RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

TRS-80®
MODEL 100
PORTABLE COMPUTER

Quick Reference Guide

CUSTOM MANUFACTURED FOR RADIO SHACK, A DIVISION OF TANDY CORPORATION

Table of Contents

TRS-80® Model 100 Quick Reference Guide

How to Use this Quick Reference Guide . . .

In this Quick Reference Guide, you'll find most everything you'll need to use in your Model 100. The first part of the book gives some general information about the Model 100, such as start-up procedures and power sources.

Following this you'll find a description of the purpose and commands of each Application Program included with the Model 100. Finally, in the back of the book you'll find several handy tables which give important Keyboard-to-ASCII Code correspondence and BASIC Error Messages.

Notations Used in this Quick Reference Guide

To describe a given command or function, we'll use an abbreviated notation as follows:

BOLDFACE CAPS	Type in the command exactly as written.
boldface italics	Insert a suitable argument.
(KEY)	Press the specified key.
(CTRL) (KEY)	Press (CTRL) and (KEY) simultaneously.
(CODE) (KEY)	Press (CODE) and (KEY) simultaneously.
(GRPH) (KEY)	Press (GRPH) and (KEY) simultaneously.
expression	Insert an argument, either numeric or string.
string expression	Insert a suitable string argument.
numeric expression	Insert a suitable numeric argument.
list	Insert one or more arguments, separated by commas.
range	Insert either a single argument, or else two arguments separated by a hyphen.

Program	(F1)	(F2)	(F3)	(F4)	(F5)	(F6)	(F7)	(F8)
BASIC	Files	Load	Save	Run	List	—	—	Menu
TEXT	Find	Load	Save	—	Copy	Cut	Sel	Menu
TELCOM	Find	Call	Stat	Term	Echo	Wait	—	Menu
ADDRSS	Find	—	—	—	Lfnd	—	—	Menu
SCHDL	Find	—	—	—	Lfnd	—	—	Menu

Function Key Definitions

Using the Model 100

Power Sources

The Model 100 can operate on 4 size AA Alkaline Manganese batteries. The Model 100 can also operate on ordinary household current (120VAC) by attaching an AC adapter (Catalog Number 26-3804).

Turning the Model 100 On

To power-up the Model 100, simply set the Power Switch (located on the right side of the Computer) to ON. On initial Power-Up, the Main Menu appears on the Display.

Setting the Date and Time

The current day, time, and date are listed on the top of the Menu. To change these, enter the BASIC Interpreter Program, and type (for example):

DAY$ = "Mon" (ENTER)
DATE$ = "03/18/83" (ENTER)
TIME$ = "13:45:25" (ENTER)

Selecting Menu Options

To access a program or file in the Model 100's memory (from the Menu level), use the arrow keys to position the Cursor on top of the appropriate file. If the file is a data file, the Model 100 enters the Text Editor. If the file is a BASIC program, the Model 100 enters BASIC and runs the program. If the file is a machine-language program (such as the BASIC Interpreter, BASIC, or the Text Editor, TEXT) the Model 100 runs the program.

Turning the Model 100 Off

To turn off the Model 100, simply set the Power Switch to OFF. RAM files currently in RAM are preserved for your access when you turn the Model 100 back on. (To insure that your files are preserved, turn the power off only from the Menu display or, if in an Application Program or file, only when the cursor is blinking.)

The Model 100 features a convenient Auto-Power Off function. The Computer will turn itself off automatically after 10 minutes of inactivity (no keyboard input or program running). To turn the Computer back on, move the ON/OFF switch to OFF, then back to ON.

Text Editor (TEXT) Quick Reference

Entering the Text Editor program

To use the Text Editor, you may either position the Menu Cursor on top of the word TEXT or on top of a text file itself (text files use the extension DO). Then press (ENTER).

Text Editor Commands

Within the Text Editor, the following keys or key combinations perform certain functions:

Cursor Control Keys

→	Moves the Cursor one position to the right.
←	Moves the Cursor one position to the left.
↑	Moves the Cursor one position up.
↓	Moves the Cursor one position down.
CTRL →	Moves the Cursor to the right end of the line.
CTRL ←	Moves the Cursor to the left end of the line.
CTRL ↑	Moves the Cursor to the top of the file.
CTRL ↓	Moves the Cursor to the bottom of the file.
SHIFT →	Moves the Cursor to the beginning of the word to immediate right.
SHIFT ←	Moves the Cursor to the beginning of the word to immediate left.
SHIFT ↑	Moves the Cursor to the top of the screen above the current position.
SHIFT ↓	Moves the Cursor to the bottom of the screen below the current position.

Text Manipulation Keys

DEL	Delete the character at the current Cursor position.
BKSP	Delete the character to the left of the current Cursor position.
F1 Find	Searches the file for the occurrence of a particular character string. Text prompts you for the match string.
F2 Load	Reads an ASCII data file into RAM. Text prompts you for the filename.
F3 Save	Writes a data file. Text prompts you for the filename.
F5 Copy	Duplicates the selected text into the paste buffer.
F6 Cut	Moves the selected text from the Screen into the paste buffer.
F7 Select	Starts definition of text for duplication or deletion.

(F8) Exit the Text Editor and return to the Menu.
Menu

(PASTE) Inserts the contents of the paste buffer, starting at the current cursor location.

(LABEL) Prints the definitions of the function keys on the bottom line of the display.

(PRINT) Prints the contents of the *Screen* onto the printer.

(SHIFT)(PRINT) Prints the contents of the *file* onto the printer. Text prompts you for the width of the printer.

(TAB) Inserts an eight-character wide tab.

(CTRL)(A) Moves the Cursor to the beginning of the word to the left from current position.

(CTRL)(B) Moves the Cursor directly to the bottom of the Screen from its current position.

(CTRL)(C) Cancels the Select, Save, Load, Find, and Print functions.

(CTRL)(D) Moves the Cursor one character to the right.

(CTRL)(E) Moves the Cursor up one line from its current place.

(CTRL)(F) Moves the Cursor to the beginning of the next word.

(CTRL)(G) Saves a data file to cassette tape.

(CTRL)(H) Deletes previous character.

(CTRL)(I) Inserts an eight-character wide tab.

(CTRL)(L) Enter Select Text mode.

(CTRL)(M) Inserts a carriage return and line feed.

(CTRL)(N) Find a Text String.

(CTRL)(O) Copy a Text String.

(CTRL)(P) Saves next keystroke as a non-printing control character (to store printer commands, etc.).

(CTRL)(Q) Moves the Cursor to the left end of the current line.

(CTRL)(R) Moves the Cursor to the right end of the current line.

(CTRL)(S) Moves the Cursor one character to the left.

(CTRL)(T) Moves the Cursor to the top of the Screen directly above its current position.

(CTRL)(U) Moves Selected string from Screen to paste buffer.

(CTRL)(V) Loads a data file from cassette tape.

(CTRL)(W) Moves the Cursor to the beginning of the file.

(CTRL)(X) Moves the Cursor down one line.

(CTRL)(Y) Prints the entire file.

(CTRL)(Z) Moves the Cursor to the end of the file.

Scheduler (SCHEDL) Quick Reference

Using the Scheduler Program

The Scheduler Program uses a data file called NOTE.DO. To insert data into the NOTE.DO file, position the Menu Cursor on top of the filename NOTE.DO (if NOTE.DO already exists) or else position the Menu Cursor on top of TEXT and press (ENTER). In either case, you'll have full use of the Text Editor.

To search for an item once you have created NOTE.DO, position the Menu Cursor on top of the word SCHEDL and press (ENTER).

Special Commands and Keys Within the Scheduler Program

(F1) **Find *string*** Finds *string* in the NOTE.DO file. If *string* occurs more than once in the file, the SCHEDL displays a screenful and prompts you for "More" or "Quit." Pressing (M) or (F3) retrieves the next screenful, pressing (Q) or (F4) ends the Find process.

(F5) **Lfnd *string*** Works exactly as Find ((F1)) with the exception that instead of displaying the results on the Screen, SCHEDL sends the data to the printer.

(F8) **Menu** Exits the SCHEDL Program and returns to the Menu.

Address Organizer (ADDRSS) Quick Reference

Using the Address Program

The Address Program uses a data file called ADRS.DO to store address data. To insert addresses into the ADRS.DO file, position the Menu Cursor on top of the filename ADRS.DO (if it already exists) or else position the Menu Cursor on top of TEXT and press (ENTER). In either case, you'll have full use of the Text Editor.

To search for an item once you have created ADRS.DO, position the Menu Cursor on top of the word ADDRSS and press (ENTER).

