TI-99/4 Extended BASIC Reference Card

A handy guide to the commands, statements, and functions of TI Extended BASIC for the TI-99/4 Home Computer. For a full discussion of these and other features, see the TI Extended BASIC manual.

Printed in U.S.A. 1043602-1

C: COMMAND **F:** FUNCTION **S:** STATEMENT

ABS(*numeric expression*)
returns the absolute value of *numeric-expression*. **F**

ACCEPT [:][AT(*row,column*)] [VALIDATE (*datatype ....*)] [BEEP,][ERASE ALL] [SIZE(*numeric-expression*)] [:] *variable*
suspends program execution until data is entered from the keyboard. Optionally, data is entered at the position specified by *row* and *column*, the data is validated, and/or option(s) are executed. **S,C**
VALIDATE *datatypes*:
UALPHA permits all uppercase alphabetic characters.
DIGIT permits 0 through 9.
NUMERIC permits 0 through 9, ".", "+", "−", and "E".
String-expression permits the characters contained in *string-expression*.
BEEP: causes an audible tone.
ERASE ALL: places the space character in all screen positions before accepting input.
SIZE(*numeric-expression*): allows only *numeric-expression* characters to be entered. If *numeric-expression* is positive, that many positions are blanked. If it is negative, no positions are blanked.

ASC(*string-expression*)
returns the ASCII code of the first character of *string-expression*. **F**

ATN(*numeric-expression*)
returns the trigonometric arctangent of *numeric-expression*. **F**

BREAK [*line-number-list*]
causes program to halt when encountered or optionally when lines in *line-number-list* are encountered. **C,S**

BYE
closes open files and leaves TI Extended BASIC. **C**

CALL *subprogram* [(*parameter-list*)]
calls the indicated *subprogram*. An optional *parameter-list* can be passed. **S**

CALL CHAR(*character-code,pattern-identifier* [,...])
defines the specified ASCII *character code(s)* using a 0 through 64 character hexadecimal coded string *pattern-identifier*. **S,C**

CALL CHARPAT(*character-code,string-variable* [,...])
returns in *string-variable* the hexadecimal code that specifies the pattern of *character-code*. **S,C**

CALL CHARSET
restores the standard character patterns and colors for characters 32 through 95. **S,C**

CHR$(*numeric-expression*)
returns the string character corresponding to the ASCII *numeric-expression*. **F**

CALL CLEAR
places the space character in all screen positions. **S,C**

CLOSE *#file-number* [:DELETE]
stops the program's use of the file referenced by *#file-number* and optionally deletes the file. **S,C**

CALL COINC(*#sprite-number,#sprite-number,tolerance, numeric-variable*)
CALL COINC(*#sprite-number,dot-row,dot-column, tolerance,numeric-variable*)
CALL COINC(ALL,*numeric-variable*)
returns in *numeric-variable* −1 if there is a coincidence and 0 if there is no coincidence. If ALL is present, a coincidence of any two sprites is reported. If two sprites are identified by number, their coincidence is reported. If a sprite and a position are identified, their coincidence is reported. Except when ALL is specified, a *tolerance* of from 0 to 255 is specified. The distance between the given sprites or sprite and position must be less than *tolerance* for a coincidence to be reported. **S,C**

CALL COLOR(*#sprite-number,foreground-color* [,...])
CALL COLOR(*character-set,foreground-color,background-color* [,...])
specifies either a *color* for *#sprite-number* or a *foreground-color* and *background-color* for characters in *character-set*. **S,C**

CONTINUE
CON
resumes execution after a break. **C**

COS(*radian-expression*)
returns the trigonometric cosine of *radian-expression*. **F**

DATA *data-list*
stores numeric and string constant data in program. **S**

DEF *function-name* [(*parameter*)] = *expression*
associates user-defined numeric or string *expression* with *function-name*. **S**

DELETE *device-filename*
removes *filename* from *device*. **C,S**

CALL DELSPRITE(*#sprite-number* [,...])
CALL DELSPRITE(ALL)
removes the specified sprite(s) from the screen. ALL removes all sprites from the screen. **S,C**

DIM *array-name*(*integer1* [,*integer2*] ... [,*integer7*]) [,...]
dimensions the listed array(s) as specified. **S,C**

DISPLAY [AT(*row,column*)] [BEEP] [ERASE ALL] [SIZE(*numeric-expression*)] :] *variable-list*
transfers *variable-list* to the display screen. Optionally, data is displayed at the position specified by *row* and *column*. **S,C**
Options:
BEEP: causes an audible tone.
ERASE ALL: places the space character in all screen positions before displaying.
SIZE *numeric-expression*): blanks *numeric-expression* characters in the indicated position.

