Coco 2 Expanded BASIC Utilities Quick Reference

Tandy Color Computer 2 Expanded BASIC
Utility Disk Package
by
Tino Debourgo
1984

Quick Reference created by HLO

For more details on any command refer to the software owners manual.

There are three different full packages of utilities plus one extra add-on package of utilities.
1.) Graphics package.
2.) Edit Package.
3.) Q screen package.
You can only run one of these packages at a time.
4.) Extra commands.
These extra commands can be run concurrently with 1-3.

To begin all utilities packages:
RUN “R”
if you have a Color Computer 2.
All utilities require version 1.0 of Extended BASIC to run. This program downgrades to version 1.0 on the Coco 2 from version 1.1. If you have a Coco 1 with version 1.0 you won’t need to run this utility.

1.) Graphics Package

This package adds graphic, sound and expanded programming commands to BASIC.

RUN “G”
This starts the graphics utilities package and sets the computer into the graphics mode screen.
Be sure to execute RUN “R” first if you are using a Coco 2 with version 1.1 of Extended BASIC.

TEXTON
Back to standard text mode

TEXTOFF
To Expanded BASIC graphics mode

Expanded Graphic and Sound Commands

Commands in RED must be executed with TEXTOFF in the graphics mode.
Commands in BLACK can be executed with TEXTON or TEXTOFF.

Screen Text Sizes
Specified by PMODE
PMODE 0 & 1 Text size is 16 x 8
PMODE 2 & 3 Text size is 16 x 16

PMODE 4 Text size is 32 x 16
- for 51 x 24 size characters in PMODE4 only, execute ***SIZE(51X24)** with the Expanded BASIC disk in the main drive
- return back to 32x16 execute ***SIZE(32X16)** also with the Expanded BASIC disk in the main drive

BORDER (X1,Y1)-(X2,Y2),edge character
This creates a border with the upper left being X1,Y1 and lower right being X2,Y2 with a border character.

REV
Reverse color of characters on screen

Redefining Characters

CHR$(char#)=d1,d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,D12

This redefines a character into a new shape. Character # 32-255 can be redefined.
d1-d12 is the hexadecimal representation of the 8 bit line in the character starting from top to bottom.
When using 51x24 characters there is 8 lines of definition.

Scrolling Characters

SCR(X1,Y1)-(X2,Y2),fill character, direction

This creates scrolling text with in the window created by (X1,Y1)-(X2,Y2) filled in with fill character in the direction specified.
(X1,Y1) is the X,Y of the upper left of the window.
(X2,Y2) is the X,Y of the bottom right of the window.
Note: the window edges are not seen. When the text scrolls beyond the edges of the window the text disappears.
Fill Character is the character that will fill in where the text was. 0 gives no fill character.
Direction is the direction of the scroll. U=up, D=down, R=right & L=Left.

Sound Commands

ENVELOPE envelope# : pitch increment, #of increments

Creates a custom wave form

BEEP envelope#, starting volume, repeat of beeps

Plays custom wave form

BASIC programming language expanded commands

REPEAT….. UNTIL <condition>

Will repeat everything inside loop till condition is meet
Ex:
10 REPEAT
20 commands
30 UNTIL <condition>

IF...THEN...ELSE...ENDIF
Enables multi-line IF statement
Ex:
10 IF <condition> THEN
20 commands
30 ELSE
40 commands
50 ENDIF

GOTO variable, GOSUB variable
Variable names for line addresses can set in GOTO & GOSUB

Memory management

FILL begin address ,end address, character
Fills a region of memory starting with begin address to end address with character

MCOPY copying address, receiving address, number of bytes
Copies a region of memory from copying address to receiving address for number of bytes

Procedures
Named Procedures can be created and called within a program

DEFPROC,name> (variable list)
Defines a Procedure
Ex:
100 DEFPROC <name> (transfer variable 1, transfer variable 2,...)
110 commands
120 ENDPROC

PROC<name>
Calls a Procedure
Ex:
10 PROC <name> (transfer variable 1, transfer variable 2,...)

Error management

ONERROR:GOTO <line number>
When an error occurs execution will be transferred to line number after GOTO

PRINTERR
Prints line# of last error to screen.