Special Commands and Keys in the ADDRSS Program

(F1) **Find *string*** — Finds the *string* in the ADRS.DO file. If *string* occurs more than once in the file, ADDRSS displays a screenful and prompts you for "More" or "Quit." Pressing (M) or (F3) retrieves the next screenful, pressing (Q) or (F4) ends the **Find** Process.

(F5) **Find *string*** — Works exactly as **Find** ((F1)) except that instead of displaying the results on the Screen, ADDRSS sends the data to the printer.

(F8) **Menu** — Exits ADDRSS and returns to the Menu.

Telecommunications (TELCOM) Quick Reference

Using the Telecommunications Program

To start the Telecommunications Program, position the Menu Cursor over the word TELCOM and press (ENTER).

Special Commands and Keys in TELCOM

When the Program displays the message TELCOM:, you may issue any of the following commands:

(F1) **Find *string*** — Finds a *string* from ADRS.DO. Within this mode:

- (F2) **CALL** — calls up the currently found number
- (F3) **MORE** — finds the next matching string
- (F4) **QUIT** — cancels the search.

(F2) **Call *number*** — Calls the phone *number*. If a *number* was just found with (F1), then TELCOM calls that number.

(F3) **Stat *config*** — Change communications configuration to the given *config*. If no configuration is given, then TELCOM displays the current configuration.

(F4) **Term** — Puts TELCOM into the Terminal Mode. Within Terminal Mode:

- (F1) Displays the previous page received.
- (F2) Transfer incoming data into a RAM file.
- (F3) Transmit a file to the host system.
- (F4) Toggles between full and half duplex.
- (F5) Echoes incoming data to the printer.
- (F8) Exits Terminal Mode and returns to TELCOM.

(F8) **Menu** — Exits TELCOM and returns to the Menu.

Communications Configuration

For RS-232C communications, the configuration description consists of a five character string of the format *rwpbs*, where:

r — **Baud Rate.** This is a number from 1 to 9, where 1 = 75; 2 = 110; 3 = 300; 4 = 600; 5 = 1200; 6 = 2400; 7 = 4800; 8 = 9600; 9 = 19200. **M** may be used. **M** sets built-in modem to 300 baud.

w **Word Length.** This is a number from 6 to 8, where 6 = 6 bits; 7 = 7 bits; 8 = 8 bits.

p **Parity.** Either E,O,N, or I where E = Even; O = Odd; N = None; I = Ignore.

b **Stop Bits.** Either 1 or 2, where 1 = 1 stop bit; 2 = 2 stop bits.

s **XON/XOFF Status.** Either E or D, where E = Enable; D = Disable.

Modem communication configuration consists of a five character string of the pattern *wpbs*, as defined for RS-232C communications. (TELCOM automatically sets the baud rate to 300 baud.)

Examples

88E1E 9600 baud, 8 bit words, even parity, 1 stop bit, XON/XOFF enable. (Uses RS-232C port.)

M7N2D 300 baud, 7 bit words, no parity check, 2 stop bits, XON/XOFF disabled. (Uses built-in modem.)

Auto Log-on Commands

You may store auto log-on information, along with phone numbers, in the ADRS.DO File. When you call the host (using Auto-dial), any characters enclosed within < and > are sent to the host as the Log-on Sequence. You may include the following abbreviations within the Log-on Sequence:

?c Wait for c to be sent from the host (c is any character)

= Pause for 2.0 seconds

!c Insure that c is interpreted as a character, not a command (c is any character).

^c Sends control character equivalent to (CTRL) c (c is any character from A-Z)

Example:

<=^C?U9857,756^M?PMICRO!?^M>

Pauses for 2.0 seconds, sends a (BREAK) (^C) the computer then waits for the host to transmit a "U". The computer then transmits 9857,756, followed by a carriage return (^M). It again waits for the host to transmit a "P", and then transmits MICRO? followed by carriage return. (the !? insures that TELCOM doesn't interpret the question mark as a "wait" command.)

Any phone number to be auto-dialed must be preceded by a colon. A second colon terminates the auto-sequence.

CIS: 555-1234:

The BASIC Interpreter (BASIC)

Starting the BASIC Interpreter Program

To use the BASIC Interpreter, you may either position the Menu Cursor on top of the word BASIC, or on top of a BASIC Program filename. Then press (ENTER).

Special Keys in the Command Mode of BASIC

(LABEL) Prints the definitions of the function keys

(PRINT) The equivalent of typing in "LCOPY" (ENTER)

(SHIFT)(PRINT) The equivalent of typing in "LLIST" (ENTER)

(F1) The equivalent of typing in Files (ENTER)

(F2) The equivalent of typing in Load "

(F3) The equivalent of typing in Save "

(F4) The equivalent of typing in Run (ENTER)

(F5) The equivalent of typing in List (ENTER)

(F8) The equivalent of typing in Menu (ENTER)

You may redefine any of the function keys within BASIC. See KEY under Keyboard Input for details.

Special Keys in the Execute Mode of BASIC

(BREAK) Stops execution of the current command. You can restart many commands where they left off by typing CONT (ENTER).

(PAUSE) Temporarily stops execution of the current command. To continue, simply press (PAUSE) again. This is particularly helpful when the Screen is changing rapidly, for example, on a LIST.

Numeric and String Operators

+ Addition or Unary Plus (Numeric) or Concatenation (String)

– Subtraction or Unary Minus

***** Multiplication

/ Division

**** Integer Division

^ Exponentiation

MOD Modulus Arithmetic

Relational Operators

<	Less than
>	Greater than
=	Equal
< = or = >	Less than or equal to
> = or = <	Greater than or equal to
<> or ><	Not equal to

Logical Operators

AND	Logical AND operation
OR	Logical OR operation
XOR	Exclusive OR operation
EQV	Equivalence operation
IMP	Implication operation
NOT	Logical NOT operation

Operator Hierarchy

Parentheses

^

+,– (unary plus and minus)

*,/

MOD

+,–

<,>,=,=,>,<=,><

NOT

AND

OR

XOR

EQV

IMP

(Note: Within an expression, operators on the same level are evaluated from left to right, with the exception of parentheses, which are evaluated from inside to outside.)

Data Ranges

Integers: –32768 to 32767

Single Precision: $\pm 10^{-64}$ to $\pm 10^{62}$ (6 Significant Digits)

Double Precision: $\pm 10^{-64}$ to $\pm 10^{62}$ (14 Significant Digits)

Strings: 0 to 255 characters.

(Note: Unless explicitly defined, the Model 100 considers all constants and variables, as well as numeric functions, as double precision.)

Declaration Tags

%	Integer
!	Single Precision
#	Double Precision
$	String

BASIC Keywords (except for Input/output)

ABS(*numeric expression*) Returns the absolute value of *numeric expression.*
A! = ABS(BAL) B = ABS(–100)

ASC(*string expression*) Returns the ASCII code for the first character of *string expression.*
A% = ASC(MN$) PRINT ASC(MN$)

ATN(*numeric expression*) Returns the arctangent (in radians) of *numeric expression.*
AN = ATN(TH) PC = ATN(3.14)

CALL *address, expression1, expression2* Calls a machine level subroutine beginning at *address.* Upon entry to the subroutine, the A register contains the value of *expression1* and the HL register contains the value of *expression2.*
CALL 60000,10,VARPTR(A%)

CDBL(*numeric expression*) Converts the value of *numeric expression* tp a double-precision number.
A# = CDBL(A%)

CHR$(*numeric expression*) Returns the ASCII character for the value of *numeric expression.*
PRINT CHR$(65)
PRINT "He said ";CHR$(34);"Hello";CHR$(34)

CINT(*numeric expression*) Returns the largest integer not greater than the *numeric expression.*
A% = CINT(45.67) B = CINT(B#)/CINT(A!)

