Xetec

Lt. Kernal®

20 or 40 Megabyte Systems

Operating Manual

READ ME FIRST

1-22-88

THANK YOU for purchasing the Lt. Kernal. Please read the following before proceeding.

1. Make sure that all functions of your computer system are working properly before adding the Lt. Kernal Disk drive to your system because any problems that your system might have, could be seen as Lt. Kernal problems after it is installed.
2. READ the Lt. Kernal manual thoroughly before doing any installation. Do not assume anything. See Section II, pages 2-1 through 2-20.
3. Do not attempt any of the installation work with the power on. Do not connect anything to your computer with the power on.
4. When doing the internal hook-up on your computer, DOUBLE CHECK the connections before closing the computer.
5. If all of the above has been done properly, you are ready to continue with the steps in the manual to bring your Lt. Kernal to life. See Sections III and on.
6. Most problems encountered on an initial system boot are due to having a C-128 computer system set for an 80 column display. This will result in a Commodore sign-on message with no apparent activity from the Lt. Kernal. This is due to the fact that the Lt. Kernal is shipped to boot-up in the *64 mode* (40 column display). Simply make sure that both the monitor and computer are set to 40 column operation OR change the boot-up default to *C-128 mode* with the CONFIG command.
7. If you have carefully followed the instructions, your Lt. Kernal should be running and ready for your use. If it is not functioning, re-check the installation and refer to the Trouble Shooting section of the manual.
8. After you have the system running, become familiar with it before making changes. Learn how to use the simple commands before attempting a system CONFIGuration, for example.
9. DO NOT DO a SYSGEN unless absolutely necessary. Read the REPLACEMENT pages 2-25 and 2-26 titled **THE SYSGEN UTILITY** to determine when and how to do a SYSGEN.
10. When making changes, record the condition the system was in prior to making the changes.
11. Most Users get into trouble *assuming*. Make sure you know what you are doing, and then question the need to do it.
12. If you have any doubts, call us or check into the BBS for help. The BBS is listed on page A-7 of your manual.
13. If possible, save the box and packing material your unit was shipped in. This box was designed to safely ship the Lt. Kernal.

LIMITED WARRANTY

Xetec, Inc. warrants this Xetec hard disk drive known as the Lt. Kernal, to be in good working order for a period of one year from the date of purchase from Xetec, Inc. or an authorized Xetec dealer. Should this product fail to be in good working order at any time during this one year warranty period, Xetec will at its option, repair or replace this product at no additional charge except as set forth below. Repair parts and replacement products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and products become the property of Xetec, Inc. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, misuse, abuse, or non-Xetec modification of the product.

Limited warranty service may be obtained by delivering the product during the one year warranty period to an authorized Xetec dealer or to Xetec, Inc., and by providing proof of purchase date. Warranty will be valid for registered owners only. If this product is delivered by mail, you agree to insure the product or assume the risk of loss or damage in transit, to prepay shipping charges to Xetec, Inc., and to use the original shipping container or equivalant. Contact Xetec, Inc., 2804 Arnold Rd., Salina, Ks. 67401, (913)827-0685 for further information.

All express and implied warranties for this product including the warranties of merchantability and fitness for a particular purpose, are limited in duration to a period of one year from the date of purchase, and no warranties, whether expressed or implied, will apply after this period. Some states do not allow limitations on how long an implied warranty lasts, so the above limitations may not apply to you.

If this product is not in good working order as warranted above, your sole remedy shall be repair or replacement as provided above. In no event will Xetec be liable to you for any damages including any lost profits, lost savings or other incidental or consequential damages arising out of the use of, or inability to use such product, even if Xetec or an authorized Xetec dealer has been advised of the possibility of such damages, or for any claim by any other party.

Some states do not allow the exclusion or limitation of incidental or consequential damages for consumer products, so the above limitations or exclusions may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which may vary from state to state.

TABLE OF CONTENTS

INTRODUCTION

SECTION I

SECTION II

SECTION III

SECTION IV

SECTION V

SECTION VI

SECTION VII

SECTION VIII

SECTION IX

The Lt. Kernal®

Welcome to the world of Serious Computing!

PLEASE READ THIS MANUAL CAREFULLY
BEFORE ATTEMPTING TO INSTALL YOUR LT. KERNAL!

If the first few pages of section II of this manual seem a little stern, please understand — we want you to have the BEST possible service from your Lt. Kernal. The only way to let you know about the potential for damage you could do to your system is to tell it like it is!

You've invested in the most advanced disk system available for Commodore™ computers, and an incorrect installation **may damage the Lt. Kernal, your computer, or both.** Read the installation portion of this manual carefully before connecting together any parts of the system.

The Lt. Kernal results from eighteen years of experience in designing large multi-user, multi-tasking mini-computer systems. We have applied the technology used in those larger systems to improve the operating characteristics and speed of the C-64® and C-128® .

Thank you for purchasing the Lt. Kernal. You now have at your fingertips really high speed computing power and a comfortable, user-friendly disk operating system that significantly upgrades the functions and usability of your Commodore computer. Quality software written with the user in mind, and rugged, conservatively designed hardware are combined to produce the best accessory ever for Commodore computers.

FCC STATEMENT

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna
- Relocate the computer with respect to the receiver
- Move the computer away from the receiver
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful: "How to Identify and Resolve Radio-TV Interference Problems." This booklet is available from the U.S. Government Printing Office, Washington, D.C., 20402, Stock No. 004-000-00345-4.

MANUFACTURER'S WARNING: Using a cable between the "Host Adaptor" and the "Hard Drive Assembly", other than that provided by Xetec Inc. may result in interference to radio and television reception.

I.

The Lt. Kernal Disk Operating System V7.1

DOS FEATURES

- Runs certain copy-protected software
- Built-in KEYED INDEXED-RANDOM ACCESS METHOD
- Supports both C-64 and 128 modes of operation and CP/M®
- 58 additional or enhanced system commands and features
- Disk access speed more than 100 times faster than the 1541 floppy
- Automatic power-up execution of any application program
- Built in CP/M™ - like command-line features in C-64 and 128 modes
- User configurable system characteristics such as screen and character colors, and logical drive sizes
- Up to eleven logical drives may be defined on the hard disk
- Up to 7 hard drives in one system
- DOS allows up to seven files to be OPEN for reading and writing simultaneously in addition to the command/error channel
- DOS differentiates between BASIC and machine language programs
- Built-in backup and restore facilities
- Direct invocation of programs from the READY prompt
- Standard capacity of 20 Megabytes configurable up to 140 Meg.
- Optional Multiplexer allows up to 16 computers to share 1 Lt. Kernal system

TECHNICAL SPECIFICATIONS

Standard capacities, Formatted	20 Megabytes
Bytes per sector	512
Sectors per track	17
Tracks per cylinder	4
Number of cylinders	626
Media size	5¼'' (13.3 cm)
Recording density	10,200 Bits/inch
Track density	300 tracks/inch
Transfer rate to C-64 memory	38,000 Bytes/sec
Transfer rate to C-128 memory	65,000 Bytes/sec
Rotational speed	3,600 RPM
Average latency time	8.3 ms
Positioning time	18 ms min. 192 ms max.
Power consumption	
20 Meg. SCSI Drive unit	
117 Volts A.C. 60 Hertz	30 w typical 40 w max.
Host Adaptor	
+5 Volts D.C.	250 ma typical
Size	
20 Meg. SCSI Drive unit	12'' x 14'' x 2.5''
Weight	
20 Meg.SCSI Drive unit	10.80 lbs.

II.

INSTALLATION

CAUTION!

Unless your Lt. Kernal hard disk system is specifically labeled otherwise, your system has been factory wired for:

115 volts A.C. 60 hertz only

Do not plug the power cord into any other voltage or frequency outlet.

For domestic American systems, the correct outlet type is the three prong grounded variety. Use of a three prong adapter in a two prong ungrounded outlet is strongly discouraged since such use presents a high shock hazard and may damage your system.

CAUTION!

Always handle your hard disk/power supply assembly with the utmost care. Mechanical bumps and shocks to the drive could irrepairably damage it.

Never move or ship the drive without first conditioning it for shipping via the "ship" system command, described later in this manual.

Never move the drive unless power has been off for at least 30 seconds.

Never ship the drive in any container except it's original carton.

INSTALLING THE LT. KERNAL

Installation of the Lt. Kernal hardware takes only a few moments, but MUST be done carefully to avoid damage. For a typical system setup refer to FIG 1. Be gentle, and work slowly and deliberately, referring to the text frequently as you go.

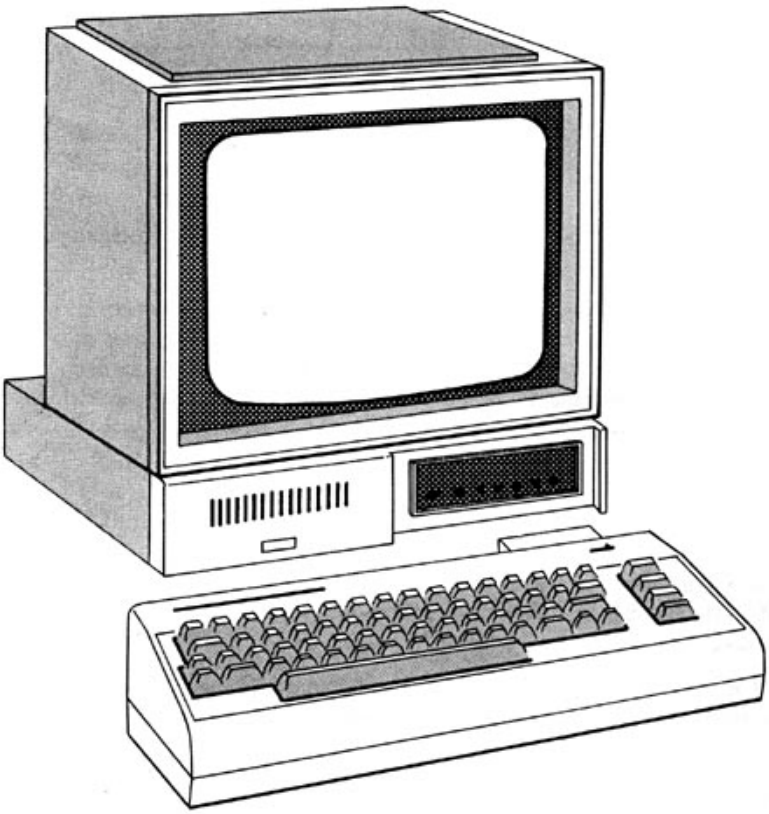


FIG. 1

FIRST
Make sure power is completely turned off to all components of your computer and the Lt. Kernal!!

SECOND
NEVER, NEVER, plug or unplug any interconnection of the system with power applied!!

THIRD
Always remember the second rule or you will eventually destroy some component of your system.

BEFORE BEGINNING YOUR INSTALLATION OF THE LT. KERNAL, CAREFULLY CHECK YOUR COMPUTER FOR PROPER OPERATION WITHOUT THE LT. KERNAL INSTALLED. THIS WILL PREVENT FALSE INDICATIONS OF TROUBLE LATER.

There are two possible installations of the Lt. Kernal. One is for the C-64 computer, and the other for the C-128. To utilize the 128 mode in the C-128, the C-128 adaptor board must be installed. Both installations will void your computer's warranty, as you will be required to open the computer case to install clips and/or an adaptor board. If you do not feel competent to do this installation properly, seek the assistance of a qualified computer technician. **CAUTION: Read each step thoroughly first before proceeding.**

TOOLS REQUIRED: A #1 phillips screwdriver and possibly a T-10 TORX driver plus needle nose pliers for the 64C inner shield and a small flat blade screwdriver.

C-64 or C-64C INSTALLATION

Step 1 - Remove screws on the bottom of the computer case, un-snap the upper keyboard section and carefully unplug the keyboard and indicator LED cables. Place this section aside for now.

Step 2 - Locate the HIRAM and CAEC cable assemblies. Refer to FIGS 2 and 3 to find your model and attach the HIRAM clip to the lead indicated of resistor R44 and attach the CAEC clip to PIN 6 of chip U27. Be sure that the clip is not shorting to any of the adjacent pins of either chip. Secure both leads with small pieces of scotch tape and dress each end out the opening for the expansion slot on rear of computer. NOTE: On some models, the metal shield must first be lifted by removing the TORX screws as needed, and then un-twist the small metal tabs around the perimeter of the shield. Replace this shield after the above clips are installed.

Step 3 - Install the keyboard section by first connecting the keyboard and LED cables and their lower section into place, snap case shut and install bottom screws into case.

Step 4 - Locate the HOST ADAPTOR and push the HIRAM connector onto the leftmost pair of pins of plug P1 as shown in FIGS 2 and 3. Push the CAEC connector onto the 4th set of pins from the left on plug P1 again as shown in FIGS 2 and 3. The jumpers on the 3rd and 5th sets of pins must also be in position as shown. The Host adaptor may now be inserted into the EXPANSION connector on the rear of the computer.

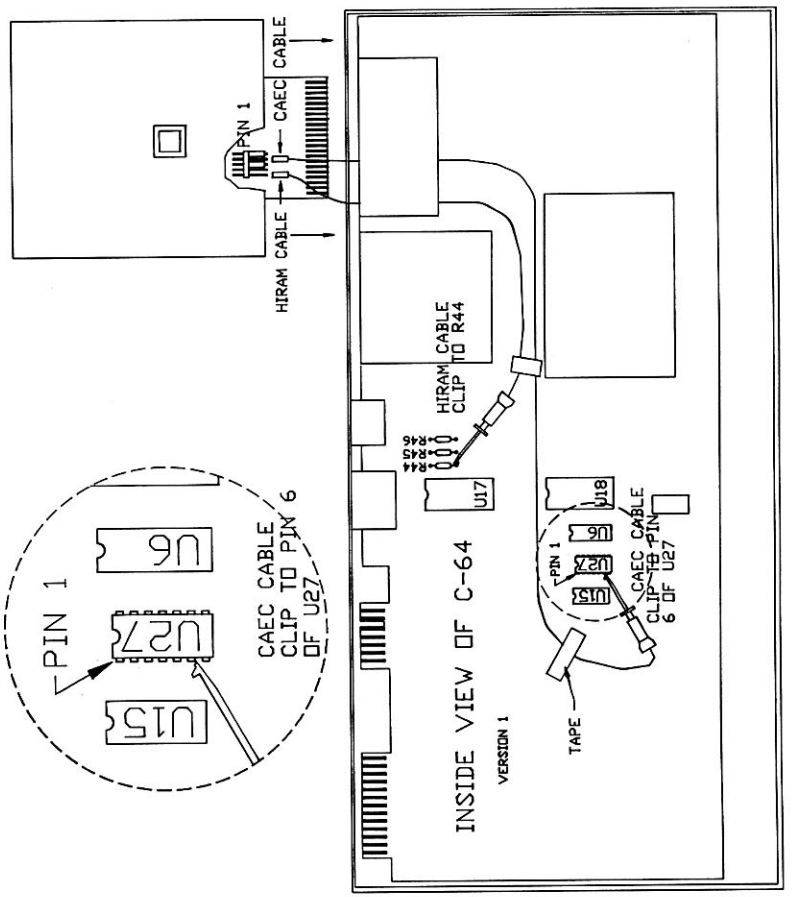

CAEC CABLE
PIN 1
HIRAM CABLE
HIRAM CABLE
TO R44
CLIP
U17
U18
U6
U27
U15
PIN 1
CAEC CABLE
CLIP TO PIN
6 OF U27
TAPE
INSIDE VIEW OF C-64
VERSION 1
PIN 1
U6
U27
U15
CAEC CABLE
CLIP TO PIN 6
OF U27

