KeyDOS

Function ROM for the Commodore 128

Version 2

Installation Guide

User's Manual

& Program Notes

Antigrav Toolkit

P.O. Box 1074, Cambridge, MA 02142

KeyDOS ROM Version 2 User's Guide

Written by Randy Winchester

FIRST EDITION

For customer service, write to:

Antigrav Toolkit
P.O. Box 1074
Cambridge, MA 02142

Warranty Information

Satisfaction Money Back Guarantee: If you are not totally satisfied with the KeyDOS ROM for any reason, return it, with all original packaging and manual, for a full refund.

Lifetime Replacement: If your KeyDOS ROM fails to operate, it will be replaced at no cost.

Upgrade Policy: Users will be notified by mail of all major KeyDOS ROM upgrades. Upgrades will be made available for a small price to cover the cost of materials and postage.

Credits: Thanks to all the people who helped make the KeyDOS ROM a real product. **Bug reports:** Joe Buckley. **ROM Guru**: Kent Sullivan. **Testers:** Rob Knop, Geoff Sullivan. **Programming:** Geoff Sullivan, *KeyBoot V2*; Eric Trepanier, *Hexpert*; and all the exceptional programmers credited in the KeyDOS documentation who wrote routines that were adapted for this project. **Documentation:** Loren Lovhaug, *RAMDOS*; Eric Trepanier, *Hexpert*. **Quick Brown Box Compatibility Consultant:** Barbara Mintz.

Table of Contents

KeyDOS ROM Specifications

The KeyDOS ROM is a 32K ROM (Read Only Memory) chip that contains new function key definitions and utilities. Here is a partial list of major features:

* Compatible with most C128 hardware and software including JiffyDOS, The Quick Brown Box, CS-DOS, and even other function ROMs.
* RAMDOS support for Commodore RAM Expansion Units from 128K to 2M.
* Full support for the creation of 1581 disk drive partitions and subdirectories.
* 20 KeyDOS Function Keys simplify disk operations on multiple drive systems.
* Load C64 programs in C128 FAST mode with one key press.
* Type SEQuential files without disturbing RAM.
* Support for batch files.
* Select multiple files from a directory list for typing, printing, copying, renaming, and scratching.
* Support for most serial bus printers.
* Convert text files between ASCII and CBM ASCII or CBM Screen code to CBM ASCII. Type CBM ASCII, CBM screen code, or ASCII files with full ASCII character set.
* GEOS SuperRBoot reboots GEOS 128 from REU.
* Edit disk sectors.
* Dump monitor output to printer or disk file.
* Advanced machine language debugger.
* Renumber or reset disk drives.
* New COLLECT command protects autoboot sectors.
* Unnew BASIC programs.
* Dual 80 column screens, screen editor, and screen dump.
* Find/Replace utility.
* Screen clock with calendar and alarm. Special versions of KeyDOS ROM that automatically set the time and date from a SmartWatch real time clock or CMD Hard Drive are available.
* Set VDC video options.
* Utility disk with sample batch files, demonstration programs, and support utilities.

Installation

The KeyDOS ROM is easily installed in one of two ways. It can be installed internally in the C128's empty ROM socket. If you own a Commodore 1700, 1764, or 1750 RAM Expansion Unit, it can also be installed in an empty socket on the REU board. The KeyDOS ROM functions identically in either location. If you will be using the KeyDOS ROM with another C128 function ROM, you must install the KeyDOS ROM in your REU. On the other hand, if you use a Quick Brown Box battery backed RAM cartridge, you must install the KeyDOS ROM in the C128's empty socket.

You should choose a clean, static free area to do the installation. The C128, REU, and KeyDOS ROM can all be damaged by static electricity. If you take reasonable precautions, such as discharging any static from your body to a cold water pipe, not walking across carpet, stay away from switched on monitors, and wear clothing that doesn't generate much static, there is little risk to your equipment. If you have a wrist grounding strap, (available from Radio Shack) wear it and follow the instructions on the package.

Keep in mind that opening your C128 or REU will void the warranty. If this is a concern, you should wait until after your warranty expires before opening your equipment.

Before disassembling your C128, remove all peripherals, power connectors, and anything else that might be connected to it.

Internal Installation - Flat 128 Disassembly

Turn the case over and remove the three Philips screws located along the front of the case. Very carefully lift the top of the case. Disconnect the small connector to the power indicator LED located on the left side of the case noting the direction the plug faces. It is usually not necessary to remove the keyboard connector on the right side of the case, but if you do remove it, make sure you note which way the plug is oriented. The top cover can now be tipped up on its right edge. Remove the screw holding the keyboard grounding strap to the front right corner of the circuit board. You can now lay the keyboard down on the right side of the computer. Remove the rest of the screws from the perimeter of the metal shield covering the circuit board and one large screw from the center of the shield. The shield is held in place by small metal tabs that are bent over its edge. Using a pair of pliers, carefully bend these tabs back. The shield is also soldered in one place. Use a soldering iron to melt this solder. Lift the shield off the board and set it aside.

C128D Disassembly

Remove the screws securing the top cover and slide it off.

Installing the ROM

Locate the empty 28 pin socket on the left side of the circuit board. It should be labeled U36 and have an indication of the orientation of the chip drawn on the board underneath it. Carefully insert the KeyDOS ROM in this socket, making sure the notch on the end of the chip matches the notch on the drawing under the socket. Make sure all pins on the ROM line up with the holes on the socket. Gently push the chip into the socket until it is seated. Check to see that all the pins on the ROM are inserted into the socket and didn't get bent or miss a hole.

Temporarily replace the top cover, keyboard connector and power LED connector. Proceed to the Testing section.

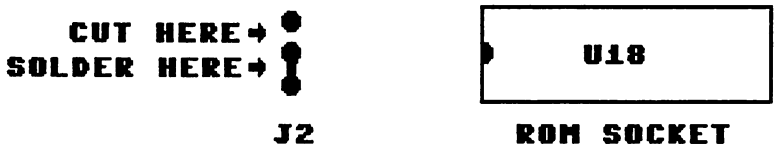
REU Installation

To install the KeyDOS ROM in empty socket inside a Commodore 1700, 1764, or 1750 RAM Expansion Unit, first open the case by carefully prying the two halves of the case apart with a small flat screwdriver or other suitable tool. The case is held together by plastic pins. The top of the case has to be lifted straight up in order not to break any of the pins. It is probably best to pry the case apart slightly in the front starting near the connector, prying both sides of the case up while working your way toward the back of the case. You'll probably have to repeat this process several times before the top of the case works its way free. Lift off the top of the case and set it aside.

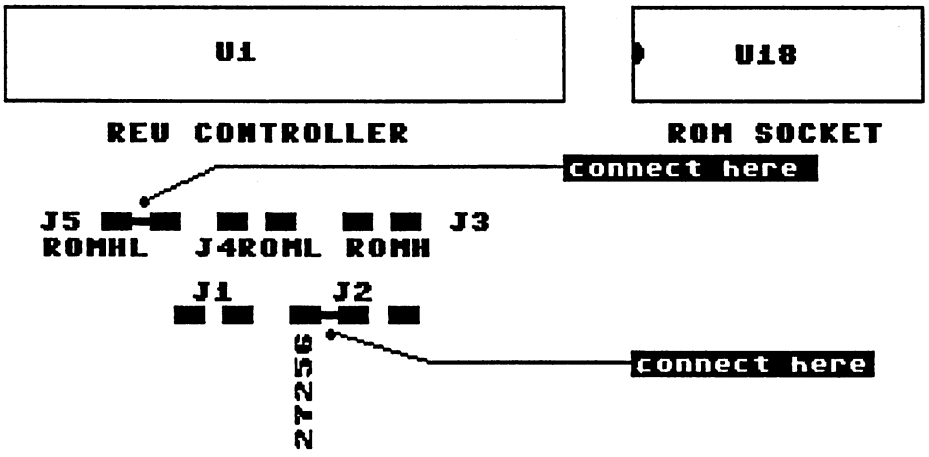
The REU circuit board is enclosed in a metal shield. Remove the circuit board and shield from the bottom of the case by lifting it straight up. Remove the shield from the circuit board by prying it apart and lifting the top half up. Remove the circuit board from the shield.

Solder a suitable 28 pin socket into the empty space available for it on the circuit board, labeled U18. Use a low wattage soldering iron, good electronics solder, and follow precautions for static electricity.

Two different versions of the REU board are known to exist. If your REU U1 controller is a square chip the J2 jumpper is located between U1 and the U18 ROM socket. Locate Jumpper J2 which is located to the left of the ROM socket. Using a sharp knife, cut the trace between the top and middle pads by carefully scraping it away. Connect the bottom and middle pads by putting a small drop of solder across them. Now insert the KeyDOS ROM in the socket following the instructions for Installing the ROM above.



If the U1 controller in your REU is a rectangular chip, then the jumppers will be located near the connector end of the board. Two jummpers on this version of the REU must be connected. Connect the two pads between **J5**. Locate jumpper **J2**. Connect the center pad of **J2** with the pad to the left of center (labeled 27256). Refer to the illustration below.Replace the circuit board in the bottom of its plastic case and temporarily set the top cover back in place. Plug the REU back into the computer.



Testing

Plug in the power supply and monitor and turn on the computer. The following screen should appear:

```
HOLD ALT KEY DURING RESET OR
SYS 65366
TO ACTIVATE KEYDOS ROM.
```

Success! Switch your computer off, unplug your equipment, and put everything back together.

If you don't see the power up message, immediately switch off your computer. Double check all your work. Make sure especially that all pins on the ROM line up with all the holes in the socket. Also check to see that the notch on

the ROM is facing the same direction as the drawing on the circuit board under the socket. If the ROM still doesn't work, contact Antigrav Toolkit at the address listed inside the front cover of this manual.

Activating the KeyDOS ROM

With the ROM installed, every time the C128 is switched on or reset, the sign-on message shown above will appear. This will serve not only to remind you of how to activate the ROM, it also types the command on the screen for you. Simply position the cursor over the **SYS 65366** line, hold down the **ALT** key, and press **RETURN**. KeyDOS ROM will activate and print the following message:

```
KeyDOS ROM activated!
Copyright 1992 by Randy Winchester.
Press ESC ← for help.
```

The KeyDOS ROM is at your service. Pressing the ESC key followed by the ← key produces the KeyDOS help screen:

```
Function Keys:
F1 LOAD Program        F2 RUN Program
F3 Disk Catalog        F4 RUN C64 Program
F5 Type SEQ file       F6 Scratch file
F7 New active drive    F8 Scratch & Save
RUN BOOT disk/file     HELP drive # + RET

ESC Keys:
←  KeyDOS ROM Help     1  KeyDOS Utility
2  KeyDOS Compiler     3  1581 Partitioner
4  Subdir key on F8    5  Batch exec on F8
6  CBM RAMDOS          7  GEOS SuperRBoot
8  Diskmon             9  Monitor dump
0  Hexpert             +  Renumber drives
-  Reset Drives        £  New Collect
*  Unnew Program       ↑  Alternate Screen
:  Find / Replace      ;  Screen Dump
=  One Drive Copy      ,  Screen Edit
.  Clock Manager       /  Video Manager

KeyDOS and KeyDOS Utilities
Copyright 1992 by Randy Winchester.
```

To select ESC commands, first press the ESC key followed by the character shown on the help screen. Do not hold the ESC key down while pressing the second key. Whenever a command is shown in this manual, it is listed as ESC x, where x is the character to press after pressing the ESC key.

KeyDOS Function Keys

KeyDOS is a new DOS interface for the Commodore 128. It was designed for ease of use and maximum compatibility with all C128 software. KeyDOS reprograms the function keys to provide a quick and simple method to minimize typing and access all disk drives on a multi-drive system.

The inspiration for KeyDOS came from using the C128's CP/M mode. With CP/M, you log onto a drive and that drive becomes the current drive. When you type a command, such as **DIR**, it will show a directory of the current drive.

That's not the case with C128 mode. To get a directory listing of a disk in drive 8, type **CATALOG**. To get a directory of a disk in any other drive, the command is different. For drive 9 it is **CATALOG ON U9**. This seems like a lot of extra typing. What is even less desirable is that the power-up function key definitions are only minimally useful on a multi-drive system. The string **DIRECTORY** is assigned to the **F3** key. The only thing that key is good for is listing the directory on drive 8. KeyDOS function keys operate on a current drive. Changing the current drive is as simple as pressing a function key and entering the drive's device number.

KeyDOS uses the C128's screen editor, to enhance the way function key definitions work with DOS commands. It makes it possible to the cursor over a command on any line of the screen, hit return, and send the command to BASIC.

KeyDOS provides a simple to solution to many common DOS problems. Now you can easily log onto any drive, scratch files, load and save programs and more, all with the function keys. The same commands work the same way with all drives on a multi-drive system. Additionally, KeyDOS commands work with the output of the CATALOG command for fast filename selection with very little typing.

Warning: Before using KeyDOS on your important disks, please read all the documentation and practice using KeyDOS with some disks you don't care about. Make sure you are familiar with KeyDOS before using it.

Running KeyDOS

KeyDOS Function Keys are installed automatically when the KeyDOS ROM is activated.

KeyDOS Commands at a Glance

Key	Function
F1	DLOAD a Basic program from a directory listing
F2	RUN a Basic program from a directory listing
F3	CATALOG listing from the active drive
F4	RUN a C64 Basic program from a directory listing
F5	Type a sequential text file from a directory listing
F6	SCRATCH a file from a directory listing
F7	Select a new disk drive
F8	SCRATCH then DSAVE a program from a directory listing
RUN	BOOT disk in the active drive / file from a directory listing
HELP	PRINT the string "ON U (active drive) <RETURN>"

Using KeyDOS Function Keys - A Quick Demonstration

After activating the KeyDOS ROM, place a disk with files in drive 8 and press the **F3** function key. Normally, this key prints **DIRECTORY** and displays a directory listing of the disk in drive 8. You'll notice that KeyDOS's **F3** key prints **cAu(peE(186))** and also produces the exact same directory listing. Big deal . . .

Here's where the fun begins. If you have more than one disk drive, make sure it is turned on and has a disk in it. Now press **F7** and enter the number of the drive you wish to work with and press **RETURN**. Press **F3** again. The directory listing is now from the newly selected drive. All KeyDOS commands will continue to use the last drive selected or used. Once you use a drive, KeyDOS will use that drive until you select or use a new drive.

