1 (START=X'9000', END=X'9050')

FORMAT Calls a utility program which organizes a diskette into tracks and sectors.
FORMAT

FREE Displays number of unused granules for each disk.
FREE

KILL Deletes file from directory; frees space allocated to that file.
KILL FL/BAS: 2

LIB Lists all TRSDOS library commands.
LIB

LIST Lists contents of a file to the display.
LIST PROG1/TXT

LOAD Loads machine-language file into memory.
LOAD GRAPHICS

PRINT Prints contents of a file on the printer.
PRINT PG7/TXT

PROT Changes file and diskette passwords. (PW, LOCK, UNLOCK)
PROT :1 (UNLOCK) PROT (PW, LOCK)

RENAME Renames a file.
RENAME MRS/BAS TO MS/BAS

TAPEDISK Copies a tape file to a disk file.
TAPEDISK

TIME *newtime* Sets the time.
TIME 13:20:00

TRACE Prints contents of PC (program counter) register on the display and updates it every 8 ms.
TRACE (ON) TRACE TRACE (OFF)

VERIFY Checks disk sector integrity after write operation.
VERIFY (ON) VERIFY VERIFY (OFF)

BASIC Functions

Argument ranges are indicated below by special symbols:
x: (−1 × 10↑38, −1 × 10↑−38), (1 × 10↑−38, 1 × 10↑38)
c: (0,255)
n: (−32768, 32767)
b: (0, 15)
str: string argument
var: variable name

ABS(*x*) Computes absolute value.
Y = ABS(X)

ASC(*str*) Returns ASCII code of first character of string.
A = ASC(T$)

ATN(*x*) Computes arctangent; value returned in radians.
Y = ATN(X/3)

CDBL(*x*) Converts to double-precision.
*X# = CDBL(N*3)*

CHR$(*c*) Returns character for ASCII, control, or graphics code.
P$ = CHR$(T)

CINT(*n*) Returns largest integer not greater than *n*.
PRINT CINT (15.0075)

COS(*x*) Computes cosine; angle must be in radians.
Y = COS(X)

CSNG(*x*) Converts to single-precision.
FC = CSNG(TM#)

CVD(*str*) Converts to double-precision after GET.
A# CVD(GRSPAY$)

CVI(*str*) Converts to integer after GET.
PRINT CVI(I1$)

CVS(*str*) Converts to single-precision after GET.
FK CVS(T$)

EOF(*b*) End-of-file detector for buffer *b*.
IF EOF(3) THEN CLOSE 3

ERL Returns the line number in which an error has occurred.
PRINT ERL

ERR If an error occurs, returns a value related to the error code: value returned = (error code − 1) * 2.
IF ERR = 12 THEN 650 ELSE 800

EXP(*x*) Computes natural antilog.
Y = EXP(X)

FIX(*x*) Truncates all digits to right of decimal point.
Y = FIX(X)

FRE(*numeric*) Finds amount of free memory.
F=FRE(X) PRINT FRE(10)

FRE(*str*) Returns amount of unused string space. *str* is any string constant or string variable.
FRE("C") FRE(C$)

INKEY$ Gets keyboard character if available.
A$ = INKEY$

INP(*p*) Gets value from specified port. *p* = 0−255.
V = INP(255)

INSTR(*pos, mainstr, substr*) Returns number which indicates the position in the main string where the substring begins. If substring not in main string, 0 is returned. If *pos* is omitted, *pos* = 1.
PRINT INSTR(S$, "VA") X = INSTR(S$, Q$)
Y1 = INSTR(8, S$, Q)

INT(*x*) Returns largest whole number not greater than *x*.
Y = INT(X)

LEFT$(*str, c*) Returns left portion of string.
P$=LEFT$(M$, 7)

LEN(*str*) Returns the number of characters in a string.
X = LEN(SEN$)

LOF(*n*) Determines number of last (highest-numbered) record in specified file.
Y = LOF(5)

LOG(*x*) Computes natural logarithm.
Y = LOG(X)

MEM Finds amount of free memory.
PRINT MEM

MID$(*string, pos, len*) Returns a substring of another string. If length option is omitted, the entire string right of *pos* is returned.
PRINT MID$(A$, 3, 2) F$=MID$(A$, 3)

MKD$(*x*) Makes double-precision number ready for disk write (random access).
LSET AVG$ = MKD$(3000.00001) LSET D$ MKD$(X#)

MKI$(*n*) Makes integer number ready for disk write (random access).
LSET AVG$ = MKI$(3000) LSET Y$ MKI$(Y%)