FIG. 2

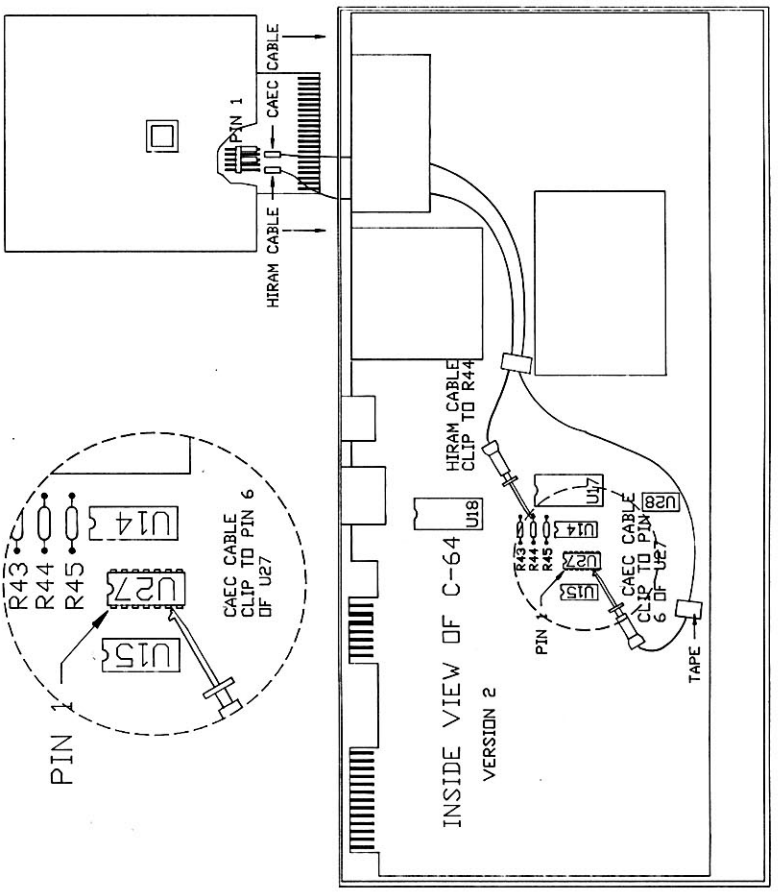


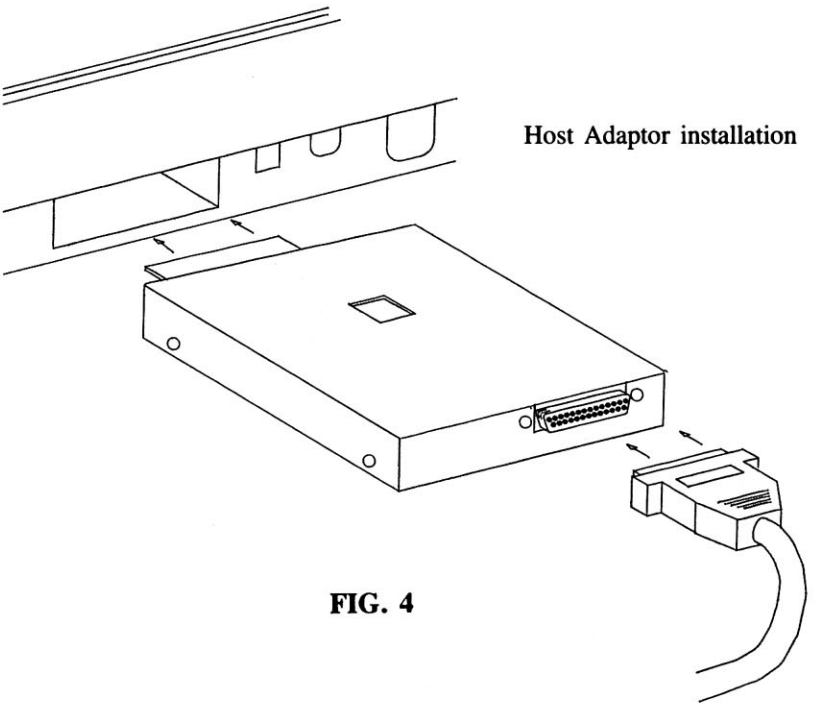
FIG. 3

You may have a newer version of the C64 computer board that is smaller at about 5'' wide. Use the information below to make your HIRAM and CAEC connections.

HIRAM cable to pin 28 of U6 (MPU) or to pin 6 of U8 (PLA)

CAEC cable to pin 5 of U6 (MPU) or to pin6 of U3.

Step 5 - Locate the 25 pin SIGNAL cable, plug one end into the connector on the rear of the Host Adaptor and secure the cable with the attached screws. Refer to FIG 4.



Host Adaptor installation

FIG. 4

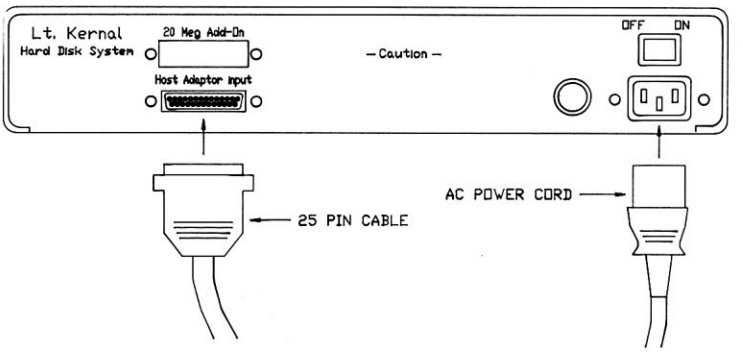
Step 6 - Attach remaining end of the 25 PIN SIGNAL cable to the HOST ADAPTOR INPUT connector of the HARD DISK enclosure. Again, secure the cable with the attached screws. Refer to FIG 5.

Step 7 - Locate the AC POWER CABLE and plug female end into the AC POWER receptable of the HARD DISK enclosure. Make sure the Power Switch is in the OFF position and plug the male end into a properly grounded 115 volt AC, 60 Hz outlet. Refer to FIG 5.

Step 8 - Re-connect any other components to your system such as printers, floppy disk and other accessories.

Step 9 - Refer to Power Application Sequence page 2-20.

Rear view of Lt. Kernal hard disk enclosure



C-128 INSTALLATION

Step 1 - Remove screws on the bottom of the computer case, un-snap the upper keyboard section and carefully unplug the keyboard and indicator LED cables. Place this section aside for now.

Step 2 - Lift the metal shield by removing the TORX screws and untwisting the metal tabs around the perimeter of the circuit board shield. Lay this shield aside for now.

Step 3 - Locate the C-128 ADAPTOR and lay on top of the SHIELD as shown in FIG 7. CAUTION: Discharge yourself from potential static electricity by touching the metal SHIELD before proceeding to the next step.

Step 4 - Locate chip U7 in FIG 6. Gently remove this chip from its socket by inserting a small flat blade screwdriver as shown and then carefully rotate or twist the blade left and right. DO NOT USE A PRYING ACTION! Once removed, check all pins for straightness, and proceed to next step.

Step 5 - Carefully insert chip U7 into the socket provided on the C-128 ADAPTOR. CAUTION: Be sure PIN 1 of chip matches PIN 1 of socket or indented end of chip matches indented end of socket. Refer to FIG 7.

Removal of chip U7 in C-128

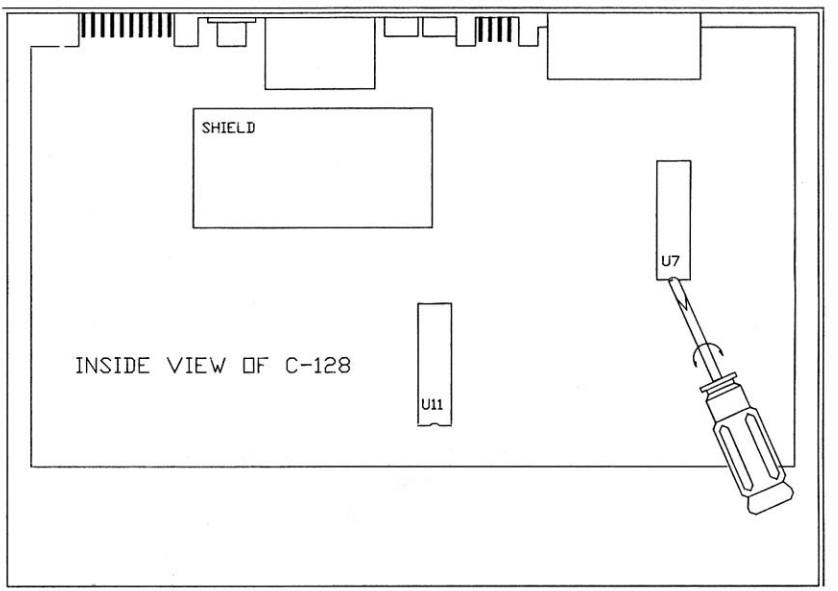


FIG. 6

C-128 adaptor board installation

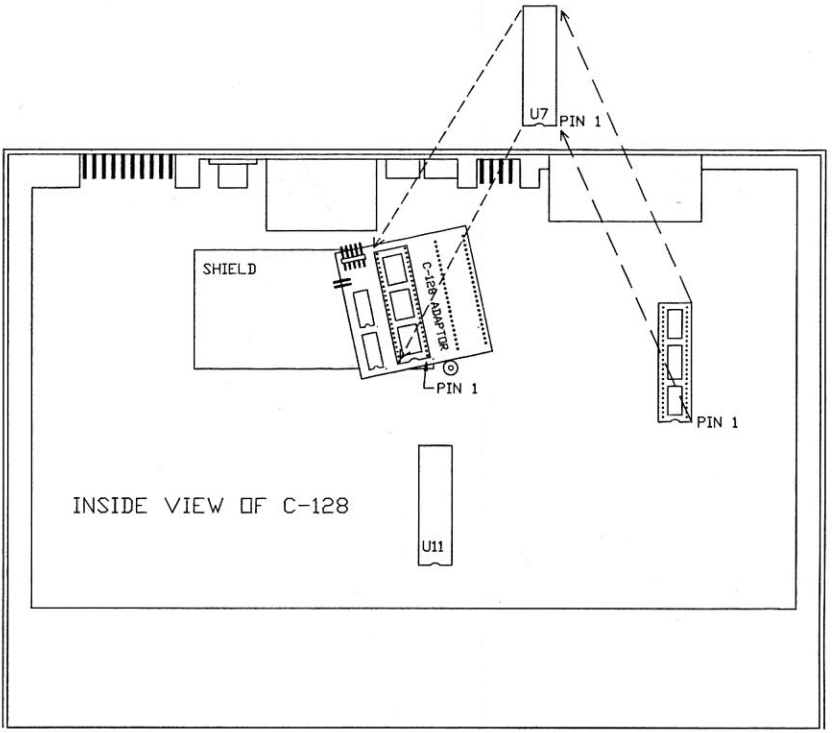


FIG. 7

C-128 cable connections with Rev C adaptor board

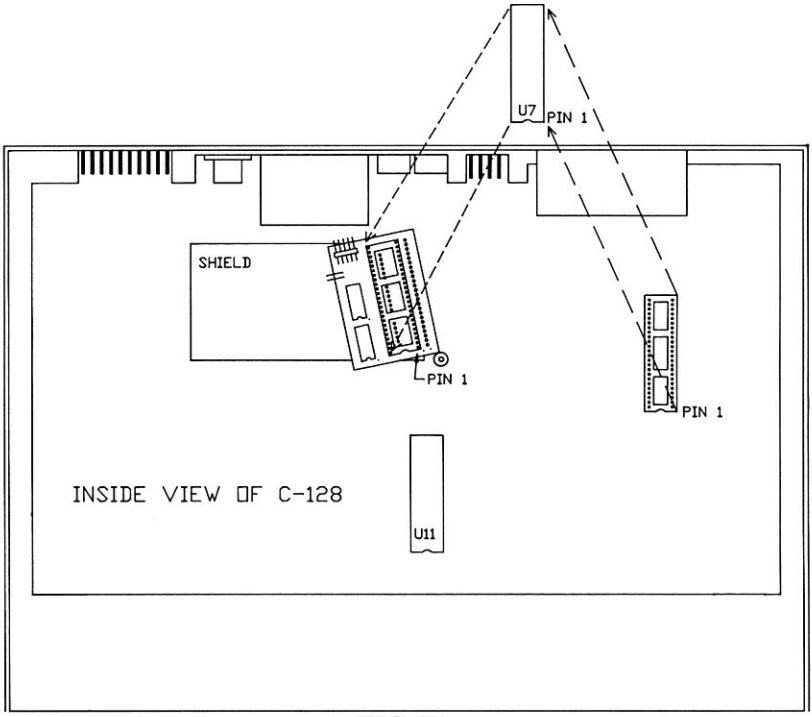


FIG 7A

The Rev B and Rev C adaptor boards are functionally identical. They vary only in size. Your Lt. Kernal 128 system may contain either Rev board.

Step 6 - Install the C-128 ADAPTOR into the socket vacated in Step 4 as shown in FIG 8. CAUTION: Make sure pins on the bottom of adaptor board are in their proper positions before firmly seating into place. Press down firmly in the area shown to firmly seat the ADAPTOR board in place.

C-128 cable connections

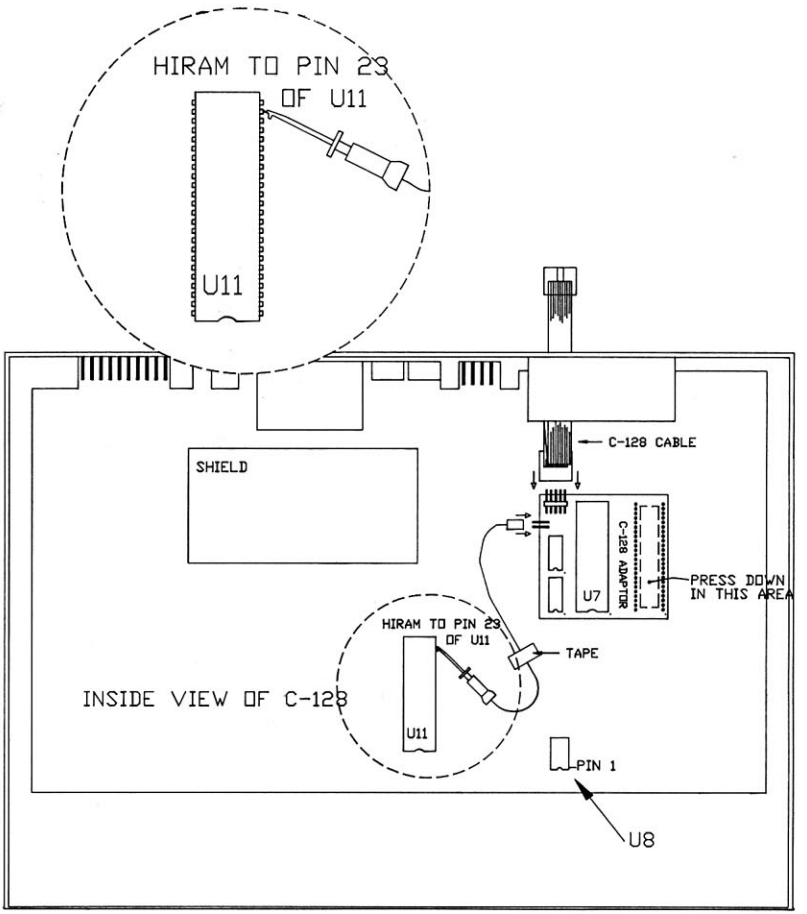


FIG. 8

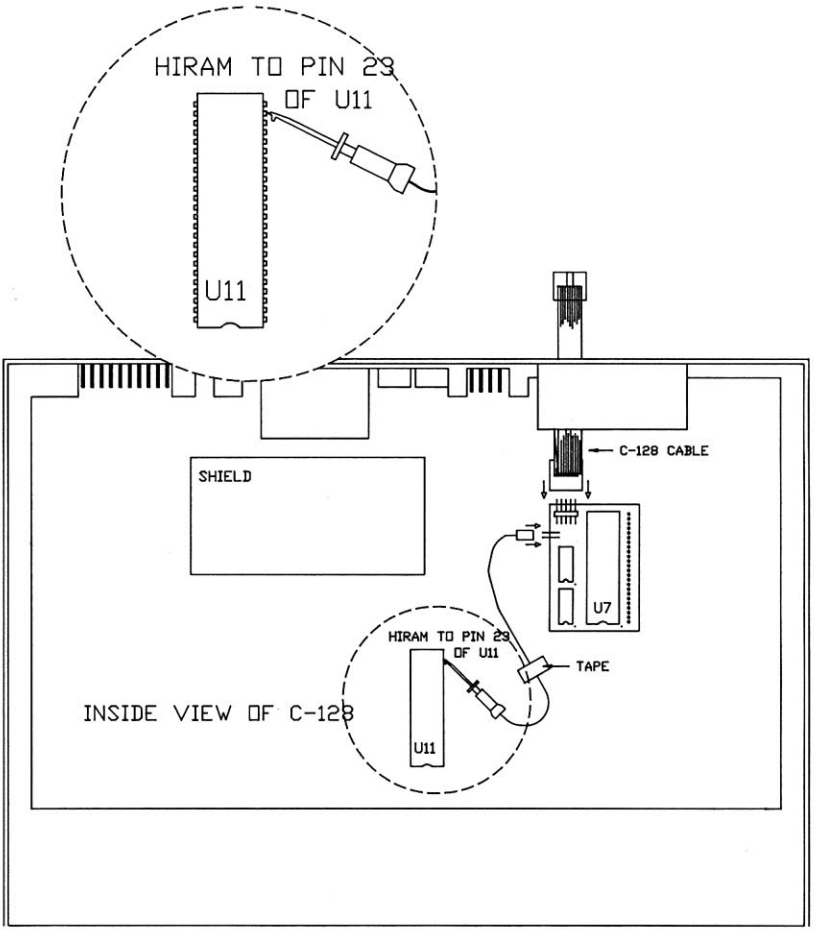

HIRAM TO PIN 23
OF U11
U11
SHIELD
C-128 CABLE
U7
HIRAM TO PIN 23
OF U11
TAPE
U11
INSIDE VIEW OF C-128

FIG 8A

Step 7 - Refer to FIG 8 and locate HIRAM CABLE. Clip this cable to PIN 23 of U11 as shown and secure with a piece of scotch tape. Be sure that the clip is not shorting to any of the adjacent pins of Chip U11. Plug the remaining end onto plug P2 of the C-128 ADAPTOR board as shown. (NOTE: The CAEC CABLE is not used in the C-128 installation)

Step 8 - Again refer to FIG 8 and locate the C-128 cable and take either one of the ends and position it so the flat ribbon is coming out of the top side of the connector. Now push this connector onto plug P1 of the C-128 ADAPTOR board as shown making sure all ten pins are properly entering each hole on the socket.