Place a disk containing a SEQ text file in one of your disk drives. Press **F3** to get a directory listing of the disk. Now use the **CRSR** up key to move the cursor to the screen line listing the text file. Press the **F5** key, and the text file will be printed to the screen. Use the **NO SCROLL** key to pause the listing. In this example, the text file could be on any disk drive in your system. The **F5** key prints a text file from a directory listing of the last used or selected drive, and it does so without disturbing anything in memory.

Other KeyDOS commands operate in a similar way. Use **F1** to DLOAD a Basic program. Once more, move the cursor key to the line in a directory listing (**F3**) containing the name of the file to load and press **F1**. Use **F2** to RUN a Basic program. The **F4** key is used to run a C64 Basic program from a directory listing. Use **F6** to SCRATCH files you no longer want. The computer will print the prompt **Are you sure?** Press **y** and **RETURN** to scratch the file. The **F8** key simulates the SAVE@ or SAVE with REPLACE command. To test it, first DLOAD a Basic program with the **F1** key. If you make changes to the program and wish to save it with the same file name, move the cursor to the line in a directory listing containing the name and press **F8**. KeyDOS will first scratch the original file, then save the new file with the same name. Be very careful with this key. It's possible to press it over the wrong file name and unintentionally replace a file with a different one from memory. If this idea frightens you, please disable this key definition by using the KeyDOS Compiler to substitute another definition for it. 1581 users might want to substitute the Subdirectory key definition by pressing **ESC 4**.

KeyDOS also reprograms the **RUN** and **HELP** keys. The **RUN** key, selected by pressing **SHIFT-RUN/STOP**, is a dual purpose key. If pressed on a blank screen line, it will BOOT an autobooting disk. The other function of the **RUN** key is to BLOAD and execute binary programs. Just press **RUN** while the cursor is on a line with the filename of a binary program from a directory listing. The binary program must be one that can be started by the BOOT command.

The **HELP** key is now truly helpful! KeyDOS places the most frequently used DOS commands on the function keys and RUN key, but there are just as many DOS commands that can't conveniently be included. For example, suppose you are working on a new program and would like to save a copy on drive 9. You've used KeyDOS and have just viewed the directory of drive 9 with the **F3** key. Just type DSAVE "program name" and press the **HELP** key. **HELP** adds the KeyDOS extension **ON U(PEEK(186))"+CHR$(13)**, which translates to the active drive number and **RETURN**. Follow commands such as DSAVE, DVERIFY, COLLECT and RENAME with the **HELP** key in order to make them access the active drive and to save yourself as much typing as possible.

1581 Subdir Key (ESC 4)

Here's a single key definition that makes it easy to select 1581 subdirectories. Many 1581 owners have been reluctant to use subdirectories due to the inconvenient commands needed to manipulate them. Now, for the first time, opening subdirectories is as easy as moving the cursor to the line in a directory listing with the name of a subdirectory and pressing **F8**.

To install the Subdir Key on **F8**, simply press **ESC 4**. To install it on any of the other function keys, use the KeyDOS Compiler.

The 1581 Subdir Key works similarly to other KeyDOS keys. It gets it's input from a directory listing, and always references the current or last used drive. Press **F7** and enter the device number of a 1581 disk drive on your system. Insert a disk that contains one or more subdirectories. Press **F3** to get a directory listing. Subdirectories appear in the directory listing as files with a file type of CBM. Move the cursor to the line containing a subdirectory name, and press **F8**. The subdirectory will be opened. Pressing **F3** for a directory listing should confirm that you're now in a subdirectory.

To get out of a subdirectory, simply move the cursor to a blank line on the screen and press **F8**. Press **F3**, and you'll see from the directory listing that you're now in the root (top level) directory.

Batch File Exec Key (ESC 5)

CP/M and MS-DOS operating systems provide a feature called batch execution which can run commands from a text file. This feature has been mostly missing from the C128, that is until now. KeyDOS 128 includes a batch execute command with all the convenience of the point and click KeyDOS user interface.

What are Batch Files?

Batch files are SEQuential text files containing BASIC direct mode commands. They can be created with any text editor that can create PETASCII SEQ files. You can use the SEQ file write feature of most word processors to make these files. Another method is to edit the batch file as if it were a BASIC program, with line numbers at the beginning of each line. To save the file to disk as a SEQ file without line numbers, use the command:

```
OPEN 1,(drive number),2,"filename,S,W":CMD 1:POKE 24,37:LIST:PRINT#1:CLOSE 1:POKE 24,27
```

Use the program **WRITE-SEQ-FILE** on the KeyDOS disk to install the above command on a function key. Batch files can contain most BASIC commands that can be used in direct mode, that is, outside of a program. Several batch files are included on the utility disk, and demonstrate some programming methods and techniques. Since so little has been done with batch files in the past, there's still a lot of territory to explore, so experiment. Interested hackers might want to read the article *SYS 65478: Taking a New Look at an Old Dog* by Miklos Garamszeghy, published in Volume 8, Issue 2 of *The Transactor*, which was the inspiration for this KeyDOS key.

Batch File Hints and Tips

Since batch files are not memory resident, they don't alter programs in memory unless you want them to. To test this, simply load a program, then use the Batch File Exec key to execute a batch file. When the batch file has finished executing, you'll find that the program has been left intact.

Batch files execute as they are read from disk. To speed things up, include the FAST command in the first line of your batch files.

After each line of a batch file executes, the operating system responds with the usual READY prompt and a carriage return. To keep the screen from getting cluttered, define: UP$=CHR$(145)+CHR$(145)+CHR$(145) and print an UP$ at the beginning of each line.

Batch files can be made to wait for user input. Placing the commands WAIT 208,1:POKE 208,0 at the end of a line will make execution wait until any key has been pressed. The ASCII value of the key pressed can be recovered with PEEK(842).

Batch Exec can also tokenize a BASIC program that has been saved as a SEQuential text file. This feature also can be used to merge a program in memory with a program saved as a text file. As each line is read in, it is entered as if it were typed in. To save a program as a text file, use:

OPEN 1,(drive number),2,"filename,S,W":CMD 1:LIST:PRINT#1:CLOSE 1

Use the program **WRITE-LIST-FILE** on the KeyDOS disk to install the above command on a function key. To make sure batch files finish in an orderly manner, end them with the line:
SLOW:POKE 153,0:DCLOSE ON U(PEEK(186)).

Installation

To install the Batch Exec Key on **F8**, simply press **ESC 5**. To install it on any of the other function keys, use the KeyDOS Compiler.

Using the Batch File Exec Key

The Batch File Exec Key works similarly to other KeyDOS keys. It gets it's input from a directory listing, and always references the current or last used drive. Press **F7** and enter the device number of drive on your system that contains batch files. Press **F3** to get a directory listing. Move the cursor to the line containing a batch file name, and press **F8**. The batch file will be opened and executed one line at a time. There are some examples of batch files on the KeyDOS ROM utility disk. Try some of them to see what they do, then print them to see how they were written.

Reload KeyDOS Function Keys (ESC TAB)

Pressing **ESC TAB** reloads the default KeyDOS function key set. This is useful for returning the KeyDOS function key definitions after running software that reprograms the function keys.

KeyDOS RAMDOS Compatibility

KeyDOS is completely compatible with Commodore RAMDOS. RAMDOS can be activated by pressing **ESC 6**. KeyDOS treats the RAM disk the same as any other drive in the system. Simply use **F7** and enter the number of the RAM disk to make it the active drive.

Quirks, etc.

It's possible to generate error messages while using KeyDOS keys. Usually these won't cause any problems and will be obvious errors. For example, press **F3** while on a screen line that contains 20 or more characters. Usually this will generate a SYNTAX ERROR, which is the same thing that would happen if you tried this stunt with the normal **F3** DIRECTORY key.

Pressing **F1**, **F2**, **F4**, **F5**, **F6** or **F8** while not on a directory listing line with a file name will usually produce a FILE NOT FOUND error. This is not a big problem. Simply list a directory, position the cursor, and try again.

It's also possible to select files that aren't on the current drive. This could happen if you changed the current or last used drive after listing a directory. If you had a catalog listing on the screen from drive 9, then scratched a file from drive 8, KeyDOS would look for all selected files on drive 8. This is another easy way to generate FILE NOT FOUND errors.

Another possible error is DEVICE NOT PRESENT. This could be caused by a number of conditions, but is usually not a problem. Any access to printers, modems or file I/O with the screen or keyboard will change the device number used by KeyDOS to a value illegal for DOS commands. It's also possible to press **F7** and enter an outrageous value, say 170. Again, KeyDOS won't function properly, but this won't cause any real problems either. The solution is to press **F7** and enter a valid drive number before proceeding.

KeyDOS Utility V2.3 (ESC 1)

KeyDOS Utility V2.3 is a multi-function program that performs all common DOS functions on selected multiple files. Similar to the KeyDOS function key definitions, KeyDOS Utility V2.3 works with any disk drive with any device number, including an REU with Commodore RAMDOS. Multiple files can be selected from a catalog listing, similar to KeyDOS function keys.

Warning: Practice using this program with some disks you don't care about. Get familiar with this program before using it regularly. As with all software, you should read the documentation and practice using the software before using it with your important disks.

Running KeyDOS Utility V2.3

While in 80 column mode, press **ESC 1**. KeyDOS Utility prints it's startup message on the top right corner of the screen:

```
KeyDOS Utility V2.3
Copyright 1992 by Randy Winchester
```

The initial prompt appears below the message and allows the selection of files for the file list.

```
Select: 8>? "*
```

The Select prompt appears at various points throughout operation of KeyDOS Utility, usually after a function such as scratching or renaming files alters the disk directory. The number 8 in the prompt refers to the active disk drive. KeyDOS Utility defaults to the disk drive that was last used. The cursor is positioned over the asterisk. There are three choices at this point:

1) **RETURN** will provide a selection list of all files on the disk.
2) Any valid CBM DOS pattern can be entered to selectively list some of the files on the disk.
3) Typing a **space** to erase the asterisk followed by **RETURN** will result in a menu with the following choices:

```
Catalog Drive Quit RETURN
```

Pressing the first letter of the command name will execute the command:

C - Presents the prompt `Catalog: 8>? "*`
Press **RETURN** for a standard catalog listing of all files, or enter any valid CBM DOS pattern for a selective listing. Use the **NO SCROLL** key to pause a listing, or the **STOP** key to return to the Select prompt.

D - Presents the prompt `Drive?`
Enter the drive number of an alternate disk drive and press **RETURN** or press **RETURN** by itself to keep the same active drive.

Q - Quit KeyDOS Utility and return to BASIC.

RETURN or any other key will return to the Select prompt.

CBM DOS Wildcard Pattern Matching

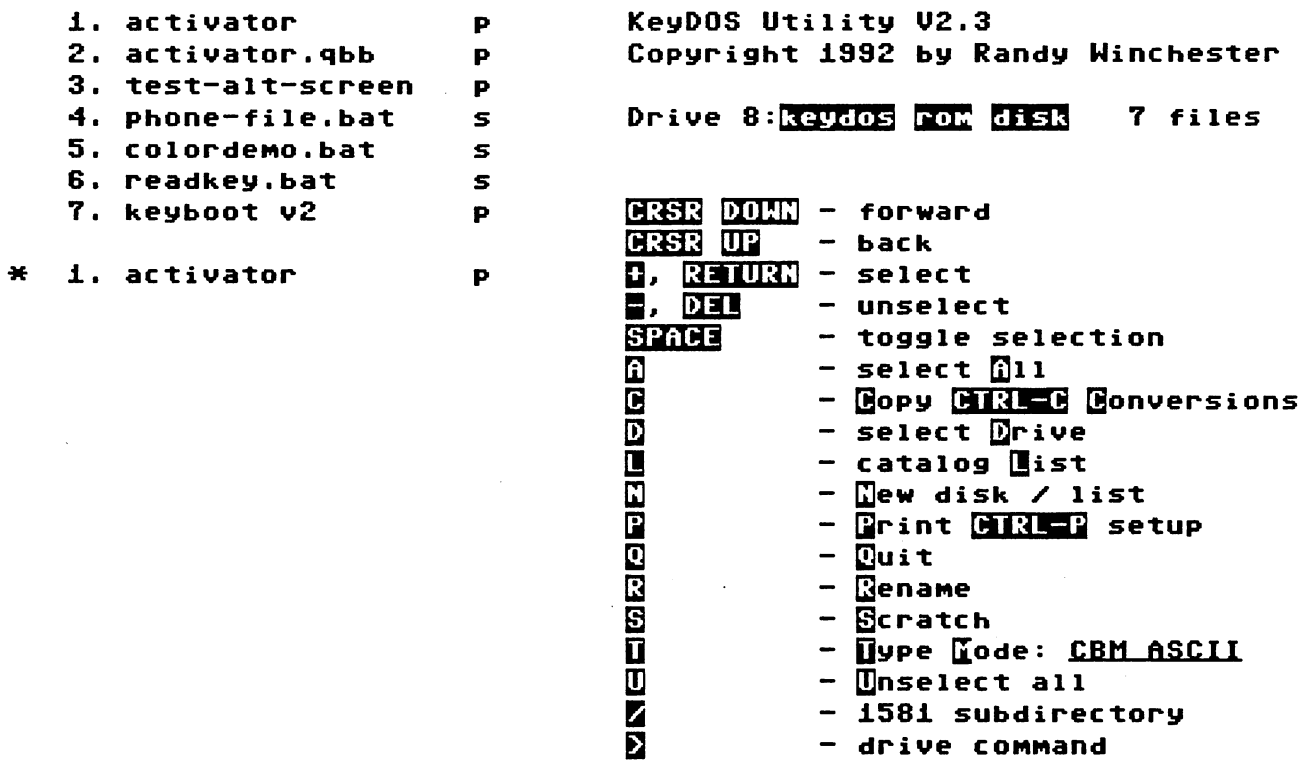
The Select and Catalog prompts accept a pattern with wildcards for file matching. This can help limit the selection list or catalog display to just the files you wish to work with. Here are some examples:

abc*	files beginning with "abc" (* matches any number of characters)
ab?	file names with three characters beginning with "ab" (? matches a single character)
a*,n*	files beginning with "a" or "n" (multiple patterns may be combined)
???de	five character files ending with "de" (? wildcards may appear in any position)
*=p	all program files (CBM file type pattern matching)
*=s	all sequential files (CBM file type pattern matching - use *=u for USR and *=r for REL)
n.*=s	sequential files beginning with "n." (standard pattern with CBM file type pattern matching)
*.doc	1581 only - files ending with ".doc" (* may appear in any position - 1581 only)

Up to four different patterns separated by commas can be entered. Pattern matching using the "*=p" format for selecting specific file types will not work with RAMDOS. For more information on pattern matching, consult the 1571 or 1581 User Guides.