CLEAR *string space, high memory* Clears the values in all numeric and string variables, closes all open files, allocates *string space* bytes for string storage, and sets the end of BASIC memory to *high memory.*
CLEAR 100,50000 CLEAR 500 CLEAR 0,MAXRAM

CONT Resumes execution of a program after you have pressed BREAK or else after BASIC has encountered a STOP command in the program.
CONT

COS(*numeric expression*) Returns the cosine of the radian angle given by *numeric expression* .
Y = COS(60*0.01745329)

CSNG(*numeric expression*) Returns the single-precision form of *numeric expression.*
A! = CSNG(0.66666666666)

DATA *constant list* Defines a set of constants (numeric and/or string) to be accessed by a READ command.
DATA 10,25,50,15,"Probabilities","Total"

DATE$ Keeps track of the current date, in string form. You may acces it like any string variable.
PRINT DATE$ DATE$ = "11/02/82'

**DAY$** Keeps track of the current day of the week, in string form. You may access DAY$ like any string variable.
PRINT DAY$ DAY$ = "Fri"

DEFDBL *letter list* Defines all of the variables which begin with the letters in *letter list* to be double precision variables. *Letter list* consists of individual letters and/or letter ranges of the form letter1-letter2.
100 DEFDBL D, X-Z

DEFINT *letter list* Defines all of the variables which begin with the leters in *letter list* to be integer variables. *letter list* consists of individual letters and/or letter ranges of the form letter1-letter2.
DEFINT D, X-Z

DEFSNG *letter list* Defines all of the variables which begin with the letters in *letter list* to be single precision variables. *letter list* consists of individual letters and/or letter ranges of the form letter1-letter2.
DEFSNG D, X-Z

DEFSTR *letter list* Defines all of the variables which begin with the letters in letter list to be string variables. *letter list* consists of individual letters and/or letter ranges of the form letter1-letter2.
DEFSTR D, X-Z

DIM *variable name(dimensions) list* Defines *variable name* as an array with the given *dimensions. dimensions* is a list of one or more numeric expressions, defining the "length", "width", and so on for the array.
DIM A$(10), BAL%(10,10)

EDIT *line number range* Enter text editing mode using the given lines.
EDIT 100-1000 EDIT EDIT-200

END Terminates execution of the BASIC program.
END

ERL Returns the line number of the last error.
IF ERL = 140 THEN RESUME 150

ERR Returns the error code number of the last error.
IF ERR = 18 THEN RESUME

ERROR *numeric expression* Simulates the error specified by *numeric expression*.
ERROR 35 ERROR ERR

EXP*(numeric expression)* Returns the exponential (antilog) of *numeric expression*.
PRINT EXP(14)

FIX*(numeric expression)* Returns the whole number portion of *numeric expression*.
10 A% = FIX(A2#)

FOR *counter variable* = *initial value* TO *final value* STEP *increment* . . . NEXT *counter variable* Executes the commands between the FOR command and the NEXT command repetitively, varying *counter variable* from *initial value* to *final value*, adding *increment* to it each time BASIC ends the loop.
FOR I = 1 TO 100 STEP 4:(...):NEXT I

FRE*(expression)* Returns the current amount of unused numeric memory in bytes when *expression* is numeric and the current total amount of unused string space when *expression* is string-type.
?FRE(0) ?FRE(" ")

GOSUB *line number* Transfers program control to the subroutine beginning at *line number*.
GOSUB 1000

GOTO *line number* Branches program control to the specified *line number*.
GOTO 1000

HIMEM Returns the top address of memory available to BASIC.
?HIMEM

IF *relational* or *logical expression* THEN *command(s)1* ELSE *command(s)2* Tests the logical "truth" of *relational* or *logical expression*. If the expression is "true", then BASIC executes *command(s)1*. If the expression is "false", BASIC executes *command(s)2*.
IF A>B THEN GOTO 100 ELSE INPUT A,B

INP *(port number)* Returns a byte from the specified CPU *port number*.
A% = INP(5)

INSTR *(start position,search string,match string)* Searches *search string* for the first occurrence of *match string*, beginning at *start position*. If the string is found, INSTR returns the position in the string where it occurs. If the string isn't found, then INSTR returns a zero.
PRINT INSTR(1,"dimenthylsulfate","sulfate")

INT*(numeric expression)* Returns the whole number representation of *numeric expression* not greater than *numeric expression*.
A# = INT(–214.995)

LEFT$*(string expression,length)* Returns the first *length* characters of *string expression*.
DAY$ = LEFT$("THURSDAY",3)

LEN*(string expression)* Returns the number of characters in *string expression*.
A% = LEN("February")

LET *variable* = *expression* Assign value of *expression* to *variable*. *variable* must be of the same data type as *expression* (that is, numeric or string). The word LET is optional.
LET A$ = "The" A$ = "The"

LOG *(numeric expression)* Returns the natural logarithm (base "e") of *numeric expression*. *numeric expression* must be greater than zero.
A = LOG(10)

MENU Exits BASIC and returns you to the Model 100 Menu.
MENU

MID$*(string expression,position,length)* Returns *length* characters from *string expression* starting at *position*.
10 HASH$ = MID$(A$,2,2)

MID$*(string expression1,position,length)* = *string expression2* Replaces characters of *string expression1* starting at *position* with *string expression2*. *length* and *position* are numeric expressions. *length* is optional and if present is ignored.
MID$(A$,5) = "FF"

NEW Erases the current program, sets numeric variables equal to zero, and sets string variables equal to null (" ").
NEW

ON ERROR GOTO *line number* Defines an error trapping interrupt.
ON ERROR GOTO 1000

ON TIME$ = *"time"* GOSUB *line number* Defines an interrupt for a clock condition. *time* is a string expression in the form HH:MM:SS.
ON TIME$ = "14:20:00" GOSUB 1000

ON *numeric expression* GOTO *line number list* Evaluates *numeric expression* to an integer *n*, and then branches to the *n*th *line number* in the list.
ON X GOTO 100,200,300

ON *numeric expression* GOSUB *line number list* Evaluates *numeric expression* to an integer *n*, and then calls the subroutine beginning at the *n*th *line number* in the list.
ON X GOSUB 100,200,300

OUT *port number, byte value* Outputs *byte value* to the CPU *port number*.
OUT 55,100

PEEK *(memory address)* Returns the byte value stored at *memory address*.
A% = PEEK(16999)

POKE *memory address, byte value* Loads *memory address* with *byte value*.
POKE 32000, 104

POWER *numeric expression* Sets the automatic power down period. *numeric expression* has a range of 10 to 255. The Model 100 will automatically turn off after a period of *numeric expression times* 0.1 minutes if you are neither running a program nor entering commands.
POWER 10

POWER CONT Disables the automatic power down feature of the Model 100.
POWER CONT

POWER OFF, RESUME Turns off the power to the Model 100 immediately. If RESUME is present, upon turning the power back on, the Model 100 resumes execution of the program at the statement following the POWER OFF. RESUME, if not present, then the Model 100 returns to the Menu upon power up.
IF TIME$>"11:30:00" THEN POWER OFF
POWER OFF, RESUME

READ *variable list* Reads an appropriate number of values from a DATA statement and stores the values in the variables of *variable list*.
120 READ A,B%,C$

REM *comment statement* Signifies to the BASIC interpreter that the remainder of the line as comment. You may abbreviate REM with an apostrophe.
REM This program finds the standard deviation
100 AVE = SUM / TT 'Calculate the average

RESTORE *line number* Resets the DATA statement pointer to the first item in the DATA statement on line number so that a READ command can access the same values more than once.
600 RESTORE 100

RESUME *line number* Ends an error handling routine by branching to *line number* where BASIC begins normal execution. If *line number* is null or 0, then BASIC returns to the line which caused the error.
1010 RESUME

RETURN Ends a subroutine by branching back to the command immediately following the corresponding GOSUB.
RETURN

RIGHT$ *(string expression,count)* Returns the rightmost *count* characters of *string expression*.
10 SEC$ = RIGHT$(TIME$,2)

RND *(numeric expression)* Returns a pseudo-random number between 0 and 1. If *numeric expression* is non-zero, then RND returns a new random number. If *numeric expression* equals 0, then RND returns the last random number generated.
PRINT RND(1) PRINT RND(0)

RUN *line number* Clears all variables and executes the current program starting at *line number*. *line number* is optional if omitted, BASIC begins execution at the first line of the program.
RUN 100 RUN

SGN*(numeric expression)* Returns a –1 for negative 0 for zero, and 1 for positive values of numeric expression.
TTL = 10 * SGN(CR)

SIN *(numeric expression)* Returns the trigonometric sine of *numeric expression*. The numeric expression must be in radians.
Y = SIN(1.5)

SPACE$ *(length)* Returns a string of *length* spaces.
B$ = SPACE$(20) + A$

SQR*(numeric expression)* Returns the square root of *numeric expression*.
C2 = SQR(A 2 + B 2)

STOP Stops execution of a BASIC program at some point other than the physical end.
STOP

STR$*(numeric expression)* Converts *numeric expression* to its string representation.
B$ = "$" + STR(BAL) + ".00"

STRING$*(length, character)* Returns a string of the given *length* composed of *character. length* may range from 0 to 255. *character* is either a string expression or numeric expression — if it is a string expression, only the first character of the string is duplicated. If it is a numeric expression, it must evaluate to a number between 0 and 255.
PRINT STRING$(20,"*") PRINT STRING$(40,239)

TAN*(numeric expression)* Returns the tangent of *numeric expression. numeric expression* must be in radians.
SLOPE = TAN(THETA)