Step 9 - Locate the HOST ADAPTOR and remove the two jumpers on plug P1 as they will not be used in your C-128 installation. Push the remaining end of the C-128 CABLE onto plug P1 again with the flat ribbon coming out on the top side of the connector. The HOST ADAPTOR may now be inserted into the EXPANSION connector on the rear of your C-128. Dress or position the flat cable so it will allow you enough slack to remove your HOST ADAPTOR if necessary. Refer to FIG 9.

Host Adaptor cable connections for C-128

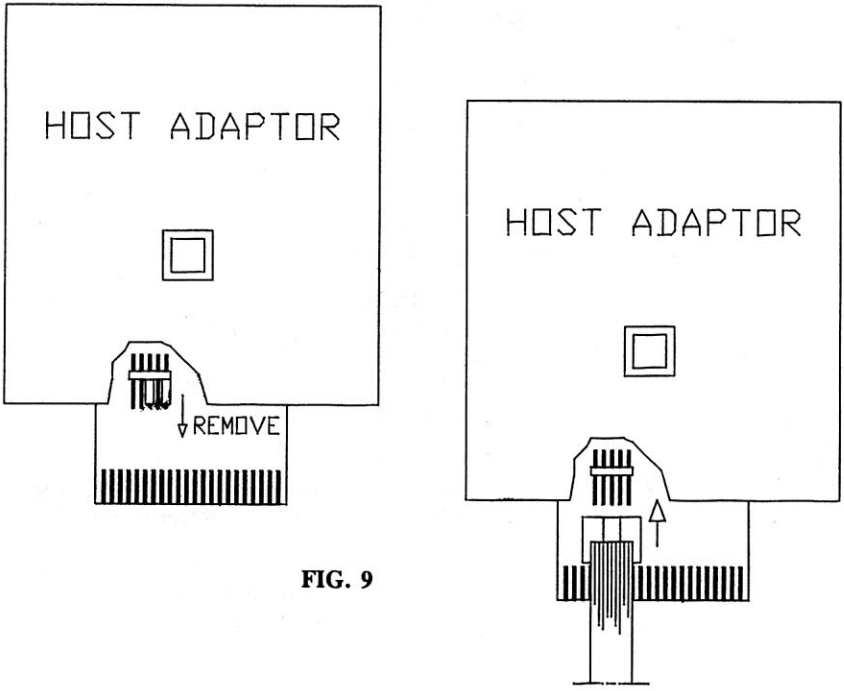


FIG. 9

Step 10 - Replace the metal shield on the C-128 main board being careful not to pinch or bind the C-128 CABLE.

Step 11 - Install the keyboard sections by re-connecting the cables, snap case halves together, and install bottom screws in case.

Step 12 - Locate the 25 pin SIGNAL cable, plug one end into the connector on the rear of the Host Adaptor and secure the cable with the attached screws. Refer to FIG 4.

Step 13 - Attach remaining end of the 25 PIN SIGNAL cable to the HOST ADAPTOR INPUT connector of the HARD DISK enclosure. Again, secure the cable with the attached screws. Refer to FIG 5.

Step 14 - Locate the AC POWER CABLE and plug female end into the AC POWER receptable of the HARD DISK enclosure. Make sure the Power Switch is in the OFF position and plug the male end into a properly grounded 115 volt AC, 60 Hz outlet. Refer to FIG 5.

Step 15 - Re-connect any other components to your system such as printers, floppy disk and other accessories.

Step 16 - Refer to Power Application Sequence page 2-20.

128D INSTALLATION

Step 1 - Unplug power, keyboard, and all other connections from the 128D computer.

Step 2 - Remove the 2 screws on the bottom and the 3 screws on the back of the computer case. Slide top back and lift up to remove. Place this aside for now.

Step 3 - Locate the 128D Adaptor Board and place it on a firm, flat surface. CAUTION: Discharge yourself from potential static electricity by touching the metal case of the computer before proceeding to the next step.

Step 4 - Locate U7 on the mother computer board. Refer to FIG 10. Gently remove this chip from its socket by inserting a small flat blade screwdriver as shown and then carefully rotate or twist the blade left and right. DO NOT USE A PRYING ACTION! Once removed. check all pins for straightness, and proceed to the next step.

Step 5 - Carefully insert chip U7 into the socket provided on the 128D Adaptor Board. CAUTION: Be sure pin 1 of chip matches pin 1 of socket or indented end of chip matches indented end of socket. Refer to FIG 11.

Step 6 - Install the 128D Adaptor Board into the socket vacated in Step 4 as shown in FIG 12. CAUTION: Make sure pins on bottom of adaptor board are in their proper positions before firmly seating into place.

Removal of chip U7 in 128D

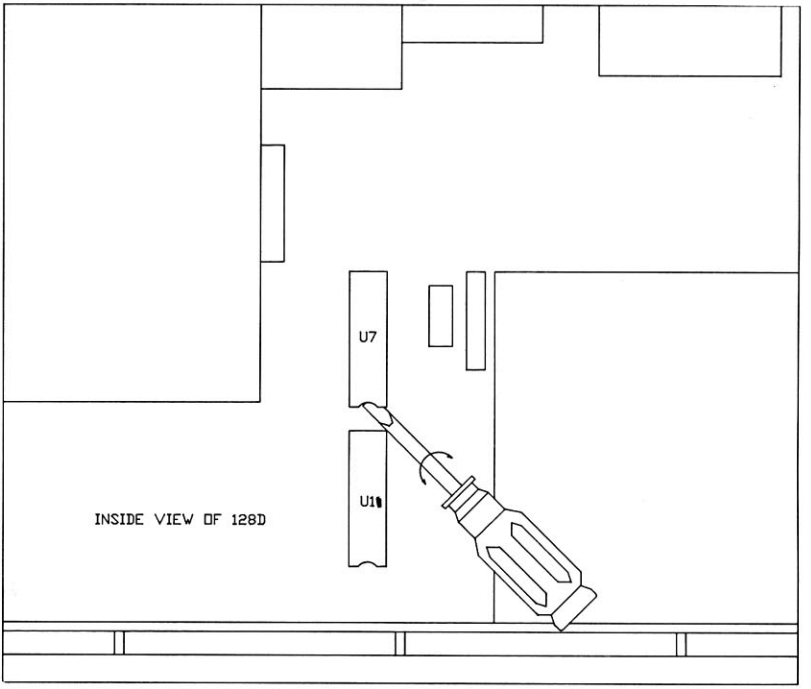


FIG 10

128D adaptor board installation

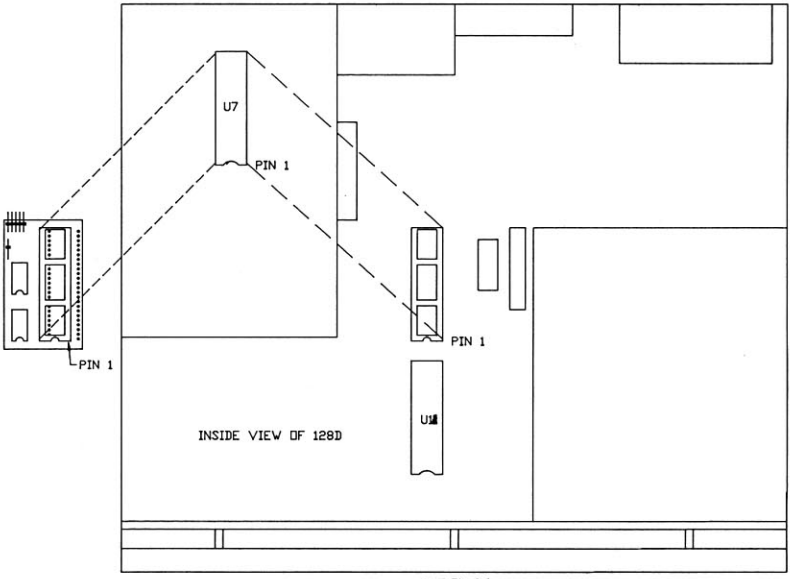


FIG 11

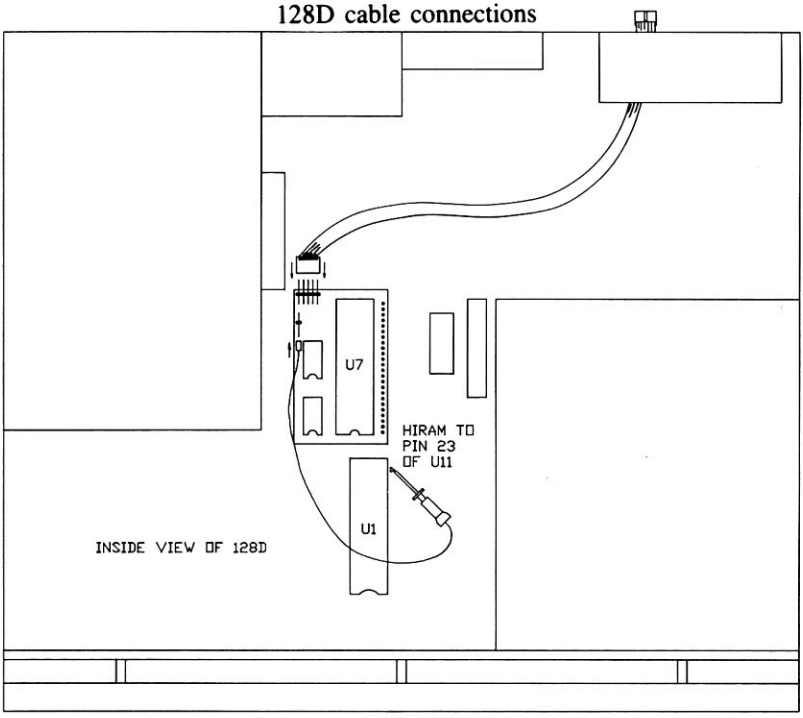


FIG 12

Step 7 - Refer to FIG 12 again and locate HIRAM Cable. Clip this cable to Pin 23 of U11 as shown. Be sure that the clip is not shorting to any adjacent pins of chip U11. Plug the remaining end onto plug P2 of the 128D Adaptor Board as shown. (Note: The CAEC Cable is not used in the 128D installation)

Step 8 - Again refer to FIG 12 and locate the 128 cable. Take either one of the ends and position it so the flat ribbon is coming out of the top side of the connector. Now push this connector onto the plug P1 of the 128D Adaptor Board as shown making sure all ten pins are properly entering each hole on the connector.

Step 9 - Repeat the applicable steps 9 through 16 of the C-128 Installation starting at page 2-11.

Burst Mode Modification

The Lt. Kernal does not support the "FAST" modes of the 1571 drive. If using a 1571 with a 128 computer, you must use the serial cable included with the Lt. Kernal, or else directory listings, programs, etc. will not load without being scrambled. With this cable, your 1571 will operate at the speed of a 1541.

The following steps instruct how to modify your 128 computer to take advantage of the burst mode. If you do not have a need for the burst mode, we suggest that you do not perform this modification. It will VIOLATE your Commodore warranty. If you do not have any technical skills and yet would like to make this modification, we suggest you have a qualified person perform it.

Step 1 - Remove screws on the bottom of the computer case, unsnap the upper keyboard section and carefully unplug the keyboard and indicator LED cables. Place this section aside for now.

Step 2 - Lift the metal shield by removing the TORX screws and untwisting the metal tabs around the perimeter of the circuit board shield. Lay this shield aside for now.

Step 3 - Locate chip U8 on the circuit board. Refer to FIG 13 for the C-128 computer and FIG 14 for the 128D computer. For the C-128, cut pin 9 of U8 and for the 128D, cut pin 13 of U8 as close as possible to the circuit board and CAREFULLY bend the pin upward so that it is parallel to the circuit board. Refer to FIG 15.

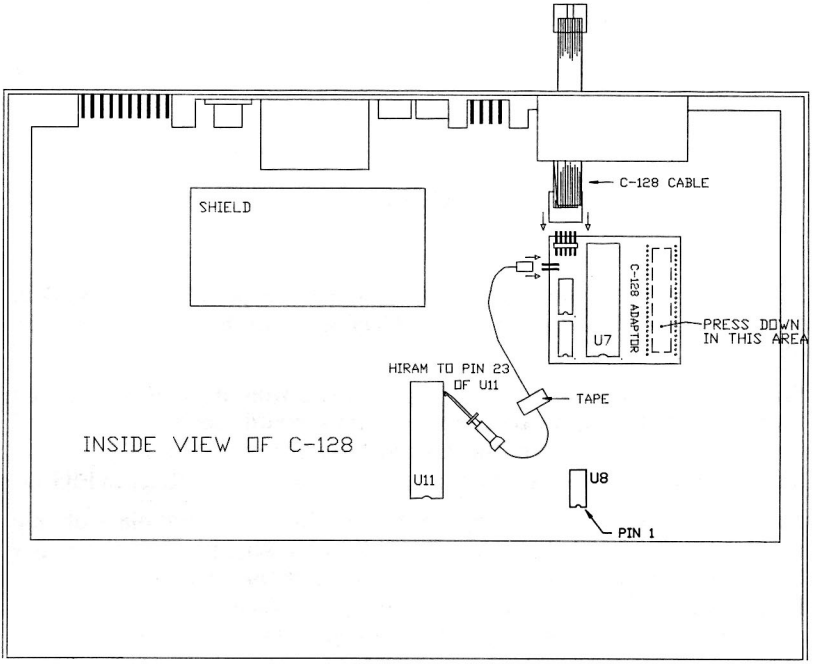


FIG 13

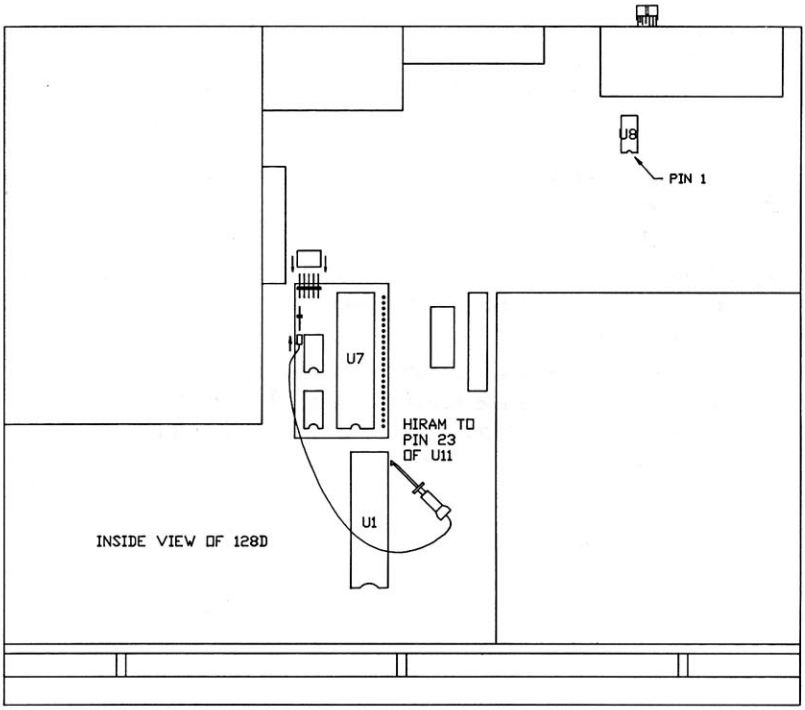


FIG 14

Step 4 - Solder one end of an insulated wire to the pin of U8 modified in step 3. Refer to FIG 16. Be careful NOT to solder bridge adjacent pins of U8.

Step 5 - Locate and unplug the Adaptor Board from the computer circuit board. Carefully remove U7 from the adaptor board. Solder the other end of the wire to the bottom of the Adaptor Board at pin 47 of the U7 socket being careful NOT to solder bridge pin 47 to pin 46 or 48. Refer to FIG 17.

Step 6 - Re-install U7 into the socket. CAUTION: Be sure pin 1 of chip matches pin 1 of socket. Re-Install the Adaptor Board into the computer circuit board making sure pins on the bottom of the adaptor board are in their proper positions before firmly seating into place. Refer to C-128 and 128D INSTALLATION—pages 2-7 through 2-14.

Step 7 - Re-assemble the computer as previously outlined refering to pages 2-7 through 2-14.

