APPLE ProDOS APPENDIX

Notes

APPLE ProDOS APPENDIX

ProDOS 8 Version

For the Apple II+, //e, //c, IIGS and Laser 128™

by
Greg Branche
Original DOS 3.3 version by Dave Overton

© Copyright 1985, 1986, 1987

ZEDCOR, INC.

All Rights Reserved

ZBasic is a trademark of Zedcor, Inc.

Apple, IIGS, Imagewriter, and ProDOS are registered or licensed trademarks of Apple Computer, Inc.
Zedcor, Inc. is not affiliated with Apple Computer, Inc.

APPLE ProDOS APPENDIX

TABLE OF CONTENTS

TABLE OF CONTENTS D3

HARDWARE REQUIREMENTS D5
- 64K Version
- 128K Version

FILES INCLUDED ON ZBASIC DISKETTE D6
- 64K Versions D6
- 128K versions D7

GETTING STARTED D8

NOTES ON THE ProDOS VERSION D9
- Boot-up Process D9
- Note to main reference section D9
- Importance of using a RamDisk D10
- ProDOS PATHNAMES D10
- File BUFFER size (how to get an extra 2048 bytes) D10
- List Keys D11
- Help File D11
- Reset Key D11
- ProDOS Disk Error Codes D11
- Hexadecimal Constants D12
- Relative Coordinates versus Pixel Coordinates D12
- MOUSE D12
- IMPORTANT NOTES about video/system problems D13
- Using the Super Serial Card D13
- Commands not supported in this version D13
- Integration of Text and Graphics D14
- 80 Column Card Control Codes D14
- INVERSE text D14
- Mouse Text Characters D15
- Custom Character Sets D15

SPECIAL "CONFIGURE" OPTIONS D16
- "Printer Slot 1-7" D16
- Setting up a Printer initialization sequences D16
- LOCATE order D17
- CONFIG D17
- Note on Case Conversion D17

APPLE ProDOS APPENDIX

TECHNICAL NOTES D18
- ProDOS Machine Language Interface D18
- Entry Points D18
- ZBasic to ProDOS interface D18
- Using MACHLG D19
- MEMORY Usage D19
- Zero Page Memory Map D19
- 64K Memory Map D20
- 128K Memory Map D21

CONVERTING Applesoft PROGRAMS TO ZBasic D22
- Applesoft commands and ZBasic Equivalents D23
- Applesoft file commands and ZBasic Equivalents D24
- Converting DOS 3.3 ZBasic programs to ProDOS D25

REFERENCE D26
- CLS command D27
- COLOR statement D28
- DATE$, TIME$ functions D29
- DEF LPRINT statement D30
- DEF MOUSE statement D31
- DIR command D32
- EDITOR command D34
- INSLOT statement D35
- MEM command D36
- MODE statement D37
- ONLINE command D38
- OUTSLOT statement D39
- PATH command D40
- POINT function D41
- RENAME command D42
- RUN command D43
- USR function D44
- USR5 function D45

FULL SCREEN EDITOR D46
- Difference between the Full Screen E ditor and the Standard Line Editor D46
- 80 Column Editor D47
- 80 column cursor movement DIAGRAM D47
- 40 column Editor D48
- 40 column cursor movement DIAGRAM D48
- Full Screen Editor Quick Reference Page D49
- Cursor key definitions (40 and 80 column) D50
- Editor Command definitions D51

APPLE ProDOS APPENDIX

HARDWARE REQUIREMENTS

Apple //c, IIGS and Laser 128

The 64K and 128K ProDOS versions of ZBasic function with a standard Apple //c and IIGS. A disk drive is required (5.25 or 3.5 inch). ProDOS provides a /RAM disk, size depending on available memory. An Apple Mousetm with interface, Joystick and Super Serial Card are supported but are not required.

IIGS Note: The IIGS emulates the //e, //c modes with this version of ZBasic. Super High-Res graphics are not supported on this version directly.

Apple //e

64K Version

The 64K ProDOS version runs with a standard Apple //e. A disk drive is required (5.25 or 3.5 inch). An AppleMousetm with interface, Joystick and Super Serial Card are supported but are not required.

If you have an Extended 80 column card, or other memory board, ProDOS provides a /RAM disk, size depending on additional memory. If you have only 64K there will be more disk accesses and compilation will take longer.

//c,//e,//GS Note: Code can be generated which will run on the older Apple][+ or][if certain restrictions are observed; Avoid MODE 3 or 7 as they require an extended 80 column card which will not function in an Apple][+.

128K Version

The 128K version of ZBasic requires an Extended 80 Column Card and a 65C02 or 65802 microprocessor.

Apple][or Apple][+

64K Version

If you have an Apple][or][+, you MUST have a 16K bank-switched memory card installed (giving you at least 64K memory). If you have a ProDOS compatible memory board that allows ProDOS to create a /RAM disk, ZBasic will take advantage of it. If you have only 64K there will be more disk accesses and compilation will take longer.

A disk drive is required (5.25 or 3.5 inch). ZBasic requires a minimum of 64K memory to create and execute programs. An AppleMousetm with interface, Joystick and Super Serial Card are supported but are not required.

128K Version

The 128K ProDOS version of ZBasic will not operate on an Apple][or][+.

OLDER 80 COLUMN CARDS

Older style 80 column cards may or may not function.

The Videx 80 column card works in mode 2 although you will have to do some manual switching. When typing CLS from the Standard line editor ZBasic will sense the Videx board and clear the screen automatically.

APPLE ProDOS APPENDIX

FILES INCLUDED ON THE MASTER DISKETTE
64K VERSION

The following files are included with the 64K ProDOS version of ZBasic.

File	Description
ZBASIC.SYSTEM	The boot program and low-memory subroutines.
RUNTIME.OBJ	High-memory runtime subroutines. This program MUST accompany stand-alone programs you create with ZBasic.
EDITOR.OBJ	The ZBasic command environment and Standard line editor.
COMPILER.OBJ	The ZBasic compiler.
FSEDIT.80.OBJ	80-column full screen editor for //c, IIGS and 80-col //e. May be deleted if not used.
FSEDIT.40.OBJ	40-column full screen editor for 40-col //e,][and][+. May be deleted if not used.
INIT.64.OBJ	Contains a stand-alone program initialization sequence.

THE FILE ABOVE ARE REQUIRED WHEN CREATING ZBASIC PROGRAMS.

THE FILES BELOW ARE OPTIONAL OR EXAMPLE PROGRAMS

ZBASIC.HLP	Help file accessed with the "HELP" command.
DISKIO.BAS	Sample program demonstrating ZBasic file commands.
GRAPH.BAS	Sample program demonstrating ZBasic graphics.
SORT.BAS	Program to illustrate the use of the QUICK.APP and SHELL.APP sorting programs. Load this program first then type: APPEND 1000 QUICK.APP (or SHELL.APP).
QUICK.APP	Append file containing a quicksort subroutine.
SHELL.APP	Append file containing a shell sort subroutine.
SIEVE	The SIEVE benchmark program from BYTE magazine.
GRAPHICS.COLORS	Demonstrates the colors available in each of the graphics modes.
BLOAD.SAMPLE	Demonstrates the use of the BLOAD and BSAVE functions.
BSAVE.FN	Function to simulate the ProDOS BASIC.SYSTEM BSAVE command.
BLOAD.FN	Function to simulate the ProDOS BASIC.SYSTEM BLOAD command.
DHRBSAVE.FN	Double Hi-Res BSAVE function saves Double Hi-Res Graphic screen.
DHRBLOAD.FN	Double Hi-Res BLOAD function loads Double Hi-Res Graphic screen.
DRAW.FN	Function to simulate the Applesoft DRAW command.
PREFIX.SAMPLE	Sample program demonstrating the use of the PREFIX function.
PREFIX.FN	Function to set or retrieve the ProDOS default prefix at runtime.
CREATE.FN	Function to create a ProDOS subdirectory from within a ZBasic program.
DATETIME.FN	Function to manually set the date and time.

APPLE ProDOS APPENDIX

FILES INCLUDED ON THE MASTER DISKETTE
128K VERSION

The following files are included on the 128K ProDOS version of ZBasic (flip side of the diskette):

FILE	Description
ZBASIC.SYSTEM	The boot program and low-memory subroutines.

*The following **three** MUST accompany stand-alone programs you create with ZBasic.*

RT.MAIN.OBJ1	High-memory runtime subroutines.
RT.AUX.OBJ0	Low auxiliary memory routines.
RT.AUX.OBJ1	High auxiliary memory routines.

EDITOR.OBJ0 EDITOR.OBJ1 EDITOR.OBJ2	The ZBasic command environment and Standard line editor and Full Screen Editor.

COMPILER.OBJ0 COMPILER.OBJ1	The ZBasic compiler.

INIT.128.OBJ	128K Stand-alone program initialization sequence.

Use the example programs on the 64K side of the diskette (see previous page for details). There will also be a couple examples on this side of the diskette. These programs will not work with the 64K version.

APPLE ProDOS APPENDIX

GETTING STARTED

1. Make a BACKUP of your master ZBasic diskette. Store the master in a safe place (refer to the ProDOS reference manual for backup methods).

   Note: There are two versions of ZBasic for ProDOS; a 64K version and a 128K version. On 5.25" diskettes they occupy opposite sides. On 3.5" diskettes they are in two different sub-directories. If using 5.25" diskettes make sure to backup both sides.

2. Due to storage limitations, the ZBasic disk does not contain ProDOS operating system. Therefore you must create a ProDOS environment. There are a couple of ways to do this:

   a. **BOOT FROM A ProDOS Master Disk** (/USERS.DISK). Then type "B" from the menu to enter Applesoft BASIC.

   From the prompt(]), enter: "PREFIX /ZBASIC", then:"-ZBASIC.SYSTEM".

   b. **CREATE A ZBASIC BOOT DISK:** Format a blank disk (using the FILER utility). Copy the file "PRODOS" from a ProDOS disk to your freshly formatted disk. Transfer the following files from you ZBasic Master Disk to your new copy:

   64K VERSION

ZBASIC.SYSTEM	EDITOR.OBJ
RUNTIME.OBJ	INIT.64.OBJ
COMPILER.OBJ	

   FSEDIT.80.OBJ (use FSEDIT.40.OBJ if using a 40 col Apple][,][+ or //e)

   128K VERSION

ZBASIC.SYSTEM	COMPILER.OBJ0
RT.MAIN.OBJ1	COMPILER.OBJ1
RT.AUX.OBJ0	INIT.128.OBJ
RT.AUX.OBJ1	EDITOR.OBJ0
EDITOR.OBJ1	EDITOR.OBJ2

   CTRL <OPEN APPLE> RESET will now load and execute ZBasic for this disk.

