TRS-80® MODEL III 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? Compares program on tape byte-for-byte with resident program.
```
CLOAD?      CLOAD? "MIXIT"
```

CLS Clears the display.
```
CLS
```

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"
```

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
```

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
```

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)
```

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

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 # – 1 Inputs data from cassette.
```
INPUT #-1, A
```

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

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

LLIST Lists program lines to the line printer.
```
LLIST       LLIST 50-
```

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
```

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
```

OUT*p, v* Sends value to specified port; *p* and *v* = 0–255
```
OUT 255, 0
```

POKE *n, v* Puts value *v* (0–255) into location *n* (15360 to end of memory). See **POKE Addresses**.
```
POKE 15872, 255
```

PRINT Prints an item or list of items on the display at current cursor position.
```
PRINT X↑2 + Y↑2       PRINT"U.S.A."
```

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

PRINT# – 1 Writes data to cassette.
```
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 "#####"; 58.2
```
. Decimal point.
```
PRINT USING "##.#"; 158.76
```
, 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. Press ↑ to generate this character.
```
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
```
– 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 plus 2.
```
PRINT USING "% %"; "BLUE"
```

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
```

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

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. In response to "*?", type *filename* or */address*.
```
SYSTEM
```

TROFF Turns off the trace.
```
TROFF
```

TRON Turns on the trace.
```
TRON
```

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
21	15	Switches special/compression characters
22	16	Switches alternate characters
23	17	Shifts to 32-character mode
24	18	Backspaces cursor without erasing
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

Operators

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

↑ or [	Exponentiation (returns single-precision). Press ↑ to generate this operator; *it will be displayed as a left bracket "["*
–, +	Unary negative, positive
*, /	Multiplication, division
+, –	Addition and concatenation, subtraction
<, >, =, <=, >=, <>	Relational tests
NOT	
AND	
OR	

Special Characters

'	Abbreviation for REM
%	Makes variable integer-precision
!	Makes variable single-precision
#	Makes variable double-precision
$	Makes variable string type
:	Separates statements on the same line
?	Same as PRINT (but L? can't be substituted for LPRINT)
,	PRINT punctuation: spaces over to the next 16-column PRINT zone
;	PRINT punctuation: separates items in a PRINT list but does not add spaces when they are output

Edit Commands

A	Cancels changes and starts again
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 insert subcommand
ENTER	Records all changes and exits edit mode
n SPACEBAR	Moves cursor *n* spaces to the right
n ←	Moves cursor *n* spaces to the left

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
SPACEBAR	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 ↓	"Control" key—hold down these two and press any key A-Z for control A - control Z
SHIFT ↓ *	Copies the display contents to the printer
SHIFT @	Causes currently executing program to pause (press any key to continue)

Functions

Argument ranges are indicated below by special letters:
x: (-1.70141E 38 - -2.93874E -39, 2.93874E -39 - 1.70141E 38) × 10^(±38)
c: (0,255)
n: (-32768, 32767)
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.
`W# = CDBL(W+3)`

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

CINT(x) Returns largest integer not greater than x.
`PRINT CINT (15,#875)`

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

CSNG(x) Converts to single-precision.
`PC = CSNG(T#+W)`

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 850 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(Z#)`