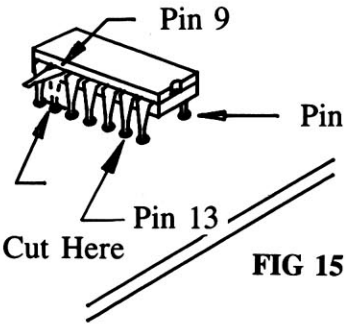


FIG 15

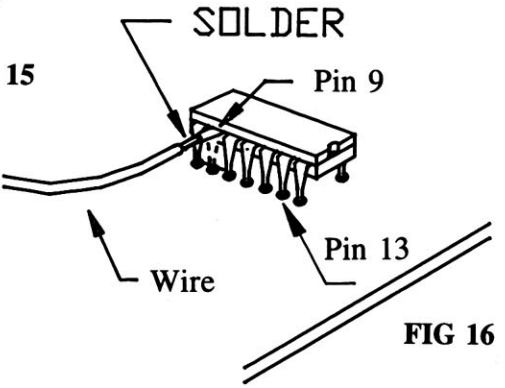
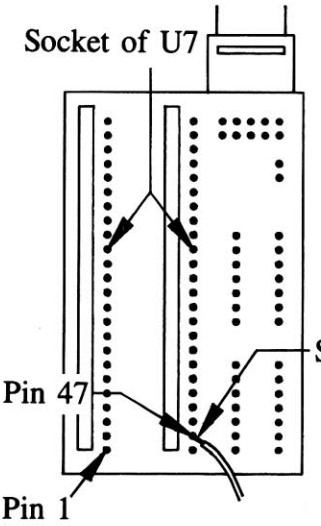
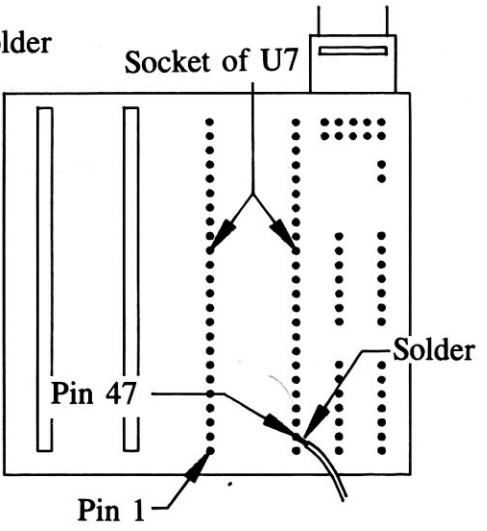


FIG 16



128 Rev. C Adaptor Board
Bottom View

FIG 17



128 Rev. B Adaptor Board Bottom View

I/O Modification for CP/M

CP/M operation requires the Host Adaptor to be set for I/O-1. The current version of the Lt. Kernal should already be set for I/O-1. Refer to FIG 18. The current version Host Adaptor (Rev. C) is the only one that has the I/O selectable pins. If you have an older version system, you will have to do a slight modification to the Host Adaptor. Refer to FIGs 19 and 20.

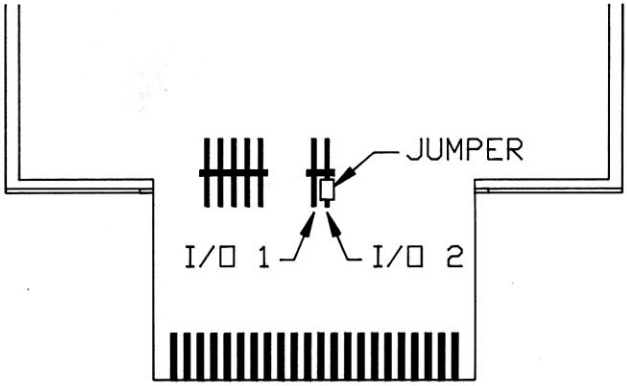


FIG 18
Host Adaptor Rev. C

To select I/O-1, remove the jumper from I/O-2 pins and slide on I/O-1 pins.

The Rev. B Host Adaptor is manufactured to operate in I/O-2 mode. To convert to I/O-1 follow the steps below. If you do not have technical skills, we suggest you have a qualified person perform this modification.

Step 1 - Cut trace on component side of Host Adaptor as shown that connects gold pad 10 with feed through hole.

Step 2 - Solder an insulated wire (approx. 22 ga.) as shown from feed through hole to gold pad 7. Be careful not to solder 'bridge' the wire to any other surrounding pads.

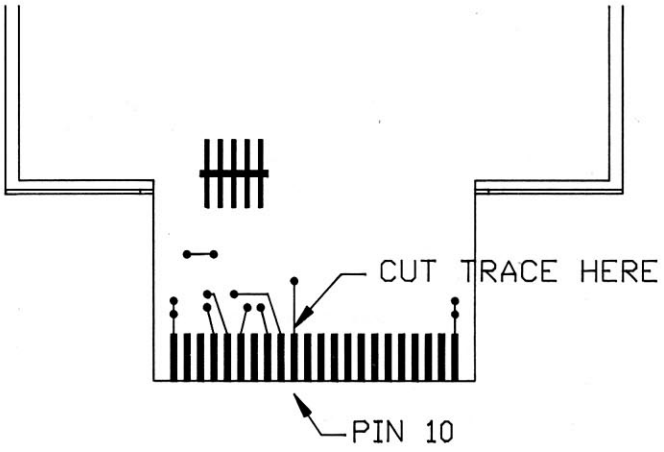


FIG 19
Host Adaptor Rev. B

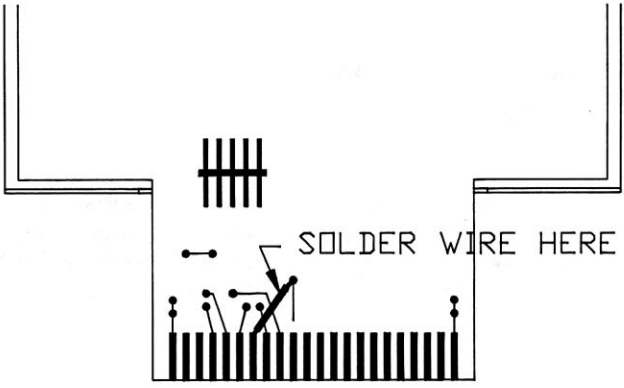


FIG 20
Host Adaptor Rev. B

POWER APPLICATION SEQUENCE

Power should be applied to your Lt. Kernal/Commodore combination in a specific manner.

Before you do power up your system, please remember that things are going to act a little differently than what you accustomed to seeing, so read this whole section before actually applying power. We want you to know what to expect BEFORE it happens.

Follow the steps in this order:

1. Monitor or television set.
2. Printers, floppy disks, and any other accessories EXCEPT the Lt. Kernel hard-disk.
3. The Lt. Kernal hard disk system.
4. Finally, the Commodore computer itself.

POWER REMOVAL SEQUENCE

1. The Lt. Kernal hard disk system
2. Printers, floppy disks, and any other still powered accessories, including your monitor or TV set.
3. The computer.

Read Section IX before continuing.

A demo program called PXE is located on LU0, USER00, and may be run in the C-64 mode by simply typing PXE and then press the "RETURN" key. Ignore the message "HIT ANY KEY WITHIN 5 SECONDS" or it will take you to an un-documented editor of the demo program. ENJOY!

III.

ACTIVATING THE SYSTEM

The Lt. Kernal is configured to come up in the 64 mode. (This can be changed). Therefore, your computer system must be ready to use the 64 mode which is a 40 column screen.

> With the 128 computer, DO NOT HOLD the C= key down during power up. The Lt. Kernal will automatically take control and place the 128 in a 64 mode. If you do not see any results at this point, you are probably in an 80 column condition.

NOW IT ALL COMES TOGETHER! When you apply power to your system without the Lt. Kernal present, you ordinarily would expect to see the Commodore BASIC power-up messages and then the BASIC ready prompt within just a couple of seconds of turn-on.

That's not going to happen with the Lt. Kernal ... at least not instantly. What will happen instead is this:

As soon as the drive has run up to speed, you should see the red indicator on the front of the drive blink twice, just briefly. About three seconds after that, the light will come on solidly for one second while the Lt. Kernal copyright messages appear. If the volume control on your monitor or TV is turned up, you'll hear a beep and see the new READY prompt with the drive's device number and logical drive number included. On a C-128, you may see even more activity. If the Lt. Kernal is configured to power up as a C-64, you'll see the screen blank again. The whole process repeats, finally to arrive at the C-64 mode of operation.

A lot went on to get to this point, and that's why there's a delay after you turn on the system. While you were waiting for the system sign-on message, and while the disk drive was running up to speed, a long series of system diagnostics took place — checking the Lt. Kernal Host Adaptor — testing the drive's controller electronics — and finally, even testing the Disk Operating System software installed on your drive.

If any one of the diagnostics along the way should fail, you'll just see the regular Commodore sign-on without the Lt. Kernal message, and without the beep. If the tests do fail, please TURN THE SYSTEM

OFF and turn directly to "TROUBLE-SHOOTING" in this manual.

NOW CHECK YOUR CABLING AND WIRING — AND TURN ON YOUR SYSTEM.

Then we'll introduce you to your new DOS.

IV.

OPERATING CONCEPTS

HOW IT WORKS

The Lt. Kernal DOS was written with the business user in mind. Many comfortable and easy-to-use new features have been added to your Commodore disk operating system while still supporting most of the existing 1541 floppy commands. The only commands not supported have no appropriate use in this environment. Most existing applications written in BASIC will run unmodified under the Lt. Kernal DOS. Many machine language applications and utilities such as assemblers, editors, and 'wedges' will also operate normally under the Lt. Kernal's control — but some programs will not run in cooperation with the Lt. Kernal. The reason lies in how the Lt. Kernal got its name, and in how it operates.

In order to support the tremendous speed at which the Lt. Kernal operates, it was essential that the cartridge/expansion port be used to communicate with the hard disk. To do that, there has to be a body of programs run by the Commodore computer to control the cartridge port and the Lt. Kernal Host Adaptor itself. But we wanted the Lt. Kernal to operate without making you sacrifice any of the memory you were accustomed to using. That has been accomplished by making the DOS support programs run in RAM (a modified Kernal) on the Lt. Kernal Host Adaptor. Since the Commodore computer itself is running the DOS, a few "programming considerations" (discussed in a later section) must be observed in order not to disrupt the always-running DOS.

IX

PROGRAMMING CONSIDERATIONS FOR THE LT. KERNAL DOS

The Lt. Kernal DOS was written specifically to satisfy the needs of the business or scientific software developer and to supply an excellent target system on which that new software might be run. In almost every instance, BASIC programs developed for the 1541 floppy disk will operate under control of the Lt. Kernal DOS. Machine language programs require some special precautions to be completely interchangable between the Lt. Kernal system and a 1541 environment.

Both BASIC and machine language programs can benefit from the 'speed tip' offered later in this section. Limitations on machine language and BASIC programs are few and well defined at the time of the writing of this manual.

Here are some general precautions to observe when programming the Lt. Kernal system:

1. BASIC programs should be modules which DO NOT contain machine language 'auto-boot' code at addresses lower than the 'normal' BASIC start address. Low memory auto-loaders can be used, but programs containing them CANNOT BE DIRECTLY INVOKED; they must be loaded via LOAD or L to function.

   Machine language 'tails' appended to the end of BASIC programs are perfectly acceptable. BE AWARE, however, that various commands such as RENUM, MERGE, and FETCH will destroy or modify the machine language parts of such 'hybrid' programs. The machine language portions of such software will have to be relinked to the BASIC portion after the BASIC portion has been edited. (This same limitation occurs in a 1541 environment)

2. Machine language programs should not modify the stack pointer. This practice, although frequently used, is generally considered by professionals to be a programming 'trick', and is not considered to be good programming practice. With the Lt. Kernal in control, modifying the stack pointer (other than by balanced pushes and pulls) will nearly guarantee that the Lt. Kernal cannot properly intercept disk service requests. There ARE ways around this and we encourage you to experiment CAREFULLY.

3. Machine language programs should always use the 'KERNAL VECTORS' to request system or disk service. The KERNAL VECTORS are a set of indirect jumps or JSR's to various KERNAL ROM routines. When the computer ROM operating system was written, it was intended that programmers use these vectors if they wished to use ROM subroutines. The Lt. Kernal DOS supports that convention.

   Although you are free to use ROM routines by JMPing or JSRing to undocumented entry points within ROM, we cannot guarantee that hard disk requests will be properly intercepted by the Lt. Kernal DOS unless you use ONLY the KERNAL VECTORS for such requests.

4. The Lt. Kernal DOS intentionally does NOT support RANDOM reads and writes to the hard disk by track and sector. The 'U' error channel commands are not supported to prevent damage to the DOS itself. We can provide other secure ways to protect proprietary software on the Lt. Kernal to authorized third-party software developers.

5. Presently, the Lt. Kernal DOES NOT support the *Serial Vectors* ($FF93-FFB4).

Backup Copying

Any system of software worth using is also worth protecting. Backup copying is the only method available to secure your programs and data against loss.

We have expended every effort to make sure that the Lt. Kernal DOS and the hard disk system hardware will be reliable. Even with that effort, WE CANNOT ASSURE that your system will not fail someday. If the system does fail, it is possible that any data or programs on the hard disk at the time of the failure will be lost.

COPY YOUR IMPORTANT FILES TO DISKETTE as often as necessary to enable you to recover in case your files are lost or accidentally erased. We have provided methods for you to perform backup copying.

For a moment, let's discuss the PROPER way to keep backups of your system in order to minimize the chances of losing any data. The correct process is sometimes known as 'DOUBLE GRANDFATHERING' or 'ALTERNATE' backup.

Say for discussion's sake that you do daily backup's (if you're keeping business data on your hard disk, you should!). On Monday evening, you copy your system and label the diskettes 'MONDAY'. After Tuesday's work, it's time for another copy to be done. On Tuesday evening, you MUST USE A NEW SET OF DISKETTES.

Rule 1

NEVER, NEVER, NEVER, NEVER DESTROY your MOST RECENTLY SUCESSFUL BACKUP!!!

Once you have completed Tuesday's copy, you label the diskettes 'TUESDAY' and store them as well in a different physical location than Monday's copy.

Rule 2

ALWAYS STORE ALTERNATE BACKUPS IN DIFFERENT PHYSICAL LOCATIONS (preferrably in different buildings).

This is simply to ensure that even a catastrophy like a fire would only lose for you at the most TWO working days of data.

On Wednesday, you can re-use Monday's diskettes; Tuesday's copy is now your most recent backup and is the one which must be protected most carefully. Each day you alternate sets of diskettes.

If you do serious business processing on your Lt. Kernal, make up two sets of labels for your two (at least two) backup copies. Make each set a different color. Next, make a calendar of your copying schedule, marking the COLOR of the set to be used on each data processing day. Make sure you truly alternate colors.

For instance: If you started this Monday with RED and you work five days a week, NEXT Monday had better be a BLUE day (no pun intended). Mark your calendar for about a month in advance then stick to it faithfully.

IF YOU HAVE A REAL DATA EMERGENCY AND HAVE FOLLOWED YOUR BACKUP SCHEDULE, YOU ARE RELATIVELY SAFE — BUT — IF A HARDWARE PROBLEM IS WHAT DESTROYED YOUR DATA — DON'T RISK A GOOD COPY OF YOUR DATA FOR ANY REASON UNTIL YOU ARE SURE THE PROBLEM IS COMPLETELY SOLVED AND THE SITUATION IS SAFE.

Floppy disk copying is just plain slow. To help relieve the slow disk pains, we have provided a program called FASTCOPY which is 1541/1571 specific to copy to and from floppy disks more quickly. FASTCOPY will copy a full diskette by filename in about two minutes. Even at that speed, and swapping diskettes as fast as your hands can move, it takes about four HOURS to copy a full twenty megabytes. Usually, though, your disk won't be completely full, so the copy will take less time than that. FASTCOPY is self documenting: Just follow its instructions. Remember, though that FASTCOPY is

designed just for Commodore 1541 and 1571 disk drives. If you own another brand or model, you may have to resort to a program such as 'COPY-ALL'.64L (supplied on the Lt. Kernal). COPY-ALL works fine but takes a
L O N G time to copy 20 megabytes (we won't scare you with HOW long).

If your Lt. Kernal disk system is full, our FASTCOPY utility offers a nice alternative copying feature which is safe as long as it is used with discretion. This feature is ARCHIVAL COPYING.

Archival copying is nothing more than the process of copying only those files which were modified since the last copy was done. Your copy takes less time because there are fewer files to copy. ARCHIVAL copying does have a pitfall.

Unless you keep EVERY consecutive archival copy you've ever made since the last COMPLETE copy of your system, you might not be able to recover from a major data loss. The solution is simple: Establish a regular schedule during which you normally do ARCHIVAL copies, but in which you also do COMPLETE copies at regular intervals.

A good regimen to follow is:

A COMPLETE copy of ALL FILES once a week (or two weeks according to the importance of your data

and

Archival copies every day or so between the complete copies.

Make sure to label and retain ALL the archival copies made until you have safely made a complete copy.

Again, FASTCOPY is self-documenting, and will take you through either a complete or archival copy automatically.

PLEASE BACK UP YOUR DATA AND PROGRAMS REGULARLY

With a hard disk's tremendous capacity, any data loss can be a HUGE data loss. Backing up regularly will protect you.

Directly Invoked Applications

With the Lt. Kernal DOS in effect, you can cause any program to be loaded and run simply by typing its name and pressing 'RETURN'. There are just a few precautions you must observe in order to make sure your programs may be directly invoked.