TIME$ Keeps track of the current time, in the form of a string variable. You may access it like any string variable.
PRINT TIME$ TIME$ = "10:00:00"

TIME ON or **OFF** or **STOP** Enables or disables ON TIME$ interrupting.
TIME$ ON

VAL*(string expression)* Converts *string expression* to a numeric representation of the string. If *string expression* contains non-numeric characters, VAL returns only the value of the leading number, if any.
A = VAL(B$)

VARPTR*(variable name)* Returns the memory address of *variable name.*
LINK(I) = VARPTR(B$)

Keyboard Input

Keyboard Input Commands and Functions

INKEY$ Returns the string value of the key currently pressed, if any. If no key is pressed, the function returns a null character (" "). In either case, BASIC doesn't wait for keyboard input, but goes to the next statement.
A$ = INKEY$

INPUT *"prompt";variable list* Prints prompt on the screen, then stops execution of your program until you enter data from the keyboard.
INPUT "X,Y Values";X,Y

INPUT$*(numeric expression)* Returns a string of *numeric expression* characters from the keyboard. INPUT$ accepts all keys as input except (BREAK) and doesn't echo (print on the screen) your input.
A$ = INPUT$(5)

KEY *function key number, string expression (in BASIC only)* Defines *function key number* as *string expression.*
KEY 6,"?TIME$" + CHR$(13)

KEY LIST Lists the current definitions for the function keys on the screen.
KEY LIST

KEY *(function key number)* ON or **OFF** or **STOP** Enables or disables the *function key* interrupt.
KEY (2) ON KEY ON KEY (4) OFF

LINE INPUT *"prompt", string variable* Prints *prompt* on the screen, then stops execution of your program until you enter a string from the keyboard, then assigns that string to *string variable.*
LINE INPUT "Enter Name and Address:";NA$

ON KEY GOSUB *line number list* Defines interrupts for the function keys. Upon pressing the *n*th function key, BASIC jumps to the *n*th line number in *line number list.*
ON KEY GOSUB 1000,2000,3000,,5000

The Screen (LCD)

The LCD screen consists of 15,360 (240 x 64) individual dots, or "pixels" which you may turn on ("set") or turn off ("reset") from BASIC. These pixels can also be grouped into 320 (40 x 8) positions at which you can display any of the Model 100's printable characters.

The Screen Commands and Functions

CLS Turns off all of the LCD pixels on the screen and moves the cursor to the upper left corner of the screen.
CLS

CLOSE *file number list* Closes the files OPEN'ed as *file number.*
CLOSE 1,2,3 CLOSE

CSRLIN Returns the vertical position (line number) of the cursor, where 0 is the top line and 7 is the bottom line.
A% = CSRLIN

LCOPY Prints the text on the screen onto the printer. LCOPY ignores non-text data.
LCOPY

LINE *(x1,y1)–(x2,y2), color switch*,BF Draws a line from coordinates *x1,y1* to *x2,y2*. If *color switch* is an odd number, BASIC sets the points of the line, and if *color switch* is even then BASIC resets the points on the line. **B** tells BASIC to draw a box with corners at *(x1,y1)* and *(x2,y2)*. **BF** tells BASIC to fill in the box with *color switch*. Both **B** and **BF** require that you specify *color switch*.
LINE–(30,30) LINE (20,20)–(50,63),0
LINE (0,0)–(239,63),1,B LINE (0,0)–(239,63),1,BF

LIST *line number range* Lists the *line number range* of the current program on the screen.
LIST 100-300 LIST

MAXFILES Stores the current maximum number of files. You may access MAXFILES like any numeric variable.
10 MAXFILES = 5 PRINT MAXFILES

OPEN "LCD:" FOR OUTPUT AS *file number* Allocates a buffer for a screen file and assigns it the given *file number.*
OPEN "LCD:" FOR OUTPUT AS 1

POS*(dummy numeric expression)* Returns the current cursor position.
R% = POS(0)

PRESET *(x-coordinate, y-coordinate)* Turns off the LCD pixel at *(x-coordinate, y-coordinate)* may range from 0 to 63.
PRESET (55,10)

PRINT *expression list* Prints the data in *expression list* onto the screen, starting at the left most end of the line.
PRINT "Menu #";I PRINT I%,J%,K%

PRINT @ *screen position* Prints at the given screen position.
PRINT @140,TIME$

PRINT USING *"format"; expression list* Prints the data in *expression list* using the specified *format. format* consists of one or more of the following "field specifiers";

"!" Prints first string character.
PRINT USING "!";"Tandy"

"\ *n* spaces \" Prints 2 + *n* characters from a string.
PRINT USING "\ \";"Tandy"

Prints one digit for each #.
PRINT USING "#####";5

+ Inserts the algebraic sign of the number.
PRINT USING "+#####";–13

– Inserts a minus sign either at the beginning or end of negative numbers.
PRINT USING "–######";14

****** Converts leading blanks to leading asterisks blanks.
PRINT USING "**#####";145

$$ Inserts a dollar sign to the immediate left of the formatted number.
PRINT USING "$$#####";450

****$** Changes leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign.
PRINT USING "**$###";12

`.` Inserts a decimal point.
PRINT USING "#####.##";14.5

`,` Inserts a comma before every three printed digits to the left of the decimal point.
PRINT USING "#########,";14432

`^^^^` Prints the number in exponential format.
PRINT USING "###.#^^^^";342200

PRINT # *file number, expression list* Prints the values of *expression list* to the LCD file OPEN'ed as *file number*.
PRINT #1,A$ PRINT#4,10,20,30

PRINT #*file number*,USING*"format";expression list* Formats the data in expression list and sends it to the LCD file OPEN'ed as *file number*. See **PRINT USING** for an explanation of *format*.
PRINT #1,USING "#####.##";A PRINT #5,USING " ";A$

PSET *(x-coordinate,y-coordinate)* Turns on the LCD pixel at *x-coordinate,y-coordinate*, where *x-coordinate* is a numeric expression ranging from 0 to 239, and *y-coordinate* is a numeric expression ranging from 0 to 63.
PSET (40,45)

SAVE "LCD:" Lists the current BASIC program onto the screen. (Note: Pressing (PAUSE) has no effect on this command.)
SAVE "LCD:"

SCREEN *on* or *off* Turns the LABEL line on or off. *On* is 0,0; *off* is 0,1.
SCREEN 0,0 SCREEN 0,1.

TAB *(numeric expression)* Skips *numeric expression* spaces before printing the next data item. *numeric expression* ranges between 0 and 255.
PRINT TAB(30);"Table 1"
PRINT #1,"Total";TAB(10);"Number";TAB(10);"Balance"

Printer (LPT)

Printer Commands and Functions

CLOSE *file number list* Closes the specified file numbers.
CLOSE 1,2,3, CLOSE

LCOPY Prints the text on the LCD screen onto the printer. LCOPY ignores non-text data.
LCOPY

LLIST *line number range* Lists the *line number range* of the current program onto the printer.
LLIST 100-300 LLIST

LPOS *(dummy numeric expression)* Returns the current position of the printer print head within the printer buffer.
LPRINT "Printer head position:";LPOS(0)

LPRINT *expression list* Prints out the values of *expression list* on the printer.
LPRINT "The total for ";A$;" was "; TT
LPRINT X,Y,Z

LPRINT USING *"format string";expression list* Formats the data in *expression list* then prints it on the printer. *format* consists of one or more of the following:

`"!"` Prints first string character.
LPRINT USING "!";"Tandy"

`\ nspaces \` Prints 2 + *n* characters from a string.
LPRINT USING " \ \";"Tandy"

`#` Prints one digit for each #.
LPRINT USING "#####";5

`+` Inserts the algebraic sign of the number.
LPRINT USING "+#####";–13

`–` Inserts a minus sign either at the beginning or end of negative numbers.
LPRINT USING "–######";14

`**` Converts leading blanks to leading asterisks blanks.
LPRINT USING "**#####";145

`$$` Inserts a dollar sign to the immediate left of the formatted number.
LPRINT USING "$$#####";450

$$ Changes leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign.
LPRINT USING "**$###";12

. Inserts a decimal point.
LPRINT USING "#####.##";14.5

, Inserts a comma before every three printed digits to the left of the decimal point.
LPRINT USING "#########,";14432

^^^^ Prints the number in exponential format.
LPRINT USING "###.#^^^^";342200

MAXFILES Stores the current maximum number of files. You may access MAXFILES like any numeric variable.
MAXFILES = 5 ?MAXFILES

OPEN "LPT:" FOR OUTPUT AS *file number* Allocates a buffer *file number* for a printer file.
OPEN "LPT:" FOR OUTPUT AS 1

PRINT *# file number, expression list* Prints the values of *expression list* into the printer file OPEN'ed as *file number*.
PRINT #1,A$ PRINT #4,10,20,30

PRINT *#file number,* USING *"format"; expression list* Formats the data in *expression list* and sends it to the printer file OPEN'ed as *file number*. For a description of format, see LPRINT USING.