If there are no files on the disk or no files matching the pattern you've entered, the message **No files** will appear in reverse video in the upper right corner of the screen and the Select prompt will be redisplayed.

After selecting files from the Select prompt, you'll see the KeyDOS Utility main screen:



The line replacing the Select prompt shows the active drive followed by the name of the disk and the number of files matching the Select pattern.

On the left side of the screen is the file selection list. Each file is numbered. Following the file name is a letter indicating the file type; c for CBM, p for PRG, r for REL, s for SEQ, and u for USR. The flashing asterisk before file #1 is a cursor for selecting files. The menu is listed on the right side of the screen under the status line.

Cursor Movement and File Selection

Use the **CRSR** up and down arrow keys to move the flashing asterisk cursor up and down the file selection list. As the cursor goes past the end of the list, the file names repeat themselves in sequence. Moving the cursor up the screen will repeat the file names in reverse order. If the cursor is on the top line of the screen, pressing the cursor up key will scroll the file selection list downward.

To select a file for an operation press + or **RETURN**. The file will be selected and the cursor will move to the next file name. Selected files are displayed in reverse video. Pressing the **A** key will select all files in the list. Pressing **SPACE** will select a file without moving the cursor. Pressing **SPACE** again will deselect the file.

To deselect a file, press - or **DEL**. The file name will be redisplayed in normal video and the cursor will move to the next file name. To unselect all selected files, press the **U** key. **SPACE** will unselect a single file without moving the cursor.

C - Copy

To copy selected files to another disk drive, press the **C** key. The status prompt is replaced with:

```
Destination?
```

Enter the number of the drive you want the files copied to. As each file is copied, the status line will display:

```
Copying: filename,p
```

followed by the file name, a comma, and the file type.

KeyDOS Utility will copy all standard CBM SEQ, PRG, and USR files. It will not copy REL or GEOS USR files. KeyDOS Utility can also copy files to or from open 1581 subdirectories. Read the following section on how to select subdirectories on 1581 disks for more details.

CTRL-C - Conversions

Pressing **CTRL-C** (press the CONTROL key and C at the same time) will display the conversion submenu:

```
Copy conversion:
*1) None
 2) ASCII to CBM ASCII
 3) CBM ASCII to ASCII
 4) Screen to CBM ASCII
```

The default selection is **1) None** and is highlighted. Normally, all files are copied exactly without any changes. The file conversion options are recommended for converting SEQ text files from CBM ASCII (also known as PET ASCII) to or from the standard ASCII used by most other computers. Often, files downloaded from BBS systems, copied from MS-DOS or CP/M disks, or transferred from GEOS will be written as standard ASCII files. If ASCII files are viewed with the KeyDOS F5 key, they will appear with the cases of the characters reversed - capital letters will be exchanged with lower case letters and vice versa. To translate ASCII files to CBM ASCII, select option **2) ASCII to CBM ASCII**, from the submenu. Press **RETURN** to go back to the main menu, then press **C** to copy the files. Files can also be converted from CBM ASCII to ASCII or CBM screen code to CBM ASCII by selecting option 3 or 4, pressing **RETURN**, then pressing **C** to copy files.

File conversion is not recommended for any file containing binary material, such as programs or other data files which are not specifically text files. If you are in doubt, view the file with the **T** - type file option. If the screen becomes scrambled, most likely the file is a binary file.

D - select Drive

Pressing **D** will display the **Drive?** prompt on the status line. Enter the number of the new drive you want to work with. If you'd like to keep the current drive active, simply press **RETURN** without entering a drive number. After selecting a new drive, the Select prompt will appear on the status line for that drive.

L - catalog List

Pressing **L** will display a standard CBM catalog listing. This will display the **Catalog:8>? "*** prompt on the status line. Press **RETURN** to see all files, or enter a CBM DOS pattern to see a list of pattern matching files. Use the **NO SCROLL** key to pause the list or the **STOP** key to stop listing. When the list has finished, press any key to return to the main menu and file selection list.

N - New disk / list

Use this command after changing the disk in the active drive or to enter a new CBM DOS pattern for a new file selection list. It is essential that this command is used whenever a different disk is placed in the active drive. This command is also useful to narrow down the search for specific files on a disk using DOS wildcard pattern matching. After pressing **N**, the Select prompt is displayed on the status line.

P - Print

KeyDOS Utility will print standard ASCII or CBM ASCII files on both CBM ASCII (Commodore serial bus direct connect) or ASCII (Centronics parallel interface) printers.

After pressing **P**, KeyDOS utility will give a quick preview of each of the selected files. The purpose of this display is so that ASCII translation can be selected if needed. The request **Translate? n** follows the text sample. If the text appears normal, simply press **RETURN** to select the default of no translation. If the cases of the characters are reversed (upper case characters appear where lower case characters should be and vice versa), answer **y** to the **Translate?** prompt followed by **RETURN**. After the translation request is answered for the last selected file, KeyDOS Utility will print each of the files.

CTRL-P - setup

This presents a submenu where you can set up KeyDOS Utility for your printer. The default printer is an ASCII printer, device number 4, secondary address 7 (upper/lower case text mode). If you need to change any of these settings, enter the new value and press **RETURN**. Values of 4 or 5 are accepted for the device number. Any secondary address between 0 and 255 can be entered. Refer to your printer's (or printer interface's) manual for more information on secondary addresses. To select an ASCII printer, press **A**; to select a CBM ASCII printer, press **C**. After setting up the printer, press any key to return to the main menu and file selection list.

Q - Quit

Press **Q** to quit KeyDOS Utility and return to BASIC.

R - Rename

To rename selected files, press **R**. The file selection list will be replaced with the following display:

```
Rename: filename1
to?
```

Enter a new file name at the **to?** prompt to rename the file. To retain the same file name, press **RETURN** by itself without entering a new name. After renaming all selected files, the Select prompt is presented.

S - Scratch

To scratch selected files, press **S**. The status line will be replaced with the following display:

```
SCRATCH marked files! Are you sure?
```

Enter **Y** and press **RETURN** to scratch the selected files. As each file is scratched, the status line will display **Scratching:** and the name of the file. If you've answered yes, and KeyDOS Utility is scratching files but you've changed your mind, press the STOP key to stop scratching files. After scratching all selected files or pressing **STOP**, the Select prompt is presented.

T - Type, Mode

Press **T** to type selected files to the screen. After pressing **T**, the top line of the screen is replaced with a display of the file name and the available commands:

NO SCROLL - will pause file typing until any key is pressed.
SPACE - will stop typing the current file and start typing the next selected file. If a space is entered while the last file is being typed, you will be returned to the main menu and file selection list.
STOP - will stop all file typing and return to the Select prompt.

Press **M** to switch the display mode among CBM ASCI, standard ASCII, and CBM Screen code. Selecting ASCII mode downloads an alternate ASCII screen character set. The following characters are substituted:

Value	Name	ASCII	CBM ASCII
$5C	Backslash	\	£
$5E	Carat	^	↑
$5F	Underscore	_	←
$60	Grave Accent	‘	SHIFT *
$7B	Left Brace	{	SHIFT +
$7C	Bar	|	C= -
$7D	Right Brace	}	SHIFT -
$7E	Tilde	~	SHIFT ↑

Screen code mode is useful for viewing PRG text files produced by SpeedScript, The Write Stuff, and many other popular CBM word processors. It can also be used to display most of the text inside binary programs data files, and converted GEOS programs and files. Keep in mind that screen code and binary files are not usually intended for screen display, and may not be formatted neatly when typed to the screen.

/ - 1581 subdirectory

KeyDOS Utility offers full 1581 subdirectory support. To open a 1581 subdirectory (file type CBM, listed as "c" in the file selection list), select the name of the subdirectory partition and press /. The subdirectory will be opened and the Select prompt presented. If more than one subdirectory partition is selected, only the first is opened. To return to the root partition, press **U** to unselect files and press /. The select prompt is then presented.

It is possible to bypass the / command and open or close subdirectory partitions with direct disk commands. This is not recommended, since the file selection list will not match the disk directory.

> - drive command

This command is used for miscellaneous DOS functions. It can be used for such things as formatting disks or changing modes on a 1571. Pressing **>** will clear the status line and give the prompt: **Command 8> "**. Enter the command followed by **RETURN**. The disk status is displayed on the status line after completion of the command. The following commands are known to work. Please refer to the manual for your disk drive for more information.

These commands work with all CBM drives:

RETURN	- show disk drive status (error channel)
n0:diskname	- erase disk
n0:diskname,id	- format (new) disk
c0:newfile=oldfile	- copy "oldfile" to "newfile" on same disk
c0:newfile=old1,old2,old3,old4	- combine files old1 - old4 in newfile
v0	- validate disk
i0	- initialize disk
uj	- reset drive

1571 Commands:

u0>m1	- select 1571 mode
u0>m0	- select 1541 mode
u0>h0	- select 1541 mode side zero
u0>h1	- select 1541 mode side one

KeyDOS Compiler (ESC 2)

The KeyDOS Compiler is used to define C128 function keys and save the definitions to a file. Different sets of function keys can be predefined and reloaded whenever needed. Additionally, keys that are defined before running the KeyDOS Compiler and are not redefined can be saved along with redefined function keys. This enables inclusion of custom key definitions along with KeyDOS keys.

Twenty KeyDOS key definitions are provided with the KeyDOS Compiler. Definitions 1 - 12 are documented in the section of this manual on KeyDOS Function Keys. Definitions 13 - 20 are new with the KeyDOS Compiler, and are similar in operation to other KeyDOS Function keys.

Running KeyDOS Compiler

While in 80 column mode, press **ESC 2.** You should then see the KeyDOS Compiler menu. The KeyDOS Compiler runs in 80 column mode only.

The KeyDOS compiler includes the following KeyDOS key definitions:

0	Null key (clears a key definition)
1	DLOAD a Basic program from a directory listing
2	RUN a Basic program from a directory listing
3	CATALOG listing from the active drive
4	RUN a C64 Basic program from a directory listing
5	Type a sequential text file from a directory listing
6	SCRATCH a file from a directory listing
7	Select a new disk drive
8	SCRATCH then DSAVE a program from a directory listing
9	BOOT disk in the active drive or file from a directory listing
10	PRINT the string "ON U (active drive) <RETURN>"
11	OPEN / CLOSE 1581 Subdirectory from directory listing
12	Execute batch (SEQ) file from directory listing
13	BLOAD file from directory listing
14	COLLECT on active drive
15	DCLEAR on active drive
16	DCLOSE on active drive
17	DSAVE with replace from directory listing
18	DVERIFY BASIC PRoGram from directory listing
19	VERIFY binary file from directory listing
20	DSAVE with replace and DVERIFY from directory listing

The menu presented offers three choices: **Define, Save,** or **Quit.** Pressing **D** for define will prompt you to enter a number from 1 to 10 for the key to define. Keys 1 through 8 are the **F1** through **F8** function keys. Key 9 is the shifted **RUN/STOP** key, and Key 10 is the **HELP** key.

After selecting a key, you will be prompted to enter a definition. Select a definition from 0 to 20 from the list. Keys that are redefined are displayed along with their new definitions on the bottom half of the screen.

The compiler displays the number of bytes used in the function key buffer. The sum of all ten function key definition can not exceed 246 bytes. If the length of the selected function key definition puts the total length over 246 bytes, the compiler will refuse to add it to the buffer and will display a **Buffer overflow** error message. To remove a function key definition from a key, select definition **0**, which has no length. This will allow for including several long definitions by removing definitions that not in use. It can also be used to disable a key.

There is no need to define all ten keys. Keys that are not defined by KeyDOS Compiler will retain the definitions they had before running the program. This allows you to include your own key definitions along with KeyDOS keys. For example, to retain an existing key definition, simply do not redefine it with the KeyDOS Compiler.

After redefining the keys as desired, you can either save the definitions as a binary file, or quit to BASIC. Pressing S will ask you to enter the number of the drive on which to save your key definitions. The next prompt asks for a file name. The default filename is **KEYDOS.KEY**. You can press **RETURN** to save the file with this name, or edit the name to something else. Insert a @ before the name in order to save the keys to a file with the same name.

Using KEYDOS.KEY Definition Files

To load a **KEYDOS.KEY** file, enter:

```
BLOAD "KEYDOS.KEY",U(drive number)
```

The file can also be loaded from a catalog listing using the KeyDOS BLOAD definition (definition 13). This opens up the possibility of changing function key definitions "on the fly" by using the same key for definition 13 in several different KEY files. Changing key definitions is as simple as listing a directory, moving the cursor to the line with the name of the KEY file to load, and pressing the BLOAD F key. BLOADING KEY files does not alter BASIC or ml programs in memory.

1581 Visual Partitioner (ESC 3)

KeyDOS Visual Partitioner V2 makes it easy to format and partition 1581 subdirectories. Completely menu driven, Partitioner gives a visual representation of the tracks on a 1581 disk, and checks for valid input. Partitioner can also create partitions and subdirectories within subdirectories to any depth as long as there are tracks available. Partitioner can be used to make partitions on a disk that has already been in use and contains files. Although this feature is reliable, you should always have a backup copy of a disk before attempting to create subdirectories on it. It's a good policy to make backup copies of disks whether you use this program or not.

Warning: Visual Partitioner is intended for use on blank disks or backup copies only. Please do not use this program on disks which contain valuable data unless they are properly backed up. There is a menu selection which formats the disk, erasing everything it contains. Please read all program documentation and heed all warnings the program might give you.

Using Visual Partitioner

While in 80 column mode press **ESC 3**. Visual Partitioner starts up with a main menu:

```
Catalog, Drive, Format, Show tracks, Quit
```

Catalog - Shows a catalog listing of the active drive. The prompt **Catalog: 8>? "*** will appear. Press **RETURN** for a catalog of all files on the disk, or enter a CBM DOS pattern for listing a selective directory. Use the **NO SCROLL** key to pause the listing or the **STOP** key to quit and return to the menu.

Drive - Selects a new active drive. Enter the number of the 1581 drive containing the disk you wish to format or add partitions to after the **Drive?** prompt.