BASIC programs are no problem at all. Any BASIC program (and most compiled BASIC programs) will run unmodified under the direct invocation feature of the Lt. Kernal DOS. Machine language programs require that three constraints be observed.

1. Machine language routines must have the same LOAD address and ENTRY POINT (the address to which you would normally SYS). Even if a routine you wish to directly invoke does not meet this requirement, it is a simple task for any machine language (assembly language) programmer to 'patch' your program to meet this specification.

2. The routine MUST NOT over-write the 'stack' (memory addresses $0100-$01FF). Some 'AUTO-BOOT' loader programs do write over this area of the computer's memory, and they cannot be directly invoked. They will only work by LOADing them with the FULL Commodore LOAD syntax.

ANY routine must use the 'KERNAL VECTORS' (between $FFCO and $FFFF) for any system calls which require disk access. This requirement was set down by Commodore Business Machines when they provided the BASIC operating system in your computer and the Lt. Kernal DOS adheres to that standard.

AUTOSTART applications should follow the same guidelines as Directly Invoked applications.

'STACK' manipulation

It is usually considered a programming 'trick', and not good practice to alter the 6510's stack pointer in machine language routines. Sometimes, however, stack pointer manipulation is the most expedient way to assure that subroutines with multiple exit points stay in control. If your routines must alter the stack pointer, then you must preserve the return pointer residing on the stack when your routine first gets control. The Lt. Kernal DOS does a 'JSR' to any routine directly invoked, and expects the return address to be on the stack at the appropriate place when your routine does its final 'RTS' back to system control.

Reserved Memory Areas

Although the 'stack' should not be altered without observing the previously mentioned constraints, no other single byte of RAM is ever required or modifed by the Lt. Kernal DOS (without replacing its EXACT contents).

Speed Tips

The Lt. Kernal is faster than any other Commodore-compatible drive because of its rapid parallel transfer scheme. However, the 'KERNAL' operating system which Commodore provided with the computer does introduce substantial overhead which slows access data that is read from the files. The same holds true for the machine language routines that use BASIN (Commodore's GET character routine) or BSOUT to write data to files. There is a TREMENDOUSLY FASTER method you may use in NEW applications.

Since the Lt. Kernal can access data files so rapidly, it is in the best interest of speed that you save DATA as PRoGram files rather than as SEQuential or RELative files. By LOADing or SAVEing data, you may realize the full increase in speed that the Lt. Kernal offers over the 1541 Commodore disk drive.

Even if you can't save data as PRoGram type files, the Lt. Kernal DOS provides extremely fast access to records of data in RELative files, and you may create indexes to individual records within RELative files using the KEY file structure to further speed access. In every instance possible, the use of RELative files instead of SEQuential files will ensure that your application runs as F A S T as possible. The Lt. Kernal is faster than ANY other Commodore compatible disk drive on the market today. Make best use of its speed in your new programs.

Disk Partitioning

As you receive your new Lt. Kernal system, the disk will be set up with only four logical drives (LU 0, 1, 2, and 10). The Lt. Kernal DOS ALWAYS resides on LU 10. The rest of the possible eleven logical drives (LU 1 - LU 9) will not be configured.

When you require more space and more logical drives for your applications, we suggest that you initially CONFIGure the rest of the hard disk into equal sized partitions until you get a good feel for the space required by your applications.

You may subdivide the space beyond logical unit 10 (which always remains fixed in size) into as many as ten logical drives, or as few as one logical drive. The CONFIG processor is the method by which you may accomplish this. Study the CONFIG section of this manual carefully and make sure you understand it all before you attempt your first CONFIG. Be sure to RECORD AND SAVE your logical unit parameters on paper: You'll need those parameters in order to preserve files on logical units 0-9 if you ever do a SYSGEN to upgrade to a newer DOS.

KEY FILE usage

This is one of the most complex subjects dealt with in this manual. You do not have to be an expert computer programmer to use the Lt. Kernal KEY files system: If you aren't an expert, though, you are going to need to absorb a lot of new information in order to make full use of this wonderfully powerful feature of the Lt. Kernal DOS.

We will supply you with examples of KEY file use, but it would require another manual the size of this one to explore all the possible uses for KEY files. If you wish to use KEY files on the Lt. Kernal to their fullest potential, study one of the several fine DATABASE MANAGER software packages available for Commodore computers, then return to this section.

The KEY file system on the Lt. Kernal makes available from BASIC a complete KEYED INDEXED-RANDOM ACCESS METHOD for accessing data.

YOU WILL BE ABLE TO BUILD VERY COMPACT AND EXTREMELY POWERFUL CUSTOM DATABASE MANAGEMENT PROGRAMS in BASIC using this file access system.

The Lt. Kernal KEY file system consists of two utility commands and one system call (SYS) with seven modes of operation which can be used to very rapidly search for and find specific data based on varied and complex search criteria.

Originally the KEY file system was written for use in conjunction with RELative files. Its use is not restricted, though. You can use KEY files in any application where you need to associate TEXT STRINGS with specific numeric values.

In the very simplest of terms, the KEY file system allows you to supply a text string and associate a numeric value directly with that string. When you later SEARCH for that string in the file, the SEARCH command will return to you the value you previously associated with the string.

That may sound very simple, indeed, but the applications to which that can be put are amazing. Additionally, the Lt. Kernal KEY file operations are VERY fast. Some remarkable database management programs can be written in just a few lines of BASIC, and they operate at nearly the speed of machine language programs.

Let's **define** a few terms, then go on to actual uses of KEY files.

KEY - is a literal string or string variable containing from one to thirty characters. The KEY is the basic 'element' of KEY files. Keys are the strings for which you will SEARCH in your applications. Depen-

ding upon the methods you use, a KEY may end up being used to refer to specific data, or may be used to point to yet another list of KEYS.

DIRECTORY - When used in the context of KEY files, a directory is a list of UNIQUE keys within a KEY file. A KEY file may contain up to five DIRECTORIES of keys. Keys within a single directory must be UNIQUE.

Record Number - is the NUMERIC value in the range 0-65535 directly associated with each KEY within a DIRECTORY. This is the number returned to you when a SEARCH for a particular KEY is successful. It is supplied BY you when INSERTing a new KEY in a DIRECTORY, or when DELETEing a KEY.

Recl and **Rech** - are the low-byte and high-byte representation of the Record Number.

```
Rech = INT(Record Number / 256)
Recl = Record Number-(Rech * 256)
```

INSERT - means to place a new, unique KEY in a DIRECTORY.

DELETE - means to remove a KEY from a DIRECTORY.

SEARCH - means to attempt to find a particular KEY within a DIRECTORY.

lfn - refers to the logical file number of a KEY file already OPENed on the hard disk.

Status - is a single precision value returned as a result of the various KEY file commands and which reflects the success or failure of the command.

> For really advanced programmers, only:
>
> The structure of a KEY file is a B-tree with unidirectional links between each of three search levels: coarse, medium, and fine. The coarse and medium levels are NOT accessible via the KEY commands: only the results of the operation after passing to the fine level are passed back to your application. You can, however, create your own coarse-to-fine levels of search by utilizing multiple DIRECTORIES within KEY files.

A KEY file example

First, let's set up a purely literal example of how KEY files work. After that, we'll attack the actual commands, and give programming examples. If you are an advanced programmer, and are already familiar with keyed indexed-random file structures, you can skip this section, and refer to the programming examples now.

We'll use a simple catalog/cross-reference as an example.

Assume you have just three items to catalog: a camera, a dishwasher, and your pet dog.

First, in order to catalog these items, you need to decide what features about them are important enough that you can remember to refer to them by those characteristics in the future. Here's a list of unique features you decide to use, and the KEYS you wish to create to describe those features.

item	characteristic	key
camera	Uses film	film
	takes pictures	picture
	very small	small
dishwasher	holds dishes	dish
	washes them	wash
	very large	large
dog	eats	eat
	makes a mess of the yard	mess
	medium size	medium

The 'keys' we decided upon above are all single words, because they're easy to remember. They don't have to be, though. The whole statment ''makes a mess of the yard'' could be treated as a KEY also: any text string up to 30 characters long can be a KEY.

Look at the keys above. The longest is seven (7) characters in length. That becomes the 'key length', since all KEYs in a DIRECTORY must be the same length.

So what do we do with the keys which are shorter than seven characters? We 'pad' them with some known character that makes them all seven characters long. As an example, let's use spaces (noted by the ' ' character) to pad our keys. Padded, our keys look like this:

item	characteristic	key
camera	Uses film	film' '
	takes pictures	picture
	very small	small' '
dishwasher	holds dishes	dish' '
	washes them	wash' '
	very large	large' '
dog	eats	eat' '
	makes a mess of the yard	mess' '
	medium size	medium' '

Now, we'll leave it up to your imagination to create a lengthy and VERY complete description of each item above (no cheating, now; describe the camera down to the last screw!), and save your descriptions in three boxes (disk files). Place the description of the camera in BOX #1, the dishwasher description in BOX #2, and a loving description of your pet (including the cost of sod replacement last year) in BOX #3.

We're ready now to create a KEY file. We do it here with another box: a box labeled ITEMKEYS. We will INSERT keys into that box. We'll need nine slips of paper to handle the nine 'keys' we defined above.

On the left of each slip write the KEY (including any spaces padding it to 7 characters), and on the right of the slip note the RECORD NUMBER (the number of the item) to which the KEY applies (i.e. eats applies to item #3, the dog). As soon as you complete the first slip, drop it into the box. You've just INSERTed your first key.

One of the intentional constraints imposed on you when using KEY files is that no two identical KEYS may be present in the same DIRECTORY: every key must be unique. From now on, we're going to have to carefully check every KEY already in the ITEMKEYS box, to make sure we don't duplicate one, before we may INSERT another.

So write the next KEY slip, take ALL the slips already in the box out, check them to make sure none EXACTLY MATCH the one you're about to INSERT, and if the new key is unique, drop it and all the other older slips back into the box. Repeat that boring cycle until all nine keys are INSERTED.

Sound like a chore? It is, but in the Lt. Kernal KEY file system, that check is made for you, instantaneously, every time you attempt to INSERT a key in a KEY file directory.

Now that you've built a KEY file FORGET EVERYTHING YOU EVER KNEW ABOUT DOGS, DISHWASHERS, and CAMERAS. Remember only this —

You know you've got three boxes describing things, and one box with characteristics of things in it. Besides, you're not so forgetful that you don't remember what a mess, or a picture, or a picture of a mess is.

Hmmm... 'picture' - a seven letter word. Let's look in the characteristics box and see if there are any words spelled EXACTLY like that (exact-match SEARCH). Yep, and it says to look in box #1.

Now you have recovered your memory about cameras! So let's try 'mess', a four letter word (indeed). Nope? No, but there was one close: only it was seven characters long, padded with spaces. But 'mess' and 'mess ' MEAN the same thing, don't they? Sure, so

let's find the first KEY in that box higher alphabetically than 'MESS' and see where it takes us (greater-than SEARCH).

Well, we came up with 'mess ', and it pointed to box #3. By gosh, now that your memory returns about your pet, you remember the yard!

Now you're getting the feel of it. All the characteristics in that box are seven letters long. You remember the word 'dishes' and remember to pad it out to seven characters, then go searching for it. No 'dishes' slip is in there (exact-match failed). You do a 'greater-than' search, and come up with 'eat '. Hey! that sounds good - but when you look in the box it points to, you get the dog, again.

Next you look for the first slip LOWER alphabetically than 'dishes '(less than search), and this time you come up with 'dish ', pointing to box #2. That box describes the dishwasher, and you've completely recovered your memory.

That's a pretty simple example, sure, but imagine it with hundreds of items each with THOUSANDS of characteristics describing them. It would take quite a while to search all those boxes, wouldn't it?

That's where the KEY file comes in. It takes only a fraction of a second to search through any list of keys (DIRECTORY) using the Lt. Kernal KEY commands. You can single out a box (data record pointed to by a RECORD NUMBER) almost instantaneously, no matter how many keys you have to search.

Already, you're probably seeing potential applications for this technique —

You could build a list of all your friends and business acquaintances, and by building several DIRECTORIES of KEYS (a single KEY file will hold five DIRECTORIES), you could cross-reference them by hobby, birth month, city, state, and marital status—

and that is an almost trivial example!

Enough with rudiments. We know you want to get on to using KEY files, so some rules for use and the command descriptions follow.

As we said before, there are two utility programs, and one SYS with seven modes comprising the KEY files commands. The utilities are Direct-Mode commands, the SYS's are designed to be used primarily in the Run-Mode.

BUILDINDEX is the command which allows you to create a new KEY file, and establish the KEY characteristics for it. BUILDINDEX is self-documenting. There are just a few **constraints** to creating a KEY file.

- A KEY must be no longer than thirty characters long: All keys within a single DIRECTORY are built to the same length.
- The file may contain no more than five (5) DIRECTORIES of KEYS.
- ALL DIRECTORIES within a given KEY file will contain the same number of KEYS (max. 65535).

This needs explaining. Other than the fact that up to five KEY DIRECTORIES may exist within a single KEY file, there is no FUNCTIONAL relationship among them. They may contain KEYS which are entirely unrelated.

However, because of the way in which KEY files are built, all DIRECTORIES within a single KEY file will be built for the SAME NUMBER OF KEYS ... That's important, because the longest DIRECTORY you can create within a key file is directly dependent on the length of the LONGEST KEY defined for that file.

A brief example will suffice to illustrate. If you built a KEY file with FIVE DIRECTORIES, each with KEYS 13 characters long, you could request a 'number of keys' as large as 65535. That's the maximum number of KEYS you are ever allowed to build in any DIRECTORY.

At a key length of 14 characters, the number of KEYS PER DIRECTORY begins to diminish. If you defined the key length as thirty (30) characters, you could have no DIRECTORY longer than 6750 KEYS.

Now here's the rub. If four of the five defined DIRECTORIES within a KEY file had key lengths of only 13 characters (allowing 65535 keys/directory), but the fifth had a key length of thirty, then the maximum number of KEYS PER DIRECTORY will still be diminished to 6750 for ALL DIRECTORIES within that one file. That doesn't mean that any space is wasted in the file. It's just a functional limitation of BUILDINDEX.

Here's a table of maximum DIRECTORY lengths for different lengths of KEYS.

KEY length	Max KEYS DIRECTORY	KEY length	Max KEYS DIRECTORY
1-13	65535	22	18522
14	59582	23	16000
15	48778	24	13718
16	43904	25	11664
17	35152	26	11664
18	31250	27	9826
19	27648	28	8192
20	24334	29	8192
21	21296	30	6750

The table is computed from the formula —

$$Nkeys = (INT(507 / (L+))\uparrow 3) * 2$$

where L is the length in characters of the longest key, and Nkeys(max) = 65535.

Another constraint on KEY files.

- Because of the manner in which KEYS are INSERTed in a B-tree, a DIRECTORY should be constructed for about 20% MORE KEYS than you expect to use.

  This will GREATLY affect the speed of insertions when you have many similar KEYS in a single DIRECTORY. Be conservative in your estimates of needed keys: Twenty megabytes will hold a lot of data.

The last limitation of KEY files —

- The space required to build a KEY file must be available in CONTIGUOUS blocks on the LU on which it is to be built.

  If there is enough space, but not sufficient CONTIGUOUS space available on an LU for the KEY files you wish to build, the method to get it all contiguous is:

  AUTOCOPY all files to another LU.
  ACTIVATE the LU you wish to clean up.
  AUTOCOPY all the files back.

This performs a quick equivalent of the 1541 VALIDATE command.

Now a brief discussion of the other utility for KEY files. DI, then on to the Run-mode commands.

DI (dump index) is provided as a convenience to KEY-file programmers. The same function could be written in BASIC, as we will show later.

DI simply lists all active keys in a KEY file. DI will prompt you for the name of the KEY file you wish to dump.

Key-File Run-Mode Commands

The general form of the KEY commands is:

SYS 64628:mode,lfn,directory,Stringvar,recl,rech,status

The parameters list may consist of variables or literal values, or a mixture of both. The colon following the SYS address is required in order to maintain C-64/C-128 compatibility.

The mode parameter defines what KEY command will be performed. The modes are:

mode	function
0	BUILD new KEY file
1	INSERT key
2	DELETE key
3	exact-match SEARCH for key
4	greater-than SEARCH for key
5	less-than SEARCH for key
6	not used
7	SHUFFLE directory

general: SYS 64628:mode.lfn.directory.Stringvar.recl.rech.status

The other parameters are. again:

lfn - refers to the logical file number of a KEY file already OPENed on the hard disk.

directory - The number (1-5) of the directory within the file which you wish to access.

Stringvar - A literal or variable string containing the KEY with which you wish to operate. For SEARCH functions, this key should be padded to the KEY-LENGTH of the directory chosen (usually with spaces or nulls). INSERT will pad the KEY with nulls if you supply a Stringvar shorter than the KEY length for that directory.

Recl and **Rech** - are the low-byte and high-byte representation of the Record Number.