MKS$(*x*) Makes single-precision number ready for disk write (random access).
LSET AVG$ = MKS$(3000.1) LSET M$ = MKI$(N)

PEEK(*I*) Gets value in location *I* (*I* = 0 to end of memory).
V=PEEK(18520)

POINT(*x, y*) Tests whether specified graphics block is on or off. *x* (horizontal) = 0−127; *y* (vertical) = 0−47.
IF POINT (13, 35) THEN PRINT "ON" ELSE PRINT "OFF"

POS(*x*) Returns column position of cursor (0−63). *x* is a dummy argument.
PRINT TAB(40) POS(0)

RIGHT$ (*str, c*) Returns right portion of string.
ZIP$ = RIGHT$ (AD$, 5)

RND(*n*) Generates a pseudorandom number between 1 and *n* if *n* > 1, or between 0 and 1 if *n* = 0.
Y = RND(100) PRINT RND(0) R = RND(X)

SGN(*x*) Returns sign component: −1, 0, 1, if *x* is negative, zero, positive.
*X = SGN(A * B)*

SIN(*x*) Computes sine; angle must be in radians.
Y = SIN(X)

SQR(*x*) Computes square root.
Y = SQR(A + B)

STR$(*x*) Converts a numeric expression to a string.
S$ = STR$(X)

STRING$(*I, c*) Returns string of characters of length *I*. Character *c* can be specified as an ASCII code or as a string.
B$ = STRING$ (125, "?") B$ = STRING$ (125, 63)

TAN(*x*) Computes tangent; angle must be in radians.
Y = TAN(X)

TIME$ Returns the time (in 24-hour format) and the date as a 17-character string.
A$ = TIME$

USR(*x*) Calls a machine-language subroutine.
PRINT USR(−1) X = USR(Y)

USR*n*(*x*) Calls any one of up to 10 machine-language subroutines, *n* = 0−9. If *n* is omitted, 0 is assumed.
X = USR0(T1) F = USR7(Y)

VAL(*str*) Evaluates a string as a number.
V% = VAL("100 DOLLARS")

VARPTR(*var*) Gets address where variable contents are stored.
Y = USR1 (VARPTR (X))

Control Keys

←	Cancels last character typed; moves cursor back one space.
SHIFT ←	Erases current line.
BREAK	Interrupts anything in progress and returns to command level.
CLEAR	Clears the screen.
ENTER	Signifies end of current line.
SPACE BAR	Enters a space (blank) character and moves cursor one space forward.
→	Advances cursor to next tab position.
SHIFT →	Puts display in 32-character mode.
↓	Line feed and carriage return.
SHIFT @	Causes currently-executing program to pause (press any key to continue).

BASIC Operators

Each operator or group of operators is precedent over the group below it.

↑	Exponentiation (returns single-precision)
−, +	Unary negative, positive
*, /	Multiplication, division
+, −	Addition and concatenation, subtraction
<, >, =, <=, >=, <>	Relational tests
NOT	
AND	
OR	

BASIC Special Characters

%	Makes variable integer-precision.
!	Makes variable single-precision.
#	Makes variable double-precision.
$	Makes variable string type.
&H	Indicates number following is hexadecimal constant.
&O	Indicates number following is octal constant.
:	Separates statements on the same line.
?	Same as PRINT (but L? can't be substituted for LPRINT).

BASIC Edit Commands

A	Cancels changes and starts over.
*n*C	Changes *n* characters.
*n*D	Deletes *n* characters.
E	Ends editing and saves all changes.
H	Hacks line and inserts at end.
I	Inserts characters.
*n*K*c*	Kills all characters up to *n*th occurrence of *c*.
L	Lists the line.
Q	Quits edit mode and cancels all changes.
*n*S*c*	Searches for *n*th occurrence of *c*.
X	Extends line (inserts at end).
SHIFT ↑	Causes escape from command.
ENTER	Records all changes and exits edit mode.
n SPACE BAR	Moves cursor *n* spaces to the right.
n ←	Moves cursor *n* spaces to the left.

Video Control Codes

Dec	Hex	PRINT CHR$ (*code*)
8	08	Backspaces and erases current character.
10	0A	Line feed with carriage return.
13	0D	Line feed with carriage return.
14	0E	Turns on cursor.
15	0F	Turns off cursor.
23	17	Shifts to 32-character mode.
24	18	Backspaces cursor.
25	19	Advances cursor.
26	1A	Downward line feed.
27	1B	Upward line feed.
28	1C	Homes cursor.
29	1D	Moves cursor to beginning of line.
30	1E	Erases to end of line.
31	1F	Clears to end of screen.