3. Read this appendix, making notes of any variations.

4. Now read "Getting Started" in the main reference section.

APPLE ProDOS APPENDIX

BOOT-UP PROCESS

When the ZBASIC.SYSTEM program is loaded from ProDOS, it does several things prior to putting you into the editor:

- o ZBasic Title page displayed during the boot process.
- o Zero page locations are initialized.
- o The low-memory runtime module is moved into place.
- o ZBasic looks for a volume with the first four characters "/RAM". If found, it will copy the necessary system files into the ram disk. If you do not wish to have ZBasic use the /RAM disk, simply rename it prior to loading ZBasic.
- o The command environment and standard line editor overlay is loaded into memory (to invoke the full screen editor type EDITOR or EDITOR+).

NOTE TO THE MAIN REFERENCE SECTION

Wherever there are notable differences between the text and the Apple ProDOS version you will see an Apple ICON that will tell you the difference or refer you to the correct section. The icon looks like this:

Occasionally the icon refers to the Apple // DOS 3.3 version. In those instances simply ignore this icon.

THE IMPORTANCE OF USING A RAM DISK

In order to leave as much free memory as possible for program development, there is a lot of overlay swapping and other disk access involved while editing and running a program interactively (like an interpreter).

For example, if you type "PRINT 2.345*32" from the editor command line, quite a number of events take place:

o the editor saves whatever program you have in memory to the disk (/RAM disk, if enabled).
o loads and runs the compiler from disk (/RAM disk, if enabled).
o compiles the command and stores the object code in memory.
o loads and runs the runtime system (/RAM disk, if enabled).
o the runtime executes the object code (which is this example prints 75.04).
o then reloads and executes the editor (/RAM disk, if enabled) and waits for the next command.

Phew! As I said, a lot of disk access! It should be obvious that a /RAM disk will speed up the whole process 10-15 times since disk access is nearly eliminated.

APPLE ProDOS APPENDIX

USING THE RAM DISK

These versions of ZBasic require 64K and 128K of memory, respectively. If your system has more then the minimum amount of memory required, and the extra memory is configured as a ProDOS /RAM disk, ZBasic will use it to store some system files and overlays so that overall program development time will be reduced and system speed will be improved.

In addition, a temporary file used to hold your source code is also saved to disk during the overlay swapping. This file is named ZTEMP.ZBS.

If there is no /RAM volume, the ZBasic disk MUST remain in the drive for normal operation.

If the /RAM disk is not large enough to hold ZTEMP.ZBS, ZBasic returns a DISK FULL error and returns to the editor. You should save the file to a diskette and compile from disk at this point (RUN*) or exit ZBasic, disable the /RAM disk by renaming it from ProDOS, then re-enter ZBasic without rebooting.

Warning: DO NOT RENAME THE /RAM DISK WHEN IN USE!

ProDOS PATHNAMES

The filenames used in ZBasic are standard ProDOS pathnames. ProDOS pathnames can consist of up to 64 characters, including separating slashes. Individual filenames can be up to 15 characters long, and can consist of alphanumeric characters and periods only.

Pathnames may be used with OPEN,RENAME,SAVE,LOAD and all other disk commands and statements. See your ProDOS manual for more information about pathname syntax.

FILE BUFFER SIZE--- OR HOW TO GET AN EXTRA 2048 BYTES

Each file opened by a ZBasic program requires a 1024 byte file buffer. ZBasic defaults to two file buffers (2048 bytes).

If you configure ZBasic for one file buffer, 1024 bytes is freed for program or variables (configuring for no open files would free 2048 bytes).

See "Configure" in main manual.

APPLE ProDOS APPENDIX

LIST KEYS

The following is a list of additional keys which can be used in the command mode editor to list lines of source code (as well as those described in the main manual):

Up Arrow	List previous line
Down Arrow	List next line
Left Arrow	List first line of the file
Right Arrow	List last line of the file

HELP

The file used by the HELP command is named "ZBASIC.HLP". If you so desire, this file can be deleted to allow more storage room on the disk.If ZBasic is not able to find this file in it's system directory, it will look in the user's currently logged directory (see the PATH command). If ZBasic still cannot find the help file, you will get a "File-Not-Found" error.

CONTROL-RESET VERSUS CONTROL-C

This version allows you to use either CTRL-C or CTRL-RESET to exit a running program. If the computer should "lock up" for some reason, or you are faced with the monitor prompt (*), you can press CTRL-RESET to restart the ZBasic editor. Your source program should remain intact. If you press CTRL-RESET while executing a stand alone program, the program will be terminated, and you will be allowed to load and execute another ProDOS system program. If you are faced with the monitor prompt anywhere within the ZBasic system, pressing CTRL-Y will also return you to the editor.

ADDITIONAL DISK ERROR CODES

Error Code	Error Message
9	Position Error
10	No Device Connected Error
11	Disk Switched Error
12	Duplicate Filename Error
13	Incompatible File Format Error
14	Access Error
15	File Already Open Error
16	Directory Structure Damaged Error
17	Not a ProDOS Volume Error
18	Duplicate Volume Online Error
19	File Structure Damaged Error
20	I/O Error
21-255	Disk Error

The actual disk error code will be the filenumber time 256 plus the number above. See disk error in the main reference manual for more information.

APPLE ProDOS APPENDIX

HEXADECIMAL CONSTANT INDICATORS ($ and &)

In addition to the "&" prefix signifying a hexadecimal constant (as in &FF69), the "$" character may also be used (as in $FF69). This is so that Apple users will feel more at home. Remember that if this character is used the program will not be directly transportable to the Apple DOS 3.3, IBM, Macintosh, CP/M, or other versions of ZBasic.

RELATIVE GRAPHIC COORDINATES VERSUS PIXEL COORDINATES

The standard ZBasic graphic coordinate system is great for porting programs between the various computers that run ZBasic or between Hi-Res and double Hi-Res. Occasionally you may need to switch to PIXEL coordinates. Use this statement:

POKE WORD &85, 0

After this statement is executed, the following screen dimensions will be in effect with the different graphics modes:

MODE 1 40 x 40
MODE 3 80 x 40
MODE 5 280 x 192
MODE 7 560 x 192 (not available with the Apple][+ or //e without an extended 80 col. card)

MODE automatically resets to the device independent coordinate system, so you must use the POKE WORD &85, 0 statement immediately after setting MODE to re-enable the pixel coordinates above.

MOUSE

If your program uses the MOUSE function to receive input from the mouse (DEF MOUSE=0), you MUST use a MOUSE(0) function at the beginning of the program prior to any other MOUSE call.

MOUSE(0) forces ZBasic to scan the slots for a mouse interface card, and then initialize the mouse properly. If the mouse is not initialized prior to accessing it, your program may die a horrible death (crash)!

In addition, if a mouse interface could be found and initialized properly, MOUSE(0) will return a value of -1 (true,) otherwise a value of zero (false) will be returned.

APPLE ProDOS APPENDIX

IMPORTANT NOTES ABOUT VIDEO/SYSTEM PROBLEMS

ZBasic allow you to set many different graphics and text modes. This feature lets you jump form one MODE to another as your program requires. This does introduce a unique potential for confusing video problems that are easily mistaken for system errors.

o While programs compiled in the interactive method (RUN) of ZBasic will usually operate correctly even if MODE is not set at the beginning of a program, a program compiled to disk as a stand-alone program (RUN* or RUN+) may appear to "Hang the system" if MODE is not set. To solve this problem; BE SURE TO SET THE MODE AT THE BEGINNING OF EVERY STAND-ALONE PROGRAM. If using an Apple][+ (or //e without an extended 80 column card) be sure to avoid MODE 3 and 7.

o Sometimes when typing programs in the editor, especially after pressing CTRL-C or CTRL-RESET from a running program, you may experience an unresponsive screen or keyboard. Nine times out of ten what has happened here is that the MODE has been changed in the compiled program and needs to be reset in the editor (your keys are actually appearing on an invisible page of another MODE). Just type:

<RETURN> **MODE 2** <RETURN>

Even though you will not see the keys being typed, the screen will return to normal when you're finished typing. Do not REBOOT the system, as you will lose the program in memory. Remember: You can't see the keys being pressed until you press <RETURN>.

o CONTROL KEYS IN LISTINGS: The 80 column card responds to certain control codes. Sometimes a REM or quoted string may contain a control character that may set the 80 column card to 40 characters or to a different mode. Use the example above to correct the setting and delete the control character from the offending line.

USING THE SUPER SERIAL CARD

The file number specified in serial I/O must be the negative slot # in which an Apple Super Serial Card is installed. The Apple IIc has the equivalent of a Super Serial Card installed in slot # 2. This card would be accessed by:

```
OPEN "C",-2,300...
```

ZBasic communication commands only support the Apple Super serial card and compatible serial interfaces.

Note: The IIGS serial port is not yet supported. A Super Serial Card or compatible card or modem will function properly.

COMMANDS NOT SUPPORTED IN THIS VERSION OF ZBASIC

The following two functions are not supported: INP() and OUT(). See the notes at the bottom of the pages in the main reference section for commands that may not be fully compatible.

APPLE ProDOS APPENDIX

INTEGRATION OF TEXT AND GRAPHICS

Unlike Applesoft, ZBasic allows you to integrate text and graphics on the screen.

This permits porting of programs over to the Apple from the IBM PC and many others (MODE 5 only on the][+ and 64K //e. MODE 5 and 7 only on the 128K //e and //c).

80 COLUMN CARD CONTROL CODES

The Apple 80-column text card firmware supports many control codes to perform special operations, such as screen scrolling up and down. These control codes are available in modes 2 and 6. Simply print CHR$(x), where x is the code for the function you want to perform (see the 80-column text card manual for a listing of these codes.)