DISPLAY [*option-list*:] USING *string-expression* [: *variable-list*]
DISPLAY [*option-list*:] USING *line-number* [: *variable-list*]
has the same options as DISPLAY with the addition of the USING clause, which specifies the format. If *string-expression* is present, it defines the format. If *line-number* is present, it refers to the line number of IMAGE statement. See IMAGE. **S,C**

CALL DISTANCE(*#sprite-number,#sprite-number, numeric-variable*)
CALL DISTANCE(*#sprite-number,dot-row,dot-column, numeric-variable*)
returns in *numeric-variable* the square of the distance between the sprites or the sprite and the location. **S,C**

END
terminates program execution. **S**

EOF(*file-number*)
returns the end-of-file condition of *file-number*. **F**
0: not end-of-file.
1: logical end-of-file.
– 1: physical end-of-file.

CALL ERR(*error-code,error-type* [*,error-severity,line-number*])
returns the *error-code* and *error-type* of the most recent uncleared error. Optionally, returns the *error-severity* and *line-number* in which the error occurred. **S,C**
Error-code: consult manual.
Error-type: Negative number: execution error. Positive number: number of file in which the error occurred.
Error-severity: 9, indicating that the error is not recoverable.

EXP(*numeric-expression*)
returns exponential value (e^X) of *numeric-expression*. The value of e is 2.718281828. **F**

FOR *control-variable* = *initial-value* TO *limit* [STEP *increment*]
repeats execution of statements between FOR and NEXT until the *control-variable* exceeds the *limit*. STEP *increment* default is one. **S,C**

CALL GCHAR(*row,column,numeric-variable*)
returns in *numeric-variable* the ASCII code of the character located at *row* and *column*. **S,C**

GOSUB *line-number*
GO SUB *line-number*
transfers control to a subroutine at *line-number*. **S**

GOTO *line-number*
GO TO *line-number*
unconditionally transfers control to *line-number*. **S**

CALL HCHAR(*row,column,character-code* [*,repetition*])
places the ASCII pattern of *character-code* at *row* and *column* and optionally repeats it *repetition* times horizontally. **S,C**

IF *relational-expression* THEN *line-number1* [ELSE *line-number2*]
IF *relational-expression* THEN *statement1* [ELSE *statement2*]
IF *numeric-expression* THEN *line-number1* [ELSE *line-number2*]
IF *numeric-expression* THEN *statement1* [ELSE *statement2*]
transfers control to *line-number1* or performs *statement1* if *relational-expression* is true or *numeric-expression* is not equal to zero. Otherwise control passes to the next statement, or optionally to *line-number2* or *statement2*. **S**

IMAGE *format-string*
specifies the format in which data is PRINTed or DISPLAYed when the USING clause is present. *Format-string* may be any or all of the following:
Letters, numbers, characters not listed below: transferred directly.
#: replaced by the *print-list* values given in PRINT or DISPLAY.
^: replaced by the E and power numbers. Must be four or five of these. **S**

CALL INIT
prepares the computer to load and run assembly language subprograms. **S,C**

INPUT [*input-prompt*:] *variable-list*
suspends program execution until data is entered from the keyboard. The optional *input-prompt* may indicate what data is expected. **S**

INPUT *#file-number* [,REC *record-number*] :*variable-list*
assigns data from the indicated file to the variables in *variable-list*. Records are read sequentially unless the optional REC clause is used. **S**

INT(*numeric-expression*)
returns the greatest integer less than or equal to *numeric-expression*. **F**

CALL JOYST(*key-unit,x-return,y-return*)
accepts data into *x-return* and *y-return* based on the position of the joystick labeled *key-unit*. Values are −4, 0, and 4. **S,C**

CALL KEY(*key-unit,return-variable,status-variable*)
assigns the code of the key pressed on *key-unit* (0 to 5) to *return-variable*. Status information is returned in *status-variable*. 1 means a new key was pressed. −1 means the same key was pressed. 0 means no key was pressed. **S,C**