FRE(str) Returns amount of unused string space; x$ 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)`

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

LEFT$(str, c) Returns left portion of string.
`P$=LEFT$(M$, 3)`

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

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$, 3, 2)` `P$=MID$(A$, 3)`

PEEK(n) Gets value in location n (n = 0 to end of memory).
`W=PEEK(16520)`

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$(str, c) Returns right portion of string.
`ZI$ = RIGHT$(A0$, 5)`

RND(n) Generates a "random" 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.
`S = 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, 45)`
`B$ = STRING$(125, B$)`

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 whose address is stored at 16526 - 16527.
`PRINT USR(-1)` `X = USR(Y)`

VAL(str) Evaluates a string as a number.
`VZ = VAL("100 DOLLARS")`

VARPTR(var) Gets address where variable contents are stored.
`Y = USR (VARPTR (V))`

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

POKE Addresses

By POKEing various values into the addresses listed below, you can activate or control many of the Model III's special features. See the Owner's Manual for details.

Sample Use

To select the High cassette rate, execute:
`POKE 16913, 1`

Address Dec	Address Hex	Contents	Initial Contents
16409	4019	Caps Lock Switch 0 = "Upper and Lower Case" Not 0 = "Caps Only"	'Caps'
16412	401C	Cursor Blink Switch 0 = "Blink" Non-Zero = "No-Blink"	0
16416	4020	Cursor Address Two bytes: LSB, MSB	N/A
16419	4023	Cursor Character ASCII Code 0 - 255	176
16424	4028	Maximum Lines/Page plus one	67
16425	4029	Number of lines printed plus one	1
16427	402B	Line Printer Max. Line length less two 255 = "No Maximum"	255
16526	408E	Address of USR Routine Two Bytes: LSB, MSB	7754
16872	41E8	$RSRCV Input Buffer One byte	0
16880	41F0	$RSTX Output Buffer One byte	0
16888	41F8	$RSINIT Baud Rate Code TX Code = Most Sig. Nibble RCV Code = Least Sig. Nibble	85
16889	41F9	$RSINIT Parity/Word Length/ Stop-Bit Code	108
16890	41FA	$RSINIT Wait Switch 0 = "Don't Wait" Non-Zero = "Wait"	'Wait'
16913	4211	Cassette Baud Rate Switch 0 = 500 Baud Non-Zero = 1500 Baud	N/A
16916	4214	Video Display Scroll Protect From 0 to 7. Greater values are interpreted in modulo 8	0
16919	4217	Time-Date Six binary bytes: SS MM HH YY DD MM	0

Z-80 ROM Subroutines

The following ROM subroutines may be used by Z-80 programs; some may also be used by BASIC programs via the USR function. **Before trying to use any of these, read the Technical Information Section of your Owner's Manual.**

Address Dec	Address Hex	Contents	Function
0	0000	$RESET	System reset
43	002B	$KBCHAR	Check for keyboard character
51	0033	$VDCHAR	Display a character
59	003B	$PRCHAR	Print a character
64	0040	$KBLINE	Wait for a keyboard line
73	0049	$KBWAIT	Wait for a keyboard character
80	0050	$RSRCV	Receive character from RS-232-C
85	0055	$RSTX	Transmit character to RS-232-C
90	005A	$RSINIT	Initialize RS-232-C
96	0060	$DELAY	Delay for a specified time
105	0069	$INITIO	Initialize all I/O drivers
457	01C9	$VDCLS	Clear the screen
473	01D9	$PRSCN	Print screen contents
539	021B	$VDLINE	Display a line
565	0235	$CSIN	Input a cassette byte
612	0264	$CSOUT	Output a cassette byte
647	0287	$CSHWR	Write the cassette header
653	028D	$KBBRK	Check for BREAK key only
662	0296	$CSHIN	Read the cassette header
664	0298	$CLKON	Turn on the clock display
673	02A1	$CKLOFF	Turn off the clock display
6681	1A19	$READY	Jump to BASIC 'READY'
12339	3033	$DATE	Get the date
12342	3036	$TIME	Get the time
12354	3042	$SETCAS	Set cassette baud rate
14312	37E8	$PRSTAT	Printer status (Read Only) 'Go' only if: Bit 7 = 0 'NOT BUSY' Bit 6 = 0 'NOT OUT OF PAPER' Bit 5 = 1 'DEVICE SELECT' Bit 4 = 1 'NOT PRINTER FAULT' Bits 3,2,1 and 0 are not used.

TRS-80® MODEL III MICRO-COMPUTER SYSTEM

Start-Up

The entire system (Computer and peripherals) should be OFF.

1. Turn all peripherals ON.
2. Turn the Computer ON.
3. The message:
   `Cass?`
   should be displayed. To select the High cassette speed (1500 baud), press H or ENTER. To select the Low cassette speed (500 baud), press L.
   For general purposes, use High. To load or save Model I Level II BASIC programs, you must use Low.
4. The message:
   `Memory Size?`
   will be displayed. To use all available memory, press ENTER. To reserve some high memory, type in the highest address (in decimal) that you want to use, then press ENTER.
5. The start-up message, followed by the READY prompt, will be displayed. The computer is now ready to use.

TRS-80® MODEL III BASIC

Radio Shack
The biggest name in little computers™

TRSDOS Commands and Utilities

APPEND Adds one disk file onto the end of another
APPEND FTM/TXT NOFT[illegible]

ATTRIB Changes protection of specified file (I,V, ACC, UPD, PROT)
ATTRIB OLD/DAT [illegible]
PROT=READ)

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

BACKUP Duplicates a system or data diskette
BACKUP BACKUP :0 :1

BUILD Creates an automatic command input file
BUILD JOBFILE

BASIC Loads Disk BASIC interpreter. BASIC * allows recovery of the program that was in memory before the return to TRSDOS
BASIC BASIC *

CLEAR Clears user memory and sets top memory address
CLEAR (START=[illegible],END=[illegible],MEM=[illegible])
CLEAR

CLOCK Turns real-time clock display on/off
CLOCK (ON) CLOCK CLOCK (OFF)

CLS Clears the screen
CLS

CONVERT Model I to Model III program/data file conversion
CONVERT

COPY *oldfile newfile* Copies a file
COPY FILE1/BAS UPDFL/BAS
COPY FILE/A FILE/A:1 COPY [illegible]

CREATE *filename*(LRL = *aaa*, REC = *bbb*) Creates a preallocated file
CREATE JOBFILE (LRL=[illegible],REC=[illegible])

DATE *newdate* Sets or displays the current date
DATE [illegible] DATE

DEBUG Starts debug monitor
DEBUG (turns monitor ON) & (turns monitor OFF)

DIR :*d*(INV, SYS, PRT) Lists the diskette directory (INVisible or SYStem) on drive *d* on the Display or Printer (PRT)
DIR :1 (SYS) DIR :0 (PRT)

DO *command-line* Begin auto command input from disk file
DO [illegible]

DUAL (*switch*) Duplicates output to video and printer
DUAL (ON) DUAL DUAL (OFF)

DUMP *file* Dumps content of RAM into a machine-language program disk file (START = *aaaa*,END = *bbbb*,TRA = *cccc*,RELO = *dddd*)
DUMP [illegible] (START=[illegible],END=[illegible])

ERROR *number* Displays an error message
ERROR [illegible]

FORMS (WIDTH = *aaa*,LINES = *bbb*) Set printer parameters
FORMS (WIDTH=[illegible],LINES=[illegible])

FORMAT Initializes a diskette into tracks and sectors
FORMAT :1 FORMAT

FREE Lists a diskette's allocation map to the Display or Printer (PRT)
FREE :1 FREE :0 (PRT)

HELP *command* Explanation of TRSDOS command
HELP BACKUP

KILL *file*/EXT:*d* Deletes a file from directory, frees space allocated to that file
KILL [illegible] KILL [illegible]

LIB Lists library commands
LIB

LIST *file* (PRT, SLOW, ASCII) Lists contents of a file to the Display or Printer
LIST [illegible] LIST JOBFILE/BLD (ASCII)

LOAD *file* Loads a machine-language file into memory
LOAD [illegible]

LPC Special printer driver for some printers
LPC

MASTER (DRIVE = *a*) Forces a drive to be the Master Read/Write drive. MASTER releases any drive defined as Master Drive
MASTER (DRIVE=1) MASTER

MEMTEST Test memory (ROM and RAM)
MEMTEST

PATCH *file* (ADD = *aaaa*,FIND = *bb*,CHG = *cc*) Change the contents of a disk file
PATCH JOBFILE/BLD (ADD=[illegible],FIND=[illegible],CHG=[illegible])

PAUSE *message* Pauses for operator action or message
PAUSE INSERT DISKETTE #2

PROT :*d* (PW, LOCK) Changes file and diskette passwords
PROT :1 (PW,LOCK)

PURGE :*d* (*file-type*) Deletes files (SYS, DATA, ALL, INV)
PURGE :1 (INV) PURGE :0

RELO *file* (ADD = *aaaa*) Changes location where program loads into memory
RELO JOBFILE/BLD (ADD=[illegible])

RENAME *file* TO *file* Renames a file
RENAME [illegible]/BAS TO [illegible]/BAS

ROUTE (SOURCE = *aa*,DESTIN = *bb*) Routes I/O devices
ROUTE (SOURCE=[illegible],DEST[illegible])

SETCOM (OFF,WORD = *a*,BAUD = *bbb*,STOP = *c*, PARITY = *d*, *mode*) Sets up RS-232C communications or display status
SETCOM (WORD=7,BAUD=300,STOP=1, PARITY=[illegible]) SETCOM

TAPE (S = *a*,D = *b*) Executes tape transfer operation
TAPE (S=D,D=T)

TIME *hh:mm:ss* Resets or gets the time
TIME [illegible] TIME

WP (DRIVE = *a*) Write-protects a diskette
WP (DRIVE=1) WP

XFERSYS Transfers system files
XFERSYS

TRSDOS Error Messages

0	No Error Found
1	CRC Error During Disk I/O
2	Disk Drive Not In System
3	Lost Data During Disk I/O
4	CRC Error During Disk I/O