In addition, several of these codes are available in Modes 5 and 7. The codes and the function they perform in MODES 5 and 7 are listed in the following table:

CHR$ Code	Function
7	Beep the Apple speaker
8	Moves the cursor position one space to the left; from left edge of window, moves to right end of line above
10	Moves cursor position down to the next line, scrolls if necessary
13	Moves cursor position to left end of next line, scrolls if necessary
14	Sets display format normal (white on black)
15	Sets display format inverse (black on white)
22	Scrolls the display down one line, leaving the cursor at the current position
23	Scrolls the display up one line, leaving the cursor at the current position
24	Turns the MouseText off
27	Turns the MouseText on
28	Moves cursor position one space to the right; from right edge of window, moves it to left end of line below

Other Apple screen control codes are not implemented (as control codes) for graphics MODE 5 and 7.

INVERSE TEXT

To shift to the inverse character set, print a CHR$(15). All characters printed after this will be in inverse text.

To switch back to normal text, print a CHR$(14). These are the same control codes that Apple's 80-column card uses to switch modes. As mentioned before, this works with the 40-column screen also (a slight enhancement to Apple's firmware done by our software).

APPLE ProDOS APPENDIX

MouseText CHARACTERS

In addition to the MouseText characters available in 40 and 80 column modes of the new Apple // machines, MouseText is available in Modes 5 & 7. To shift the character set to MouseText, print a CHR$(27) and a CHR$(15).

To de-select MouseText, print a CHR$(14) and a CHR$(24). Since Apple's procedure for printing MouseText requires you to shift to inverse mode (the CHR$(15)), you might think that inverse MouseText isn't possible. Not so with ZBasic! If you want to experiment a little, just use a CHR$(27) to select inverse MouseText, and CHR$(24) to select normal alphanumerics again!

CUSTOM CHARACTER SETS

The character set that is included with your ZBasic system and used by the graphics character driver is the standard ASCII character set with the addition of the MouseText characters (MODE 5 and MODE 7 only).

If you wish, you can customize the character set to your liking. Space does not permit getting into the specifics of how each character is defined or used, but I can tell you how to change the character set to a pre-defined set. Our character set is defined in exactly the same way as the character sets included on the DOS Toolkit disk, available from Apple Computer, Inc. To change the character set, follow these instructions:

1. From Applesoft BASIC, with ProDOS active, insert a BACKUP COPY of your ZBasic master disk in the drive.

2. Type: 64K: `BLOAD "/ZBASIC/ZBASIC.SYSTEM, A$2000,TSYS"`
   128K: `BLOAD "/ZBASIC/RT.AUX.OBJ0, A$2800"`
   This loads the character set (and some other stuff) into memory.

4. Load your character set by typing:
   BLOAD <your character set pathname>, A$3900"
   This loads your character set over our character set. Since the DOS Toolkit character sets are only 768 bytes long, (characters 32-128) and only contain definitions for the standard ASCII characters, you will not be overwriting the MouseText (0-31).

5. Re-insert your ZBasic master disk, and type:
   64K: `BSAVE "/ZBASIC/ZBASIC.SYSTEM, A$2000,L$4000,TSYS"`
   128K: `BSAVE "/ZBASIC/RT.AUX.OBJ0, A$2800"`

APPLE ProDOS APPENDIX

SPECIAL ProDOS CONFIGURATION OPTIONS

ZBasic can be configured by typing "C" at the initial prompt screen (see the "Configure" section of the main reference manual), or by typing "CONFIG" while in the editor (see "CONFIG" on the next page). In addition to the standard configuration parameters, there are two more parameters which you can set for the Apple //.

PRINTER SLOT? 1-7

This allows you to specify which slot your printer interface is in. This number must be from 1 to 7 (slot 1 is the standard printer slot for Apples). As in the rest of the configuration questions, pressing <RETURN> as a response will accept the default and skip the initialization string configuration.

If you type a number from 1 to 7, you are telling ZBasic that your printer card is in that slot and you will be given an opportunity to specify a printer initialization string (the //c has the equivalent of an Apple Super Serial Card in slot 1):

SETTING UP A PRINTER INITIALIZATION SEQUENCE

ENTER THE EXACT KEYSTROKES REQUIRED BY YOUR PRINTER AND/OR INTERFACE CARD ("^" TO END):

The printer initialization string can be any sequence of up to 12 ASCII characters that can be typed from your keyboard (end input with the "^" symbol (caret).

To enter the initialization string, type the EXACT keys required by your printer and/or interface card. The keys will appear on the screen as you type them. Unprintable control characters will appear prefixed by a caret(^) character on the screen. Once set, this string is sent to the printer prior to anything else being sent out (such as LLIST,LPRINT, or ROUTE 128). Be sure to see the <S>ave option under "Getting Started" in the front of this manual.

Some common control codes may be entered from the keyboard using:

CTRL H	=8	TAB or CTRL |	=9	CTRL J	=10
CTRL L	=12	RETURN	=13	DELETE	=127
CTRL \	=28	CTRL]	=29	CTRL ^	=30
CTRL _	=31	ESC or CTRL [	=27		

See your Apple reference manual for other character sequences.

This is most useful for those users who have an older interface card that does not interface correctly with the 80-column screen. These cards will echo characters to the screen using the 40-column screen firmware, instead of the 80-column firmware when the 80-column card is active (usually messing everything up).

One solution is to tell the interface card to NOT echo characters by using the following initialization sequence: <CONTROL I> 80N

This would instruct the interface to turn off the screen, and allow up to 80 characters per line on the printer. See your interface card manual for more details. You can also send printer configuration characters to your printer for all kinds of fancy printing, if your printer is capable of it. Your printer manual will list printer control codes that are applicable.

continued...

APPLE ProDOS APPENDIX

continued from previous page

LOCATE order X,Y? <Y/N> Y_

This option allows you to configure the order of the coordinates in the LOCATE statement.

Normally ZBasic expects the horizontal (X) coordinate first. By answering "N" to this question you can make the vertical (Y) coordinate first and the horizontal (X) coordinate second.

Note: This also alters the coordinate base of the screen to make the upper-left hand corner character position 1,1 instead of 0,0 (only affects LOCATE).

This option is provided to maintain compatibility with the IBM/MSDOS versions of ZBasic which have this option so that BASICA programs are easier to convert. This makes porting BASIC programs from other computers much easier.

CONFIG

You may re-configure the system any time from the standard line editor by using the CONFIG command. Use caution when doing this while working with CHAIN programs or programs that will be sharing data (especially floating point numbers).

Each CHAINed program must be compiled using the same configuration as the other programs in the overall CHAINed system. Otherwise, you will get a chain error when attempting to run them.

If you elect to (S)ave your custom configuration, ZBasic will ask you to enter the complete pathname of the ZBASIC.SYSTEM file. This will normally be "/ZBASIC/ZBASIC.SYSTEM",

unless you have installed ZBasic on a hard disk and/or changed the name of this file.

If ZBasic has trouble saving your configuration, it will give you an error message and wait for a keypress. After pressing a key, you will be returned to the configure menu.

If no error is encountered, you will automatically be put into the line editor.

NOTE ON CASE CONVERSION

During boot-up, the system checks to see if it is running on an Apple][+ or a newer machine. If you are using a][+ the system will automatically convert from lower to upper case for both keyboard input and screen output.

If it's a newer machine, the system will skip the conversion. Upper/lower case conversion can be configured separately by the user from the configure menu. See "Configure" in the front of this manual.

APPLE ProDOS APPENDIX

THE ProDOS MACHINE LANGUAGE INTERFACE

These versions of ZBasic have been written with the ease of direct access to ProDOS in mind. This section of the manual describes how a ZBasic program can talk with ProDOS directly.

MLI INTERFACE

First of all, this is NOT a tutorial on how to use the ProDOS Machine Language Interface. For more information on that subject, consult the ProDOS Technical Reference Manual.

ENTRY POINTS

All parameters for ProDOS calls that are made by ZBasic are located in a parameter block at $1F00 (all addresses are in HEX). There is an 18 byte buffer here that the ZBasic system uses for all MLI calls (18 being the maximum length of any MLI parameter block).

In addition, the entry point for a ProDOS call with the 64K version is at $803 ($865 for the 128K version). One more buffer that might be useful is the file name buffer. It is located at $1F12, and is 64 bytes long (the maximum length of a ProDOS pathname).

ZBasic TO ProDOS INTERFACE

To use the ZBasic to ProDOS interface, first set up the parameter list for the MLI call that you wish to make. If you need to, you can set up the pathname pointer with:

```
POKEWORD &1F01, VARPTR (name$)
```

since ZBasic strings conform to the ProDOS pathname standards (a count byte followed by the string). Next, you must load the 6502 Accumulator with the MLI command code, and then JMP or JSR to location $803 ($865 for the 128K version). The ProDOS call will be performed, and the carry flag will have the status of the call upon return. If the carry is clear, then the call returned with no error. If, on the other hand, the carry is set, then there was an error and the error code can be retrieved from location $A2. The ProDOS error that is returned by the MLI is translated into the appropriate ZBasic error code, if possible. If not, then the actual MLI error code will be returned. If you wish to use the standard ZBasic error handler, then you can perform a JMP to location $809 ($87F for the 128K version) if the carry is set upon return from the ProDOS interface.

For example, to use this interface to set the ProDOS system prefix to "/ZBASIC":

```
PATH$ = "/ZBASIC"
POKE &1F00, 1               <--- Sets up parameter block
POKE WORD &1F01, VARPTR(PATH$)
MACHLG &A9, &C6 &20, $803    <--- 128K version change to $865
MACHLG &90, 3, &20, $809     <--- 128K version change to $87F
END
```

Note: Either "&" or "$" may be used to denote Hex numbers (ProDOS version only).