Rech = INT(Record Number / 256)
Recl = Record Number-(Rech * 256)

The Record Number is the NUMERIC value in the range 0-65535 directly associated with each KEY within a DIRECTORY. This is the number returned to you when a SEARCH for a particular KEY is successful. It is supplied BY you when INSERTing a new KEY in a DIRECTORY, or when DELETEing a KEY.

Status - is a single precision value returned as a result of the various KEY file commands and which reflects the success or failure of the command.

```
DESTBL   .byte MODE      ; same MODE values as BASIC version
         .byte LFN       ; logical file-number of OPENed KEY file
         .byte DIR#      ; DIRECTORY # of KEY file to acess
                         ; (see note below)
         .byte KEYLEN    ; length in bytes of KEY addressed next
         .word KEYADR    ; memory address of KEY string
                         ; (low/high order)
         .word REC#      ; RECORD NUMBER associated with KEY
                         ; (low/high order)
```

Just as in the BASIC versions of KEY file access, any portion of the Descriptor Table considered a 'return' value from the KEY operation will be filled in by the Lt. Kernal during Key operation.

The carry flag is cleared if there is no error, and set if an error occurred. If any error occurred, then the accumulator will contain an error value identical to the BASIC versions of the operation.

All registers are USED for communication to KEY operations, and you may consider that ALL REGISTERS ARE MODIFIED by KEY operations.

In C-64 operation, the Decsriptor Table MUST reside in BANK0, but the KEY string may reside in either bank.

By OR'ing the DIR# value with $80, the Lt. Kernal is directed to seek the KEY in ram-BANK1. A DIR# value less than $80 directs the KEY to be found in BANK0.

This is the general form of machine language KEY file access. The MODE value and expected returns will change from mode to mode, but will match those expected in the BASIC versions of KEY file access.

The **INSERT** key command - mode=1

SYS 64628:1,lfn,directory,Stringvar,recl,rech,status

This command inserts a new, unique key into the selected directory. In addition to the lfn # and directory #, you must supply:

The EXACT key to insert (length = key-len)

The record number in recl and rech to be associated with the KEY.

The variable status returns from INSERT are:

0	INSERT successful
2	invalid key length (key supplied is longer than key length for directory)
4	DIRECTORY full - insert cannot occur
5	KEY already exists, cannot duplicate keys

If directory full status (4) occurs, a SHUFFLE may fix the condition: more on that later.

If status 5 (key exists) is returned, you must either change the key you supply, or DELETE the key from the directory, then re-insert it.

The **DELETE** key command -mode=2

SYS 64628:2,lfn,directory,Stringvar,recl,rech,status

This command attempts to delete a key from the selected directory. In addition to the lfn # and directory #, you must supply:

The EXACT key to delete (length = key-len)
The EXACT record-number in recl and rech already associated with the KEY.

The variable status returns from DELETE are:

0	DELETE successful
2	invalid key length (key supplied is longer than key length for directory)
5	KEY not found, or record number supplied does not match record number already associated with KEY found.

The **SHUFFLE** directory command - mode=7

SYS 64628:7,lfn,directory,Stringvar,recl,rech,status

Because of the manner in which B-tree key insertions occur, sometimes there will not appear to be an available slot for a new key, even when sufficient space exists in a directory. SHUFFLE attempts to re-order the KEYS in an existing directory to make slots available.

If an attempt to INSERT a key returns a status of 4 (directory full), you should SHUFFLE the directory, and attempt the INSERT again. If, after TWO ATTEMPTS to SHUFFLE and INSERT, the directory full status is still returned, you may actually consider the directory to be full.

You must supply:

The logical file number of the KEY file already OPEN.
The directory number of the directory you wish to SHUFFLE.

There are only two variable status returns from SHUFFLE:

0	SHUFFLE completed (NOT an indication that slots were freed up)
2	No keys in directory (SHUFFLE not done)

The SEARCH commands

The SEARCH key command - mode=3, 4, or 5

exact-match SYS 64628:3,lfn,directory,Stringvar,recl,rech,status
greater-than SYS 64628:4,lfn,directory,Stringvar,recl,rech,status
less-than SYS 64628:5,lfn,directory,Stringvar,recl,rech,status

The SEARCH commands allow you to very rapidly search a directory for a particular key, based on variable criteria.

In addition to the logical file and directory numbers, you must supply:

The SEARCH mode
and
The KEY for which to search (length <=key-len)
for exact-match, length = key-len

The variable status returns from SEARCH are:

0	KEY satisfying criteria found
2	invalid key length (key supplied is longer than key length for directory)
5	KEY satisfying search criteria NOT found

SEARCH ALWAYS RETURNS THE KEY FOUND WHICH SATISFIES THE SEARCH CRITERIA IN Stringvar, and the record number associated with that key in recl and rech.

If no satisfactory key is found, Stringvar is unmodified, and recl and rech are meaningless.

An exact match is satisfied when the KEY (including pad characters) exactly matches the Stringvar in length, and character-by-character.

A greater-than match is satisfied when the first KEY logically greater than the supplied Stringvar is found.

A less-than match is satisfied when the first KEY which is logically less than the Stringvar supplied is found.

As an example of how to use KEY files on the Lt. Kernal, we're going to build a dictionary application in BASIC. The dictionary will hold up to 6750 words of up to 30 characters in length.

Each word will have a text definition keyed to it. The dictionary will permit us to have a total of 65535 lines of 40 characters assigned to definitions of words. Any single word may use up to 20 forty-character lines for its definition.

When searching for a word in the dictionary, if the word is not found, the dictionary will display the words alphabetically surrounding it, and give you an opportunity to display the definition of one of them, or to enter the new word into the dictionary.

If all that sounds like it will take a pretty large, slow BASIC program to accomplish, you're in for a pleasant surprise! Not only does it take only 39 lines of program to build such a dictionary, but it takes less than one second to find any word already in it, or to determine that the word is not there!

Because of the length of the example, the actual program appears in APPENDIX i.

The CONFIG processor

CONFIG is a powerful (and potentially dangerous!) utility program which allows you to set many system defaults. DO NOT use CONFIG until you have become comfortable with the LU and User structure of the Lt. Kernal. You may change several power-on characteristics of the system including screen, border, and character colors, the hard disk device number, the logical unit (logical drive) and user (subdirectory) onto which the system first 'logs' or establishes operation, and the sizes of the various logical drives which the system may emulate.

For the most part, CONFIG is self-documenting, and will prompt you through the changes it can accomplish. Once you exit CONFIG properly, any characteristics of the system you have set will remain that way EVERY TIME you turn on the power to your system.

Only a few points of CONFIG need explaining. The first is the 'beep flag'. When the beep flag is set to 1 (one) the beeper is enabled. After that, any time you issue a CHR$(7) through a BASIC print-statement or a byte of value 7 through the KERNAL's BSOUT routine, the Lt. Kernal will issue an audible 'beep' through the monitor's speaker. This feature is provided simply as a convenience to programmers so that you do not have to maintain SID drivers in your C-64 programs just to issue audible prompts.

When the beep flag is set to 0, CHR$(7) has no effect in the C-64 mode (the C-128 does its own beeps). This character is NOT reserved in the Commodore-64 character set for any special purpose and is not a printable character. Usually, it is desirable to keep the beep flag enabled because all Lt. Kernal DOS error messages use the beep, when enabled, to alert you that an error has occurred.

The second point of CONFIG explanation concerns the default logical unit number. This point is only critical inasmuch as it affects the AUTOSTART and CP/M features of the Lt. Kernal DOS.

When the Lt. Kernal first powers up, or after a hardware reset, it FIRST switches to the logical drive defined in CONFIG and THEN searches for a file by the name of AUTOSTART. If you wish to use the AUTOSTART feature of the Lt. Kernal, make sure that the program you have named 'AUTOSTART' resides on the same logical drive that you have named as the default in CONFIG.

The CP/M DEFAULT LU is that 'CP/M-type' LU which is presently accessible via the GOCPM command.

You may define several of these LU's on your system, but only ONE is accessible to GOCPM at a time.

The third point concerns the hard drive's default hardware device number. The only special situation here is when the hard disk and a floppy drive BOTH carry the same device number.

This is a special and very desirable situation, contrary to first appearances. In this situation, any COMMODORE-SYNTAX LOAD request directed to the system either through BASIC or the KERNAL vectors will be directed first to the hard disk. If the file requested is not found on the hard disk, the request will be AUTOMATICALLY referred to the floppy disk (neat, huh?). Requests to load a file via the Lt. Kernal's abbreviated 'L' load command, or by direct invocation will not be referred to the floppy disk. The automatic referral of LOAD commands only applies when both the hard disk and the floppy disk carry the SAME hardware device number (usually 8).

If this 'AUTOACCESS' feature interferes with a particular application, you may turn it off via CONFIG.

The last point of CONFIG to discuss is the most complex, and to a degree, somewhat hazardous. The point at issue is the CONFIGuration of logical unit (logical drive) parameters.

Briefly, let's discuss how the Lt. Kernal (and most hard disk systems) allocate space on the disk.

In a floppy disk environment, space on a diskette is usually parcelled out in units known as SECTORS or BLOCKS. A sector is the smallest unit of data which a floppy disk may read or write at one time.

A hard disk also may read or write as little as one SECTOR of data at one time: Hard disks, however, have HUGE numbers of sectors available. In order to make those numbers more managable, most hard disk systems allow you to allocate (set aside for future use) sectors in groups. The largest number of sectors which may be accessed on a hard disk without requiring any mechanical repositioning of the 'heads' (the part which physically reads and writes data) is known as a CYLINDER.

Thus, the CYLINDER becomes the smallest unit of storage which may be allocated on the Lt. Kernal. Remember that allocating space doesn't use it up; It only sets that space aside for use later. When programs or files use that space, they use it on a sector-by-sector basis.

On the Lt. Kernal system, a CYLINDER contains 68 SECTORS, each of which is twice the size of a 1541 sector (or block). The Lt. Kernal stores data in 512-byte sectors, and the 1541 in 256-byte sectors. Since a cylinder contains sixty-eight sectors, that is the minimum increment of space you may allocate to a logical drive (LU).

The Lt. Kernal DOS resides on LU 10, and that logical drive is always fixed in size at 30 cylinders; You may not change its size. All other LU's (0-9) sizes are user-definable via CONFIG.

An LU or logical drive must contain at least enough space for the 'BAM' and 'INDEX' which constitute the directory storage area for that LU. Since any Lt. Kernal LU may hold up to 4000 directory entries, the MINIMUM amount of space you must allocate for a new LU is 16 cylinders, or 1088 hard disk blocks. After the creation of the BAM and INDEX, which collectively use 272 blocks, that leaves 816 blocks available on the LU for your files. That works out to about as much space as 2-½ full 1541 diskettes.

That's the minimum space you may allocate to a new logical drive. You can allocate more space in CYLINDER increments, up to a maximum 911 cylinders. Since that's more than the number of cylinders available on a 20 megabyte drive, there is no practical limitation to the size of a single LU, other than the remaining un-allocated capacity of your drive.

CONFIG will not allow you to allocate less than 16 cylinders to a new LU, nor will it permit you to allocate more space than that which is available. Those restrictions are automatic - you don't need to do any math to use CONFIG.

When CONFIG is caused to enter the SET LU PARAMETERS mode, it will respond with the CURRENT logical unit boundaries in table form. You will be given the current total cylinders available on the hard disk, along with the current cylinder boundaries of any already defined logical units. CONFIG will allow you to establish new cylinder boundaries for any logical unit with the following constraints—

No logical unit may ever overlap an existing logical unit. CONFIG will prevent you from declaring any LU boundary which overlaps an existing LU boundary. If you wish to expand a logical unit to a point which will overlap an existing logical unit, you must first shrink the conflicting LU in the appropriate direction. To shrink an LU, you must first delete the LU's entry, then re-declare its boundaries.

IF YOU MOVE THE LOWER BOUNDARY OF AN ALREADY USED LU, ALL FILES ON THE LU WILL BE LOST. If you shrink an LU by moving its upper boundary, SOME files may be lost, depending on how full (and how old) the LU is, and a NEW LU created just above it may also be endangered.

The only safe way to re-define an LU's size is to AUTOCOPY all of its files to another LU, redefine its size, ACTIVATE the LU, then AUTOCOPY the files back.

The total cylinders ascribed to all logical units defined in CONFIG may not exceed the total cylinders available on the hard disk. CONFIG will prohibit this.

Now the one point of danger. CONFIG REQUIRES that YOU record the logical unit parameters you have established. You may direct your LU parameters to the printer or the screen, but once you have selected a destination for the parameters, you MUST record and save them for future reference!

RECORD AND SAVE your LU parameters list every time you change LU parameters —

THIS IS **YOUR** RESPONSIBILITY!!!

When you establish a NEW LU via CONFIG, you will be given the option to create a 'DOS image file'. The DOS image file is an optional feature of the Lt. Kernal which tremendously enhances the speed of file access on LU's physically distant from the DOS LU.

When the DOS is running, it must continually refer back to the DOS LU to bring in DOS overlays (the term we use to describe the various program modules of the DOS). Every time an overlay is required, the disk drive must physically reposition its 'heads' from the LU you are using to the DOS LU (lu 10), and then back again.

The further (in cylinders) your working LU is from the DOS, the longer that process takes. In order to speed it up, we offer you the ability to build a COPY of the run time modules of the DOS on each LU you create. The pay off is a dramatic increase in speed of Lt. Kernal file access. The cost is space.

A DOS image file uses 222 hard disk blocks. In most instances, you won't even miss this small (?!) amount of space. If your system is full right up to the gills, though, you may opt not to create DOS image files.

You may FORCE an update of the Lt. Kernal DOS image files on all LUs at any time. This is provided so that you may 'refresh' the DOS images if you ever suspect one of them may have been corrupted (by a power failure during a 'write' operation, as an example).

Simply issue the CHECKSUM command, and once it has been completed, reset your system. As soon as the system boots back up, all Lt. Kernal DOS image files will be updated automatically.

Once you have defined an LU, you may assign a 'type' to it. The types are *regular* and *CP/M*. Since the ACTIVATE process operates differently on the two types, you must assign a 'type' to an LU before you ACTIVATE it.

Once you have defined a new LU or changed the size of an old one, and exited CONFIG, you must run ACTIVATE which always totally erases all files except DOS image files, and creates a new BAM and INDEX on the LU. It is not safe to use an LU which has not been ACTIVATEd, since an old BAM and INDEX may still exist which do not properly reflect the new physical size of the LU. THIS COULD EVEN ENDANGER AN LU PHYSICALLY ADJACENT TO THE 'BAD' LU. The only exception to this is when the EXACT original parameters of an LU are re-established after a SYSGEN, as discussed below.

One final note about CONFIG concerning doing a SYSGEN. The files on LUs 0-9 will remain intact. Logical unit 10 is ALWAYS completely overwritten by a SYSGEN.

We do not prohibit your using logical unit 10. In fact, the utility programs you wish to be accessible from ALL other logical units should be placed on LU 10. Remember, however, that LU 10 is COMPLETELY ERASED AND REPLACED during a SYSGEN. Any programs you wish to preserve on LU 10 should be copied to another LU or to floppy disk before you do the SYSGEN.

THE SYSGEN UTILITY

Your Lt. Kernal package contains a SYSGEN disk which has the same DOS on it that is *already installed* on your Lt. Kernal on LU10. The SYSGEN disk is provided as a back-up for the DOS already on the Lt. Kernal. **Do NOT do a SYSGEN unless you have corrupted your DOS on the Lt. Kernal.**

To receive updates to the SYSGEN disk, you must have registered your Lt. Kernal with Xetec, Inc. and Fiscal Information. Do not fail to register.

Use the following steps to perform a successful SYSGEN on your Lt. Kernal. Do not omit any step.

1. Make sure you have a 1541 or 1571 floppy drive (set as device #8) connected to your computer.

2. If you have any of your own files or programs on LU10, use AUTOCOPY to temporarily move them to any other LU.

3. If your system is operational, type D 9 and carriage return and skip to step 11 below.

4. Turn you entire system off.

5. Make sure the power switch to your Lt. Kernal is off.

6. Unplug the 25 pin cable from the rear of the Host Adaptor. Leave the Host Adaptor plugged into the expansion port of your computer.

7. Leave the Lt. Kernal off, and power up the rest of your system. After about one minute, your monitor should show a normal Commodore sign-on screen.

8. If you are using a C-128 computer, you must now GO 64.

9. After the C64 sign-on screen appears, connect the 25 pin cable back into the rear of your Host Adaptor.

10. Power up your Lt. Kernal and allow about 30 seconds for it to come up to speed.

11. If you are using a 1571 drive, type the following command:

    OPEN15,8,15,"U0>M0":CLOSE15 then hit RETURN

12. Place the SYSGEN disk in your floppy drive and type:

    LOAD"*",8,1 then hit RETURN

13. It will take 10-15 minutes to complete the SYSGEN process. Progress messages (please wait...check summing the DOS) will show up periodically.

14. After completion, the SYSGEN process will display the following message:

    "SYSTEM INITIALIZATION COMPLETE"
    "NOW DO A FULL SYSTEM RESET"
    "THANK YOU"

15. To RESET your system, turn off power to the computer for a short period of time, and then turn it back on.

16. After running SYSGEN, check your LU CONFIGuration parameters. A normal SYSGEN will NOT change any of your CONFIGured defaults nor destroy any files except those on LU10.