5	Disk Sector Not Found
6	Disk Drive Hardware Fault
7	"Undefined Error Code"
8	Disk Drive Not Ready
9	Illegal I/O Attempt
10	Required Command Parameter Not Found
11	Illegal Command Parameter
12	Time Out On Disk Drive
13	I/O Attempt To Non System Disk
14	Write Fault On Disk I/O
15	Write Protected Disk
16	Illegal Logical File Number
17	Directory Read Error
18	Directory Write Error
19	Invalid File Name
20	GAT Read Error
21	GAT Write Error
22	HIT Read Error
23	HIT Write Error
24	File Not Found
25	File Access Denied Due To Password Protection
26	Directory Space Full
27	Disk Space Full
28	Attempt To Read Past EOF
29	Attempt to Read Outside of File Limits
30	No More Extents Available
31	Program Not Found
32	Invalid Drive Number
34	Attempt To Use Non Program File As A Program
35	Memory Fault During Program Load
36	"Undefined Error Code"
37	File Access Denied Due To Password Protection
38	I/O Attempt To Unopen File
39	Invalid Command Parameter
40	File Already In Directory
41	Attempt To Open File Already Open

Disk BASIC Functions

CVD(*str*) Converts to double-precision after GET
[illegible]

CVI(*str*) Converts to integer after GET
PRINT CVI([illegible])

CVS(*str*) Converts to single-precision after GET
[illegible]=CVS([illegible])

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

INSTR(*pos, mainstr, substr*) Returns number which indicates the position of the main string where the substring begins. If substring not in main string, zero is returned. If *pos* is omitted, *pos* = 1
PRINT INSTR([illegible]) [illegible]=INSTR([illegible])
[illegible]=INSTR([illegible])

LOC(*n*) Gets current record number
PRINT LOC([illegible])

LOF(*n*) Determines number of last (highest-numbered) record in specified file
[illegible]=LOF([illegible])

MKD$(*x*) Makes double-precision number ready for disk write (random access).
LSET AVG$=MKD$([illegible])

MKI$(*n*) Makes integer number ready for disk write (random access).
LSET AVG$=MKI$([illegible]) LSET [illegible]=MKI$([illegible])

MKS$(*x*) Makes single-precision number ready for disk write (random access).
LSET AVG$=MKS$([illegible]) LSET [illegible]=MKS$([illegible])

USR*n*(*x*) Calls any one of up to 10 machine-language subroutines, *n* = 0-9. If *n* is omitted, zero is used. See DEFUSR*n*.
[illegible]=USR0([illegible]) [illegible]=USR5([illegible])

PRINTED IN U.S.A.
[illegible] 381-NP

Disk BASIC Statements

CLOSE Closes all open file-buffers or specified buffer(s).
```
CLOSE     CLOSE 1,2,6     CLOSE N
```

CMD "A" Returns to TRSDOS on error.
```
CMD "A"
```

CMD "B" Enable/Disable (BREAK) key.
```
CMD "B","ON"     CMD "B","OFF"
```

CMD "C" Deletes program remarks (R) or spaces (S).
```
CMD "C",R     CMD "C",S     CMD "C"
```

CMD "D" Displays directory for specified drive.
```
CMD "D:1"
```

CMD "E" Displays previous TRSDOS error.
```
CMD "E"
```

CMD "I", *command* Executes a command to TRSDOS; may overwrite BASIC.
```
CMD "I","HELP"
```

CMD "J" Changes calendar date from *source* to *destination*. mm/dd/yy can be changed to ddd.yy; yy.ddd can be changed to mm/dd/yy.
```
CMD"J","08/12/81",D$
CMD"J","-64/281",D$
```

CMD "L", *routine* Loads Z-80 routine or program file into RAM.
```
CMD "L",JOBFILE
```

CMD "O", *s, array (start)* Alphabetizes (sorts) contents of an array. *s* is the number of items to be sorted; *start* is where the sorting process begins.
```
CMD"O",50,A(1)
```

CMD "P", *status* Checks printer status. *status* is a string variable.
```
CMD "P",A$
```

CMD "R" Turns real-time clock display ON.
```
CMD "R"
```

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

CMD "T" Turns real-time clock display OFF.
```
CMD "T"
```

CMD "X", *target* Cross-references program lines and line numbers. *target* can be a reserved word, string, or string variable.
```
CMD "X",GOTO     CMD "X","PRINT"
```

CMD "Z" Simultaneous output to Printer and Display (dual routing).
```
CMD "Z","ON"     CMD"Z","OFF"
```

DEFFN Defines a user-created function.
```
DEF FNA$(X)=STRING$(X,45)
```

DEFUSR*n* Defines entry point for machine-language subroutine called by USR*n*. If *n* is omitted, zero is used.
```
DEFUSR=&H5500     DEFUSR1=&H7D7E
```

FIELD Organizes a random file buffer into fields.
```
FIELD 3,10 AS NM$,25 AS AD$
```

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
```