APPLE ProDOS APPENDIX

USING MACHLG

The assembly language source for the MACHLG statements would look something like this:

```
SET.PREFIX          EQU         *
                    LDA         #$C6    ;MLI CODE FOR SET_PREFIX
                    JSR         $803    ;CALL THE INTERFACE
                    BCC         DONE    ;NO ERROR
ERROR               JSR         $809    ;LET ZBASIC HANDLE THE ERROR
DONE                EQU         *
```

ProDOS ERROR CODES

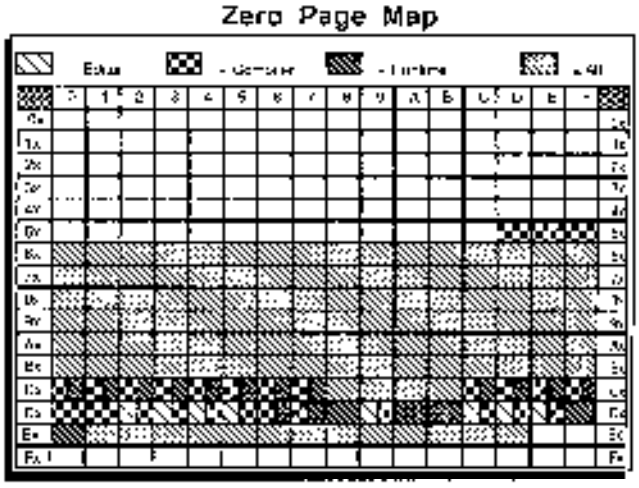
For the 64K version only; another location of interest is $806. This is the entry point for the subroutine that translates ProDOS error codes into ZBasic error codes. If you wish to access the MLI directly, but still want ZBasic error codes returned, you can perform a JSR to this subroutine with the ProDOS error code in the accumulator. The translated error code will be stored in location $A2.

For more examples of how to use the ProDOS-ZBasic interface, see the CREATE, PREFIX, BLOAD, and BSAVE functions included on your master diskette.

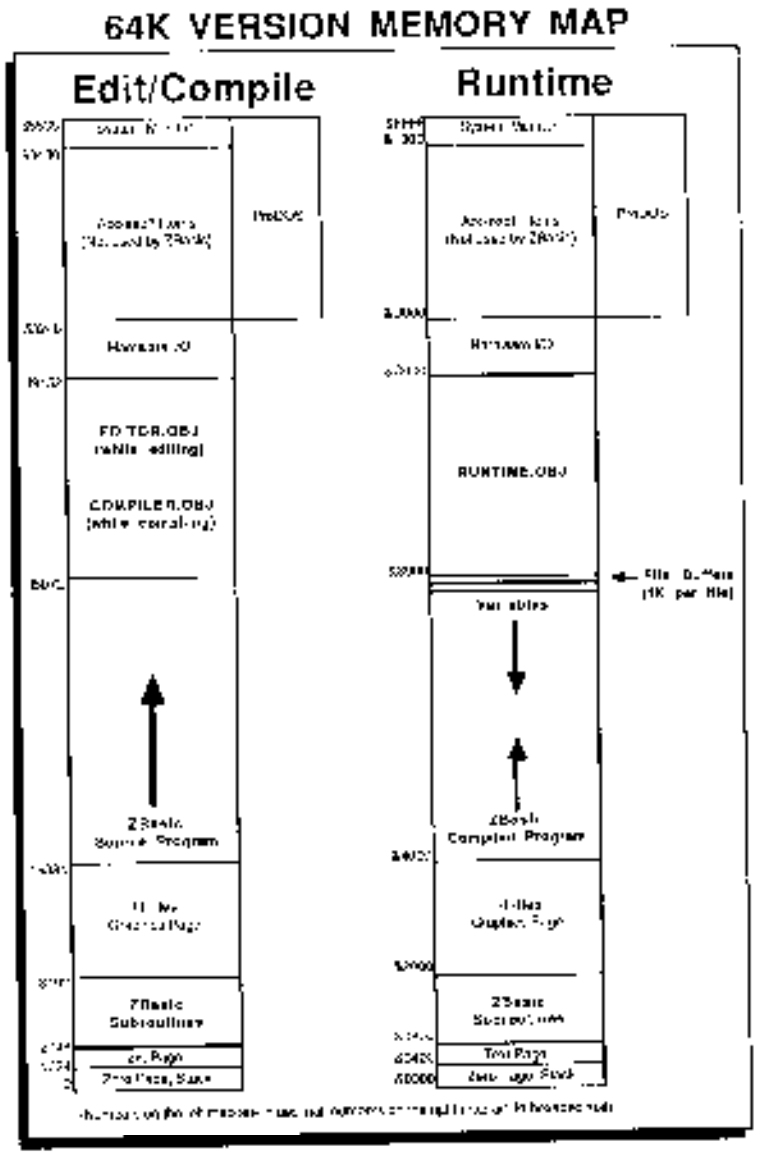
MEMORY USAGE

The following diagrams illustrate memory usage for the various phases of operation of the ZBasic system.

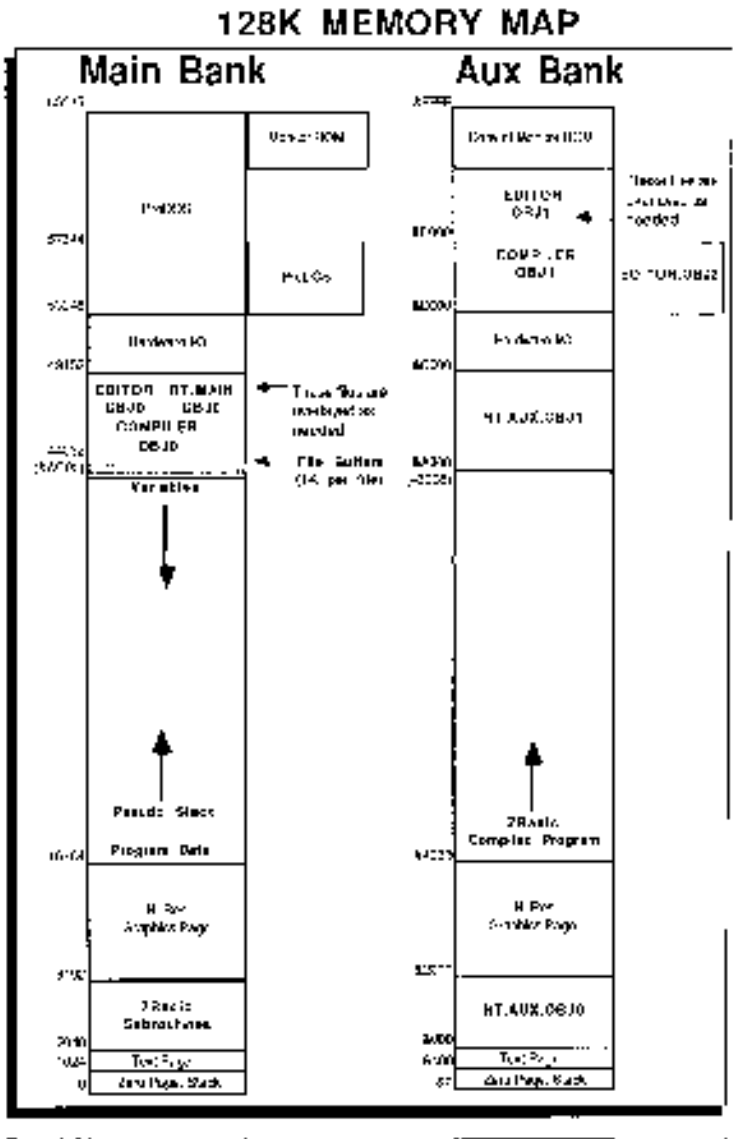
Note: Memory locations 768-975 (page 3) are not used by the ZBasic system. This would be a good place to store short machine language subroutines.



APPLE ProDOS APPENDIX



APPLE ProDOS APPENDIX



APPLE ProDOS APPENDIX

CONVERTING APPLESOFT PROGRAMS TO WORK WITH ZBASIC

ZBasic is an advanced version of BASIC. While it shares many of the commands and syntax of Applesoft, it is not exactly the same in many areas, such as graphics, disk file handling and such.

CONVERTING APPLESOFT FILES FOR LOADING INTO ZBASIC

ZBasic source code files and Applesoft files are not compatible. To convert an Applesoft program so you can load it into ZBasic:

1. Make sure you have a Backup of your Applesoft program then load the Applesoft program into Applesoft. Make sure your program doesn't have a line zero then add the following line to the program.

```
0F$="FILENAME":PRINTCHR$(4) "OPEN";F$
:PRINTCHR$(4) "WRITE";F$:POKE33,33:PRINT"0";
:LIST 1-:PRINTCHR$(4)"CLOSE":TEXT:END
```

   Note: The program above is one line. Enter without spaces or <RETURN>.

2. Change "FILENAME" above to the name of the file you wish to create for loading into ZBasic. Then type RUN.

3. Load ZBasic, press "E" for edit, and then load the program using LOAD. To compile the program type RUN. When errors occur use the chart on the next few pages to convert syntax to ZBasic syntax.

CONFIGURING ZBASIC FOR COMPATIBILITY WITH APPLESOFT

ZBasic allows you to configure the system for your preferences. To make ZBasic as compatible as possible to Applesoft, set the following configurations. See "Configure" in the front of this manual for details about setting configuration options:

Default Variable type:	S (avoid doing this whenever possible)
Convert to Uppercase Y/N:	Y
Optimize Expressions for integer Y/N:	N (avoid doing this whenever possible)

STRING LENGTH NOTE

ZBasic uses strings differently than Applesoft. See "Converting Old Programs" and DIM and DEF LEN in the front section of this manual for more information.

COMMANDS THAT ARE DIFFERENT

The following commands are different and will require converting.

The list includes hints on how to convert the various Applesoft statements to ZBasic equivalents.