Format - Formats the disk in the active drive. This choice first presents an **Are you sure?** warning. Press **y** to format the disk, or any other key to return to the main menu. If you press **y**, you will be asked to supply a disk name (up to 13 characters) and a disk ID (2 characters). Partitioner will then format the disk. If there are any errors during formatting, the program will return to the main menu. The format command can also be used to erase a previously formatted disk. If you enter a disk name without an ID (press RETURN at the ID prompt) the disk will be erased without performing a complete format.

Show tracks - This will display the tracks in use on a previously used disk which contains files. It also takes you to the next menu where you can add partitions to the disk.

Quit - Ends the program and returns to BASIC.

Once the disk has been formatted or erased, or Show tracks selected, Partitioner displays the available tracks and the the partition menu.

```
KeyDOS Visual Partitioner V2                    0 "keydos           " td 3a
C128 80 Columns                                 3160 blocks free.
Copyright 1992 by Randy Winchester

Catalog, Partition, Select, Main, Quit

Status: 00, ok,00,00

Track:
111111111122222222223333333333444444444455555555556666666666777777777
123456789012345678901234567890123456789012345678901234567890123456789012345678
                                                  B
                                                  A
                                                  M
                                                  /
                                                  D
                                                  I
                                                  R
```

The upper right corner of the screen is the area where the catalog listing is displayed. The menu line now shows three different choices, **Partition**, **Select**, and **Main**. A status line appears several lines below the menu line. It displays the latest disk status or error. Below that is a visual representation of the tracks on the 1581 disk. Used tracks are marked with a check mark. BAM/DIR marks the directory track. As new partitions are opened, the BAM/DIR changes locations.

Catalog - Shows a catalog listing of the active drive

Partition - Displays the disk tracks used so far along with the first seven characters of the partition name. Check marks under a track number mean the track is in use and cannot be included in a partition. Partition names are displayed vertically under the first track of the partition. The program prompts for a partition name, the number of the starting track, and the number of tracks to include. You must not include the number of any track already in use. The partition will be generated if there are no errors, and the track display will be updated to show the new tracks included in the partition. Errors will be reported on the status line and control passed to the partition menu. If the partition contains three or more tracks, the program will offer the choice of formatting a subdirectory in the partition. A subdirectory name (up to 13 characters) and an ID (2 characters) will be requested. After formatting, the catalog display appears and the status line will report the results. Press any key to return to the main menu.

Select - Opens a previously created subdirectory and updates the track display to show only the tracks contained in the partition. The BAM/DIR mark moves to show the location of the subdirectory's directory track. Partitions can be created inside an existing subdirectory if there are free tracks available. To select the root (main) directory, choose Select and press **RETURN** without entering a name.

Main - Returns to the main menu. Use Main if you wish to change disks or format or erase a disk.

The KeyDOS Subdir function key makes getting into and out of partitions a snap. See the section on the Subdir key on page 7.

RAMDOS 128 (ESC 6)

Version 4.4 documentation
Program by: Hedley Davis & Fred Bowen

2MB Patch by Andrew E. Mileski, 1991
Documentation by: Loren Lovhaug & Fred Bowen

RAMDOS 128 wedges into your C-128's operating system to facilitate the usage of the 1700 or 1750 RAM expansion unit as a fast pseudo-disk drive (RAMdisk). Once RAMDOS is installed, you can use most Commodore DOS commands to read, write, and manipulate programs and data with your RAM expansion unit.

Installing RAMDOS 128

Press ESC 6. Upon execution you will be asked what unit number you wish to assign to the RAM expander. The default unit number is 9, however you can use any device number from 4 to 30. Once you have responded to the device number prompt you will be asked where you would like the RAMDOS interface code to reside, the default is page 14. You should not alter this value unless you are trying to make RAMDOS work with an application that must utilize the default area. (The default area is the lower half of the sprite definition area so obviously you cannot use page 14 if you wish to use sprites or a program which uses sprites). If you must relocate the interface page, note that this code must be placed on a page boundary and must not be placed at an area that is overlapped by ROM. Once you have responded to the interface prompt, you will be asked if you want to initialize the RAM disk. Respond by typing "y" if you have not previously saved data in the RAM expander or if do not want to access any existing data currently stored in the expander via RAMDOS. The default response is "y". Once you have responded to the initialization prompt your RAMdisk will be installed.

Using the RAMdisk

The RAMdisk is ideal for data intensive applications since you can read and write data to the RAM expander much faster than would be possible with a conventional physical disk drive. However, you must remember that unlike a physical disk drive, the data you store in your 1700 or 1750 is by no means permanent and will be erased when you turn off your C-128, so make sure that you store any important data to disk before you turn off your machine.

To copy programs and data to the RAMdisk, you can simply write it to the RAMdisk just as you would normally, using its designated unit number. However, at times it is handy to copy a large number of files from a conventional 1541, 1571, or 1581 disk to the RAMdisk. The KeyDOS Utility is ideal for this task.

As alluded to above, a subset of Commodore DOS commands can be used in conjunction with the RAMdisk. Most normal file creation and accessing commands are available including: OPEN, DOPEN, APPEND#, CLOSE, DCLOSE, LOAD, DLOAD, SAVE, DSAVE, BLOAD, BSAVE, VERIFY, DVERIFY, PRINT#, INPUT#, RECORD# and GET#. Likewise most DOS housekeeping commands are implemented such as: SCRATCH, RENAME, HEADER, DCLEAR, COPY, and COLLECT. However the COLLECT and DCLEAR commands are not really necessary since the RAMdisk code does the equivalent of COLLECT and DCLEAR with each operation. The CONCAT command is not implemented. The error channel, as well as the BASIC 7.0 variables DS and DS$ can be examined to determine disk operation status.

Most direct access commands (i.e. B-A, B-R, B-W, M-R, M-W etc.) are not implemented since these involve either the accessing of physical tracks and sectors on a disk or the reading or writing of a physical disk drive's memory. The B-P (buffer pointer position) command has been implemented for the positioning of the relative file pointer ala BASIC 2.0 (in most cases 128 owners will want to use the RECORD# command for this purpose). Another special exception is the use of the USER command for changing device number as found in the 1571 and 1581 versions of Commodore DOS. You may change the unit number of the RAMdisk at will by using the following syntax:

OPEN 15,old unit number,15,"u0>"+chr$(new unit number) You may also use the 1541 equivalent M-W command (this is the only M-W command that is implemented in RAMDOS).

Unlike physical disk drives you can change the unit number of the RAMdisk to a unit number that is already in use without changing the unit number or shutting down the physical drive that already has that unit number. For instance you may have a physical drive such as the 1541, 1571, or 1581 in use as unit 9 and define the RAMdisk as unit 9 also. In this example, your C-128 would "see" the RAMdisk as unit 9, and subsequent DOS operations directed at unit 9 would operate on the RAMdisk and not the physical drive set as unit 9. As a rule, the RAMdisk takes precedence over a physical drive with the same unit number as far as DOS operations are concerned.

When the RAMdisk is installed, BASIC 7.0's STASH, FETCH, and SWAP commands are disabled. BASIC programs using these commands will not function correctly with the RAMdisk installed although their use will not cause a BASIC error per se.

Pressing the reset button on the left side of your keyboard will not erase or corrupt the data stored in the RAM expansion units. However, pressing the reset key while the RAMDOS is effecting an operation on the contents of your RAMdisk will most certainly corrupt the data in the RAM expander, so you should make sure that all DOS operations are completed and all open files closed BEFORE you hit the reset button.

If you are viewing the C-128's 40 column bitmapped graphics screen during the execution of a DOS command to the RAMdisk, you will probably notice a momentary flicker of the screen and the appearance of random pixels on the graphics screen for a brief instant. This is caused by a normal function of the RAMDOS and should not be cause for alarm. Upon execution of a DOS command, RAMDOS automatically retrieves some transient operation code placed in the RAM expander upon installation. This code is momentarily placed in the 40 column graphic area for execution, however the contents of the 40 column graphics screen are not corrupted because it is temporarily stored in the RAM expander and then retrieved when the execution of the DOS command is completed.

Finally, although DMA transfers to the RAM expansion units cannot be done in FAST mode, there is no need for the user to put the C-128 into slow mode before accessing the RAMdisk. RAMDOS will automatically place the C-128 in slow mode while DMA operations are taking place, and return the C-128 to FAST mode if the C-128 was in FAST mode before the operation was executed.

We hope you enjoy RAMDOS!

GEOS SuperRBoot (ESC 7)

SuperRBoot is the ultimate reboot program for GEOS 128 with a RAM Expansion Unit. SuperRBoot and ACTIVATOR (on the KeyDOS ROM Disk) provide the most reliable method available for switching between GEOS and native mode, and for rebooting GEOS. Use SuperRBoot to reboot GEOS quickly after returning to C128 mode, or after a GEOS software crash. SuperRBoot will reboot GEOS 128 regardless of how disk drives were set before leaving GEOS or renumbered while away from GEOS. ACTIVATOR returns the C128 to native mode from GEOS and at the same time corrects several native mode problems caused by GEOS.

The value of SuperRBoot and ACTIVATOR will be obvious to anyone who has had GEOS 128 crash. Maybe you've just finished the 65th page of that novel you're writing with geoWrite. It's then that you realize that geoWrite doesn't allow more than 63 pages and something must be terribly wrong. You try to select quit from the file menu, but before you get there, a dialog box pops up: **System error near $*!!!%*#!** There is no **OK** to click Things aren't rosy. It's a good thing you were using your REU, you think. Everything should be safe, you think. All you have to do is hit the RESET switch. Aha! Great! The screen cleared, and the drives are spinning, and, and . . . NOTHING! Hit RESET again. Still nothing. You recall that after booting GEOS you had moved Drive C to the Drive A position. There isn't much to do except to stare at the blank screen and curse.

Now that your KeyDOS ROM is installed, there's hope.

Some preparation is necessary to insure that SuperRBoot will work properly. After booting GEOS, use the Configure program to select **Fast Reboot**. You must also select either a 1541, 1571, or 1581 RAM Drive. Select **save configuration** from the file menu. Exit Configure and copy the deskTop and ACTIVATOR (from the KeyDOS ROM Disk) to the RAM drive.

Instead of selecting BASIC from the deskTop options menu to switch from GEOS 128 to native C128 mode, double click on ACTIVATOR. The computer will switch to native mode and you'll be prompted to press the **ALT** to activate KeyDOS ROM or **STOP** to go to BASIC without activating KeyDOS ROM. Activating KeyDOS ROM will automatically run the Reset 15xx Drives utility from the ROM. This is done because on exit from GEOS 128, 1571 disk drives are left in single-sided slow mode. ACTIVATOR has some added benefits that aren't immediately obvious. It fixes the **RESTORE** key problem caused by GEOS 128, and it returns the **RESET** switch to its normal function.

Although it facilitates rebooting GEOS to have the disk drives in their default positions before returning to native mode, it is no longer necessary. The only minor problem you are likely to encounter is that when ACTIVATOR resets the drives, you'll be unable to access one of the lower numbered drives, either 8 or 9. The solution is to switch the drive off, then back on.

You also should not alter the contents of the REU in any way while in native mode. Doing so will most likely overwrite the copy of the GEOS system stored there and you will not be able to reboot. Among other things, this includes running RAMDOS or other programs which use the REU such as CP/M or CS-DOS.

It's also possible to recover from a GEOS software crash similar to the deadly example above, even if you haven't run ACTIVATOR during your session. Reset into C64 mode by holding the Commodore logo key while pressing **RESET**. Enter **POKE 996,96<RETURN>**. Press **RESET** again while holding the **ALT** key to activate KeyDOS ROM. You can now press **ESC 7** for SuperRBoot and be back in business.

Diskmon128 (ESC 8)

by Anton Treuenfels, from *The Transactor*, Volume 9, Issue 5

Diskmon128 adds disk sector read, write, and editing capabilities to the C128's built-in monitor. A sector can be read from the disk, displayed with the monitor's **m** command, modified, disassembled, and written back to disk.

Diskmon can be started from either BASIC or within the monitor. Pressing **ESC 8** will result a sign on message from Diskmon:

```
diskmon128 v092388
by anton treuenfels
```

If you are not already in the monitor, type **MONITOR**, and press **RETURN**. Diskmon is now ready to use.

Command Syntax	Description	Examples	
`/r` [<track> <sector>]	- read sector	`/r b a`	- read track 11, sector 10
		`/r`	- read the next linked sector
`/w` [<track> <sector>]	- write sector	`/w +10 5`	- write track 10, sector 5
		`/w`	- write the last sector read
`/#` <device> [<drive>]	- set device and drive numbers	`/# +11`	- select device 11
`/q`	- quit Diskmon 128 wedge		

Do not reload Diskmon if you have loaded it earlier and have not used the /q command to quit. All parameters are numeric and may be specified in any convenient base (hexadecimal, decimal, octal, or binary). Hexadecimal is the default. Square brackets indicate optional parameters.

Displaying Sectors

The sector is read into a buffer from $b00 to $bff in RAM bank 0. The command to display an entire sector is:

`m b00 bff` - display sector contents

The buffer can also be disassembled. Modifications can be made to data in the sector and written back to disk using the /w command.

If a track and sector are not specified, both the read and write commands use default track and sector values. In the case of the read command the contents of the first two bytes of the disk sector buffer are taken to be the track and sector to read. This is based on two assumptions: that the contents of the buffer represent the link to the track and sector of the next sector of the file. These assumptions are often true, making it possible to easily trace forward through the system of links tying files together under Commodore DOS. Particular conditions under which the assumptions are untrue include reaching the last sector of a file and before the program has read its first sector.

The write command defaults to using the values found in the current track and current sector variables maintained by the program. Since these variables will usually have last been set by the most recent read command, the default action normally amounts to putting the (possibly modified) sector currently in the disk sector buffer back where it came from.

The /# command is used to set the device and drive number. The default value is device 8, drive 0.