17. AUTOCOPY any files you moved in step 2 back to LU10.

18. This completes the SYSGEN process.

X

ADDENDA/ERRATA and BUG FIXES AND PATCHES

This section is reserved for addenda describing discovered software problems, the fixes, and DOS versions reflecting the fixes. Please insert addenda supplied with any DOS updates in this section.

XI

TROUBLE-SHOOTING

We hope you never have to trouble-shoot your Lt. Kernal hard disk system. but shipping, even in the best of containers, can be hard on mechanical devices. Your Lt. Kernal system was fully tested and burned-in at the factory before it was shipped. It should arrive in perfect working order.

General Procedures

Trouble shooting any system should follow a strictly ordered procedure. Only after verifying that each prerequisite feature is in order should the next feature be checked. The following guide will step you from the most elemental causes of system failure through to some more complex possible causes. Usually the cause of failure WILL be one of the simpler ones.

Regardless of how ridiculously simple an item might seem, check each item carefully and in the order indicated. Doing so will lead you to a quick resolution of the problem. You may fail to find the problem if you skip steps, or be led down a long and fruitless path making you retrace many previously checked points.

TROUBLE—SHOOTING GUIDE

I. CABLING
- A. DATA CABLE and ADAPTOR BOARD
  1. Check cable connection at Lt. Kernal Host Adaptor and hard disk enclosure
     a. Check for proper pin 1 alignment
     b. Check for full and even insertion
  2. Check Lt. Kernal Host Adaptor insertion into computer
     a. Check that Host Adaptor nose is properly aligned with expansion port slot inside computer
     b. Check that Host Adaptor is fully and evenly inserted into computer expansion slot
     c. Check for proper connections of jumpers, HIRAM and CAEC cables, and the C-128 cable.
- B. POWER CONNECTIONS
  1. Check cord entry to Lt. Kernal hard disk enclosure
     a. Check that power cord is fully inserted
     b. Check that power cord is plugged into a functional outlet for the correct voltage and frequency of power
     c. Check that hard disk power switch is ON at appropriate point in power-up sequence

II. FUNCTIONAL TESTS

FIRST — Check your computer without the Lt. Kernal Host Adaptor plugged in. If it performs normally, then reinstall the Host Adaptor and proceed.

A. SYMPTOM — SCREEN remains blank indefinitely, or remains blank for about 1 minute then the normal Commodore sign-on messages appear WITHOUT the Lt. Kernal messages.

1. A blank screen is normal on the C-128 if the 2 mhz fast mode is selected in the C-64 mode. Remedy by returning to the 1 mhz mode.

1. Check the internal fan to see if the Lt. Kernal is running — IF NOT:
   a. Check that the Lt. Kernal power switch is ON
   b. Check that the POWER CORD is firmly seated in the receptacle on the rear of the Lt. Kernal hard disk enclosure
   c. Check that the POWER CORD is firmly seated in the outlet into which it is plugged
   d. Check with another device (such as a lamp) that the power outlet used for the Lt. Kernal IS ACTUALLY supplying power
   e. Check the Lt. Kernal's fuse
      - remove the power cord from the Lt. Kernal's power receptacle
      - gently unscrew the cap of the fuse holder until the cap and fuse together are removed
      - carefully pull the fuse and the cap apart
      - inspect the fuse. If necessary, check the fuse with a continuity checker. If it is blown, replace it ONLY with the exact original type.

2. Check the ribbon cable
   a. Check for full, FIRM, and even insertion of the cable into its connections at both ends
   b. Check that pin 1 of the cable corresponds to pin 1 of the connectors on both the Host Adaptor and the drive enclosure
   c. Check that the cable has not been torn or frayed at any point especially CLOSE to its connectors
   d. ONLY IF THE POINTS ABOVE CHECKED OK, then remove the ribbon cable from its connections and check the pins inside each connector to see that no pins have been bent or broken by improper insertion of the cable
      — if any pins are bent, GENTLY staighten them with a small, flat tool like a cosmetic 'orange stick' or a very small screw driver

— if any pins are broken, you may have to return your Lt. Kernal to Xetec, Inc. for repair. Call our technical support for service.

3. Check computer's power supply
   a. Check the red POWER indicator on your computer. If it seems dim or is not on at all, suspect either a bad computer power supply, or a defective Lt. Kernal Host Adaptor
   b. Test you computer / Lt. Kernal combination with another known-good computer power supply (preferrably one of the NEWER ones)
   c. If the above checks do not resolve the problem, check your computer and power supply with any large cartridge (i.e. Commodore's CP/M cartridge)

B. SYMPTOM — Your Lt. Kernal works properly for a time (seconds to hours), then begins to behave erratically, or screen colors begin to change for no apparent reason, or the system ceases to function entirely

THIS IS THE MOST COMMONLY REPORTED PROBLEM WITH THE LT. KERNAL SYSTEM (and all systems which use large, active cartridge modules)

The Lt. Kernal Host Adaptor has been designed very conservatively. It draws less than two-thirds the amount of power from the computer that Commodore specifies it may. Yet MANY power supplies do not meet even Commodore's own specifications for supplying extra power to accessory devices.

Try a different power supply. The newer 'potted' version seems a little better at supplying extra power than the original design. The IDEAL solution is to obtain one of the excellent third-party designed supplies which GUARANTEE to meet or exceed Commodore's specifications.

C. SYMPTOM — The Lt. Kernal does a 'double boot'. The screen briefly displays the sign-on message at the top, then it shrinks horizontally and returns to a NORMAL sign-on screen WITHOUT the Lt. Kernal messages.

Your DOS software resident on the hard disk HAS BEEN DAMAGED. The DOS loader has sensed the damage and turned itself and the Lt. Kernal Host Adaptor off.

AT THIS TIME YOU HAVE ONLY ONE METHOD OF RECOVERY AVAILABLE.

YOU MUST PERFORM THE 'SYSGEN' PROCESS. SYSGEN will not by itself destroy any of the software you have on logical drives 0—9, but will destroy all files on logical drive 10.

Turn to the section 'THE SYSGEN UTILITY' if your Lt. Kernal does a 'double-boot'.

D. SYMPTOM — The system displays the first couple of lines of sign-on messages, then just 'hangs' and never displays the 'READY' prompt.

Your DOS software has been corrupted, as previously described under 'double-boot'symptoms.

Turn to the section 'THE SYSGEN UTILITY' if your Lt. Kernal displays this symptom.

E. Your hard disk begins to make a high-pitched whistle which continues, but the system appears to behave normally otherwise.

This is an annoying symptom which is present to some degree on almost every brand of small hard disk. We have chosen a brand which is not inclined to 'squeal', but still, it might happen.

The whistle or squealing noise is caused by dust particles contaminating one of the rotating parts of the hard disk. It will not interfere with normal use and the noise will usually subside within minutes (to hours) after it begins. Drives allowed to sit for several hours between uses are most apt to display this symptom. Drives in continuous duty (like BBS systems) almost never make this sound.

F. SYMPTOM — Your hard disk enclosure gets very warm or hot to the touch.

QUICKLY, SHUT THE SYSTEM DOWN!! Your Lt. Kernal is fan-cooled. Normally, the only warm area will be around the ventilation exhaust slots. If the top or sides of the hard disk enclosure begin to get warm, it's a good indication that the ventilation slots in the enclosure have either been blocked, or that the fan has failed (possible due to dust accumulation).

NEVER RUN THE LT. KERNAL WITHOUT
THE FAN IN PROPER WORKING ORDER!

RETURN POLICY

Do not return any Lt. Kernal to Xetec without a RETURN MATERIAL AUTHORIZATION number (RMA#). If a RMA# is not clearly marked on the outside of the shipping carton, the product will be refused.

Call (913) 827-0685 to obtain a RMA#.

Prepare the drive for shipment using the SHIP command descibed on page 8-47.

Use the original carton or equivalent, insure shipment or assume the risk of loss or damage in transit, and pre-pay the shipping charges.

Send a letter with the Lt. Kernal detailing the specific problem. This will speed up the return of your unit.

If the unit is under warranty, include a copy of the proof of purchase. If this is not included, you will be billed for the repair.
NOTE: The limited warranty will be honored only if the XETEC, Inc. registration card is completed and mailed to Xetec.

REMEMBER, follow these steps:

1. Obtain a RMA #
2. Prepare drive for shipping using "SHIP" command
3. Use the original carton or equivalent.
4. Insure shipment.
5. Pre-pay shipping charges.
6. Include a letter describing the problem
7. Include proof of purchase, if under warranty

XII

DOS SYSTEM UPDATES and ENHANCEMENTS

To qualify for DOS updates, you must have REGISTERED with FISCAL INFORMATION, Inc. by filling out the Fiscal registration card.

Certain enhancements are to follow and will carry charges. Notices will be sent to registered owners only.

The limited warranty will be honored only if Xetec, Inc. registration card is completed and mailed to Xetec.

'BUG' REPORTING

Unfortunately, every system of software will eventually be discovered to contain 'bugs' or errors in programming. We have made every effort to ensure that the Lt. Kernal DOS is bug-free. If you do find a software defect, PLEASE REPORT THE PROBLEM AS SOON AS POSSIBLE to us on a copy of the form which follows. We truly wish to make the Lt. Kernal error free, and will give every formal 'bug' report close and careful consideration.

The bug reporting form appears separately on the next page so that you may photo copy it as necessary.

PLEASE LIMIT your trouble reports to ONE problem per form.You may, however, send more than one bug report per envelope.

LT. KERNAL DOS REPORT FORM

THANK YOU FOR YOUR ASSISTANCE IN IMPROVING THE LT. KERNAL!

Please report only ONE problem per form, however, you may send more than one form per envelope.

Please answer all questions, if possible.

SERIAL NUMBER _____

DOS VERSION/REVISON # _____

Date of problem / / Approx. time of day _____

C-64 or C-128 (circle one) & computer serial # _____

Approx. how long have you owned your Lt. Kernal system? _____

Does the problem occur under identical circumstances WITHOUT the Lt. Kernal Host Adaptor module plugged in? (circle) Y or N

Approx. how long had the system been turned on before the problem was detected? _____

Does the problem appear immediately with a cool system and power supply, or only after a period of warm up?

(circle) COOL WARMED UP

If you circled WARMED UP, how long must the system warm up before the problem FIRST begins to occur? _____

Please describe the problem and all surrounding circumstances in as much detail a spossible. Include NAMES and VERSIONS if other commercial software/hardware products are involved. Use the back of this form if more space is required.

_____

_____

_____

_____

_____

XIII

Installing CP/M™ on the Lt. Kernal

Using CP/M™

CP/M operates EXACTLY the same on the Lt. Kernal as it does on floppy disks, except for the speed of disk access. Depending upon the model of Lt. Kernal system you own, CP/M will run on the hard drive at a speed of 80% to 103% the *speed of RAM disk!* That's FAST!

The Lt. Kernal defaults to the 'L' drive under CP/M and supports all existing floppy types as well as RAM disk. It is invoked simply by typing GOCPM from the C-64 or 128 modes, or by CONFIGuring CP/M as the default mode on power-up.

That's all you need to know to use CP/M on the Lt. Kernal. This is not intended to be a CP/M manual. If you do not already know CP/M, you need to study how to use it before attempting to BUILDCPM on your Lt. Kernal.

Building CP/M™

We cannot distribute CP/M on the Lt. Kernal without infringing on the copyrights of Digital Research, Inc., the authors of the product. However, what we can provide is a mechanism whereby you may copy the CP/M system to the hard disk and enable THAT disk to become the system's default drive.

FIRST OF ALL— because of the way Commodore chose to implement CP/M, the Lt. Kernal CP/M support modules are *version dependent.* There is only ONE version of Commodore's 128 CP/M which will run on the Lt. Kernal. It is the

December 1985 Release which supports RS-232 devices

You can only tell the Release Date by 'Booting' CP/M. This release is the most recent U.S. one of Commodore CP/M at the time of the writing of this manual, and is widely distributed. Again, it is the ONLY version which will work with the Lt. Kernal.

SECOND— you will have to enable the 'I/O-1 page' option on your Lt. Kernal Host Adaptor. CP/M expects RAM disk to be present in the 'I/O-2 page' which is the *other* way the Host Adaptor can be set. If you have not already enabled the 'I/O-1 page' option, do it now before proceeding.

THIRD— if you desire to use RAM disk with your Lt. Kernal CP/M, you MUST build it on the CP/M LU with RAM disk PRESENT. CP/M built without a RAM disk installed will not support one later. It is possible, however, to re-build CP/M on an LU at a later date WITHOUT disturbing any of the files already saved on that LU. CP/M built with RAM disk may be used without it.

Steps for the building of a CP/M LU:

1. You will need to CONFIGure an LU for CP/M use and ACTIVATE it. That entirely erases the LU, and prepares it for the CP/M operating system. You may rebuild the CP/M system on that SAME LU later without ACTIVATEing the LU.
2. Create a CP/M diskette containing CP/M and the following additional files:

   PIP.COM
   SAVE.COM
   ERASE.COM

   The files CCP.COM and CPM+.SYS will have been placed on the diskette already by the COPYSYS command which create it.

   You may build CP/M on the Lt. Kernal using either a 1541 or a 1571 floppy drive. Even though it takes a bit longer on the 1541, it will not affect the final speed of the Lt. Kernal version.
3. From the 128 mode of operation, type BUILDCPM and press return if you have prepared the LU and Host Adaptor properly, you will soon be prompted to insert your CP/M 3.0 diskette. Do so and press return again.
4. You will see the familiar CP/M boot message appear, and soon the system will issue the CP/M 'A' prompt. DO NOTHING ELSE EXCEPT EXACTLY the steps below:

type	SAVE <return>
wait for the 'A' prompt, then type	CCP <return>
wait for the 'A' prompt, then press	CONTROL and C keys down together

5. The SAVE processor will now be invoked, and will prompt you for:

prompt	you enter
FILE—NAME	T.COM <return>
Starting hex Address	4000 <return>
Ending hex Address	40ff <return>

   The floppy disk will churn awhile, then you will be returned to the 'A:' prompt again.

Warning: We are approaching the only unreliable part of the process. Up until now, CP/M has been in *exclusive control* of the computer. Now we have to 'trap' it and place the Lt. Kernal back in control. If the following step fails, you will have to re-boot CP/M and return directly to THIS next step. You DO NOT have to re-SAVE the T.COM file above.

6. Now type T <return>

The message will prompt you to press your I.C.Q.U.B. button located on your Host Adaptor. *Firmly* and *quickly* press and release the button. Within ten seconds you should see the Lt. Kernal hard disk access light come on almost solidly. If the light does not come on within that time, re-boot your system and try this step again.

It works perfectly the first time 9 out of 10 tries, but occasionally CP/M just will not yield control to the Lt. Kernal. When that happens, we find it is MOST likely to work on the next try if you turn power to your 128 off for at least 10 seconds before trying again.

If the Lt. Kernal has gained control of CP/M, you will see the access light stay on solidly for about five seconds and your system will respond with the 'L' (Lt. Kernal) drive prompt.

7. Follow the next few steps EXACTLY:

type	A: <return>
wait for 'A' prompt, then press	CONTROL and C keys together
wait for 'A' prompt, then type	PIP L:=*.*[rv]<return>

All files will be copied from the 'A' drive to the 'L' drive, and they will be verified after copying.

If PIP returns without errors, you have successfully built a CP/M operating system on this LU. You may now PIP any files you choose to the 'L' drive. We suggest that you copy all your utility software to USER 0, and use SET.COM to make all your utilities *SYSTEM* type files. It is also a good idea to use SET.COM to force all utilities to RO (Read Only) so you cannot accidentally erase them.

Operating Speed

The Lt. Kernal cannot speed up the floppy disk, but you will be amazed at the speed of loading and copying files on the Lt. Kernal. You can even further speed up the operation of a 40-column system, if you do not mind a little (very unattractive) screen flickering.

The Lt. Kernal CP/M BIOS looks at the 40/80 column key on the 128 as a signal for which mode to boot in. If you wish to use CP/M in the 40 column mode, be in the 40 column mode when you GOCPM. Likewise for the 80 column mode.

Once the the Lt. Kernal has gone to CP/M, it then looks at the 40/80 key as a *speed switch*. If it is UP (40 column setting), the Lt. Kernal accesses the disk at 1 MHz. If it is in the 80 column position, the Lt. Kernal automatically shifts to 2 MHz during disk operations. The 40 column display cannot OPERATE at 2 MHz but the Lt. Kernal ONLY shifts speed DURING disk access, so 2 MHz operation causes violent screen flickering WHILE READING FILES. If you do not mind the flickering, you can make the Lt. Kernal go *Full Speed Ahead* even with a 40 column display.

Remember, you can BUILDCPM in either 40 or 80 column modes and still use it in either mode simply based upon the position of the 40/80 key when you issue the GOCPM command.