APPLE ProDOS APPENDIX

Applesoft Commands	ZBasic Equivalent
BLOAD/BSAVE	See the BLOAD and BSAVE functions on the master disk.
CALL	ZBasic uses a constant as an address (not a variable). Parameters not allowed.
CLEAR	See CLEAR in the main reference section for ZBasic's additional options.
COLOR	ZBasic uses this statement for all graphics modes (not just low-res).
CONT	Not supported (ZBasic is a compiler).
DEF FN	More options in ZBasic. See DEF FN and LONG FN in the main reference section.
DIM	ZBasic only allows constants in DIM expressions. See DIM.
DRAW	Not available (see DRAW.FN example on the master disk).
FLASH	Not available.
FRE	Not applicable (and not necessary since ZBasic doesn't do "Garbage collection").
GET	Not available. See Get.FN on the master disk.
GR	Use: MODE 1:CLS
HCOLOR	Use: COLOR.
HGR	Use: MODE 5 (also see MODE 7 for double hi-res).
HGR2	Not applicable
HIMEM	Not applicable
HLIN	Use PLOT,xy TO x2,y2
HOME	Use CLS.
HPLOT	Use PLOT
HTAB	Use LOCATE x, PEEK(37) (also see PRINT@/% and INPUT@/%)
IN#	Use INSLOT
INVERSE	Use CHR$(15). See "Inverse Characters" in this appendix.
LOMEM	Not applicable
NORMAL	Use CHR$(14). See "Inverse Characters" in this appendix.
ON ERR GOTO	See ZBasic's ON ERROR GOSUB statement.
PDL	See DEF MOUSE and MOUSE in this appendix and the main reference section.
POP	Use RETURN nnnn instead.
POS(expr)	Expr=0 for default device, 1 for printer and 2 for disk.
PR#	See OUTSLOT, LPRINT, OPEN"C" and ROUTE.
RECALL	Not available.
RESUME	Use RETURN with ON ERROR GOSUB
ROT	Not available.
RUN	See RUN in this appendix and in the main reference section for other options.
RND(n)	ZBasic returns an integer number between one and n.
SCRN	Use POINT
SCALE	Not available.
SHLOAD	Not available.
SPEED	Not available.
STORE	Not available.
TEXT	Not available. Use MODE 0,2,4 or 6 instead . See MODE.
TRACE	Use TRON or TROFF (also see TRONX, TRONS).
VLIN	Use PLOT x,y TO x2,y2
VTAB y	Use LOCATE PEEK(36),y (see PRINT@/% and INPUT@/%)
WAIT	Not available.
XDRAW	Not available.

Many of the commands Applesoft supports have extensions in ZBasic. For instance; ELSE is supported with IF THEN. RESTORE will allow you to position the DATA pointer to a specific item, PRINT USING is supported, etc.

Note: When converting programs a word processor with FIND and REPLACE is very handy.

continued...

APPLE ProDOS APPENDIX

DIFFERENCES IN DISK FILE COMMANDS

Applesoft File Commands	ZBasic Equivalents
OPEN A FILE FOR INPUT	
PRINTCHR$(4)"OPEN filename"	
PRINTCHR$(4)"READ filename"	OPEN"I",filenum, "filename"
OPEN A FILE FOR OUTPUT	
PRINTCHR$(4)"OPEN filename"	
PRINTCHR$(4)"WRITE filename"	OPEN"O"filenum, "filename"
OPEN A FILE FOR READ/WRITE	
PRINTCHR$(4)"OPEN filename, L100"	OPEN"R",filenum,"filename",100
PRINTCHR$(4)"OPEN filename, R10"	RECORD#filenum,10
CLOSE FILES	
PRINTCHR$(4)"CLOSE filename"	CLOSE#filenumber
PRINTCHR$(4)"CLOSE"	CLOSE

Note: Also see "Files" in the front section of this manual for more information about ZBasic's powerful file handling commands. Also see: RECORD,READ#,WRITE#,DIM,PRINT#,INPUT# and LINEINPUT#.

PEEKS, POKES, AND SYSTEM CALLS

Applesoft	ZBasic Equivalents
CALL -958	CLS PAGE
CALL -868	CLS LINE
X=PEEK(-16336)	See SOUND in reference section.
X=PEEK(-16287), Y=PEEK(-16286)	See MOUSE(3) and DEF MOUSE

Other PEEK and POKE statements should work as expected except those dealing with Applesoft.

Also see MACHLG, USR, CALL and LINE in the main reference section.

APPLE ProDOS APPENDIX

TRANSFERRING ZBasic DOS 3.3 FILES TO THE ProDOS VERSIONS OF ZBasic

The file format for the ProDOS version of ZBasic is different than the file format for the DOS 3.3 version of ZBasic. Therefore; follow these instructions to convert files:

1. LOAD your program into the DOS 3.3 version of ZBasic.

2. Use the SAVE* command to save the source code in ASCII.

3. Exit the DOS 3.3 version of ZBasic and boot your favorite DOS 3.3 to ProDOS conversion program; such as CONVERT found on your ProDOS /USERS.DISK or APPLE SYSTEMS UTILITIES.

4. Copy the file just saved in ASCII to a ProDOS formatted diskette.

5. Execute either the 64K or 128K ProDOS versions of ZBasic and LOAD the program.

Programs created in the DOS 3.3 version should run with few, if any, changes; although you may want to modify the programs to take advantage of the ProDOS /RAM disk or the 1 colors available in Double Hi-Res.

APPLE ProDOS APPENDIX

REFERENCE SECTION

This section of the appendix discusses commands unique to the ProDOS version of ZBasic and commands that may have other meanings other than those described in the main reference section.

CLS command

FORMAT **CLS** [n]

DEFINITION Clears the screen. Same as the standard CLS statement with the following variations:

o If you are currently editing in one of the text modes, the screen will be cleared immediately (without going through the compiler).

o If you are currently in one of the graphics modes, the command must first be compiled before it is executed by the runtime system, and takes longer.

EXAMPLE See CLS in the main reference section for detailed information.

REMARK This command works correctly with the standard Apple 80-column card and Videx 80-column cards (and compatible). Any control key typed at the keyboard that is not defined as n editor command will be passed through unchanged to the 80-column firmware. What this means is that if your card requires a CHR$(26) to clear the screen you can press CTRL-Z to accomplish the same thing.

To use any of the other CLS options from within the editor, such as CLS nn, precede the command with a colon. e.g. `:CLS ASC("A")`

COLOR statement

FORMAT **COLOR** [=] n

DEFINITION The **COLOR** codes for the ProDOS version of ZBasic are:

Modes 0, 2, 4, & 6: Text Characters only, no color.

Modes 1, 3, & 7:

NUMBER	COLOR
0	Black
1	Magenta
2	Dark Blue
3	Purple
4	Dark Green
5	Grey
6	Medium Blue
7	Light Blue
8	Brown
9	Orange
10	Grey
11	Pink
12	Green
13	Yellow
14	Aqua
15	White

Mode 5:

NUMBER	COLOR
0	Black 1
1	Green
2	Violet
3	White 1
4	Black 2
5	Orange
6	Blue
7	White 2

IIGS Note: The IIGS Super Hi-Res graphics mode is not supported directly (the //e, //c modes are emulated).

APPLE ProDOS APPENDIX

DATE$**, **TIME$ function

FORMAT **DATE$**
TIME$

DEFINITION See the main reference manual.

EXAMPLE See the main reference manual for details of usage.

REMARK These functions behave exactly as described in the standard reference section if your system has a ProDOS compatible clock installed.

The system performs a ProDOS call to retrieve the date and time from a clock card. If no card is installed, then the strings that are returned will be set to whatever the current values are of the ProDOS date and time locations on the global page (00/00/00 and 00:00 normally).

If your system has no clock, and you wish to set the date and time manually, you can include the DATETIME function in your program (from your master disk).

Since ProDOS does not have any storage space for seconds, the TIME$ seconds field will always be "00".

DEF LPRINT statement

FORMAT DEF LPRINT [=] Slot number

DEFINITION This command is used to configure the printer slot during runtime. After this command is used, all printer output will be diverted to the selected slot.

The slot number may be specified by any numeric expression but the value of Slot number MUST be between one and seven.

EXAMPLE

```
DEF LPRINT = 1
```

REMARK This command supersedes the configuration value, except for the initialization string. See the notes on configuration for more info.

If value exceeds the range of 1-7, the number will be masked to stay in range.

DEF MOUSE statement

FORMAT DEF MOUSE [=] *expression*

DEFINITION This statement defines which device (MOUSE or JOYSTICK) will be use for the MOUSE function call.

expression=0	APPLE MOUSE INTERFACE
expression<>0	JOYSTICK

EXAMPLE

```
DEF MOUSE=1: REM Define as a JOYSTICK
DO
  PRINT MOUSE(1), MOUSE(2)
UNTIL MOUSE(3)
END
```

This program will print the positions of the joystick
until you press the joystick button.

REMARK The default is equivalent to DEF MOUSE=0. The Apple //c has the equivalent of a mouse card built-in.

If DEF MOUSE=1 is used to activate the joystick, the function MOUSE(3) will return a value corresponding to which joystick button was pressed.

Value	**Meaning**
0	Neither button pressed
1	Button 0 pressed
2	Button 1 pressed
3	Both buttons pressed

DIR command

FORMAT [L]DIR [+] [*pathname*]
[L]CAT [+] [*pathname*]

DEFINITION These commands display a directory of a ProDOS volume, as explained in the reference section. DIR and CAT are interchangeable. CAT was implemented to make conversion easier for Applesoft programmers.

When the command DIR is given by itself, ZBasic will display a directory of the currently logged ProDOS pathname (see PATH command) in the standard 40 column format.

DIR+ operates in the same way as DIR without the "+", and will produce the ProDOS standard 80-column display format (more information is shown). If in 40-column mode the output will wrap to the second line.

The optional pathname specifies a directory to be displayed. The pathname can be a full or partial ProDOS pathname. Full pathnames start with a slash ("/"), and specify the root volume. If a partial pathname is specified, ZBasic will append this pathname to the currently logged pathname, and display the contents of this sub-directory. Pathnames can be any legal ProDOS pathname.

The optional "L" preceding the command will direct output to the printer. There must not be a space between the "L" and the "DIR".

REMARK As you can see, the first line of the directory contains the name if the directory which the listing is produced from. A slash preceding the directory name (as in the example) signifies that this is a root (volume) directory. Sub-directory names are not preceded by the slash.

The heading line is pretty much self-explanatory, except the ENDFILE (found on a DIR+ listing). The figures in the ENDFILE column represent the total number of bytes in that file.

The TYPE column represents the ProDOS standard file types, with one exception -- ZBS. This file type is a ZBasic tokenized source file.

An asterisk (*) preceding a file name signifies that this file is locked. It can not be modified in any way from within the ZBasic system.