Diskmon Example

The following is a sample of a typical session using Diskmon to change a disk's name. Commands entered by the user (followed by RETURN) and the response from the monitor are shown along with comments.

```
monitor                                          User enters monitor command

monitor                                          Monitor displays register values
    pc  sr ac xr yr sp
; fb000 00 00 00 00 f8

diskmon 128 v092388                              Press ESC 8 to start Diskmon
by anton treuenfels

/# a                                             Select drive 10
/r +18 0                                         Command to read track 18, sector 0
m b90 b9f                                        Display portion of buffer that contains the
>00b90 4b 44 52 4f 4d 20 57 4f:kdrom wo          disk name. (m b00 bff displays the
>00b98 52 4b a0 a0 a0 a0 a0 a0:rk                entire sector.) Cursor to the second line and
                                                 make the underlined changes to add the
>00b98 52 4b 20 4f 4e 45 a0 a0:rk one            word one after the disk name.
/w                                               Write sector back to disk
x                                                Exit monitor
ready.                                           All done!
```

Monitor dump (ESC 9)

The KeyDOS Monitor dump will send the output of the monitor to the current device, either a PETASCII disk file or a PETASCII printer. The disk files produced by Monitor dump can be loaded into any text editor that accepts PETASCII files and edited to produce source code from disassemblies. Quick print outs are available by using the KeyDOS F7 key to select a printer device before running Monitor dump. To run Monitor Dump, press ESC 9. Monitor dump redefines the F1 and F2 function keys to begin and end the monitor dump. The F1 key is redefined to open a file named .mon. on the current active device and enter the monitor. After pressing F1, monitor output is redirected to the file or printer. When you are finished, pressing F2 will exit the monitor, close the file and restore the default KeyDOS function key definitions.

Monitor Dump Example

```
F1-Open .mon.                                    Press ESC 9 to Set up
F2-Close .mon.                                   Monitor Dump
ready.
d=peE(i86):oPi,(d),7,"@0:.mon.,s,w":cmdi:moN     Press F1 to open .mon.
m 1000 1fff                                      Display function key buffer
gf4dc6                                           Press F2 to close .mon. and exit
pRi:c101                                         monitor.
ready.
KeyDOS F keys installed.                         Default KeyDOS F keys installed.
```

geoDebugger is a registered trademark of Berkeley Softworks
TurboDebugger is a registered trademark of Borland International

Manual written by Eric Trepanier (release 1.0)

Hexpert designed and written by Eric Trepanier

A Few Words from the Author

As an active Commodore 128 programmer, I have every intentions of improving Hexpert and make it more and more versatile and powerful. This software is released as Public Domain, which means I am granting you the privilege of using it at your discretion. I am not expecting any financial compensation out of this program. You can freely distribute it, and what the hell, you can even sell it if you can find someone silly enough to pay anything for it. This is not to say that Hexpert is not a good program. In fact, I am not sure they are other debuggers available for the C128, besides geoDebugger. The only thing I expect from the programmers who will actually use it is some feedback, either good or bad. I want this program to be the "BEST" debugger for the C128.

This is only an initial release, but future versions will include much more advanced features. However, if I don't get any feedbacks, I'll have to conclude that it's not worth spending time on it. Any suggestions, criticism, or simply a wish-me-luck card will be appreciated...

Eric Trepanier

NOTE: If you do distribute this program, please distribute it in its entirety, with all the documentation files. Needless to say that a modified copy of Hexpert should not be distributed.

Credits

This program might never have seen the light of day without the presence of a good friend of mine named Pedro, who happens to be an excellent machine language programmer himself. Without his help, I would never have mastered machine language so easily. I would also like to thank a couple of my friends for their encouragements and support. These friends are Sylvain Tremblay, Philippe Blais and Marc Veilleux.

Disclaimer

Let's get this over with... This program and accompanying documentation come with no warranty, either express or implied, with respect to accuracy or suitability for any particular application. The author assumes no responsability whatsoever for any damage, direct or incidental, resulting from their use. Good, now on to the more interesting stuff!

Chapter 1: Introduction to Hexpert

What is Hexpert?

Hexpert is a simple program, designed to be used in conjunction with the C128's built-in machine language monitor. This is part of the reason why even though it is relatively small (2K), it still has a lot of features to offer. Hexpert makes as much use as possible of the monitor's own routines (e.g. symbolic disassembly, numeric conversions, etc.). The main purpose of Hexpert is to allow you to monitor your programs as they are executed. This is what a debugger is for. Several commands are available to allow a close watch of the program's state at any time during its execution.

Manual Organisation

This manual is organised in the following manner:

Chapter 1 is the introduction chapter. This chapter needs only to be read if you wish to know more about the development of Hexpert. It is not essential for making good use of Hexpert.

Chapter 2 presents a brief discussion on the concept of "program debugging." It presents some of the major features of Hexpert. At the end of the chapter, there is a complete "hands-on" sample session to illustrate these functions. Although not essential, this chapter will make a good introduction on how to actually use Hexpert. It would be a good idea to read it through before using the debugger.

Chapter 3 is the reference chapter. This is the most important chapter, it presents all the debugger's commands as well as examples on how to use each of them.

Appendix A explains all the debugger's information and error messages, appendix B contains all the technical information concerning Hexpert and appendix C presents a small C128 machine language bibliography.

The Origin of Hexpert

Recently, I started working on a window package, with commands for opening, closing, moving and resizing windows. I quickly realized I could hardly debug the subroutines without the help of something more evolved than the C128's monitor. This is how I got the idea for Hexpert. At first, I didn' t think of releasing it in the Public Domain. I was doing this program for my personnal needs, end of the line. A friend of mine who saw an early version of Hexpert running was impressed. He suggested that I add some features and release it as a shareware program. So I did add a few options here and there, and made it what it is now. However, I felt kind of shamed to release it as shareware. There are so many good PD programs out there. I just felt I owed it to them to release my own program as Public Domain. Besides, it didn't take me more than a few weeks to finish the program. Some shareware programs take years to create (e.g. DesTerm 128 v2.00 from Matthew E. Desmond) and yet their authors only charge a ridiculously small fee for their use.

It's up to you, the users, to turn Hexpert into a real bug killer. Just send in your comments on Hexpert. Let me know if you discover any "undocumented features". Every suggestions will be considered. If I see that it is used, I might rework it over and turn into a serious TurboDebugger competitor.

Chapter 2: Hexpert Usage

This chapter introduces the major features of Hexpert as well as overviewing how to run and use the debugger. It begins by describing the concept of debugging and finishes with a short tutorial covering the basic features of Hexpert. This chapter does not cover aspects of the debugger in exhaustive detail. Refer to the following chapter for more complete coverage.

What Is A Debugger?

Unless you are an exceptional programmer, your programs will seldom work right away. Most of the time, they will be "bugged". Bugs are the little mistakes and errors that you haven't noticed while coding your program. It might be an incorrect jump or a bad branch, or it could be something much less obvious. In such cases, it's very hard to know exactly what is going wrong. Maybe your algorithm has a logical error, but unless you can trace your program step by step and monitor all important variables, you will have a very hard time fixing the problem. This is where a debugger comes in handy. With it, you can set up breakpoints in your program and then execute it. When your program reaches the breakpoints, it will stop and you can monitor all the 8502 registers and important variables. You can also watch your program's execution instruction by instruction. The debugger is perhaps the programmer's most important tool.

Hexpert Features

Hexpert is a software debugger that is accessed via the C128's monitor. A lot of work has been put to make Hexpert as small as possible, yet big enough to hold a complete arsenal of useful debugging commands. The result is a surprisingly versatile 2K debugger with all the important features one could expect from an expensive commercial debugger package.

Hexpert offers a plethora of commands to help you monitor your program and memory, giving you full access to both C128's RAM bank, I/O registers, Kernal ROM and all 8502 registers. You can disassemble, modify and run your program interactively, setting break points, displaying and changing register values.

Dual Displays

Hexpert makes use of both the C128's forty and eighty column display. This allows you to debug on the forty column screen a program which accesses the eighty column screen, so the program's output will not be disturbed by Hexpert's output. This feature works best if you have access to two monitors (CRTs). Hexpert could be run on one monitor while the program's output appears on the other monitor. Even geoDebugger doesn't allow this possibility! The screens may be swapped at anytime during the program's execution.

Breakpoints

Hexpert supports a user-breakpoint facility in a limited fashion. Breakpoints may be set anywhere in RAM memory. Once a breakpoint is encountered during program execution, the program stops and Hexpert is given control. Due to memory conflicts, this version of Hexpert doesn't allow the use of a single break point more than once. This means that once a breakpoint has been reached, it is removed from the program. Hexpert allows ten simultaneous break points.

Running Hexpert

To activate Hexpert from the KeyDOS ROM, simply press ESC 0. If your assembler/linker doesn't alter memory in the range $1300-$1BFF, you can have Hexpert activated before calling the assembler. Merlin 128 works fine this way. If you want to use Hexpert with Merlin 128, just add PRINT CHR$(27)"0" to the "MERLIN" basic boot program from the Merlin boot disk. When you use Merlin's "Exit to BASIC" option, you only need to type SYS 4864 to activate Hexpert.

You should see the "HEXPERT Vx.x" copyright message appear when the SYS 4864 is executed.

Hexpert should work fine with other monitor enhancements provided that the other programs do not alter memory range $1300-$1BFF and that Hexpert is booted first.

Once you've seen the "HEXPERT Vx.x" message, the debugger can be accessed via the "E" command in the C128's monitor.

Sample Hexpert Session

This section is a hands-on tutorial. It is designed to familiarize you with the debugger environment. You will need a disk with the HEXPERT program file on it.

Begin by resetting your C128. When the "**READY.**" prompt appears, type the following:

```
ESC 0
```

A message like this should appear:

```
HEXPERT V1.1 (C) 1990
BY ADVANCED SOFTWARE SYSTEMS

READY.
```

Again, at the "READY." prompt, type **MONITOR** + **[RETURN]**, or hit the **F8** key. You will enter the C128's monitor which should look like this:

```
MONITOR
   PC  SR AC XR YR SP
;FB00 00 00 00 00 F8
```

When the cursor starts flashing, type **E** + **[RETURN]**. This will call Hexpert which will respond with the following:

```
DEBUGGER
   PC  AC XR YR SP NV-BDIZC  MEMMAP
;1B00 00 00 00 F8 00000000 01001110
. 1B00 xx xx xx AAA 0000
DEBUG>
```

This is a representation of the 8502 registers and of the next instruction to be executed in mnemonic format at the current value of the program counter. The xx xx xx refers to the binary format of the instruction, opcode first, operand second in a low/high byte fashion. The AAA is the mnemonic for the opcode and OOOO is the operand.

The cursor should be flashing after the "DEBUG> " prompt. This means that Hexpert is awaiting a debugger command. Unfortunately, we do not have a program to run yet... This is not too bad though, we will use the monitor to assemble a small program which we may then execute step by step. You should now exit using the **Q** command. It is not necessary to hit the **[RETURN]** to execute a command while you are under Hexpert's "DEBUG> " prompt, Hexpert knows when a command is complete. Now that we are back in the monitor, we will assemble a small program at $2000. You should refer to the C128 manual for help with the C128 monitor. Enter the following lines:

```
A 02000 A9 00     LDA #$30
A 02002 A2 00     LDX #$00
A 02004 9D 20 20  STA $2020,X
A 02007 38        SEC
A 02008 69 00     ADC #$00
A 0200A E8        INX
A 0200B E0 04     CPX #$04
A 0200D D0 F5     BNE $2004
A 0200F 00        BRK
```

This program performs a small loop which stores the ASCII codes of 0,1,2,3 consecutively at $2020. Before we enter Hexpert, let's take a look at what's in $2020. Type **M2020 + [RETURN]**. You should get a following display:

```
>02020 xx xx xx xx xx ...
```

Remember the first five bytes starting at $2020. Write them down if you have too, we will execute the program and verify that the values have effectively been changed. Now let's enter Hexpert again, type **E + [RETURN]** again.

Once you get the "DEBUG> " prompt, type **02000; S**. Once you get used to Hexpert you will notice that many commands may be abbreviated. The previous command could've been abbreviated to "**02S**" with the same result. This sets the program counter at $02000 which is where our program resides. You can hit the ESCAPE key at any time to cancel a command. Now that we have something like this:

```
. 02000 A9 00     LDA #$30
DEBUG>
```

We will start executing the program step by step. Hit the **S** key. The register will appear again. This time, AC will hold the value 30. The next instruction at $2002 is displayed. Hit **S** again to execute it. The registers will appear again unchanged, except for the Z bit in the status register which should have been set to one. This is ok, since we loaded the value zero in the x register. The next instruction stores the contents of the accumulator at $2020 + x, where x is the value of the x register. Then the carry is set. The corresponding bit of the status register should be set to one accordingly. The ADC #$00 will add the carry to the contents of the accumulator. Result: AC = $31 and bit C is cleared. Then the X register is incremented and compared with the value 4. If it is not equal, the program branches back to $2004 and the loop is executed until the x register holds a value of 4 at which time a BRK

instruction is encountered which indicates the end of the program.

This is a very simple program which doesn't do much, but it shows the main components encountered in a program: loops, tests and jumps. You might think that it is annoying to always have the registers displayed after every instruction (I do!). You can hit the D key to toggle the register display option on/off. When it is off, you will have to hit the R key to see the registers.

Now we want to confirm that the program did store the values at $2020. We'll use the SHOW BYTE command "**£**", to do this. Type **02020; £**. You will get:

```
DEBUG> 02020; £02020 = 30
```

With the cursor positioned to the right of the 30 value. Indeed, this is the value which we had stored at $2020. Hit the **LINE FEED** key to see the next byte, which will show a value of 31, this is also good. The next two bytes will have also changed to 32 and 33. The fifth byte however has not been modified, so its value should be intact. If you think Hexpert is playing a magic trick on you, you can use the monitor's M command to make sure.

And Now On To More Powerful Manipulations

Now that you have completed the sample session with Hexpert, you should be beginning to feel more comfortable with the debugging environment. You will find a complete discussion of Hexpert's command set in the following chapter.

Chapter 3: Hexpert Reference

This chapter contains a complete reference of Hexpert's command set. It covers every aspect of using the debugger such as jumps, step by step, breakpoint and view options, including a detailed description of each command. Although this is primarily a reference chapter, it would be a good idea to read it through completely at least once. For a general overview and information and information on using the debugger, refer to the previous chapter.

Command Syntax

Hexpert uses an interactive parser which knows what to expect from a command. This is why it is seldom required to hit RETURN to confirm, or to add separator characters like spaces or the likes. In addition, every command which requires (or accepts) an address can be abbreviated if the address has terminating zeros. For example, if you want to start execution at $3000 in bank 1, the entire command should look like this:

```
DEBUG> 13000; G
```

However, the space and the ';' character are added automatically by Hexpert's parser if you omit them. The zeros can be omitted too, thus you could simply type **13G** and obtain the same result. At any time while entering a command, use the ESCAPE key to cancel the command. It is not possible to use the delete key however...