That is all there is to it! Enjoy your new CP/M power and speed — only on the *Lt. Kernal.*

APPENDIX I

KEY File Programming Example

The following program implements a complete 'dictionary' on the Lt. Kernal. This is NOT intended to be an example of good programming practice, or even to demonstrate the limits of the KEY file system, but rather to serve as a fairly good example of a (simple) KEY file application.

Lines 1-89 aren't even really part of the dictionary application. They comprise a BASIC full-screen editor to make your life a little easier when entering the word definitions into the dictionary's 'definitions' file.

Lines 91-249 actually demonstrate KEY files. The editor code is placed at the beginning only to make it run faster.

This is a very simple, single KEY system. The organization is:

- One KEY file with one directory of 6750, 30 character 'word' keys. The record numbers associated with the keys point directly to a specific record in the definitions (RELative) file.
- A definitions file. This RELative file is intially OPENed for a record length of 41 characters. Each screen line entered in a word's definition occupies a single record, with the leading character of each record set to ' " ' (quotation mark) to allow punctuation to be INPUT from BASIC.

The record number fetched from a SEARCH of the KEY file points to the FIRST record of definition. That record contains a single numeric value defining how many lines of text definition follows.

That means, of course, that one record is 'wasted' for every definition in the file. In a two-tiered KEY system, the definitions' lengths could be contained in yet another KEY file with keys of only 5 characters in length. Remember that KEYs may be ANY text, even print images of numeric values. With that in mind, the record number derived from the word search could BOTH point to the first line of definition in the RELative file, AND be used as a KEY to find the length of the definition in another KEY file. The multi-tiered keying can be carried on infinitely, for very complex search criteria.

An organization diagram for a short, theoretical version of our dictionary follows:

The file 'DICTKEYS' contains the words. The file 'DEFINITIONS' contains the text of the definitions. Record #1 of that file also contains a pointer to the NEXT available record in itself.

	KEY FILE	RELative file
name	'DICTKEYS'	'DEFINITIONS'
org.	6750 keys (words)	up to 65535 lines
	of 30 char.	of 41 char.

KEY	REC—NUM	RECORD	DEFINITION
APPLES	2	1	11 (next avail. rec.)
FIGS	4	2	1 (length in lines)
MONEY	9	3	"RED fruit"
ZEBRA	6	4	1
		5	"YELLOW fruit"
		6	2 (2 lines of def.)
		7	"A horse-like,"
		8	"African animal."
		9	1
		10	"coinage"

In order to use the program which follows, you must first build the KEYs file. This needs to be done only once.

Enter the command BUILDINDEX
Supply the file name 'DICTKEYS'
Request 6750 keys
Request ONE directory, and
a KEY length of 30.

The program will automatically create and intialize 'definitions', the RELative file. You will supply the words and the definition text when the program is run.

The program follows:

```
1 GOTO 173:REM SETUPS
3 REM BASIC FULL-SCREEN EDITOR
5 REM RETURNS UP TO 20 FULL LINES IN E$(1-20)
7 REMEMBER TO DIM E$(20) BEFORE CALLING
9 REM
11 REM
13 REM
15 PRINT"{clear}";:FORI=1TO20:E$(I)="{40 spaces}":
   PRINT
17 NEXT I:X=1:L=1
19 PRINT"    ENTER          PRESS 'RETURN' HERE
   ^";
21 PRINT" DEFINITION        WHEN DONE EDITING.
   {home}";
23 GETG$:IFG$=""THENGOSUB85:GOTO23
25 IFG$="{clear}"ORG$="{home}"THENX=1:L=1:
   PRINT"{home}";:GOTO23
27 IFG$="{insert}"THEN23
29 IFG$="{right}"ANDL=20ANDX=40THEN23
31 IFG$=CHR$(13)ANDL=20ANDX=40THENRETURN
33 IFG$="{right}"THENPRINTMID$(E$(L),X,1);:
   GOSUB67:GOTO23
35 IF G$="{up}"ORG$="{down}"THENPRINTMID$(E$(L),X,
   1)+"{left}";:GOSUB75:PRINTG$;:GOTO23
37 IFNOT(X=40ANDL=20)ANDG$<>"{left}"AND-
   G$<>CHR$(13)ORG$=CHR$(20)THEN PRINT" {left}";
   G$;
39 IF G$="{left}"ANDL=1ANDX=1THEN23
41 IF G$="{left}"THENPRINTMID$(E$(L),X,1);G$;
   "{left}";:GOSUB59:GOTO23
43 IFG$=CHR$(20)THENGOSUB55:GOTO23
45 IFG$=CHR$(13)ANDL=20THEN23
47 X1=0:IFG$=CHR$(13)THENPRINTMID$(E$(L),X,1);
   "{left}":X1=X:X=1
49 IFG$=CHR$(13)ANDX1=40 THENPRINT"{up}";
51 IFG$=CHR$(13)THENGOSUB77:GOTO23
53 GOSUB63:GOTO23
55 IFX=1THENE$(L)=" "+RIGHT$(E$(L),38):GOTO59
57 E$(L)=LEFT$(E$(L),X-1)+" "+RIGHT$(E$(L),40-X)
59 X=X-1:IFX<1THENX=40:L=L-1:GOTO79
61 RETURN
63 IFX=1THENE$(L)=G$+RIGHT$(E$(L),38):GOTO67
```

```
65 E$(L)=LEFT$(E$(L),X-1)+G$+RIGHT$(E$(L),40-X)
67 IFL=20THENIFX<40THENX=X+1:RETURN
69 IFL=20THENG$="":RETURN
71 IFL<20THEN X=X+1:IFX>40THENX=1:GOTO77
73 RETURN
75 IFG$="{up}"THENL=L-1:GOTO79
77 L=L+1
79 IFL<1THENL=1:G$=""
81 IFL>20THENL=20:G$=""
83 RETURN
85 B=B+1:IFB=10THENPRINTMID$(E$(L),X,1);+"{left}";
   :RETURN
87 IFB=20THENB=0:PRINT"{rvs on}"+MID$(E$(L),X,
   1)+"{rvs off}{left}";
89 RETURN
91 REM
93 REM
95 REM
97 REM PAD W$ TO 30 CHARS & SET IN K$
99 REM
101 REM
103 K$=LEFT$(W$+"{30 spaces}",30):RETURN
105 REM
107 REM
109 REM
111 REM ROUTINE TO FIND ONE WORD
113 REM
115 REM
117 GOSUB97:SYS64628:3,1,1,K$,L,H,S
119 IFS=0THENK$="FOUND {rvs on}"+W$:RETURN
121 K$="{rvs on}"+W$+"{rvs off} NOT FOUND":RETURN
123 REM
125 REM
127 REM
129 REM ENTER A NEW KEY IN DICTKEYS
131 REM
133 REM
135 GOSUB97:SYS64628:1,1,1,K$,L,H,S:IFS<>0THENSTOP
137 RETURN
139 REM
141 REM
143 REM ROUTINE TO FIND FOUR SURROUNDING WORDS
```

```
145 FORI=1TO5:W$(I)="{30 spaces}":NEXTI
147 REM
149 R=2:GOSUB97
151 SYS64628:5,1,1,K$,L,H,S
153 IFS=0THENW$(R)=K$
155 R=R-1:IFR<>0THEN151
157 R=4:GOSUB97
159 SYS64628:4,1,1,K$,L,H,S
161 IFS=0THENW$(R)=K$
163 R=R+1:IFR<>6THEN159
165 GOSUB111:W$(3)=K$:RETURN
167 REM
169 REM
171 REM
173 DIM E$(20):OPEN1,8,3,"DICTKEYS":OPEN2,8,2,
    "DEFINITIONS,L,"+CHR$(41)
175 OPEN 15,8,15
177 R1=1:GOSUB209:IFE<>0THENR1=1:GOSUB209:R2=2:
    PRINT#2,R2:REM INITIALIZE DEFS
179 R1=1:GOSUB209:INPUT#2,R2
181 REM
183 REM GET A WORD TO LOOK UP
185 REM
187 PRINT:PRINT:INPUT "WORD TO SEARCH OR {rvs
    on}Q{rvs off}UIT";W$:PRINT:IFW$="Q"THENSTOP
189 GOSUB 111:IFS<>0THENPRINT K$:GOSUB143:
    FORI=1TO5:PRINTI;" ";W$(I):NEXT
191 IFS=0THENU$="3":PRINTK$:GOTO203
193 IFS<>0 THEN PRINT"# OF WORD OR {rvs on}A{rvs
    off}DD WORD OR {rvs on}N{rvs off}EW WORD";
195 GETU$:IFU$=""THEN195
197 IFU$="N" THEN 187
199 IFU$="A" THEN PRINT:R1=R2:GOSUB209:GOSUB 129:
    GOSUB235:GOTO183
201 IFU$<"1"ORU$>"5"ORU$="3"THEN195
203 PRINT:U=VAL(U$):W$=W$(U):GOSUB111:R1=H*256+L:
    GOSUB209:GOSUB221:GOTO183
205 REM
207 REM
209 REM POSITION IN DEFS FILE
211 REM
213 REM
```

```
215 H=INT(R1/256):L=R1-(H*256)
217 PRINT#15,"P"+CHR$(2)+CHR$(L)+CHR$(H)+CHR$(1)
219 INPUT#15,E,T$,T,S:RETURN
221 REM
223 REM PRINT DEF FOR WORD
225 REM
227 PRINT
229 INPUT#2,L1:FORI=1TOL1:INPUT#2,P$:PRINTP$;:
    NEXT:RETURN
231 REM
233 REM
235 REM
237 REM
239 REM GET NEW DEFINITIONS
241 REM
243 R1=R2:GOSUB209:GOSUB3:PRINT#2,0:FORI=1TO20
245 IFE$(I)="{40 spaces}"GOTO249
247 PRINT#2,CHR$(34)+E$(I):R2=R2+1
249 NEXT:GOSUB209:PRINT#2,R2-R1:R1=1:GOSUB209:
    R2=R2+1:PRINT#2,R2:RETURN
```

APPENDIX II

Bulletin Board

Xetec, Inc. now has a bulletin board in operation to provide support for all Lt. Kernal owners. *Fiscal Information* also has a bulletin board and the two numbers are:

Xetec board number	913 827 1974
Fiscal board number	904 252 8179

We encourage your use of these boards and welcome any input as to how we can improve our service to you.

APPENDIX III

20 Meg Add-on Drive

Please use the following steps to connect your 20 Meg Add-on Drive enclosure to your Lt. Kernal.

Hardware connection

1. Remove 25 pin signal cable from the back of the Lt. Kernal and plug into the back of the Add-on enclosure (either slot).
2. Plug one end of second 25 pin cable (included with Add-on) into the second slot on the back of the Add-on Drive enclosure.
3. Plug the other end of the second cable into the slot on the back of the Lt. Kernal enclosure labeled *Host Adaptor*.

Setting up LU parameters

1. Type *CONFIG*
2. Select *F6* for DOS 6.3
   F1 for DOS 7.0
3. For physical controller answer *1*.
4. For physical drive answer *0*.

At this point, your screen should display the LU parameter set-up table. (NOTE: All the LUs on other drive(s) are not shown). Now you can add LUs that have NOT BEEN USED on the original drive.

The first LU assigned can use cylinders beginning with cylinder 0 (zero) and up.

5. UPDATE the new parameters.
6. Exit CONFIG by using *F8* for DOS 6.3
   F7 for DOS 7.0

ACTIVATE new LUs and perform a power down, power up sequence to Update all LUs.

This completes the process.

Note: If you have DOS V7.1, a separate SYSGEN disk and set of instructions will be sent with your add-on drive for installation purposes.

INDEX

Xetec Lt. Kernal operating manual

THIRD PRINTING

XETEC, Inc. 2804 Arnold Rd. Salina, Ks. 67401
(913) 827-0685

QUICK COMMANDS REFERENCE

DIRECT—MODE Commands

↑↑ restores defaults
p. 8-1

ACTIVATE erase and re-creates bam and index for existing LU
p. 8-2 ACTIVATE

AUTOCOPY copy file(s) between LUs
p. 8-3 AUTOCOPY

AUTODEL delete file(s) from LU
p. 8-4 AUTODEL

AUTOMOVE move files between USER partitions
p. 8-5 AUTOMOVE

BUILD create 'formatted' RELative file
p. 8-6 BUILD_filename,nrecs,recl

BUILDCPM construct CP/M on the CP/M LU
p. 8-7 BUILDCPM

BUILDINDEX builds a KEY file for indexed RELative files
p. 8-8 BUILDINDEX

CHANGE change file's characteristics
p. 8-9 CHANGE_[lu:[filename

CHECKSUM verify DOS integrity
p. 8-10 CHECKSUM

CLEAR clear archive bits for file(s)
p. 8-11 CLEAR

CONFIG change power-up default system settings and LU parmeters
p. 8-12 CONFIG

COPY copy oldfilename into newfilename
p. 8-13 COPY_['']['lu:][newfilename]=[lu:]oldfileneme['']

D set temporary hardware device # for hard disk
p. 8-14 D[_drynum]

DEL delete lines of BASIC program in memory
p. 8-15 DEL_line number or DEL_[beg.line]—[end.line]

DI lists keys within the specified KEY file
p. 8-16 DI

DIR list directory of files on hard disk
p. 8-17 DIR_[[lu:[user:]]:[Tfiltyp][P][S][S][G][C]_][filename]

DUMP write editable text image of BASIC in memory to disk file
p. 8-19 DUMP_[range_][lu:]seqfile

ERA
p. 8-20
erase one file from disk
ERA_[lu:[user:]]filename

FASTCOPY
p. 8-21
fast 1541 back-up/restore utility
FASTCOPY

FETCH
p. 8-22
create memory BASIC programs from disk resident text file
FETCH [lu:]filename

FIND
p. 8-23
text or tokens in BASIC
FIND_delimSTRINGdelim[,<line-range>

GO64
p. 8-24
sends CPU to the C-64 mode
GO64

GO128
p. 8-25
sends CPU to 128 mode
GO128

GOCPM
p. 8-26
sends CPU to the CPM mode
GOCPM

ICQUB
p. 8-27
capture or run captured copy protected software
ICQUB or filename

L
p. 8-30
abbreviated LOAD file from disk
L_["][lu:]filename["]

LOAD
p. 8-30
load file from disk
LOAD "[lu:]filename",dev[,sa]

LKREV
p. 8-31
reports DOS version
LKREV

LU
p. 8-32
changed logged logical device #
LU[_unum]

MERGE
p. 8-33
merge disk based BASIC programs into memory program
MERGE_[lu:]filename

OOPS
p. 8-34
recover last erased file
OOPS

QUERY
p. 8-35
list characteristics of file
QUERY_[lu:]filename

RENUM
p. 8-36
renumber BASIC program in memory
RENUM[_incr[,newstart[,oldstart-oldend]]]

S
p. 8-42
save & replace BASIC programs in memory under same name
S

S
p. 8-42
abbreviated SAVE program or range of memory to disk
S_[range][lu:filename

SAVE
p. 8-43
SAVE program or range of memory to disk
SAVE "[range][lu;]filename",dev

DIRECT—MODE Commands continued

SHIP p. 8-44
locks head in preparation to transport hard disk
SHIP

TYPE p. 8-45
list disk resident File program to screen
TYPE_[lu;]filename

UPDATEDOS p. 8-46
refresh all DOS image files with new version of DOS (automatic)

USER p. 8-47
change logged subdirectory number
USER[_user#]

RUN—MODE Commands

BUILDKEY p. 7-4
create key files
SYS64628:0,lfn,directory,Stringvar,recl,rech,status

COPY p. 8-13
copy sourcefile to destfile
OPEN #lfn,dev,15,"C[lu]:dfilename=[lu:]sfilename"

DELETE p. 7-5
key from specified directory of a KEY file
SYS 64628:2,lfn,directory,Stringvar,recl,rech,status

INPUT# p. 7-6
read string up to 254 chars from file
INPUT# lfn,stringvar

INSERT p. 7-7
new key into specified directory of a KEY file
SYS 64628:1,lfn,directory,Stringvar,recl,rech,status

LDLU p. 7-8
change operating characteristics
OPEN #lfn,dev,15,"ldev#LU#USR#"

LOAD p. 7-9
load program file from disk
LOAD "[lu:]filename",dev[,sa]

OPEN p. 7-10
open disk file for i/o
OPEN #lfn,dev,sa,"[lu:]filename"

PRINT# p. 7-11
write string up to 254 chars to file
PRINT#lfn,stringvar or PRINT#lfn,"literal string"

SAVE p. 7-12
save program (or memory range) to disk
SAVE "[<range>][lu:]filename",dev

SCRATCH p. 7-13
scratch (erase) file from disk
OPEN #lfn,dev,15,"S[lu]:filename

SEARCH p. 7-14
search KEY file for key entry match, > condition, or < condition

Match	SYS 64628:3,lfn,directory,Stringvar,recl,rech,status
Greater-than	SYS 64628:4,lfn,directory,Stringvar,recl,rech,status
Less-than	SYS 64628:5,lfn,directory,Stringvar,recl,rech,status

SHUFFLE p. 7-15
re-distributes keys within a directory of a KEY file for optimum space usage within the file
SYS 64628:7,lfn,directory,Stringvar,recl,rech,status