As with the editor "LIST" command, the directory can be temporarily halted by pressing the space bar once. Pressing the space bar again will advance the directory one line. Pressing any other key will restart the listing. Pressing CTRL-C will abort the directory listing.

To read a directory from within a running program, simply OPEN the directory file as you would any other, then read the necessary information from it. See the ProDOS Technical Reference Manual, Appendix B, for information concerning the format of directory files.

Also see the special ZBasic ProDOS command: ONLINE.

APPLE ProDOS APPENDIX

EXAMPLE

```
ZBasic Ready
DIR

/ZBASIC
NAME                 TYPE   BLOCKS  MODIFIED

 ZBASIC.SYSTEM       SYS    33            10-FEB-87 12:02
*RAM.FILLER          BIN    17            31-JAN-86 11:40
*RUNTIME.OBJ         BIN    28            29-JAN-87 15:49
*ZBASIC.HLP          TXT    57            12-OCT-86 13:18
 DISKIO.BAS          TXT    7              5-DEC-86 14:26
 GRAPH.BAS           ZBS    3              6-NOV-86 10:25
 CREATE.FN           TXT    1             20-JAN-87 11:31
 DRAW.FN             TXT    3             29-DEC-86 17:08

BLOCKS FREE:  26     BLOCKS USED:  254

ZBasic Ready


ZBasic Ready
DIR+

/ZBASIC
NAME           TYPE   BLOCKS  MODIFIED             CREATED              ENDFILE

 ZBASIC.SYSTEMSYS    33      10-FEB-87 12:02      10-FEB-87 12:51      16384
*RAM.FILLER    BIN    17      31-JAN-86 11:40      10-FEB-87 12:51      9384
*RUNTIME.OBJ   BIN    28      29-JAN-87 15:49      10-FEB-87 12:51      7644
*ZBASIC.HLP    TXT    57      12-OCT-86 13:18      10-FEB-87 12:51      384
 DISKIO.BAS    TXT    7       5-DEC-86 14:26       10-FEB-87 12:51      844
 GRAPH.BAS     ZBS    3       6-NOV-86 10:25       10-FEB-87 12:51      1982
 CREATE.FN     TXT    1       20-JAN-87 11:31      10-FEB-87 12:51      123
 DRAW.FN       TXT    3       29-DEC-86 17:08      10-FEB-87 12:51      456

BLOCKS FREE:  26     BLOCKS USED:  254

ZBasic Ready
```

Note: endfile numbers may not be actual.

EDITOR command

FORMAT EDITOR [+]

DEFINITION This command is used to enter the full screen text editor from the Standard line editor.

Typing **"EDITOR"** on the ZBasic command line will transform any program currently in memory from ZBasic tokenized format to full ASCII format and enter the full screen editor.

If you use the optional "+", the program currently in memory will have the line numbers stripped prior to entering the full screen editor. Be sure that you have not used any line number references in your program (such as GOTO or GOSUB). Use label references instead.

REMARK See the section entitled "Full Screen Editor" in this appendix for a complete description of editor commands and operation.

Note: To get back to the standard line editor press ESC (or CTRL-K CTRL-Q on][+).

APPLE ProDOS APPENDIX

INSLOT statement

FORMAT INSLOT(*Slot Number*)

DEFINITION This statement will allow you to specify the slot number of an interface card which your program is to receive input from.

This is supplied so that you can use non-standard interface cards (i.e. other than those supported directly by ZBasic, such as a graphics tablet).

Do not use this command to access a Super Serial Card; use the OPEN"C" command instead.

INSLOT(0) will "re-attach" the keyboard for input.

EXAMPLE

```
CLS
DO
  INPUT"Which slot is the widget?";Slot
UNTIL (Slot>0) AND )Slot<8)
:
INSLOT(Slot)
:
do something with the slot here...
:
INSLOT(0)                    <--- Set the slot back to normal.
END
```

(example only do not use)

REMARK See your hardware technical reference manuals for details.

Note: Any interface card that attempts to store a value into an Applesoft variable (such as some clock cards) will not work correctly with ZBasic since there are no "Applesoft variables" in ZBasic.

Check the technical reference manual of the device to set it to some other parameters.

MEM command

FORMAT **MEM**

DEFINITION This command is used to show the amount of memory remaining for text and object code remaining and the amount of text and object code space used during each phase of operation.

#####	**Text**	o Shows amount of text space used
#####	**Text Mem**	o Amount of text room remaining
#####	**Object**	o Size of object code generated*
#####	**Variable**	o Amount of variable space used*
#####	**Code Mem**	o Object and variable space remaining*

*Values returned only correct immediately after compiling (RUN).

REMARK See Memory map in this appendix for the 64K or 128K ProDOS versions of ZBasic. Also see MEM in the main reference section.

APPLE ProDOS APPENDIX

MODE statement

FORMAT **MODE** [=]*expression*

DEFINITION ZBasic uses MODE to define the characteristics of a screen. ZBasic allows a program to integrate text and graphics anywhere on the screen in MODE 5 and 7. This feature allows ZBasic programs from an IBM PC and other computers to run on your Apple.

ProDOS™ Version
MODE CHART

MODE	TEXT	GRAPHICS
0,8	40 x 24	Character
1	None	40 x 48
2,10	80 x 24	Character
3	None	80 x 48
4,12	40 x 24	Character
5	40 x 24	280 x 192
6,14	80 x 24	Character
7*	80 x 24	560 x 192
9	40 x 4	40 x 40
11	80 x 4	80 x 40
13	40 x 4	280 x 160
15*	80 x 4	560 x 160

REMARK

character	= Graphics are defined as text characters
40 x 80	= Low Resolution Color Graphics
80 x 48	= Medium Resolution Color Graphics
280 x 192	= High Resolution Color Graphics
*560 x 192	= Double High Resolution. For IIe, IIc and //GS only.

Modes 9, 11, 13 and 15 have graphics at the top of the screen and text at the bottom, similar to Applesoft GR and HGR commands.

MODE will set COLOR to default, while in most modes. See COLOR for the other colors available in each mode.

*Double Hi-Res does not function in the 64K version (it requires 128k).

ONLINE command

FORMAT **ONLINE**

DEFINITION When the ONLINE command is issued in the Standard Line Editor, a list of all ProDOS volumes currently connected to the system will be displayed on the screen.

Each entry will display the slot, drive, and volume name of the device.

EXAMPLE

```
ZBasic Ready
ONLINE

S6,D1 = /ZBASIC
S3,D2 = /RAM
```

REMARK Since ZBasic will only operate on volumes by using pathnames, this is supplied so that you can identify a particular volume in a drive.

APPLE ProDOS APPENDIX

OUTSLOT statement

FORMAT **OUTSLOT**(*slot number*)

DEFINITION This command allows you to redirect output to the interface card located in the slot number specified. This is not intended as an alternative to the existing ZBasic commands (such as LPRINT or ROUTE).

This command is supplied only to allow you to interface your program with those cards that ZBasic does not support directly (such as graphics tablets, etc.).

OUTSLOT(0) will "re-attach" the screen for output.

EXAMPLE

```
CLS
DO
  INPUT"Which slot is the widget?";Slot
UNTIL (Slot>0) AND (Slot<8)
:
OUTSLOT(Slot)
:
do something with the slot here...
:
OUTSLOT(0)              <--- Set the output back to normal.
END
```

(example only do not use)

REMARK See INSLOT statement and your hardware technical reference manuals for details using slots.

PATH command

FORMAT **PATH** [[-][-] ...][*pathname*]
PREFIX [[-][-]...][*pathname*]

DEFINITION The PATH command allows you to set and/or display the currently logged ProDOS pathname. PREFIX is also provided for compatibility reasons.

PATH without any parameters will display the current ProDOS prefix.

"PATH pathname" will set the current ProDOS prefix, then display it as a verification that it was indeed set. Pathname can be either a full or partial pathname. If it starts with a slash ("/"), ZBasic will treat it as a full pathname and reset the prefix appropriately. If you specify a partial pathname, ZBasic will append it to the current prefix to create the new prefix.

The "-" parameter will "step-back" the current prefix by one directory. If a pathname is specified after the "-", ZBasic will then set this as the current prefix. You may use more than one "-" parameter to specify stepping back multiple directories.

EXAMPLE

```
ZBasic Ready
PATH                              (display current prefix)
/PROFILE

ZBasic Ready
PATH ZBASIC/SOURCE                (append ZBASIC/SOURCE)
/PROFILE/ZBASIC/SOURCE            (to current prefix)

ZBasic Ready
PATH-OBJECT                       (remove SOURCE and)
/PROFILE/ZBASIC/OBJECT            (append OBJECT)

ZBasic Ready
```

REMARK ZBasic will not allow you to remove the prefix entirely. The system MUST have a prefix set at all times.

POINT function

FORMAT **POINT** (*expression1* , *expression2*)

DEFINITION This function will return either a 0, signifying that the pixel is "off", or a 1, signifying that the pixel is "on."

EXAMPLE

```
PLOT 0,0
PRINT POINT(0,0)
:
CLS
PRINT POINT(0,0)
:
END

RUN

1
0
```

REMARK In modes 5 and 7, the POINT function cannot return the color of the pixel at the specified coordinates, due to the method that the Apple uses to create colors on the screen.

RENAME command

FORMAT **RENAME** ["]*pathname1*["],["]*pathname2*["]

DEFINITION Renames the file specified by pathname1 to the name specified by pathname2. The comma separating the names IS required.

This command is supplied as an editor command in addition to the ZBasic statement so that the compiler does not have to be accessed every time you wish to rename a file.

EXAMPLE

```
RENAME ZBASIC.SYSTEM, ZBASIC
```

REMARK See RENAME in the main reference section for more information about using RENAME.

APPLE ProDOS APPENDIX

RUN* command

FORMAT See the main reference manual for syntax.

REMARK When saving your compiled programs to disk with the RUN* command, ZBasic will create a SYS type file that can be executed directly from ProDOS.

64K VERSION
In addition to your object file, the file "RUNTIME.OBJ" ***MUST*** be in the same directory.

128K VERSION
In addition to your object file, the following three files must be in the directory:

```
RT.MAIN.OBJ1
RT.AUX.OBJ0
RT.AUX.OBJ1
```

As part of it's initialization, your program attempts to load the runtime modules from disk (you don't have to do this; the compiler will generate the necessary code automatically).