NOTE: All commands are entered UNSHIFTED.

The correct syntax for an address is the following: Bhhhh. All addresses require four hexadecimal digits h, though

the zeros will be appended by Hexpert if necessary. The B stands for BANK number. The current version of Hexpert offers a limited BANK support. Only RAM bank 0 or 1 may be accessed. Memory in the range $D000-$DFFF is always mapped as I/O registers. Memory beneath it is currently inaccessible, this will probably change in an upcoming version. Memory in the range $C000-$FFFF can be toggled as either RAM (0/1) or ROM Kernal. Keep in mind that the ROM kernal can not be single-stepped.

Some commands will require an additional parameter following the command code. This parameter should be a decimal number (0-9).

The Concept of Memory Banking on the C128

The C128 is an eight bit machine. This means that it's processor (8502) can handle information that is 8 bits wide. The C128 address bus handles 16 bit addresses, so the C128 can only address up to 64K of memory. You are probably wondering how the C128 can have 128K of RAM memory, if it can only handle up to 64K, right? Well, it just so happens that a special device on C128 mother board is in charge of handling all the memory switching of the C128. This device is called the Memory Management Unit (MMU).

The RAM memory of the C128 consists of two 64K RAM banks. Only one of these banks can be accessed at a time. Additionnaly, the C128 has a fair amount of ROM memory (BASIC 7.0, Machine Language Monitor, Kernal) as well as a 4K block of input/output registers. All of these need to be accessed by the C128 too. The MMU is in charge of establishing different memory configurations that will enable the 8502 to access all of that RAM/ROM memory. You might be familliar with BASIC's BANK command which allows you to choose one of fifteen pre-set memory configurations. These configurations are only a subset of all the possible combinations that might be selected. The machine language programmer has full control over the MMU through a dozen of I/O registers. We are not going to expand any more on the MMU, please refer to the C128 Programmer's Reference Guide for more information on the use of the MMU's input/output registers.

Hexpert acknowledges some of the different configurations that might be selected by a program, notably it will permit a program to freely access RAM memory from bank 0 or 1. It allows the Kernal ROM to be accessed as well as the input/output registers at $D000. It does not allow access to the BASIC & Monitor ROM, nor does it allow you to access the 32K ROM sockets.

The programmer can select where in memory the program is located via the first digit of every addresses specified at the "**DEBUG>**" prompt. This digit should be a 0 or 1 and corresponds to one of the two 64K RAM memory blocks. The "**MEMMAP**" register display is an image of the program's actual MMU configuration.

Hexpert Command Summary

General Commands

Q	Exits Hexpert and returns to the C128's monitor
D	Register display toggle for step by step execution
K	Kernal toggle (memory range $C000-$FFFF)
^	Hexpert screen toggle (40/80 columns)

General Display Commands

R	Display processor registers
A	View/Modify contents of accumulator
X	View/Modify contents of x register
Y	View/Modify contents of y register
£	View/Modify contents of a byte
/	View/Modify contents of a word

Execution Commands

G	Start full speed execution of program
J	Execute subroutine at PC (must be on a **jsr** instruction)
S	Single step through current level and subroutines, also used to set new program counter
T	Top step through subroutines using inline parameters

Breakpoint Commands

B	Set/Remove breakpoint

General Commands:

Command:	**Q**
Purpose:	leave the debugger and return to the monitor.
Usage:	**Q**
Note:	takes no parameters.

Q leaves the debugger and returns to the C128's monitor. The programs space remains intact as do the 8502 registers, so it is possible to toggle between the monitor and the debugger during the execution of a program to use the monitor's extended memory view capabilities.

Command:	**D**
Purpose:	toggle the display of 8502 registers during step by step execution.
Usage:	**D**
Note:	takes no parameters.

D toggles the automatic display of the 8502 registers during step by step execution. The automatic display might be practical during critical program sections, but most of the time, it is mostly annoying. This option uses the BEEPER flag in zero page $F9 to determine whether it's on or off. This has an effect on the output of the bell (CHR$(7)). When the automatic display is turned off, the bell can't be heard.

Command: K
Purpose: Toggle contents of memory in range $C000-$FFFF between RAM and ROM.
Usage: K
Note: takes no parameters.

K toggles what is found by Hexpert in memory range $C000-$FFFF for program execution. This means that it is possible to run programs which use the Kernal in RAM BANK 0 or 1. This is reflected by bit 4 and 5 in the MEMMAP display. When both bit are set to zero, ROM will be accessed. When both bit are set to one, RAM will be accessed. The RAM BANK will depend on bit 6 of the MEMMAP. A clear (zero) bit 6 means RAM BANK 0 is accessed, while a set (one) bit 6 means RAM BANK 1 is accessed.

NOTE: It is not possible to use the **b, s, t, j, £, /** commands in ROM. Such commands are not forbidden by Hexpert, but they will however have unpredictable results and may very well crash your system. See the **S** command for information on how to single-step through the Kernal's routines.

Command: ^
Purpose: Toggles Hexpert's active screen between forty and eighty columns.
Usage: ^
Note: takes no parameters.

^ toggles Hexpert's active screen (the one in which Hexpert's "**DEBUG>** " prompt appears. If you want to debug an 80 column-specific routine, then Hexpert's active screen should be in forty columns. The same holds true for the opposite situation. You can make the best use of this feature by connecting two monitors to your C128. One monitor should be connected to the RGB port, while the other is connected to the Composite output.

NOTE: While you are debugging applications that use the VDC, select the 40 column's screen as the active Hexpert screen. This way, Hexpert's output will not affect the READY bit of the VDC register ($D600).

Display Commands:

Command: R
Purpose: Display processor registers.
Usage: R
Note: Takes no parameters.

The **R** command displays all the processor registers, including the MMU pseudo-register. The output has the following format:

```
   PC   AC XR YR SP NV-BDIZC   MEMMAP
;1B000 00 00 00 F8 00000000 01001110
```

The PC, AC, XR, YR and SP indicators refer to the program counter, accumulator, x register, y register and stack pointer. NV-BDIZC are the seven bits of the status register. The "-" bit is not used. Do not pay attention to it. The most complicated of the registers displayed is the MEMMAP, which refers to the C128 MMUCR register. Refer to the C128 PROGRAMMER'S REFERENCE GUIDE for information on the MMU. This indicator shows the actual MMU configuration. Below the registers, the next instruction to be executed at the current PC is also displayed.

Command: A
Purpose: View/Modify current value of accumulator.
Usage: A
Note: Takes no parameters.

The A command (as well as the X and Y commands) is used to view and modify the current value of this 8502 register. When used, it will look like this:

```
DEBUG> AC = xx
```

The cursor will appear after the xx value (hexadecimal). This value may then be changed or cleared using the DEL key and hexadecimal keys (0-9,A-Z). To confirm the new value, simply hit [RETURN]. You can abort modifying the current value using the ESCAPE key. The value needs not to have two hexadecimal digits. If no digits are entered, the value zero will be given.

Command: X
Purpose: View/Modify current value of x register.
Usage: X
Note: Takes no parameters.

This command is similar in use to the A command. Refer to the A command description for more information.

Command: Y
Purpose: View/Modify current value of Y register.
Usage: Y
Note: Takes no parameters.

This command is similar in use to the A command. Refer to the A command description for more information.

Command: £
Purpose: View/Modify byte value at specified address (PC if none provided).
Usage: [addrexp] £
Note: addrexp is the address of the byte to view. If no address is specified, the current address pointed by the PC will be used. This command is similar in use to the A command.

The £ command is the general view and modify data command. When you open a memory location with £, you are placed in an interactive mode. You can then either change the current value and confirm with the return key, abort changing the value using the escape key, or leave this value unchanged and view the next byte using the LINE FEED key. You can cycle through the entire memory, viewing one byte at a time using the LINE FEED key.

It is also possible to switch to the VIEW WORD mode which is described next by hitting the / key.

Example:

04000; £	view byte at address $4000 in BANK 0
£	view byte at current PC

Command:	/
Purpose:	View/Modify word value at specified address (PC if none provided).
Usage:	**[addrexp]/**
Note:	addrexp is the address of the word to view. If no address is specified, the current address pointed by the PC will be used. This command is similar in use to the A command.

The / command is much like the previous command, VIEW BYTE. However, this one allows you to view a whole word in its correct representation. Thus if bytes C5 and 60 are stored consecutively at $3000, the **03000; /** command will give the following display:

```
DEBUG> 03000; /03000 = 60C5
```

Notice how the two bytes have been inverted. This is consistent with the 8502's LO/HI byte representation. Other than that, this command behaves much like the VIEW BYTE command, although it has an added option: @. When the @ key is typed, you get the value of the word at the address that you were previously viewing as a word. This sounds complicated but it really isn't. Let's say you would press the @ key while viewing the contents of $3000, then you would get the word value at address $60C5 (provided that $3000 holds the $60C5 value). This is useful when you are viewing complex list structures with lots of pointers to sub-lists and so on.

It is also possible to switch to the VIEW BYTE previously described by hitting the **£** key.

Example:

04000; /	view word at address $4000 in BANK 0
/	view word at current PC

Execution Commands:

Command:	G
Purpose:	Begin full-speed execution of program.
Usage:	**[addrexp]G**
Note:	addrexp is the address to begin execution. If no address is specified, the current address pointed by the PC will be used.

The G command starts full-speed execution of the program. Control will not return to Hexpert unless a breakpoint or a brk instruction is encountered.

Example:

```
02000; G
```
begin execution at address $2000 in BANK 0

```
G
```
begin execution at current PC

Command:	J
Purpose:	Execute a subroutine at current PC.
Usage:	J
Note:	Takes no parameters.

The J command is only a T0 command in disguise. It will only have effect if the instruction at the current PC is a jsr instruction. If so, Hexpert will set a brk instruction after the jsr and will perform a JMPFAR at PC. This may sound a bit complicated. What it means is that the subroutine at the current PC will be executed and control will return to Hexpert after the routine is executed.

NOTE: It is important that the return address on stack is not modified by the subroutine or else, control will not return to Hexpert, which could have unpredictable results.

Example:

```
. 02000 20 D2 FF JSR $FFD2
DEBUG> J                          execute routine at $FFD2
. 02003 xx xx xx AAA 0000
DEBUG>                            control returns to Hexpert after RTS
```

Command:	S
Purpose:	Single step through instructions and in and out of subroutines, also allows to set a new PC value when used with an address.
Usage:	[addrexp]S
Note:	addrexp is used to set a new program counter value. If no address is specified, the current address pointed by the PC will be used.

The S command, when used with no prefixed address will single-step the processor, executing one instruction at a time. All 8502 registers and modified memory locations are updated and the following instruction is printed. The display of the new registers depends on the BEEPER flag in $F9. This flag can be toggled on/off with the D command.

The S command when used with a prefixed address will make this address the new program counter value, without executing any instructions.

The SINGLE STEP command works by inspecting the instruction to be executed and determining where the following instruction is located; it then places a temporary breakpoint at that location. For most instructions this is a trivial process because the next instruction to be executed will be the next instruction in memory. However, for instructions which transfer control by reloading the program counter (e.g., jmp, jsr, rts, branches, etc.), the address of the next instruction must be calculated accordingly.
IMPORTANT: You cannot step through ROM. Hexpert will allow you to step through ROM code, however the

program will probably crash right away or something weird might happen. Hexpert will be modified to prevent SINGLE STEPping through ROM code in an upcoming version.

If, while single-stepping a breakpoint is encountered, a message will be displayed and the breakpoint will be removed from the breakpoint table.

HINT: It is possible to single-step through some of CBM's Kernal routine by first transfering the entire ROM Kernal to the underlying RAM, using the **T** command from the monitor. The **T** command should look like this: **T FC000 FFFFF 0C000 + [RETURN]** This will move ROM code in RAM BANK 0. Then using the debugger's **K** command, switch the ROM off and you will be able to single-step through the pseudo-ROM routines.

Example:

```
DEBUG> 02000; S                    set new program counter at address
. 02000 A9 41     LDA #$41         $2000 in BANK 0
DEBUG> S                           execute the LDA instruction AC = $41
. 02002 20 D2 FF JSR $FFD2
DEBUG> J                           execute subroutine at $FFD2
. 02005 00        BRK
```

Command:	T
Purpose:	top-step through subroutines.
Usage:	**Tparam**
Note:	param refers to the number of bytes to skip when setting the return temporary breakpoint.

The **T** command will only have effect if the instruction at the current PC is a **jsr** instruction. If so, Hexpert will set a **brk** instruction after the **jsr** with an offset of param bytes, where param is a numeric value in the range 0 through 9. Then Hexpert will perform the JMPFAR at PC. What this means is that the subroutine at the current PC will be executed and control will return to Hexpert after the routine. The use of the param parameter is to allow routines which are passed parameters via the return address on stack to be top-stepped too. Here is an example of such a routine:

```
          jsr       i_DoWindows   ; Call the window display routine
table     .word     WindowTable   ; Address to window table
          .byte     Screen1       ; Screen in which to draw window
done      bcc       error         ; Window display was not successful
```

In this situation, you would use a **T3** command so that the return breakpoint would be set at 'done' instead of at 'table'. The **J** command mentioned earlier is nothing more than a **T0** command, provided for practical reasons. Most programmers will make use of the **T** command, because few programmers use this inline parameter passing technique, but the command is there for your convenience anyway.

NOTE: It is important that the return address on stack corresponds to the correct value of param or else, control will not return to Hexpert, which could have unpredictable results.

In the example above, you would use the T command like this:

```
. 03000 20 00 C0 JSR $C000
DEBUG> T3                          Execute subroutine and set return
. 03006 90 0A    BCC $3012         break point at address $3006 instead
DEBUG>                             of $3003.
```

Breakpoint Commands:

When debugging a program, it is often desirable to stop program execution at a specific point so that you can check variables, flags or registers, making sure they contain the correct and expected values. The debugger implements this mechanism with breakpoints. A user-defined breakpoint, or breakpoint for short, can be set at a specific point in the program. When the breakpoint is encountered, control is transfered to the debugger, a message is printed, the instruction at the breakpoint is disassembled and the breakpoint is removed from the table, which also means that the original instruction is restored. This current version of Hexpert removes the breakpoint because of memory conflict, but an upcoming version of Hexpert should allow breakpoints to be reused any number of times.