LEN(*string-expression*)
returns the number of characters in *string-expression*. **F**

[LET] *numeric-variable* [,*numeric-variable*, ...] =*numeric-expression*
[LET] *string-variable* [,*string-variable*, ...] =*string-expression*
assigns the value of an expression to the specified variable(s). **S,C**

CALL LINK(*subprogram-name* [,*argument-list*])
passes control to an assembly language subprogram. **S,C**

LINPUT [[*#file-number*] [,REC *record-number*] :] *string-variable*
LINPUT [*input-prompt*:] *string-variable*
assigns data from the indicated file to *string-variable* or suspends program execution until data is entered from the keyboard. If data is assigned from a file, records are read sequentially unless the optional REC clause is used. If data is entered from the keyboard, the optional *input-prompt* may indicate what data is expected. **S**

LIST ["*device-name*":] [*line-number*]
LIST ["*device-name*":] [*start-line-number*] − [*end-line-number*]
sequentially displays program statements or optionally a single line number or all lines between specified line numbers. **C**

CALL LOAD ("*access-name*" [,*address,byte1* [,...] ,*file-field*, ...])
loads an assembly language subprogram. **S,C**

CALL LOCATE(*#sprite-number,dot-row,dot-column* [,...])
moves the given sprite(s) to the given *dot-row(s)* and *dot-column(s)*. **S,C**

LOG(*numeric-expression*)
returns the natural logarithm of *numeric-expression*. **F**

CALL MAGNIFY(*magnification-factor*)
sets the size and magnification of all sprites. **S,C**
Magnification-factors:
1: single size unmagnified.
2: single size magnified.
3: double size unmagnified.
4: double size magnified.

MAX(*numeric-expression1,numeric-expression2*)
returns the larger of *numeric-expression1* and *numeric-expression2*. **F**

MERGE ["] *device-filename* ["]
merges lines in *filename* from the given *device* into the program lines already in the computer's memory. **C**

MIN(*numeric-expression1*,*numeric-expression2*)
returns the smaller of *numeric-expression1* and *numeric-expression2*. **F**

CALL MOTION(*#sprite-number,row-velocity, column-velocity* [,...])
changes the motion of a sprite(s) to the indicated *row-velocity* and *column-velocity*. **S,C**

NEW
clears the memory and screen and prepares for a new program. **C**

NEXT *control-variable*
See FOR statement. **S,C**

NUMBER [*initial-line*] [,*increment*]
NUM [*initial-line*] [,*increment*]
generates sequenced line numbers starting at 100 in increments of 10. Optionally, you may specify the *initial-line* and/or *increment*. **C**

OLD ["] *device-program-name* ["]
loads *program-name* from *device* into memory. **C**

ON BREAK STOP
ON BREAK NEXT
determines the action taken if a breakpoint is encountered either in the program or by **SHIFT C** (CLEAR). The default is STOP, which halts execution of the program. The keyword NEXT causes breakpoints to be ignored and execution of the program to continue. **S**

ON ERROR STOP
ON ERROR *line-number*
determines the action taken if an error occurs. The default is STOP, which halts execution of the program. If *line-number* is given, control is transferred to it when an error occurs. See RETURN. **S**

ON *numeric-expression* **GOSUB** *line-number* [,...]
ON *numeric-expression* **GO SUB** *line-number* [,...]
transfers control to the subroutine with a beginning line number in the position corresponding to the value of *numeric-expression*. **S**

ON *numeric-expression* **GOTO** *line-number* [,...]
ON *numeric-expression* **GO TO** *line-number* [,...]
unconditionally transfers control to the line number in the position corresponding to the value of *numeric-expression*. **S**

ON WARNING PRINT
ON WARNING STOP
ON WARNING NEXT
determines the action taken if a warning condition occurs. The default is PRINT, which prints a message and continues with the program. The keyword STOP causes the warning message to be printed and execution of the program to stop. The keyword NEXT causes no message to be printed and the program to continue. **S**

OPEN *#file-number:"device-filename"* [,*file-organization*] [,*file-type*] [,*open-mode*] [,*record-type*]
enables the program to use the given *filename*. **S,C**
File-number: 0-255
File-organization: RELATIVE or SEQUENTIAL.
File-type: DISPLAY or INTERNAL.
Open-mode: INPUT, OUTPUT, UPDATE, or APPEND.
Record-type: FIXED or VARIABLE.