If the required runtime files cannot be found, a ProDOS error message will be generated, and you will be left in the Apple system monitor.

USR function

FORMAT See the main reference manual.

REMARK When your USR subroutine is entered, the value that was in the parentheses in the ZBasic program can be found at zero page locations $64 and $65. This value will be a 16-bit integer in standard least-significant-byte/most-significant-byte order.

If your subroutine is to pass a 16-bit value back to the ZBasic program, it should place the value in locations $64 and $65, again in lsb/msb order.

128K Note: Be aware that with this version the USR routine must be located in the program auxiliary bank of memory. This is most easily accomplished by using"

```
DEF USRx=LINE nnnn
```

USR5 function

FORMAT *variable* = **USR5(*slot*)**

DEFINITION This pre-defined USR function returns the status byte of an Apple Super Serial Card in slot number slot.

The status byte will be returned in the lower 8 bits of variable.

The variable must be an integer variable.

If no Super Serial Card is installed in the system, the value returned will be undefined.

REMARK See OPEN"C" for more details about using communication functions and the Super Serial Card reference manual for the format of the status byte.

APPLE ProDOS APPENDIX

FULL SCREEN EDITOR

This version includes an easy-to-use, full screen text editor. It can be used to enter and edit ZBasic source program files, or any other text file. Some of it's features include full screen cursor movement, long distance cursor movement, split screen operation, cut/copy/paste/replace lines, global search, automatic indentation, full scrolling capabilities up/down and left/right, and some other goodies.

DIFFERENCE BETWEEN THE FULL SCREEN EDITOR AND STANDARD LINE EDITOR

ZBasic comes with a Standard line editor, as described in the main reference section, that works the same way on all versions of ZBasic. From this editor you can do direct commands as described in the main reference section. You cannot do direct commands from the Full Screen Editor (other than those defined).

INVOKING THE FULL SCREEN EDITOR

To enter the full screen editor, type"EDITOR" from the Standard line editor ("EDITOR+" if you want to strip line numbers). If you currently have a file in memory, the file will be converted to a text file and transferred. If no file is in memory, you will enter the full screen editor without text.

RETURNING TO THE STANDARD LINE EDITOR

To return to the Standard line editor (command environment), press <ESC> (CTRL-K D in the 40-column editor). The file that you were editing will be re-loaded into the line editor (with line numbers added if the file did not contain any).

APPLE ProDOS APPENDIX

80-COLUMN EDITOR

If you are using and Apple //e with an 80-column text display, most of the functions are accessed by pressing one of the or keys in combination with one of the numeric keys.

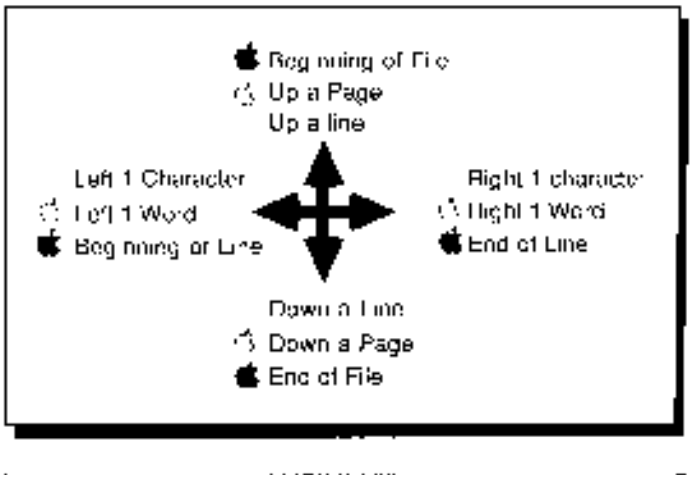
While the editor is waiting for you to enter a character, you have the option of using one of the commands available.

HELP LINE

When you press one of the Apple keys, a short "help" line will appear on the bottom line in place of the status line. This help line will remain on the screen for as long as you keep pressing an or key.

The help lines are not meant to be complete descriptions of the commands available, just memory joggers.

80 COLUMN CURSOR MOVEMENT KEYS



APPLE ProDOS APPENDIX

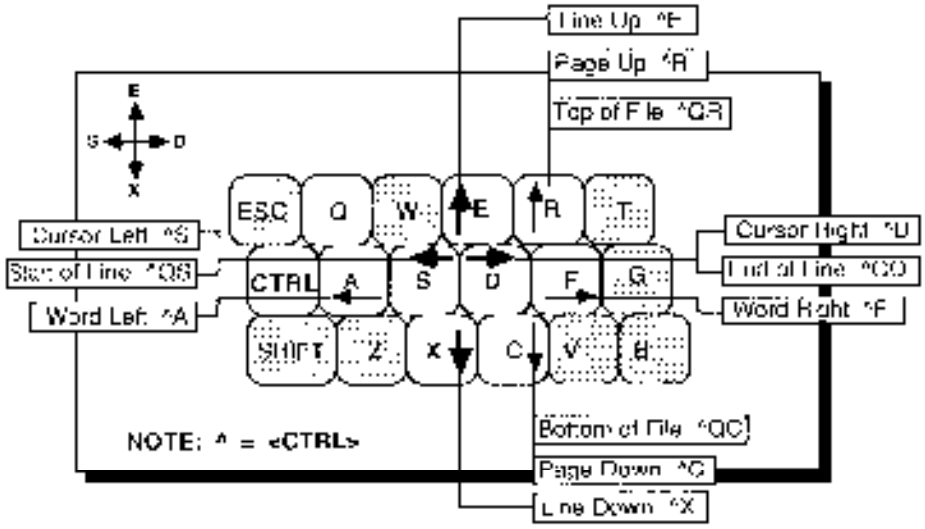
40-COLUMN EDITOR

Since Apple][+ users don't have Apple keys on their keyboard, control keys are used in place of the Apple keys. These commands have been set up to match Wordstar tm, a word processor form MicroPro where possible. For those commands that are not a part of Wordstar tm, we tried to make the command key match the command as logically as possible (the command is followed by an asterisk if it is not Wordstar compatible).

When one of the prefix keys (CTRL-Q or CTRL-K) is pressed a "^Q" or "^K" appears in the lower left corner of the screen, to remind you that one of the prefix keys has been pressed. If you change your mind, and don't want to access one of the commands, simply press the space bar to cancel the command.

The following pages describe all of the commands and cursor movements available. Each one operates exactly the same way, whether the machine is a][+ or one of the newer machines.

40 COLUMN CURSOR MOVEMENT KEYS



USING THE FULL SCREEN EDITORS

The following pages contain a complete description of the Full Screen Editor. You may want to Xerox the Quick Reference page.

APPLE ProDOS APPENDIX

FULL SCREEN EDITOR QUICK REFERENCE PAGE

40 Column	CURSOR	80 Column
CTRL-S	Left 1 Character	←
CTRL-D	Right 1 Character	→
CTRL-A	Left 1 Word	⌥ ←
CTRL-F	Right 1 Word	⌥ →
CTRL-Q CTRL-S	To Beginning of Line	⌘ ←
CTRL-Q CTRL-D	To End of Line	⌘ →
CTRL-E	Up a Line	↑
CTRL-X	Down a Line	↓
CTRL-R	Up a Page	⇧ ↑
CTRL-C	Down a Page	⇧ ↓
CTRL-Q CTRL-R	To Beginning of File	⌘ ↑
CTRL-Q CTRL-C	To End of File	⌘ ↓
40 Column	**COMMANDS**	**80 Column**
CTRL-K CTRL-O	Return to Line Editor	ESC
←	Delete Character	Delete
CTRL-Q ←	Delete to Line Start	⌥ Delete
CTRL-Q CTRL-Y	Delete End of Line	⌘ Delete
CTRL-V	Switch INSert/Overwrite	⌥ -0
CTRL-K CTRL-N	Clear Text Buffer NEW	⌘ -0
CTRL-K CTRL-L	LOAD File	⌥ -1
CTRL-K CTRL-S	SAVE File	⌘ -1
CTRL-K CTRL-X	Cut Line	⌥ -2
CTRL-K CTRL-V	Paste Line	⌘ -2
CTRL-K CTRL-C	Copy Line	⌥ -3
CTRL-K CTRL-R	Replace Line	⌘ -3
CTRL-N	Insert Line	⌥ -4
CTRL-Y	Delete Line	⌘ -4
CTRL-Q CTRL-F	Find	⌥ -5
CTRL-L	Find Next Occurrence	⌘ -5
CTRL-K CTRL-I	Set Tab Value	⌥ -6
CTRL-Q CTRL-I	Autoindent On/Off	⌘ -6
CTRL-K CTRL-I	Restore Line	⌥ -7
CTRL-K CTRL-P	PRINT	⌘ -7
CTRL-Z	Scroll Up	⌥ -8
CTRL-W	Scroll Down	⌘ -8
CTRL-K CTRL-T	Freeze Top	⌥ -9
CTRL-K CTRL-B	Freeze Bottom	⌘ -9

CURSOR KEY DEFINITIONS

This page contains the detailed descriptions of the cursor key movements for the Full screen editor:

80 COLUMN 40 COLUMN

UP A LINE

Up-Arrow CTRL-E
Moves the cursor up one line.

UP A PAGE

-Up-arrow CTRL-R
Moves the cursor on page back in the file. A page is defined as the current number of lines in the editing window minus one.

UP TO TOP

Places the cursor at the beginning of the file.

LEFT A CHARACTER

Left Arrow CTRL_S
Moves the cursor one character to the left.

LEFT A WORD

-Left Arrow CTRL-A
Moves the cursor to the beginning of the word to the left of the current cursor position.

LEFT TO START OF LINE

-Left-Arrow CTRL-Q CTRL-S
Moves the cursor to beginning of current line.

RIGHT A CHARACTER

Right Arrow CTRL-D
Moves the cursor on character to the right.

RIGHT A WORD

Right Arrow CTRL-F
Moves the cursor to the beginning of the next word to the right of the present position.

RIGHT TO END OF LINE

-Right-Arrow CTRL-Q CTRL-D
Moves the cursor to the end of the current line.

DOWN A LINE

Down Arrow CTRl-X
Moves the cursor down one line.