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

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

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

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

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

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

MERGE Merges disk program with resident program. Disk program must be in ASCII format.
```
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)="USAF"     MID$(A$,5)="!!"
```

NAME *newline, startline, increment* Renumbers program line numbers. *newline* is the new number of the first line which is to be renumbered. If omitted, 10 is used. *startline* is the line number where renumbering is to begin. If omitted, entire program will be renumbered. *increment* is the increment between successive renumbered lines. If omitted, 10 is used.
```
NAME 100,10,100     NAME     NAME ,5
```

OPEN *mode, b, file, n* Opens file, assigns mode (I = input, O = output, R = random, E = Output to end-of-file); assigns buffer number *b*, *file* specifies filename; *n* specific number of files.
```
OPEN"R",3,"CLIENTS/TXT"
```

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

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 2,N
```

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

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

SAVE *filename* Saves BASIC program on disk. A option causes file to be stored in ASCII format.
```
SAVE "PLI/BAS:3"     SAVE "PRT/TXT",A
```

Disk BASIC Debug Monitor Commands

D Displays memory contents.
D ADDRESS? = *aaaa* where *aaaa* is a hexadecimal number.

X Half-screen display.

S Full-screen display.

M Modify RAM. M ADDRESS? = *aaaa* where *aaaa* is a hexadecimal number.

R Change Register contents.
R*aa*,*bbbb* (SPACEBAR) where *aa* is one of the register pairs AF, BC, DE, HL, PC and *bbbb* is a hexadecimal value.

I Single-step.

C Single-step executing call.

U Up-dates display.

: Increment the first location on a half-screen display by 16; on full-screen, by 256.

- Decrement the first location on a half-screen display by 16; on full-screen, by 256.

J Jump the transfer of control from one location to another. J ADDRESS? = *aaaa*,*bbbb* where *aaaa* specifies the hexadecimal location where execution begins and *bbbb* specifies the hexadecimal location of the breakpoint.

Q Quits or exits from debug.

F Disk file utility which allows you to load disk file into memory and change it.

Disk BASIC Error Codes

51	Field Overflow
52	Internal Error
53	Bad File Number
54	File Not Found
55	Bad File Mode
58	Disk I/O Error
62	Disk Full
63	Input Past End
64	Bad Record Number
65	Bad File Name
67	Direct Statement in File
68	Too Many Files
69	Disk Write-Protect
70	File Access

Disk BASIC Abbreviations & Special Characters

&H	Indicates following number is a hexadecimal constant.
&O	Indicates following number is an octal constant.
↑	Lists previous line.
↓	Lists next line.
→	Lists current line.
←	Edit current line.
SHIFT ↑	Lists first line.
SHIFT ↓	Lists last line.
Lxx	List line xx.
Exx	Edit line xx.
Dxx	Delete line xx.
Axxx, xxx	Automatic line numbering beginning at line xxx, incrementing by xxx.

TRS-80® MODEL III MICRO-COMPUTER SYSTEM

Start-Up

The entire system (Computer and peripherals) should be OFF and the disk drives empty.

1. Turn all peripherals ON.
2. Turn the Computer ON.
3. Insert a System diskette into Drive 0. Close the drive door.
4. Press the RESET button. Once the system is initialized, TRSDOS will load and take control.
5. To start Disk BASIC, type BASIC (ENTER).
6. When BASIC asks HOW MANY FILES? type in the number of concurrent files you need or press (ENTER) (three concurrent files).
7. Then BASIC will ask MEMORY SIZE? Answer by typing in a specific number or press (ENTER) to enter Disk BASIC.
8. The Disk BASIC start-up message will appear followed by the READY prompt. The Computer is now ready for use.

Radio Shack®
The biggest name in little computers™

TRS-80® MODEL III DISK SYSTEM