SAVE "LPT:" Prints the current BASIC program onto the printer.
SAVE "LPT:"

TAB *(numeric expression)* Skips *numeric expression* spaces before printing the next data item. *numeric expression* ranges between 0 and 255.
PRINT #1,TAB(30);"Table 1"
LPRINT TAB(30);"Total";TAB(30);"Number";TAB(30);"Balance"

RAM Files (RAM)

RAM File Filenames and Extensions

RAM filenames consist of a string of one to six characters, the first of which is a letter. In addition, most RAM files have specific extensions which are suffixed to the filename. These extensions are:

.BA	BASIC Program File
.DO	ASCII Formatted File (a BASIC data file, a Text Editor File, certain BASIC program files)
.CO	Command File (a machine-language file)

Note: Most commands which allow you to specify a device (such as LCD,MDM, and so on) default to RAM. In addition, BASIC can often assume the extension of a file from the context of the command.

RAM I/O Commands and Functions

CLOSE *file number list* Closes the specified file numbers.
CLOSE 1,2,3 CLOSE

EOF *(file number)* Tests for an end-of-file condition on the RAM file OPEN'ed as *file number*. The function returns a "logical" answer, either "true" (–1) if you have reached the end of the file, or else "false" (0) if you have not reached the end of the file.
IF EOF(1) THEN GOTO 1000

INPUT *# file number, variable list* Inputs data sequentially from the RAM file OPEN'ed as *file number*.
INPUT #1,A,B,C$

INPUT$ *(numeric expression, file number)* Returns a string of a length given by *numeric expression* from the RAM file OPEN'ed as *file number*.
A$ = INPUT$(5,1)

IPL *"filename"* Defines the RAM filename as the warm-startup program (that is, the program which runs immediately when you turn on the Computer).
IPL "TIMSET.BA"

KILL *"filename"* Deletes the RAM file *filename*. You must include the file's extension.
KILL "BILLS.BA" KILL "MSGS.DO"

LINE INPUT *#, str var* Reads a line of text from device.
LINE INPUT #1,Z$

LOAD "RAM:*filename*",R Loads a BASIC program from RAM. If **R** is present, BASIC runs the program after it has been loaded.
LOAD "RAM:TIMSET" LOAD "TIMSET",R

LOADM "RAM:*filename*" Loads the machine-language program *filename* from RAM at the address specified when it was saved.
LOADM "MEMTST" LOADM "RAM:MEMTST"

MAXFILES Stores the current maximum number of files. You may access MAXFILES like any numeric variable.
10 MAXFILES = 5 ?MAXFILES

MERGE "RAM:*filename*" Merges the lines from the ASCII formatted RAM file called *filename* with the lines of the current program.
MERGE "RAM:ACT.DO"

NAME "RAM:*old filename*" AS "RAM:*new filename*" Renames *old filename* to *new filename*. You must include the file's extension.
NAME "ACCTS.DO" AS "OLDACT.DO"
NAME "RAM:CLS1.CO" AS "RAM:LCDCLS.CO"

OPEN "RAM:*filename*" FOR *mode* AS *file number* Allocates a buffer, *file number*, for a RAM file called *filename*. *mode* can be OUTPUT, specifying data will be written sequentially to the file, starting at the beginning of the file, INPUT, specifying data will be read sequentially from the file, starting at the beginning of the file, or APPEND, specifying that data will be written sequentially to the file, adding records to the end of the file.
OPEN "RAM: ACCT.DO" FOR APPEND AS 1
OPEN "NAMES.DO" FOR INPUT AS 4

PRINT # *file number, expression list* Writes the values of *expression list* to the RAM file OPEN'ed as *file number*.
PRINT #1,A$ PRINT #4,10,20,30

PRINT #*file number*, USING *"format"; expression list* Formats the data in *expression list* and sends it to the RAM file OPEN'ed as *file number*. *format* consists of one or more of the following:

"!"	Prints first string character. PRINT #1,USING "!";"Tandy"
"\ ***n* spaces** \"	Prints 2 + *n* characters from a string. PRINT #1,USING " \\ ";"Tandy"
#	Prints one digit for each #. PRINT #1,USING "#####";5

+	Inserts the algebraic sign of the number. PRINT #1,USING " + #####";–13
–	Inserts a minus sign either at the beginning or end of negative numbers. PRINT #1,USING "–######";14
**	Converts leading blanks to leading asterisks blanks. PRINT #1,USING "**#####";145
$$	Inserts a dollar sign to the immediate left of the formatted number. PRINT #1,USING "$$#####";450
**$	Changes leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign. PRINT #1,USING "**$###";12
.	Inserts a decimal point. PRINT #1USING "#####.##";14.5
,	Inserts a comma before every three printed digits to the left of the decimal point. PRINT #1,USING "#########,";14432
^^^^	Prints the number in exponential format. PRINT #1,USING "###.#^^^^";342200

RUN "RAM:*filename*",R Clears all variables, loads the BASIC program called *filename* from RAM, and then executes the program. If **R** is present, BASIC keeps all open files open. If **R** is omitted, BASIC closes all open files before loading file.
RUN "PART2.BA",R RUN "RAM:BILLS.BA"

RUNM "RAM:*filename*" Loads and executes the machine-language program stored as *filename*. The program must be one executable from the Menu, not a BASIC subroutine. In addition, when the program is loaded, BASIC closes all open files.
RUNM "RAM:MEMTST.CO" RUNM "CLR1"

SAVE "RAM:*filename*",A Writes the current BASIC program to the RAM file called *filename*. **A** is optional; if used, BASIC saves the file in ASCII format. Otherwise BASIC saves the file in a compressed binary format. If *filename* already exists in RAM, BASIC writes over the old file.
SAVE "TIMSET" SAVE "RAM:PART1.DO",A

SAVEM "RAM:*filename", start address, end address, entry address* Writes the machine language program stored from *start address* to *end address* into RAM, under the name *filename*. *entry address* is optional; if not present, then BASIC assumes that the program *entry address* is the same as the *start address*.
SAVEM "RAM:MEMTST",50000,50305,50020
SAVEM "MEMTST",50000,50305

TAB *(numeric expression)* Writes *numeric expression* spaces before writing the next data item.
PRINT #1,TAB(30);"Table 1"

The Cassette Recorder/Player (CAS)

File Names for Cassette Files

Cassette file names consist of a string of one to six characters, the first of which is a letter. There is no need for an extension. For example,
ACCTM MEMTST CLK100

Cassette Commands

CLOAD *"filename"*, R Clears the current BASIC program and loads a BASIC program from cassette tape. **R**, if present, tells BASIC to run the program after loading it.
CLOAD "ACCTS",R CLOAD

CLOAD? *filename* Compares the cassette file filename with the BASIC program currently in memory. If there are any differences, BASIC displays the message *Verify failed* on the screen; otherwise, BASIC simply prints OK.
CLOAD? "ACCT"

CLOADM *"filename"* Loads the machine-language program called *filename* from cassette tape into memory, at the address used when it was written to the cassette tape.
CLOADM "MEMTST" CLOADM

CLOSE *file number list* Closes the files OPEN'ed as *file number*.
CLOSE 1,2,3 CLOSE

CSAVE *"filename"*,A Stores the current BASIC program on cassette tape as *filename*. **A**, if present, tells BASIC to save the program in ASCII format. If omitted, BASIC stores the program in a compressed binary form.
CSAVE "WDC" CSAVE "PART1",A

CSAVEM *"filename", start address, end address, entry address* Writes the machine language program stored from *start address* to *end address* with the *entry address* onto cassette tape, under the name *filename*.
CSAVEM "MEMTST",50000,50305,50020
CSAVEM "CLR",39000,39030

EOF *(file number)* Tests for an end-of-file condition the cassette file OPEN'ed as *file number*. The function returns a "logical" answer, either "true" (–1) if you have reached the end of the file, or else "false" (0) if you have not reached the end of the file.
IF EOF(1) THEN GOTO 1000

INPUT # *file number, variable list* Inputs data sequentially from the cassette file opened as *file number*.
INPUT #1,A,B,C$

INPUT$ (*numeric expression, file number*) Returns a string of a length given by *numeric expression* from the cassette file opened as *file number*.
A$ = INPUT$(5,1)