DOWN A PAGE

-Down-Arrow CTRL-C
Moves the cursor down one page in the file.

DOWN TO END OF FILE

-Down-Arrow CTRL-Q CTRL-C
Moves the cursor to the end of the file.

-Right-Arrow CTRL-Q CTRL-D
Moves the cursor to the end of the current line.

APPLE ProDOS APPENDIX

FULL SCREEN EDITOR COMMANDS

This following pages contain the definitions for the full screen editor commands (cursor movement definitions are on the previous page).

80 COLUMN 40 COLUMN

DELETE CHARACTER

DELETE Left Arrow*
Deletes the character to the left of the cursor. If the cursor is currently at the beginning of the line, then the editor will assume that the user means to delete the carriage return at the end of the previous line. The current line and the previous line will be combined, and the cursor will be placed at the old end of the previous line.

DELETE TO BEGINNING OF LINE

-DELETE CTRL-Q Left Arrow*
Deletes characters from the beginning of the line through the character to the left of the cursor. The remainder of the line will be moved to the left.

DELETE TO THE END OF LINE

-DELETE CTRL-Q CTRL-Y
Deletes characters from the current cursor position to the end of the line.

QUIT THE FULL SCREEN EDITOR

ESC CTRI-K CTRL-D
CTRL-K CTRL-Q
This command quits the Full Screen Editor and returns to the Standard line editor. Any text that is currently in the text buffer will be re-loaded into the line editor, with line numbers added to each line if the text does not already contain line numbers.

INSERT / OVERWRITE

-0 CTRL-V
This command is another toggle, switching the editor between Insert and Overwrite modes of operation. The editor "wakes-up" with overwrite mode selected (as can be seen on the bottom status line). The overwrite cursor is the underline character. While overwrite mode is active, any characters that you type will replace whatever character the cursor is currently on (except for the carriage return character at the end of a line). If the cursor is at the end of a line, then any characters that you type will effectively be inserted ahead of the terminating carriage return.

When Insert mode is selected, the cursor character changes to the caret ("^") character, and any characters that you type will be inserted at the current cursor position, moving any characters at and to the right of the cursor over one position to the right. If you press RETURN while in the middle of a line, the cursor will be moved down a line and to the left margin, and the portion of the line at and to the right of the cursor will be brought down as well.

The current setting of the Insert/Overwrite switch can be seen on the status line.

NEW
(Clear Text Buffer)

-0 CTRL-K CTRL-N*
This command will clear any text from the text buffer and set the search string to null. It will then clear the active window, and lace the cursor in the upper left corner of the window. It also removes the current file name from memory, and selects overwrite mode. If no file is in memory when the editor is entered, this is the state that is set when the editor "wakes-up."

LOAD A FILE

-1 CTRL-K CTRL-L*
This commands clears any text from the text buffer, and then will prompt you for the ProDOS pathname of a file to load. The file MUST be a TEXT type file (TXT). If it isn't you will receive an error message.

If you have previously loaded a file, the system will place the file name of this file on the screen for you automatically. If this is the file that you wish to re-load (if, for example, you really botched up the file and want to start over again), simply press the return key, If you want a different file altogether, press CTRL-X to remove the old file name, and enter the new file name. The left arrow key or the delete key can be used to correct any typing errors. If you initiate this command by accident, you can press CTRL-C to return to the editor with your current file intact.

SAVE A FILE

-1 CTRL-K CTRL-S
This command will prompt you for a ProDOS pathname to save the current text. If you have previously used the Load command to load a file into the buffer, the system will place the current file name on the prompt line for you. You have the same options here as you did when you loaded the file. Use caution with this command. If a file already exists on the disk with the same file name, the editor will replace whatever was previously in the file without any warning message.

CUT LINE

-2 CTRL-K CTRL-X*
This command will remove the current line from the text buffer and place it on the clipboard (just a temporary holding area). The line will remain on the clipboard until you either Cut another line, or Copy a line. This line can be pasted from the clipboard back into the main text buffer with the Paste command. This command will only work with entire lines. There is no way to Cut in increments of less than, or more than, a single line.

(This command actually does a COPY, and then a DELETE).

PASTE LINE

-2 CTRL-K CTRL-V*
This command will copy whatever line is currently on the clipboard into the current position within the main text buffer. The current line will be moved up in the buffer (down on the screen) to make room for the new line. This action does NOT remove the line from the clipboard. Therefore, you can paste the same line as often as you like, into as many places in the text as you like.

COPY LINE

-3 CTRL-K CTRL-C*
This command will make a copy of the current line to the clipboard. It does not remove the line from the main buffer. You are then free to paste this line to your liking.

REPLACE LINE

-3 CTRL-K CTRL-R*
This command will replace the current line with the line that is currently on the clipboard. If there is no line currently on the clipboard, no action will be taken.

INSERT LINE

-4 CTRL-N
This command will insert a carriage return at the current cursor position without moving the cursor. If the cursor is at the beginning of a line, then the current line will move down on the screen, leaving a blank line for you to enter a new line on. If the cursor is in the middle of a line, the portion of the line at and to the right of the cursor will be moved down to the next line, and the cursor will remain at the (new) end of the current line.

DELETE LINE

-4 CTRL-Y
This command deletes the current line from the text. No copy of the line is retained in memory. Therefore, this is not a reversible command. Use it with caution.

FIND

-5 CTRL-Q CTRL-F
This command will allow you to enter a character sequence of up to 30 characters. The editor will then search from the current position to the end of the file for the character sequence. If it can't find the search string, a message will be displayed on the last line to this effect, and the cursor position will not change. If the string is found, the line containing the string will be placed at the current cursor position, and the cursor will be placed at the beginning of the string.

FIND NEXT OCCURRENCE

-5 CTRL-L
This command searches for the last search string that was entered using the FIND command. This command operates in exactly the same way as the find command, except that it does no prompt for the search string. (As a matter of fact, the find command prompts for the search string, and falls through to this command.)

SET TAB STOP

-6 CTRL-K CTRL-I
This command will allow you set the size of the tab stops. The editor will prompt you for the new value on the bottom line of the screen. The last part of the prompt is the current value of the tab setting, which has a default of 16. To leave this setting, simply press the RETURN key. To change it, enter the new value. The editor uses the same tab value as the rest of the ZBasic system, so this command accomplishes the same thing as the DEF TAB statement in a ZBasic program. This also implies that if the value is changed here, then the value will be changed for the rest of the system as well.

This tab value is used in the screen editor whenever you press the TAB key (CTRL-I for][+ users). The cursor will be positioned to the next calculated tab stop on the screen. If the next tab stop is beyond the end of the line, the cursor will be placed AT the end of the line. If you continue to press the TAB key, the cursor will move to the beginning of the next line (the absolute first tab position on the screen), and then continue normally.

AUTO TAB ON/OFF

-6 CTRL-Q CTRL-I*
This command is simply a toggle to turn the autotab feature on or off. The editor starts with autotab on. The current setting of autotab can be seen on the status line at the bottom of the screen. Autotab is a feature that will allow you to enter nicely formatted source code. When insert mode is on, and RETURN is pressed at the end of a line, the cursor will be moved down to the next line, and then spaces will be inserted in the new line until the cursor is in a position underneath the first non-space character in the line above. If insert is off (Overwrite mode), autotabbing is only operable when you are entering text at the end of the file. This is so that spaces are not inserted ahead of any existing text.

RESTORE LINE

-7 CTRL-K CTRL-F*
This command will restore the line at the current cursor position, deleting any changes that you have made to the line. This will only work if the cursor has NOT moved off the line since you made the changes, and/or the screen has not scrolled sideways. Normally, while you are editing a line, you are actually editing a copy of the line in an edit buffer. When the cursor is moved off the line, or if the screen is scrolled either left or right, this edit copy of the line is moved back to the main text buffer prior to moving the cursor. If you have made some changes to a line, and then change your mind, an old copy of the line still resides in the main buffer. A

copy of this old line is placed into the edit buffer over any changes that you might have made when you invoke this command.

PRINT LINE

-7 CTRL-K CTRL-P
This command will print the contents of the text buffer to your printer. This command accomplishes the same operation as the line editor's LLIST command without any parameters. The entire contents of the text buffer will be printed; no provision is provided for printing only a portion of the buffer. If no printer is connected in the slot that is currently configured, the system may "hang". Press CTRL-RESET to warm start the editor.

SCROLL DOWN

-8 CTRL-W
This command will scroll the screen down, with the cursor remaining in the same screen position (which means that it will be on the previous text line). If the cursor is currently within the first page of the text, the screen will not scroll, but the cursor will move up a line. The cursor will NOT move past the first line of the file.

SCROLL UP

-8 CTRL-Z
This command will scroll the screen up, with the cursor remaining in the same screen position (which means that it will be on the next line). If the cursor is currently within the last page of the text, the screen will not scroll, but the cursor will be moved down a line. The cursor will NOT move past the last line of the file.

FREEZE SCREEN FROM THAT LINE UP

-9 CTRL-K CTRL-T*
This command will freeze the top of the screen. A separating line will be drawn at the current cursor position, and the portion of the screen above this line will be frozen. The line that the cursor is in will remain within the new active portion of the screen (the active window). While the screen is frozen, the cursor will not move into the frozen portion, and no changes will be made within the frozen portion.

Only the screen is frozen. You can still move the lines that are in the frozen portion into the active window and make changes, but these changes will not appear in the frozen window.

To "thaw out" the window, press the control key sequence again. The separating line will be removed, and the screen will be refreshed, leaving the cursor at whatever position it was at.

FREEZE SCREEN FROM LINE DOWN

-9 CTRL-K CTRL-B*
This command freezes the bottom portion of the screen. A separating line will be drawn at the current cursor position, and the portion of the screen below this line will be frozen. The text line that the cursor was on will remain within the active window. This command is very much line the Freeze Top command. To "thaw out" the bottom window, press the command key sequence again.

These two Freeze command are entirely separate from each other. You can have up to two frozen windows on the screen; a top window and a bottom window, leaving a third, active window in the middle of the screen between the two frozen portions.

This can be handy if you have a couple of different subroutines in a couple of different sections of your program, while accessing those subroutines in a third portion of your program.