The breakpoint is triggered before the instruction at the breakpoint is executed.

NOTE: You can set up to ten user-defined breakpoints. Care should be taken to avoid setting two breakpoints at the same address. This could have undesirable consequences if the breakpoints are removed in wrong order.

How Breakpoints Work: The Ins and Outs of Breakpoints

The 8502 implements a special instruction called **brk** (for break) which generates a non-maskable interrupt. The debugger intercepts this interrupt and treats it as a software breakpoint. When you set a user-defined breakpoint, the debugger replaces the data byte at the address with a **brk** instruction ($00). By going through the debugger, the breakpoints are automatically controlled and managed. Because the debugger saves the byte that was replaced, it is capable of restoring it later on.

However, there is nothing stopping you from manually placing **brk** instructions in your code using Hexpert's VIEW BYTE option. These **brk** instructions will actually appear as **brk**'s and will not be managed by the debugger. When one of the these **brk** instructions is encountered, the debugger returns with an appropriate message. The debugger will not execute or step through a **brk** instruction.

NOTE: Because breakpoints are implemented by modifying data in memory, they cannot be set in ROM. Any attempts to set a breakpoint in ROM will cause a zero to be written to RAM mapped behind it.

Command:	**B**
Purpose:	Set/Remove a user-defined breakpoint.
Usage:	**[addrexp]Bparam**
Note:	If addrexp is specified the breakpoint will be set at the corresponding address. If addrexp is not specified, the breakpoint will be removed. The breakpoint affected by this operation depends on the value of param, which must be a decimal numeric value (0-9).

The **B** command allows you to set user-defined breakpoints in memory. If you specify an address, the breakpoint will be set at this address. If not, then the corresponding breakpoint will be removed from the table. Your

determine which breakpoint (out of ten) is affected by specifying param which is a decimal number (0-9).
NOTE: Hexpert will not allow you to set breakpoints in zero page memory and it will not allow you to remove a breakpoint that has not been set already.

Appendix A: Hexpert Messages

Information Messages

These messages will be displayed everytime Hexpert want's to tell you something has beed done.

```
HEXPERT Vx.x, (c) 1990
BY ADVANCED SOFTWARE SYSTEMS
```
This message indicates that Hexpert has been enabled. This message appears every time you will type SYS 4864 + [RETURN].

```
> breakpoint removed
```
Informs you that a breakpoint has been removed via the REMOVE BREAKPOINT command.

```
> breakpoint has been set
```
Informs you that a breakpoint has been set via the SET BREAKPOINT command.

Error Messages

These messages will be displayed everytime Hexpert refuses to do a command because of a certain condition. These messages do not necessarily mean that something has gone wrong. They can simply mean that Hexpert has stopped executing a program because a BRK was encountered.

```
> breakpoint encountered
```
A user-defined breakpoint has been encountered during the program execution.

```
> user brk encountered
```
A **brk** instruction was encountered. You should use the **brk** instruction instead of a **rts** to exit out of you program.

```
> breakpoint invalid
```
An attempt has been made to set a breakpoint in zero page memory ($0000-$00FF). This message will also appear if you try to remove a breakpoint which had not been previously set.

```
> not a jsr instruction
```
This message is displayed if you attempt to use the JUMP TO SUBROUTINE or TOP STEP commands on an instruction other than a **jsr**.

```
> end of program
```
This message is displayed if you attempt to single step your program past the last **rts** command. This can be checked in SINGLE STEP mode by comparing the current value of the stack pointer with its initial value. If the current value is equal to the initial value, the program is at the same level it was when it first started so a **rts** will not be allowed. This is only a safety feature. The stack can not be checked while the program is in FULL SPEED execution mode, this is why it's important to terminate your programs with a **brk** instead of a **rts** instruction.

Appendix B: Hexpert Technical Notes

Debugger Dependency

Hexpert depends on the ROM monitor code at $B000-$BFFF. This should not be a problem, but it explains why the debugger will not work in C64 mode. Hexpert makes as much use as possible of the monitor's own code, and will use some of the monitor's variables. The debugger also uses some Kernal routines (BSOUT, INDFET, INDSTA, etc.). This should not be a problem either.

Off Limits Memory

At no time should the application modify the following memory areas:

1: $1300-$1BFF / BANK 0	(debugger kernal)
2: $FFFA-$FFFF	(interrupt vectors)

The application should be careful, as usual, if it needs to modify memory in the range $0000-$0400. This is the C128's operating system variable space. A lot of these variables are only useful for BASIC so then can be tampered with, as long as you restore their original values upon re-entering BASIC.

Miscellaneous

With the ROM bank enabled with the MMU register, any attempt to modify the ROM memory area will actually modify the RAM which is swapped over. This is why attempting to set a breakpoint in ROM will store a $00 (a **brk** instruction) in the RAM it maps over.

The application can modify the memory map register (location $D500) directly. Hexpert will sense the state and adjust appropriately.

The Hexpert QUIT command assumes that the important system variables in zero page are intact. If they are not, the system may crash, requiring a reset.

Keep in mind that Hexpert preserves the program's state when exiting to the monitor, so it is possible to toggle in and out of the monitor to use some of its more advanced memory viewing commands while executing a program.

Appendix C: Bibliography and Further Reference

Commodore 128 and the 8502 Machine Language

Commodore 128 Programmer's Reference Guide,
Commodore Bussiness Machines, Bantam Books, Inc.,
666 Fifth Avenue, New York, NY 10103 (1986)
This is clearly a "MUST HAVE" book. No C128 programmer should be without it. It contains a bundle of OFFICIAL C128 information and it includes some information on the rather obscure CP/M mode.

Commodore 128 INTERNALS: The Authoritative Insider's Guide,
Data Becker Books, published by Abacus Software, Inc.,
P.O. BOX 7211, Grand Rapids, MI 49510 (1985)
Another "MUST HAVE" book for the serious machine language programmer. It's main interest is obviously the entire disassembled/commented ROM listing of the C128 Kernal (over 200 pages of listing). Besides the ROM listings, this books has a wealth of interesting technical information on the different 128 components such as the MMU, the VDC and the VIC chips.

Machine Language for the Commodore 64, 128, and Other Commodore Computers,
Brady books, published by Prentice Hall Press, New York, NY 10023 (1986)
Written by the Commodore GURU himself, Jim Butterfield, this book is an excellent introduction to the 6502 machine language on Commodore machines. Contains memory maps for virtually all Commodore computers. I highly recommand this book if you are a novice or intermediate machine language programmer.

This is by no means a complete 6502 machine language bibliography, several other good books are available, notably some excellent books from Compute's. Don't forget your old copies of Compute's Gazette, RUN, Transactor, etc. Some of the best machine language tutorials can be found in these monthly magazines.

Additional Notes on Hexpert - 5/1/91

Caution is advised in using Hexpert to single step through programs that redirect the input channel to a disk file. It is possible for Hexpert to begin taking input from the disk file in this situation. It is likely to result in a system crash. A suggested work-around is to set breakpoints after any disk file input code, allowing the sectction of code to execute at full speed.

Hexpert may also crash if a #$00 is stored to $DC02.

Renumber 15xx Drives (ESC +)

Renumber 15xx Drives makes it easy to change device numbers on multiple drive systems. Unlike other programs to change device numbers, Renumber does not require the drive's power be cycled off and on. Renumber will work with 1541, 1571, 1581, and CBM RAMDOS disk drives.

Press **ESC +** to start Renumber. Renumber will ask for the number of the device to change with the prompt **Renumber?**. Enter the device number followed by **RETURN**. Renumber then asks the number to change the device to with the prompt **to?** Enter the new device number and **RETURN**. Renumber then confirms the new device number and whether the device is on or off. To quit Renumber, press **RETURN** by itself at a **Renumber?** prompt.

Reset 15xx Drives (ESC -)

Reset will restore 1541, 1571, 1581 and Commodore RAMDOS disk drives to their power-up reset condition. Use Reset to recover from disk errors or disk drive crashes, to restore default drive conditions, or after entering BASIC from GEOS.

To activate Reset, press **ESC -**. Reset will print a sign on message and begin resetting drives beginning with device 8. As each drive is reset, Reset prints its status, either on or off line. Reset will reset all drives numbered 8 - 15.

KeyDOS New Collect (ESC £)

New Collect corrects a deficiency in the C128's COLLECT (Validate) command. If a disk that COLLECT is operating on contains a boot sector, the sector is deallocated. The boot sector can be overwritten if you save other files on that disk. New Collect performs a COLLECT on the disk, then allocates the boot sector if one exists.

Place the disk to validate in the active drive and press **ESC £**. New Collect will operate on that disk and report if a boot sector is present.

Do not use New Collect on a GEOS disk or a disk containing GEOS files, use only the GEOS deskTop Validate command. If you happen to use New Collect (or the old COLLECT command) on a GEOS disk, you should immediately boot GEOS and use the deskTop Validate command on that disk if you do not want to ruin your files.

Unnew (ESC *)

Use Unnew to recover a BASIC program from memory that has been erased by the NEW command. Press **ESC *.** The message **Unnew - press RETURN** will be printed on the screen. Press **RETURN** and LIST your program to verify that it has been recovered.

Alternate Screen (ESC ↑)

based on a program by D.J. Morriss. From *The Transactor*, November 1987: Volume 8, Issue 03

This feature activates a second 80 column screen that can be used to hold different parts of a program while editing, help screens, or whatever else you can think of. All screen attributes, such as colors, flashing, and underlining will remain intact as long as the memory between $b00 and $bff is not disturbed. If that memory is overwritten, you can still switch screens, but you will not be able to recover the character attributes for it. To switch between screens, press **ESC ↑**. To switch back to the first screen, press **ESC ↑** a second time.

Find/Replace/Scroll (ESC :)

by Joseph F. Caffrey. From *The Transactor*, March 1987: Volume 7, Issue 05

This programmer's aid adds find, replace, and bi-directional list scrolling for 40 or 80 columns. To activate this utility, press ESC :. The general syntax for Find/Replace is displayed. If you list a BASIC program then press the cursor down key until the cursor goes below the bottom of the screen, the program will continue to list and scroll the screen upward. If the cursor is at the top of the screen and you press the cursor up key, the program will list in reverse and the screen will scroll downward.

The syntax for the Find command is:

@GET "text to be found"

This may be followed by a line number range as:

@GET "some more text" 50-100

to find the text in only part of a program. Use double quotes to find literal text strings that are not tokenized, or single quotes to find BASIC keywords and other tokenized text as follows:

@GET 'print#2,a$'

The syntax for Replace is:

@IF "text to find" THEN "replacement text"

The Replace command will also accept a line number range and uses the same conventions as the Find command in regard to single and double quotes.

Find/Replace/Scroll will crash if other programs are loaded over it ($1300 - $1700). Programs on the KeyDOS ROM that will not run with Find/Replace/Scroll include RAMDOS, Diskmon, Hexpert, and the Clock Manager. If you would like to run one of these programs, you must first power the C128 off and back on. Additionally, Find/Replace/Scroll will crash if you attempt to activate it while RAMDOS is active.

KeyDOS Screen Dump (ESC ;)

The KeyDOS Screen Dump will dump the contents of either the 40 or 80 column screen current window to the active device, either a disk file or PETASCII printer. This is handy for getting quick printouts of parts of program listings, monitor output, or for creating brief help files. KeyDOS Screen Dump works from within KeyDOS Screen Edit, from BASIC, or from the monitor. It will not alter the contents of the screen or change the position of the cursor. To dump the current window in which the cursor is located press ESC ;. If the active device is a printer, the screen contents will be printed. If the active device is a disk drive or RAMDOS, the screen contents will be written to a PETASCII SEQuential file named **.screen.** If a **.screen.** file exists on disk it will be overwritten, so if you need to save multiple screens, you must rename each one before saving another.

Single Drive File Copy (ESC =)

KeyDOS Single Drive File Copy offers most of the file copy features of the KeyDOS Utility, but works with a single disk drive. It is useful if you need to copy files with a system that has only one disk drive or only one drive of a particular type, such as one 1581 drive.

Press ESC = to activate Single Drive Copy. You will see the sign on and prompt: **Copy: 9>? "**

Pressing **RETURN** alone at the prompt will display the menu:

```
Catalog Drive Select Translate Quit
```

Catalog - displays the prompt: **Catalog: 8>? "*** and will accept a CBM DOS pattern for displaying a catalog of files on a disk. To display all files, press **RETURN**.

Drive - displays the prompt: **Drive?** Enter the number of the new source drive and press **RETURN**.

Select - returns to the Copy prompt.

Translate - displays the menu:

```
 Select file translation:
*1) None
 2) ASCII to CBM ASCII
 3) CBM ASCII to ASCII
 4) Screen to CBM ASCII
```

The selected option is highlighted. To select a different file translation option, press the number of the option. When finished, press **RETURN**. For more information, see the section on file translations during file copying for the KeyDOS Utility.

Quit - Quit the program.

To copy a file or files, enter a file name or CBM DOS pattern at the Copy prompt:

Example	Description
`Copy: 9>? "menu`	copy "menu"
`00, ok,00,00`	disk status
`menu,s`	reading "menu" (type SEQ)
`00, ok,00,00`	disk status

```
Insert destination disk in drive 9
and press RETURN  (ESC to abort).
```

At this point, follow the directions by removing the source disk from the drive, replacing it with the destination disk. Press **RETURN** to copy the file. Pressing ESC will return to the Copy prompt. After the file has been copied, the following prompt will ask you to return the source disk to the drive:

```
Insert source disk in drive 9
and press RETURN.
```

If there are more files to be copied, the process will be repeated. After the last file has been copied the program will return to the Copy prompt.

Screen Editor (ESC ,)

The Screen Editor turns the screen into a scratchpad. After activating Screen Editor you can type anything anywhere on the screen without sending commands to BASIC, generating syntax errors, or having the screen scroll away. Screen Editor does not clear the screen when it is activated, so it is ideal for adding comments to disk directory listings or monitor output. All C128 ESC commands for screen editing can be used from within Screen Editor as well as the KeyDOS ROM commands for Alternate Screen and Screen Dump. Use the KeyDOS Screen Dump to save or print Screen Editor screens. When you enter **ESC ,** to activate Screen Editor, the only visible sign is that the cursor will move to the home position if it wasn't already there. You can then edit the screen using the cursor keys, **INST/DEL, CLR/HOME, TAB**, any of the characters, and **RETURN**. Use **ESC ;** to save the screen. To exit the Screen Editor, press **STOP**. If you'd like to exit without altering the screen you are working on, press **ESC ↑** to get the alternate screen, then press **STOP**. Press **ESC ↑** again to return to the original screen.