OPTION BASE 0
OPTION BASE 1
sets the lowest allowable subscript of arrays to zero or one. The default is zero. **S**

CALL PATTERN(*#sprite-number,character-value* [,...])
changes the pattern number of the specified sprite(s) to the specified *character-value(s)*. **S,C**

CALL PEEK(*address,numeric-variable-list*)
returns values in *numeric-variable-list* corresponding to the values in *address*. **S,C**

PI
returns the value of pi as 3.14159265359. **F**

POS(*string1,string2,numeric-expression*)
returns the position of the first occurance of *string2* in *string1*. Search begins at the position specified by *numeric-expression*. Returns zero if no match is found. **F**

CALL POSITION(*#sprite-number,dot-row,dot-column* [,...])
returns the positions in the given *dot-row(s)* and *dot-column(s)* of the specified sprite(s). **S,C**

PRINT [*#file-number* [,REC *record-number*] :] [*print-list*]
transfers optional *print-list* to the display screen or optionally to an external file. The REC clause directs *print-list* to the specified *record-number*. **S,C**

PRINT [*#file-number* [,REC *record-number*]] **USING** *string-expression:print-list*
PRINT [*#file-number* [,REC *record-number*]] **USING** *line-number:print-list*
acts the same as PRINT with the addition of the USING clause, which specifies the format. If *string-expression* is present, it defines the format. If *line-number* is present, it refers to the line number of an IMAGE statement. See IMAGE. **S,C**

RANDOMIZE [*numeric-expression*]
resets the random number generator to an unpredictable sequence. With optional *numeric-expression*, the sequence is repeatable. **S,C**

READ *variable-list*
assigns numeric and string constants from DATA statements to *variable-list*. **S,C**

REC(*file-number*)
returns the current record position in *file-number*. **F**

REM *character-string*
indicates internal program documentation with no effect on program execution. **S,C**

RESEQUENCE [*initial-line*] [,*increment*]
RES [*initial-line*] [,*increment*]
automatically renumbers lines starting at 100 in increments of 10. Optionally, you may specify the *initial-line* and/or *increment*. **C**

RESTORE [*line-number*]
indicates that the next READ operation will take data from the first DATA statement in the program or, optionally, from the first DATA statement after *line-number*. **S,C**

RESTORE *#file-number* [,REC *record-number*]
resets file pointer to the beginning of the file or, optionally, to *record-number*. **S,C**

RETURN
transfers program control from a subroutine to the statement following the corresponding GOSUB or ON...GOSUB statement. **S**

RETURN [*line-number*]
RETURN [NEXT]
controls program action after an error has occurred when an ON ERROR statement has been executed. With nothing following it, returns control to the statement which caused the error and executes it again. Followed by a *line-number*, it transfers control to the given line. Followed by NEXT, it transfers control to the statement after the one in which the error occurred. **S**

RND
generates a pseudo-random number greater than or equal to zero and less than one. **F**

RPT$(*string-expression,numeric-expression*)
returns *numeric-expression* occurances of *string-expression* concatenated together. **F**

RUN ["*device.program-name*"]
RUN [*line-number*]
starts program execution at the lowest program statement of the program currently in memory. Optionally *program-name* is loaded from *device* or execution starts at *line-number*. **C,S**

SAVE *device-program-name* [,PROTECTED]
SAVE *device-program-name* [,MERGE]
places a copy of the current program in *device* as *program-name*. PROTECTED makes it impossible to change or list the program later. MERGE enables later merging of the program with another program. See MERGE. **C**

CALL SAY(*word-string* [,*direct-string*] [,...])
causes the speech synthesizer to speak the given *word-string* or *direct-string*. **S,C**

CALL SCREEN(*color-code*)
changes the screen color to *color-code*. **S,C**

SEG$(*string-expression,position,length*)
returns a substring of *string-expression* beginning at *position* and extending for *length* characters. **F**

SGN(*numeric-expression*)
returns 1 if *numeric-expression* is positive, 0 if it is zero, and −1 if it is negative. **F**

SIN(*radian-expression*)
returns the trigonometric sine of *radian-expression*. **F**

SIZE
displays on the screen the number of unused bytes of memory. **C**

CALL SOUND(*duration,frequency1,volume1* [,...,*frequency4,volume4*])
controls up to three tone and one noise generators. Tone and noise parameters can occur in any order. Negative duration causes immediate sound update. **S,C**
Duration: 1 through 4250 ms., −4250 through −1 ms.
Frequency: 110 through 44733 Hz for tone, −1 through −8 for noise.
Volume: 0 (loudest) through 30 (softest).