ERRORER
Print error to screen and stops program. Use with ONERROR.

CONTOFF
Disables Break Key

CONTON
re-enables Break Key

CONTERROR
produces an error on Break

2.) Editing Package

This package adds editing features to BASIC.

RUN “E”

This starts the editing utilities package.
Be sure to execute RUN “R” first if you are using a Coco 2 with version 1.1 of Extended BASIC.

Copying characters

Pressing the <CLEAR> key and the up arrow puts you in copy mode and a 2nd inverse cursor appears. use the arrow keys while pressing the <CLEAR> to move the 2nd cursor over a character you wish to copy. Press the <CLEAR> <@> to copy the character to your new line. Press <CLEAR> and down arrow to exit.

Defining <CLEAR>+<number key> with a function

KEY0 “function” through **KEY9 “function”** defines number keys 0 through 9, respectively, with function such as “LIST”.

Auto key repeat

KEYSCR delay before repeat, time between repeat
sets the delay of key repeat and then the time between repeat.
KEYSCR 255,255 turns it off.

Auto line numbering

AUTO beginning line number, increment begins auto line number starting at beginning line number for increment. Press <BREAK> to end auto line numbering.

Adjusting printer width

WIDTH row width will define printer printout with a width of row width.

3.) Q-Graphics Package

This package adds semi-graphics commands to BASIC.

RUN “Q”
This starts the Q-graphics utilities package and sets the computer into the graphics mode screen.
Be sure to execute RUN “R” first if you are using a Coco 2 with version 1.1 of Extended BASIC.

PMODE 4,1 is required for all Q graphics.

QON Turns on Q graphics.
QOFF Turns off Q graphics and returns to text mode.

QCLS (even color, odd color) This will clear the screen and make the even horizontal lines the first value and the odd lines the second value. QCLS(0,0) for solid black.

QSET(x,y,color) This will put a point on the screen in the color specified. The pixel will be 4x1 and all the colors are available.

QRESET(x,y) This will put black point on the screen.

QPRINT @coordinates, “text” This print text on the Qscreen in the standard green @ text coordinates.

QSCR(X1,Y1)-(X2,Y2),fill lines 1,fill lines 2 , direction
This creates scrolling text with in the window created by (X1,Y1)-(X2,Y2) filled in with fill lines in the direction specified.
(X1,Y1) is the X,Y of the upper left of the window.
(X2,Y2) is the X,Y of the bottom right of the window.
Note: the window edges are not seen. When the text scrolls beyond the edges of the window the text disappears.
Fill lines 1 & 2 are the horizontal or vertical lines (depending on direction) that will fill in where the text was. 0,0 gives no fill.
Direction is the direction of the scroll. U=up, D=down, R=right & L=Left.

NOTE: Don’t use a variable that starts with a Q. This will confuse the program.
STOP has been renamed to QUIT.

4.) Extra Commands Package

This package adds two extra commands into the other three packages.

From within one of the other three packages enter ***EXTRA.**
This loads in the extra commands. Note: you will have 1.25k less with the EXTRA package loaded.

Local variables

You can localize variables in your procedures.

.DELLOC forget any local variables currently stored. Always put this at the beginning of a program to clear any previous variables.

DIM [ ] (max local numeric) and **DIM [ ]$ (max local string).** This sets aside memory space for local variables. Maximum is 255 numeric and 255 string variables. Space must be set aside in the main program before local variables can be defined.

.LOC variable defines the local variable within the procedure. All local variables must be defined before use.

.ENDPROC This releases the local variable from the procedure. If this version of ENDPROC is not used the variable will continue to be used even in the main program.

Strings as commands

.variable$ Certain commands can be used within a program in string form.
EX: A$=”RUN” : .A$ .A$ will now execute RUN when .A$ comes across in a program.

.+”” When adding commands to an existing string .+”” must be added to the command.
EX: .+””LEFT$(A$,3) when using with LEFT$...

These commands can not be used as strings:
IF..THEN..ELSE multi-line command
FOR..NEXT...STEP
REPEAT
UNTIL
GOTO
GOSUB
? as PRINT
‘ as REM