LINE INPUT #, *str var* Reads a line of text from device.
LINE INPUT #1,Z$

LOAD "CAS:*filename*",R Loads a BASIC program from cassette. If **R** is present, BASIC runs the program after it has been loaded.
LOAD "CAS:ACCT",R LOAD "CAS:MATH"

LOADM "CAS:*filename*" Loads the machine-language program *filename* from cassette tape, at the address specified when it was SAVE'd.
LOADM "CAS:MEMTST"

MAXFILES Stores the current maximum number of files. You may access MAXFILES like any numeric variable.
10 MAXFILES = 5 ?MAXFILES

MERGE "CAS:*filename*" Merges the lines from the ASCII formatted *filename (not compressed format!)* with the lines of the current program.
MERGE "CAS:ACCT"

MOTOR ON or **OFF** Starts or stops cassette player motor.
MOTOR ON

OPEN "CAS:*filename*" FOR *mode* AS *file number* Allocates a buffer for the cassette file *filename*. *file number* is the buffer number assigned to the file. *mode* can be OUTPUT, specifying data will be written sequentially to the file, starting at the beginning of the file, or it can be INPUT, specifying data will be read sequentially from the file, starting at the beginning of the file.
OPEN "CAS:" FOR OUTPUT AS 3
OPEN "CAS:ACTDAT" FOR INPUT AS 4

PRINT # *file number, expression list* Writes the values of *expression list* to the cassette file opened as *file number*.
PRINT #1A$ PRINT #4,10,20,30

PRINT #*file number*, USING *"format"; expression list* Formats the data in *expression list* and writes it to the cassette file OPEN'ed as *file number*. *format* consists of one or more of the following:

"!" Prints first string character.
PRINT USING "!";"Tandy"

"\ *n* spaces \" Prints 2 + *n* characters from a string.
PRINT #1,USING " \ \ ";"Tandy"

Prints one digit for each #.
PRINT USING "#####";5

+ Inserts the algebraic sign of the number
PRINT USING "+#####";-13

- Inserts a minus sign either at the beginning or end of negative numbers.
PRINT USING "-######";14

****** Converts leading blanks to leading asterisks blanks.
PRINT USING "**#####";145

$$ Inserts a dollar sign to the immediate left of the formatted number.
PRINT USING "$$#####";450

****$** Changes leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign.
PRINT USING "**$###";12

. Inserts a decimal point.
PRINT USING "#####.##";14.5

, Inserts a comma before every three printed digits to the left of the decimal point.
PRINT USING "#########,";14432

^^^^ Prints the number in exponential format.
PRINT USING "###.#^^^^";342200

RUN "CAS:*filename*",R Clears all variables, then loads and executes the cassette program called *filename*. If **R** is present, BASIC keeps all open files open. If **R** is omitted, BASIC closes all open files before loading filename.
RUN "CAS:PART2",R RUN "CAS:PART1"

RUNM "CAS:*filename*" Loads and executes the machine-language program stored as *filename*.
RUNM "CAS:MEMTST" RUNM "CAS:"

SAVE "CAS:*filename*",A Writes the current BASIC program to cassette tape. If **A** is present, BASIC saves the file in ASCII format. If omitted, BASIC saves the file in a compressed binary format.
SAVE "CAS:TIMSET" SAVE "CAS:PART1",A

SAVEM "CAS:*filename, start address, end address, entry address* Writes the machine language program stored from *start address* to *end address* onto cassette tape as *filename. entry address* is optional; if omitted, then BASIC assumes that the program entry address is the same as the start address.

```
SAVEM "CAS:MEMTST",50000,50305,50020
SAVEM "CAS:CLR1",50000,50305
```

TAB *(numeric expression)* Skips *numeric expression* spaces before printing the next data item to a cassette file. *numeric expression* ranges between 0 and 255.

```
PRINT #1, "LEADER1";TAB(30);"Table 1"
```


RS-232C Communications (COM)

Communications Configuration

Some BASIC commands require that you signify the communications configuration. It consists of a five character string of the pattern *rwpbs*, where:

r	**Baud Rate** This a number from 1 to 9, where 1 = 75; 2 = 110; 3 = 300; 4 = 600; 5 = 1200; 6 = 2400; 7 = 4800; 8 = 9600; 9 = 19200.
w	**Word Length** This is a number from 6 to 8, where 6 = 6 bits; 7 = 7 bits; 8 = 8 bits.
p	**Parity** Either E,O,I, or N, where E = Even; O = Odd; I = Ignore; N = None.
b	**Stop Bits** Either 1 or 2, where 1 = 1 stop bit; 2 = 2 stop bits.
s	**XON/XOFF Status** Either E or D, where E = Enable; D = Disable.

Communications Commands and Functions

COM ON or **OFF** or **STOP** Enables or disables the ON COM interrupt.

```
COM ON        COM OFF        COM STOP
```

CLOSE *file number list* Closes the files OPEN'ed as file number.

```
CLOSE 1,2,3        CLOSE
```

EOF *(file number)* Tests for an end-of-file condition on the communications file OPEN'ed as *file number.* The function returns a "logical" answer, either "true" (–1) if you have reached the end of the file, or else "false" (0) if you have not reached the end of the file.

```
IF EOF(1) THEN GOTO 1000
```

INPUT # *file number, variable list* Inputs data sequentially from the communications file OPEN'ed as *file number.*

```
INPUT #1,A,B,C$
```

INPUT$ *(numeric expression, file number)* Returns a string of a length given by *numeric expression* from the communications file OPEN'ed as *file number.*

```
A$ = INPUT$(5,1)
```

LOAD "COM:*configuration*",R Loads a BASIC program from communications lines. If **R** is present, BASIC runs the program after it has been loaded.
LOAD "COM:78N1E"
LOAD "COM:88E1E"

MAXFILES Stores the current maximum number of files. You may access MAXFILES like any numeric variable.
10 MAXFILES = 5 ?MAXFILES

MERGE "COM:*configuration*" Merges the lines from the incoming file with the lines of the current program.
MERGE "COM:78E1E"

ON COM GOSUB *line number* Defines a communications interrupt subroutine for incoming RS-232C communications.
ON COM GOSUB 1000

OPEN "COM:*configuration*" FOR *mode* AS *file number* Allocates a buffer given as *file number* for a communications file. *mode* can be OUTPUT, specifying data will be transmitted out the RS-232C line, or INPUT, specifying data will be received via the RS-232C line.
10 OPEN "COM:6601E" FOR INPUT AS 4

PRINT # *file number*, *expression list* Transmits the values of *expression list* to the communications file opened as *file number*.
PRINT #1,A$ PRINT #4,10,20,30

PRINT #*file number*, USING *"format"*; *expression list* Formats the data in *expression list* and sends it to the communications file OPEN'ed as *file number*. *format* consists of one or more of the following:

"!"	Prints first string character. PRINT #1,USING "!";"Tandy"
"*n* spaces \"	Prints 2 + *n* characters from a string. PRINT #1,USING " \ \ ";"Tandy"
#	Prints one digit for each #. PRINT #1,USING "#####";5
+	Inserts the algebraic sign of the number. PRINT #1,USING " + #####";–13
–	Inserts a minus sign either at the beginning or end of negative numbers. PRINT #1,USING "–######";14

******	Converts leading blanks to leading asterisks blanks. PRINT #1,USING "**#####";145
$$	Inserts a dollar sign to the immediate left of the formatted number. PRINT #1,USING "$$#####";450
****$**	Changes leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign. PRINT #1,USING "**$###";12
.	Inserts a decimal point. PRINT #1,USING "#####.##";14.5
,	Inserts a comma before every three printed digits to the left of the decimal point. PRINT #1,USING "#########,";14432
^^^^	Prints the number in exponential format. PRINT #1,USING "###.#^^^^";342200

RUN "COM:*configuration*",R Clears all variables, loads a BASIC program from communications line and then executes the new program. If **R** is present, BASIC keeps all open files open. If **R** is omitted, BASIC closes all open files before loading the new file.
RUN "COM:67E1E",R RUN "COM:67E1E"

SAVE "COM:*configuration*" Writes the current BASIC program out the communications line, in ASCII format.
SAVE "COM:38N2E"

TAB *(numeric expression)* Transmits *numeric expression* spaces before transmitting the next data item. *numeric expression* ranges between 0 and 255.
PRINT #1, TAB(30);"Table 1"

Modem Communications (MDM)

Modem Communications configuration
Some BASIC commands require that you specify the modem configuration. The baud rate is set to 300 by default. The rest of the configuration consists of a four character string of the pattern *wpbs*, where:

w	**Word Length** This is a number from 6 to 8 where 6 = 6 bits; 7 = 7 bits; 8 = 8 bits.
p	**Parity** Either E,O,I, or N, where E = Even; O = Odd; I = Ignore; N = None.
b	**Stop Bits** Either 1 or 2, where 1 = 1 stop bit; 2 = 2 stop bits.
s	**XON/XOFF Status** Either E or D, where E = Enable; D = Disable.