The Screen Editor is also compatible with the screen clock. If the screen clock is displayed, it be saved with the screen during a screen dump. This is handy for including the time and date in a file.

Clock Manager (ESC .)

Clock-Calendar 128 subroutines by William J. Brier. From *The Transactor*, Volume 8, Issue 5

The Clock Manager is used to set up and control an on-screen clock and clock utilities. Both the Clock Manager and clock run in both 40 or 80 columns. The clock uses the 24 hour time format and includes a calendar and audible alarm. To start the Clock Manager, press **ESC ..** Clock Manager's menu appears:

```
0 - Exit Clock Manager
1 - Activate Clock        off
2 - Set Time
3 - Set Alarm             ??:??
4 - Set Date
5 - Display Clock         on
6 - Adjust Color
7 - Reverse video         off
8 - Alarm Enable          off
```

0 - Exit Clock Manager. Quit the Clock Manager and return to BASIC. If the clock was activated, it will continue to run after leaving the Clock Manager.

1 - Activate Clock. The clock status is also displayed on this line. If activated and option 6 (Display Clock) is also on, the clock will appear on the upper right corner of the screen. If the clock is on, pressing 1 again will disable it. This option shuts off the clock and removes it from the computer's operating system. You should select this option before loading any other software that loads into the same memory as the clock routines ($1300 - $1800). The KeyDOS ROM has a check for the clock built into it to prevent it from loading routines over this area. If you attempt to run a program from the KeyDOS ROM that uses this memory, you'll see the message **Please disable clock**. Run Clock Manager and select this option to proceed.

2 - Set Time. Presents the prompt: Correct time: (hh:mm). Enter the time in 24 hour format: Midnitht = 00:00, 8 am = 08:00, 2 pm = 14:00. Use the **DEL** key to correct mistakes. After pressing **RETURN**, the program will ask you to confirm your entry. Press **y** if you have entered the correct time, or **n** to reenter the time.

3 - Set Alarm. This line also displays the alarm time if set. If the alarm has not been set, the alarm time will be displayed as **??:??**. Enter and confirm the alarm time in the same manner as for setting the time.

4 - Set Date. Presents the prompt: **today's date: - -** . Enter the date in the order month-day-year. Each field should be entered as a two digit number. Example: enter September 13, 1992 as **09-13-92**. Use the **DEL** key to correct mistakes. After pressing **RETURN**, the program will ask you to confirm your entry. Press **y** if you have entered the correct date, or **n** to reenter the date.

5 - Display Clock. Use this option to shut off the clock display without disabling the clock. The status of the display is also indicated on this line. The clock display should always be disabled while editing programs in 80 column mode.

6 - Adjust Color. This selection changes the color of the clock display independently of the text color. When you select this option you'll see the prompt Use + or - to change color, **RETURN** to exit. Pressing + will cycle forward through all the available colors. - cycles backward, and **RETURN** returns to the main menu.

7 - Reverse video. This toggles reverse video mode for the clock display. The status of reverse video mode is also indicated on this line.

8 - Alarm Enable. The status of the alarm is indicated on this line. If it is on, the alarm will chime for one minute at it's set time. Selecting this option will shut the alarm (and chime) off. The alarm will not chime again the next day unless it is reenabled with this option.

Clock Manager / SmartWatch (KeyDOS ROM V2.CB and V2.CE only)

This version of the Clock Manager sets the screen clock time and date automatically when menu item **2 - Set Time/Date** is selected. There is not a seperate selection for setting the date. Menu items are numbered from **1 - 7**.

Notes for Programmers - Using the clock in other programs

The following programming notes detail some of the routines and memory locations used by the clock that can be accessed by other programs.

To disable the alarm	POKE 4896,129
To reenable the alarm:	POKE 4896,0
To stop the clock:	SYS 4900
To restart the clock:	SYS 4897

$1315	- Display flag: 0 = display on 1 = display off
$1316	- Clock color
$1317	- Reverse video display flag: 0 = normal 1 = reverse
$1300	- PETASCII Date: mm-dd-yy + chr$(0)
$1309	- PETASCII Time: hh:mm:ss + chr$(0)
$1312	- BCD Date: m d y
$1318	- Signature: "CLK" + chr$(0) (present only when clock is activated)
$1327	- BCD Time: h m s

Video Manager (ESC /)

Video Manager is a menu driven program that sets various options for the 40 and 80 column screens. Not all options are available in both screen modes, and Video Manager's menu will only display those options that are available in the current mode. Press ESC / to start Video Manager. To select options from the menu, simply press the number or letter for that option.

0 - Quit. Exit Video Manager and return to BASIC.

40/80 Column Colors

1 - Text
2 - Background
3 - Border (40 columns only)

The color selection options are adjusted by pressing + or -. When the desired color has been selected, press RETURN to go to the main menu.

80 Column Screen Mode

4 - Reverse Off. Toggles the VDC reverse screen function. Reverse status (on/off) is also indicated on this line.

5 - Interlace Off. Toggles the VDC interlace function. An interlaced display produces a better picture on some high quality monitors. On some monitors, such as a CBM 1902, interlace mode looks fuzzy or unstable. Interlace status (on/off) is also indicated on this line.

6 - Height. Adjust the height of the VDC display, measured in lines. This option produces a submenu with the following choices:

0 - 200 (Normal)
1 - 225
2 - 250
3 - 275

Not all height selections will look good on all monitors. There might be lines missing from the top or bottom of the screen. You might want to adjust your monitor to compensate for the increased size if you plan on using one of these settings. The currently selected value (if known) is also shown on this line.

7 - Tabs. On screen is a visual representation of TAB settings that can be edited. The numbers across the screen are a reference to the screen column numbers. A T represents a TAB set at that position. A total of 10 TABs are available.

```
          1         2         3         4         5         6
01234567890123456789012345678901234567890123456789012345678
T       T       T       T       T       T       T       T
```

Selecting this option presents a submenu:

0 - Clear all. This option removes all TABs.

1 - Default (8 spaces). This option restores TABs to their default condition by placing a TAB every eight spaces.

2 - Edit TABs. Selecting this option moves the cursor to the TAB line. The cursor left and right keys are used to move the cursor, and the TAB key will set a TAB at that position. If there is already a TAB at that position pressing TAB will erase the TAB. Keep in mind that KeyDOS Function Keys rely on default TAB settings to operate properly and that the KeyDOS CATALOG Function Key resets TABs to their default settings.

Cursor mode

8 - Flash On. Toggle cursor flash. The status is also indicated.

9 - Rate Slow. (80 columns only). Toggles the cursor flash rate between fast and slow. The status is also indicated.

A - Shape. (80 columns only). Edit the shape of the VDC cursor as shown in the Cursor display box. Use the CRSR up and down keys to raise and lower the top line of the cursor and the CRSR left and right keys to raise and lower the bottom line of the cursor. Many different cursor effects are possible. Press RETURN to go to the main menu.

Character mode (80 columns only)

B - Pixel Double Width. This option doubles the width of the VDC's pixels and effectively emulates a 40 column screen on the 80 column screen. Press B a second time to restore the 80 column screen.

C - Blink rate Slow. Toggles the character blink rate between slow and fast. The blink rate is also displayed on this line. Blinking characters can be selected by printing a CTRL O or CHR$(15) and is disabled by printing CHR$(143).

Internal Commands

The following commands are used internally by KeyDOS ROM and are not usually called directly by the user.

ESC - CTRL/A	Used by RUN 64 F-Key (F2)
ESC - CTRL/B	Loads Screen Clock routine
ESC - CTRL/C	Loads Screen Clock Set Time/Alarm routine
ESC - CTRL/D	Loads Screen Clock Set Date routine
ESC - CTRL/D	Loads Read SmartWatch routine - (V2.CB and V2.CE only)
ESC - CTRL/E	Swapper (copy bank 0 to bank 1)
ESC - CTRL/F	Swapper (copy bank 1 to bank 0)
ESC - C=/A	Install true ASCII 80 column character set
ESC - C=/P	Download default 80 column character set

Swapper Routines

The Swapper routines allow KeyDOS ROM to operate without affecting BASIC programs in memory. There are two Swapper routines, one to swap bank 0 RAM to bank 1, and another to swap bank 1 RAM to bank 0. Swapper routines are run automatically by KeyDOS ROM. They are very fast and powerful, and if you understand how to use them you can keep more than one BASIC program in memory at the same time.

The sequence of operation is this. When an ESC key sequence calls for a KeyDOS routine that must execute in BASIC memory space, the Swapper is called first (it can be called from the keyboard with ESC CTRL/E). The first swap is to preserve the contents of BASIC space so that it can be recovered later. Swapper takes the following actions:

1) Move 37 pages of memory from the start of BASIC (pointed to by TXTTAB, usually $1C01) to the start of variable space in bank 1 (pointed to by VARTAB, usually $0400) and increments the high byte of VARTAB by 37.
2) Disables NMI (STOP/RESTORE) and STOP key. Some applications reenable the STOP key.
3) Pass control to the routine to download and start the KeyDOS routine called for by the ESC key sequence.

The normal exit for KeyDOS routines that use the Swapper is to print ESC - CTRL/F. This calls the second portion of the Swapper. The sequence of events is essentially the reverse of the above.

1) Decrement the high byte of VARTAB by 37 and copy 37 pages of bank 1memory from VARTAB to TXTTAB in bank 0.
2) Reenable STOP key and NMI vectors.
3) Call BASIC routine to relink program and return to BASIC main loop.

Swapper executes entirely from ROM.

KeyDOS ROM Utility Disk

ACTIVATOR - activate the KeyDOS ROM on returning to BASIC from GEOS, or after running software that disables the KeyDOS ROM. Use ACTIVATOR whenever it is not desirable to reset the computer or inconvenient to use SYS 65366 to activate the KeyDOS ROM. To use ACTIVATOR with GEOS 128, double click on the ACTIVATOR icon. GEOS resets the C128 to BASIC mode, then prompts:

```
KeyDOS ROM Activator
Press the ALT key or
STOP to quit.
```

If you decide not to activate the KeyDOS ROM, press the STOP key. Otherwise, pressing ALT will activate the ROM, printing the sign on message. ACTIVATOR also resets the disk drives and returns the reset switch and STOP/RESTORE keys to their normal functions. GEOS puts 1571 drives into slow 1541 mode when it exits to BASIC. ACTIVATOR resets the drives back into 1571 mode. Keep a copy of ACTIVATOR on your most often used GEOS disks so that you can easily get back to KeyDOS ROM.

ACTIVATOR.QBB - activate KeyDOS ROM when used with a Quick Brown Box battery backed RAM cartridge. Place ACTIVATOR.QBB on the QBB. If you use the QBB manager to select ACTIVATOR.QBB as a QBB autoexecuting file, it will run every time the C128 is turned on or reset. Otherwise, manually enter the command to run ACTIVATOR.QBB. This is the only way of activating KeyDOS ROM when used with a Quick Brown Box.

TEST-ALT-SCREEN - demonstrates how a KeyDOS ROM ESC command can be used in a BASIC program.

PHONE-FILE.BAT - a simple phone directory batch file.

COLORDEMO.BAT - a batch file to display all combinations of 80 column colors.

READKEY.BAT - a batch file to demonstrate keyboard input.

KEYBOOT V2 - by Geoff Sullivan. Use KeyBoot to save function key definitions in an autoboot sector on an autobooting disk. KeyBoot uses a little-known feature of Commodore DOS that allows it to work with existing autobooting disks. Follow the instructions in the program. List the program for more information on the techniques used.

KD ROM VERSION - Reports if a KeyDOS ROM is installed internall or externally, and it's version number and ID.

ASCIISET DEMO - Demonstrates 80 column true ASCII character set.

WRITE-SEQ-FILE - Installs a function key command to write a SEQ file (without line numbers) from a BASIC program. This can be used to create batch files.

WRITE-BAS-LIST - Installs a function key command to save a BASIC program as a SEQ LIST file. This can be used to create program segments to be merged with the Batch Exec F key.

KeyDOS ROM Memory Use

Function	Address (Bank 0)	Comments
KeyDOS ROM (not activated)	None	
KeyDOS ROM (activated)	$0FC0 - $0FFF	ROM Interface code
	$1000 - $1FFF	Function Key buffer
Reload F Keys	$1000 - $1FFF	Function Key buffer
Help Screen	None	
KeyDOS Utility	$0B00 - $0BFF	
	Swapper	
KeyDOS Compiler	Swapper	
	$1000 - $1FFF	Function Key buffer if used
Visual Partitioner	Swapper	
Subdir Key on F8	$1000 - $1FFF	Function Key buffer
Batch Exec Key on F8	$1000 - $1FFF	Function Key buffer
RAMDOS	Swapper	
	RAMDOS Interface page	Location selected by user
GEOS SuperRBoot	Don't care	
Diskmon 128	$1300 - $15FF	
	$0B00 - $0BFF	If /r command is used
Monitor Dump	$1000 - $10FF	Function Key buffer
Hexpert	$1300 - $1BFF	
Renumber Drives	Swapper	
Reset Drives	Swapper	
New Collect	Swapper	
Unnew Program	$0B00 - $0B0B	
Alternate Screen	$0B00 - $0B9F	
Find / Replace / Scroll	$1300 - $17FF	
Screen Dump	$0B00 - $0BFF	
One Drive Copy	$0B00 - $0BFF	
	Swapper	
Screen Edit	Swapper	
Clock Manager	$1300 - $17FF	
	Swapper	
Video Manager	Swapper	

The Swapper uses 37 pages of RAM in bank 1 starting at the address pointed to by VARTAB, usually $0400 - $2800. Space in lower bank 1 RAM can be protected so that it is not overwritten by Swapper by adjusting VARTAB to the desired address above $0400. Swapper also uses $0B00 - $0B08 in bank 0.

Compatibility with CMD RAMLink

KeyDOS ROM is compatible with CMD RAMLink when RAMLink is disabled with its disable switch. RAMLink blocks access to some ROM functions when it is active. These problems are due to the methods that RAMLink uses to manage memory and are not unique to KeyDOS ROM; function ROMs other than KeyDOS are also affected. If KeyDOS ROM is to be used with RAMLink, it is recommended that KeyDOS ROM be installed in the C128's internal ROM socket.