CALL SPGET(*word-string,return-string*)
returns in *return-string* the speech bit pattern that corrresponds to *word-string*. **S,C**

CALL SPRITE(#*sprite-number,character-value,sprite-color,dot-row,dot-column*, [,*row-velocity, column-velocity*] [,...])
specifies the existence of sprite(s) *sprite-number* with a pattern specified by *character-value*, a color of *sprite-color*, a screen position of *dot-row* and *dot-column*, and optionally a velocity of *row-velocity* and *column-velocity*. **S,C**

SQR(*numeric-expression*)
returns the square root of *numeric-expression*. **F**

STOP
terminates program execution. **S,C**

STR$(*numeric-expression*)
converts the value of *numeric-expression* to a string. **F**

SUB *subprogram-name* [(*parameter-list*)]
indicates the beginning of *subprogram-name* with optional *parameter-list*. **S**

SUBEND
indicates the end of a subprogram and transfers program control from a subprogram to the statement following the CALL statement. **S**

SUBEXIT
transfers program control from a subprogram to the statement following the CALL statement. **S**

TAB(*numeric-expression*)
controls column position of the output from a PRINT or DISPLAY statement. **F**

TAN(*radian-expression*)
returns the trigonometric tangent of *radian-expression*. **F**

TRACE
lists line numbers of lines before each is executed. **C,S**

UNBREAK [*line-list*]
removes all breakpoints or optionally those in *line-list*. **C,S**

UNTRACE
cancels the action of the TRACE command. **C,S**

VAL(*string-expression*)
converts *string-expression* into a numeric constant. **F**

CALL VCHAR(*row,column,character-code* [,*repetition*])
places the ASCII representation of *character-code* at *row* and *column* and optionally repeats it *repetition* times vertically. **S,C**

CALL VERSION(*numeric-variable*)
returns a value indicating the version of BASIC being used. TI Extended BASIC returns a value of 100. **S,C**

Character Sets

Set	*ASCII Codes*	*Set*	*ASCII Codes*
0	30-31		
1	32-39	8	88-95
2	40-47	9	96-103
3	48-55	10	104-111
4	56-63	11	112-119
5	64-71	12	120-127
6	72-79	13	128-135
7	80-87	14	136-143

Color Codes

Color	*Code*	*Color*	*Code*
Transparent	1	Medium Red	9
Black	2	Light Red	10
Medium Green	3	Dark Yellow	11
Light Green	4	Light Yellow	12
Dark Blue	5	Dark Green	13
Light Blue	6	Magenta	14
Dark Red	7	Gray	15
Cyan	8	White	16

ASCII Code	*Character*	*ASCII Code*	*Character*
30	(cursor)	63	? (question mark)
31	(edge character)	64	@ (at sign)
32	(space)	65	A
33	! (exclamation point)	66	B
34	" (quote)	67	C
35	# (number or pound sign)	68	D
36	$ (dollar)	69	E
37	% (percent)	70	F
38	& (ampersand)	71	G
39	' (apostrophe)	72	H
40	((open parenthesis)	73	I
41	) (close parenthesis)	74	J
42	* (asterisk)	75	K
43	+ (plus)	76	L
44	, (comma)	77	M
45	- (minus)	78	N
46	. (period)	79	O
47	/ (slash)	80	P
48	0	81	Q
49	1	82	R
50	2	83	S
51	3	84	T
52	4	85	U
53	5	86	V
54	6	87	W
55	7	88	X
56	8	89	Y
57	9	90	Z
58	: (colon)	91	[(open bracket)
59	; (semicolon)	92	\ (reverse slash)
60	< (less than)	93	] (close bracket)
61	= (equals)	94	^ (exponentiation)
62	> (greater than)	95	_ (underline)

The following key presses may also be detected by CALL KEY.

1	SHIFT A	3	SHIFT F
4	SHIFT G	6	SHIFT R
7	SHIFT T	8	SHIFT S
9	SHIFT D	10	SHIFT X
11	SHIFT E	12	SHIFT V
13	ENTER	14	SHIFT W
15	SHIFT Z		