Modem Communications Commands and Functions

CLOSE *file number list* Closes the specified *file numbers.*
CLOSE 1,2,3 CLOSE

EOF *(file number)* Tests for an end of file condition on the modem file OPEN'ed as *file number.* The function returns a "logical" answer, either "true" (–1) if you have reached the end of the file, or else "false" (0) if you have not reached the end of the file.
IF EOF(1) THEN GOTO 1000

INPUT # *file number, variable list* Inputs data sequentially from the modem file opened as *file number.*
INPUT #1,A,B,C$

INPUT$ *(numeric expression, file number)* Returns a string of a length given by *numeric expression* from the modem file opened as *file number.*
A$ = INPUT$(5,1)

LINE INPUT #, *str var* Reads a line of text from device.
LINE INPUT #1,Z$

LOAD "MDM:*configuration*",R Loads a BASIC program from the modem. If **R** is present, BASIC runs the program after it has been loaded.
LOAD "MDM:8N1E" LOAD "MDM:8n1e",R

MAXFILES Stores the current maximum number of files. You may access MAXFILES like any numeric variable.
10 MAXFILES = 5 ?MAXFILES

MDM ON or **OFF** or **STOP** Enables or disables the ON MDM interrupt.
MDM ON MDM OFF

MERGE *"MDM: configuration"* Merges the lines from the BASIC program file coming in over the modem with the lines of the current program.
MERGE "MDM:8EIE"

ON MDM GOSUB *line number* Defines an interrupt for incoming modem communications.
ON MDM GOSUB 1000

OPEN "MDM:*configuration*" FOR *mode* AS *file number* Allocates a buffer *file number* for a modem file. *mode* can be OUTPUT, specifying data will be transmitted out the modem, or it can be INPUT, specifying data will be received via the modem.
OPEN "MDM:6E1E" FOR INPUT AS 4

PRINT # *file number, expression list* Transmits the values of *expression list* to the modem file OPEN'ed as *file number.*
PRINT #1,A$ PRINT #4,10,20,30

PRINT #*file number*, USING *"format"; expression list* Formats the data in *expression list* and sends it to the modem file OPEN'ed as *file number. format* consists of one or more of the following:

"!"	Prints first string character. PRINT #1,USING "!";"Tandy"
"*n* spaces \"	Prints 2 + *n* characters from a string. PRINT #1,USING " \ \ ";"Tandy"
#	Prints one digit for each #. PRINT #1,USING "#####";5
+	Inserts the algebraic sign of the number. PRINT #1,USING " +#####";–13
–	Inserts a minus sign either at the beginning or end of negative numbers. PRINT #1,USING "–######";14
******	Converts leading blanks to leading asterisks blanks. PRINT #1,USING "**#####";145

$$ Inserts a dollar sign to the immediate left of the formatted number.
PRINT #1,USING "$$#####";450

****$** Changes leading spaces to asterisks except for the space to the immediate left of the number, where it inserts a dollar sign.
PRINT #1,USING "**$###";12

. Inserts a decimal point.
PRINT #1,USING "#####.##";14.5

, Inserts a comma before every three printed digits to the left of the decimal point.
PRINT #1,USING "#########,";14432

^^^^ Prints the number in exponential format.
PRINT #1,USING "###.#^^^^";342200

RUN "MDM:*configuration*",R Clears all variables, loads the BASIC program from the modem, and then executes the program. If **R** is present, BASIC keeps all open files open. If **R** is omitted, BASIC closes all open files before loading.
RUN "MDM:8E1E" RUN "MDM:7E1D",R

SAVE "MDM:*configuration*" Transmits the current BASIC program out the modem, in ASCII format.
SAVE "MDM:6N2E"

TAB *(numeric expression)* Transmits *numeric expression* spaces before transmitting the next data item.
PRINT #1, TAB(30);"Table 1"

The Sound Generator

Sound Generator Commands

BEEP Causes the sound generator to "beep" for approximately 1/2 second.
BEEP

SOUND *pitch, length* "Plays" a given *pitch* for the given *length*. *length* ranges from 0 to 255. Dividing length by 50 gives the approximate length in seconds. *pitch* ranges from 0 to 16383, with the smaller values corresponding to higher pitches.
SOUND 4500,50

SOUND ON or **OFF** Enables or disables "beep" when:
- You're loading from cassette
- The Model 100 is waiting on a carrier signal from the telephone modem lines.

SOUND ON SOUND OFF

SOUND Pitch Values Corresponding to Standard Musical notes

	Octave				
Note	**1**	**2**	**3**	**4**	**5**
G	12538	6269	3134	1567	783
G#	11836	5918	2959	1479	739
A	11172	5586	2793	1396	698
A#	10544	5272	2636	1318	659
B	9952	4968	2484	1244	622
C	9394	4697	2348	1174	587
C#	8866	4433	2216	1108	554
D	8368	4184	2092	1046	523
D#	7900	3950	1975	987	493
E	7456	3718	1864	932	466
F	7032	3516	1758	879	439
F#	6642	3321	1660	830	415

ASCII Codes/Characters

Decimal	Hex	Binary	Printed Character	Keyboard Character
0	00	00000000		PAUSE
1	01	00000001		CTRL A
2	02	00000010		CTRL B
3	03	00000011		CTRL C
4	04	00000100		CTRL D
5	05	00000101		CTRL E
6	06	00000110		CTRL F
7	07	00000111		CTRL G
8	08	00001000		CTRL H
9	09	00001001		CTRL I
10	0A	00001010		CTRL J
11	0B	00001011		CTRL K
12	0C	00001100		CTRL L
13	0D	00001101		CTRL M
14	0E	00001110		CTRL N
15	0F	00001111		CTRL O
16	10	00010000		CTRL P
17	11	00010001		CTRL Q
18	12	00010010		CTRL R
19	13	00010011		CTRL S
20	14	00010100		CTRL T
21	15	00010101		CTRL U
22	16	00010110		CTRL V
23	17	00010111		CTRL W
24	18	00011000		CTRL X
25	19	00011001		CTRL Y
26	1A	00011010		CTRL Z

Decimal	Hex	Binary	Printed Character	Keyboard Character
27	1B	00011011		ESC
28	1C	00011100		→
29	1D	00011101		←
30	1E	00011110		↑
31	1F	00011111		↓
32	20	00100000		SPACEBAR
33	21	00100001	!	!
34	22	00100010	"	"
35	23	00100011	#	#
36	24	00100100	$	$
37	25	00100101	%	%
38	26	00100110	&	&
39	27	00100111	'	'
40	28	00101000	(	(
41	29	00101001	)	)
42	2A	00101010	*	*
43	2B	00101011	+	+
44	2C	00101100	,	,
45	2D	00101101	–	–
46	2E	00101110	.	.
47	2F	00101111	/	/
48	30	00110000	0	0
49	31	00110001	1	1
50	32	00110010	2	2
51	33	00110011	3	3
52	34	00110100	4	4
53	35	00110101	5	5
54	36	00110110	6	6

Decimal	Hex	Binary	Printed Character	Keyboard Character
55	37	00110111	7	7
56	38	00111000	8	8
57	39	00111001	9	9
58	3A	00111010	:	:
59	3B	00111011	;	;
60	3C	00111100	<	<
61	3D	00111101	=	=
62	3E	00111110	>	>
63	3F	00111111	?	?
64	40	01000000	@	@
65	41	01000001	A	A
66	42	01000010	B	B
67	43	01000011	C	C
68	44	01000100	D	D
69	45	01000101	E	E
70	46	01000110	F	F
71	47	01000111	G	G
72	48	01001000	H	H

* For uppercase letters A-Z, press SHIFT or CAPS LOCK before pressing the Keyboard Character.

Decimal	Hex	Binary	Printed Character	Keyboard Character
73	49	01001001	I	I
74	4A	01001010	J	J
75	4B	01001011	K	K
76	4C	01001100	L	L
77	4D	01001101	M	M
78	4E	01001110	N	N
79	4F	01001111	O	O
80	50	01010000	P	P
81	51	01010001	Q	Q
82	52	01010010	R	R
83	53	01010011	S	S
84	54	01010100	T	T
85	55	01010101	U	U
86	56	01010110	V	V
87	57	01010111	W	W
88	58	01011000	X	X
89	59	01011001	Y	Y
90	5A	01011010	Z	Z
91	5B	01011011	[	[
92	5C	01011100	\	GRPH –
93	5D	01011101	]	]
94	5E	01011110	^	^
95	5F	01011111	_	_
96	60	01100000	\	GRAPH [
97	61	01100001	a	A

* For lowercase letters a-z, be sure CAPS LOCK is not pressed "down."

