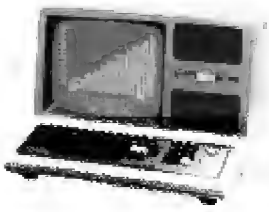
TRS-80® MODEL III MICRO-COMPUTER SYSTEM



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.

Radio Shack®

The biggest name in little computers™

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! + Y1          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 "#####"; 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. 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
```

BASIC

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 LPRINT 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.
nC	Changes n characters.
nD	Deletes n characters.
E	Ends editing and saves all changes.
H	Hacks line and inserts at end.
I	Inserts characters.
nKc	Kills all characters up to nth occurrence of c.
L	Lists the line.
Q	Quits edit mode and cancels all changes.
nSc	Searches for nth 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 ↓ P	Copies the display contents to the printer.
SHIFT @	Causes currently executing program to pause (press any key to continue).

PRINTED IN U.S.A.
8759036-581

Functions

Argument ranges are indicated below by special letters:
x: $(-1\times10\ E\ 38,-1\times10\ E\ -38),(1\times10\ E\ -38,1\times10\ E\ 38)$
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.
```
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(TH#)
```

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)
```

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$)
```

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)
```

PEEK(*n*) Gets value in location n (n = 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 "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.

```
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$(*l, c*) Returns string of characters of length l. 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 whose address is stored at 16526–16527.

```
PRINT USR(-1)        X = USR(Y)
```

VAL(*str*) Evaluates a string as a number.

```
V% = VAL("100 DOLLARS")
```

VARPTR(*var*) Gets address where variable contents are stored.

```
Y = USR (VARPTR (X))
```

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			Initial
Dec	Hex	Contents	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
16928	4220	$ROUTE Destination Device Two-byte I/O designator	N/A
16930	4222	$ROUTE Source Device Two-byte I/O designator	N/A

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	**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
108	006C	$ROUTE	Route I/O
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.