Decimal	Hex	Binary	Printed Character	Keyboard Character
98	62	01100010	b	B
99	63	01100011	c	C
100	64	01100100	d	D
101	65	01100101	e	E
102	66	01100110	f	F
103	67	01100111	g	G
104	68	01101000	h	H
105	69	01101001	i	I
106	6A	01101010	j	J
107	6B	01101011	k	K
108	6C	01101100	l	L
109	6D	01101101	m	M
110	6E	01101110	n	N
111	6F	01101111	o	O
112	70	01110000	p	P
113	71	01110001	q	Q
114	72	01110010	r	R
115	73	01110011	s	S
116	74	01110100	t	T
117	75	01110101	u	U
118	76	01110110	v	V
119	77	01110111	w	W
120	78	01111000	x	X
121	79	01111001	y	Y
122	7A	01111010	z	Z
123	7B	01111011	{	GRPH 9
124	7C	01111100	|	GRPH _
125	7D	01111101	}	GRPH 0

Decimal	Hex	Binary	Printed Character	Keyboard Character
126	7E	01111110	~	GRPH]
127	7F	01111111		DEL
128	80	10000000	☎	GRPH p
129	81	10000001		GRPH m
130	82	10000010		GRPH f
131	83	10000011		GRPH x
132	84	10000100		GRPH c
133	85	10000101		GRPH a
134	86	10000110		GRPH h
135	87	10000111		GRPH t
136	88	10001000	i	GRPH l
137	89	10001001	√	GRPH r
138	8A	10001010	≠	GRPH /
139	8B	10001011	Σ	GRPH s
140	8C	10001100	≈	GRPH '
141	8D	10001101	±	GRPH =
142	BE	10001110	∫	GRPH i
143	BF	10001111	◀	GRPH e
144	90	10010000		GRPH y
145	91	10010001		GRPH u
146	92	10010010		GRPH ;
147	93	10010011		GRPH q
148	94	10010100		GRPH w
149	95	10010101		GRPH b
150	96	10010110		GRPH n
151	97	10010111	%	GRPH .
152	98	10011000		GRPH o
153	99	10011001		GRPH ,

Decimal	Hex	Binary	Printed Character	Keyboard Character
154	9A	10011010	→	GRPH I
155	9B	10011011	←	GRPH k
156	9C	10011100	♧	GRPH 2
157	9D	10011101	♢	GRPH 3
158	9E	10011110	♡	GRPH 4
159	9F	10011111	♤	GRPH 5
160	A0	10100000	'	CODE '
161	A1	10100001	à	CODE x
162	A2	10100010	ç	CODE c
163	A3	10100011	£	GRPH 8
164	A4	10100100	`	CODE "
165	A5	10100101	µ	CODE M
166	A6	10100110	°	CODE)
167	A7	10100111	▼	CODE _
168	A8	10101000	†	CODE +
169	A9	10101001	⊶	CODE s
170	AA	10101010	®	CODE R
171	AB	10101011	©	CODE C
172	AC	10101100	¼	CODE p
173	AD	10101101	¾	CODE ;
174	AE	10101110	½	CODE /
175	AF	10101111	¶	CODE 0
176	B0	10110000	¥	GRPH 7

Decimal	Hex	Binary	Printed Character	Keyboard Character
177	B1	10110001	Ä	CODE A
178	B2	10110010	Ö	CODE O
179	B3	10110011	Ü	CODE U
180	B4	10110100	¢	GRPH 6
181	B5	10110101	-	CODE [
182	B6	10110110	ä	CODE a
183	B7	10110111	ö	CODE o
184	B8	10111000	ü	CODE u
185	B9	10111001	B	CODE S
186	BA	10111010	TM	CODE T
187	BB	10111011	é	CODE d
188	BC	10111100	ù	CODE ,
189	BD	10111101	è	CODE v
190	BE	10111110	¨	CODE =
191	BF	10111111	£	CODE F
192	C0	11000000	â	CODE I
193	CI	11000001	ê	CODE 3
194	C2	11000010	î	CODE 8
195	C3	11000011	ô	CODE 9
196	C4	11000100	û	CODE 7
197	C5	11000101	^	CODE –
198	C6	11000110	ë	CODE e
199	C7	11000111	ï	CODE i
200	C8	11001000	á	CODE q
201	C9	11001001	í	CODE k
202	CA	11001010	ó	CODE l
203	CB	11001011	ú	CODE j
204	CC	11001100	ý	CODE y

Decimal	Hex	Binary	Printed Character	Keyboard Character
205	CD	11001101	ñ	(CODE) n
206	CE	11001110	ã	(CODE) z
207	CF	11001111	õ	(CODE) .
208	D0	11010000	Â	(CODE) !
209	D1	11010001	Ê	(CODE) #
210	D2	11010010	Î	(CODE) *
211	D3	11010011	Ô	(CODE) (
212	D4	11010100	Û	(CODE) &
213	D5	11010101	Ï	(CODE) I
214	D6	11010110	Ë	(CODE) E
215	D7	11010111	É	(CODE) D
216	D8	11011000	Á	(CODE) Q
217	D9	11011001	Í	(CODE) K
218	DA	11011010	Ó	(CODE) L
219	DB	11011011	Ú	(CODE) J
220	DC	11011100	Ý	(CODE) Y
221	DD	11011101	Ù	(CODE) <
222	DE	11011110	È	(CODE) V
223	DF	11011111	À	(CODE) X
224	ED	11100000		(GRPH) Z
225	E1	11100001	▘ (upper left)	(GRPH) !
226	E2	11100010	▝ (upper right)	(GRPH) @
227	E3	11100011	▖ (lower left)	(GRPH) #
228	E4	11100100	▗ (lower right)	(GRPH) $
229	E5	11100101	▚	(GRPH) %

Decimal	Hex	Binary	Printed Character	Keyboard Character
230	E6	11100110	▟	(GRPH) ^
231	E7	11100111	▔ (upper)	(GRPH) Q
232	E8	11101000	▁ (lower)	(GRPH) W
233	E9	11101001	▏ (left)	(GRPH) E
234	EA	11101010	▕ (right)	(GRPH) R
235	EB	11101011	▛	(GRPH) A
236	EC	11101100	▜	(GRPH) S
237	ED	11101101	▙	(GRPH) D
238	EE	11101110	▟	(GRPH) F
239	EF	11101111	■	(GRPH) X
240	F0	11110000	⌐	(GRPH) U
241	F1	11110001	—	(GRPH) P
242	F2	11110010	¬	(GRPH) O
243	F3	11110011	⊤	(GRPH) I
244	F4	11110100	⊢	(GRPH) J
245	F5	11110101	|	(GRPH) :
246	F6	11110110	└	(GRPH) M
247	F7	11110111	┘	(GRPH) >
248	F8	11111000	⊥	(GRPH) <
249	F9	11111001	┤	(GRPH) L
250	FA	11111010	┼	(GRPH) K
251	FB	11111011	◤	(GRPH) H
252	FC	11111100	◢	(GRPH) T
253	FD	11111101	◥	(GRPH) G
254	FE	11111110	◣	(GRPH) Y
255	FF	11111111	▒	(GRPH) C

Model 100 BASIC Error Codes

Code	Message	Meaning
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	Bad Subscript.
10	DD	Doubly Dimensioned 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	IO	Error.
19	NR	No RESUME.
20	RW	RESUME Without Error.
21	UE	Undefined Error.
22	MO	Missing Operand.
23-49	UE	Undefined Error.
50	IE	Undefined Error.
51	BN	Bad File Number.
52	FF	File Not Found.
53	AO	Already Open.
54	EF	Input Past End of File.
55	NM	Bad file name.
56	DS	Direct Statement in File.
57	FL	Undefined error.
58	CF	File Not Open.
59-255	UE	Undefined Error.

RADIO SHACK, A DIVISION OF TANDY CORPORATION

U.S.A.: FORT WORTH, TEXAS 76102
CANADA: BARRIE, ONTARIO L4M 4W5

TANDY CORPORATION

AUSTRALIA	BELGIUM	U. K.
91 KURRAJONG ROAD MOUNT DRUITT, N.S.W. 2770	PARC INDUSTRIEL DE NANINNE 5140 NANINNE	BILSTON ROAD WEDNESBURY WEST MIDLANDS WS10 7JN

TRS-80®
MODEL 100
PORTABLE COMPUTER

Quick Reference Guide

CUSTOM MANUFACTURED FOR RADIO SHACK, A DIVISION OF TANDY CORPORATION