1
original

THE ORIGINAL LOCKSMITH USERS MANUAL

VERSION 5.0

TABLE OF CONTENTS

NOTICE

Omega MicroWare, Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

Omega MicroWare, Inc. and the author make no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. This software is sold or licensed "as is". The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Omega MicroWare, Inc., the author, their distributors, or their retailers) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Omega MicroWare, Inc., or the author be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if they have been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

NOTICE OF COPYRIGHT

INTRODUCTION

THE OMEGA POLICY

THE COPYRIGHT LAW ALLOWS THE CREATION OF ARCHIVAL COPIES OF COMPUTER SOFTWARE WHICH IS OWNED BY THE LOCKSMITH USER. LOCKSMITH 5.0 IS SOLD WITH THE UNDERSTANDING THAT THE PURCHASER WILL NOT USE THE PROGRAM TO GENERATE DISKS OF COPYRIGHTED PROGRAMS FOR SALE OR DISTRIBUTION. SHOULD THE PROGRAM BE MISUSED, OMEGA MICROWARE WILL ASSIST IN THE PROSECUTION OF VIOLATORS AT THE COPYRIGHT HOLDER'S REQUEST.

If a program is sold with an archival (or back up) disk, Omega MicroWare will make no effort to permit Locksmith to make an archival copy of that program.

THE HISTORY OF LOCKSMITH

In the early days of the Apple computer all programs were copyable. In fact, each Apple Disk Operating System (DOS) came (and comes) with a copy program permitting the owner to copy programs quickly and efficiently. DOS itself has facilities to save out a program and information to diskettes. Most of the first commercial programs for the Apple were copyable. However some software manufacturers were concerned over indiscriminate distribution of their product and began to employ methods that would not allow copying of their programs. Other manufacturers noticed that disks had a tendency to wear out or get destroyed and began charging for the privilege of getting a replacement. In some cases the backup copies were almost as much as the originals. Some manufacturers would not provide backup copies, period! Finally, from the users' standpoint, when they found out they needed a backup copy and attempted to contact the manufacturer, they sometimes found the manufacturer was out of business.

Any Apple disk can be copied! As software protection was developed, as an intellectual exercise, computer aficionados determined what was done and counteracted it. These individuals were not necessarily interested in copying the program, but in determining how the protection was done. Today, as computer knowledge grows, more and more people exist who can defeat copy protection.

Unfortunately, there was (and is) another group of people who only wish to make a copy of their program in case their disk goes bad. They don't care how it's duplicated, they have no ulterior motives, they only want to use their program. Locksmith came about as a result of this need.

In January 1981, Omega MicroWare offered a program which would copy programs for the general user.

EASE OF USE

In the original days of Locksmith, it copied virtually all programs. In this, the seventh revision, it again copies virtually everything. However it is to be expected that new techniques will be developed to prevent the user to make archival copies. As this happens, registered Locksmith owners will be provided with the necessary information to back up programs. As the state of art changes, new revisions of Locksmith may be necessary...always at a reasonable price.

Through this manual, we will introduce you to the way Locksmith works. If you're not interested, you need not go beyond the first chapter. However, it is interesting to note that classes and seminars are springing up

around the country dealing with the way Locksmith works and modifications and changes which can be made using it in conjunction with its companions, Watson and The Inspector, programs also available through Omega MicroWare.

THE FUTURE

Imitation is the sincerest form of flattery, they say. Locksmith is flattered by its imitators and it keeps us on our toes. We promise our customers to stay in the forefront of the duplication technology and provide them with the information and assistance they need to keep us number one.

SYSTEM REQUIREMENTS

You will need an Apple II, Apple II+, Apple IIe or Apple /// in emulation mode. Locksmith 5.0 will also work with all known computers running Apple software. You must have at least one disk drive and disk controller card. If you have two disk drives, both on the same disk controller card, your copying will be much easier and faster. If you have a controller card manufactured by Apple, all features of Locksmith 5.0 will work as described. Other, non-Apple manufactured controller cards, may or may not allow all the features of Locksmith 5.0 to function properly. Apple built controller cards are totally controlled by software which means that the disk controller software is modifiable. Some controller cards are controlled by firmware and are therefore not modifiable. If you tell these to do some things, via software, they aren't able to do it. Fortunately, most non-Apple controller cards are also software controlled.

A printer can also be used but is not required.

THE HISTORY OF LOCKSMITH AND COPY PROTECTION

For the past several years, there has been an intense battle being fought between software manufacturers and software users. The manufacturers, concerned about their programs being pirated or stolen, started 'protecting' their software. They did this by making their programs uncopyable. This means that normal copy programs would no longer copy their software. Since it could not be copied, it could not be passed around between users. This was to insure that anyone who wished to use a program would be required to purchase it, thereby guaranteeing that the manufacturer would receive his fair share of profits.

As in most issues, however, there are two sides. Software users, upon purchasing a program, received a disk which they could not copy. This means that they could not even make back-ups of their disks, which is a legitimate concern of everyone who has 'blown' a disk. Since some businesses day to day operations rely very heavily on their software investment, this becomes a very critical situation. Some applications simply do not allow for up to several weeks waiting time while the bad disk is sent out to be replaced. In addition, some of the manufacturers charge inflated prices to get these replacements. It seems rather unfair to charge $35.00 for a replacement of a $2.00 disk for which the user already paid $100.00 or more.

Shortly after these 'protected' disks started appearing on the market, Locksmith was made available to copy these disks. Locksmith used a new type of technology to copy protected software, known as nibble-copying. In the several years since, the manufacturers have introduced new, more sophisticated methods of copy protection. As a result, Locksmith has been updated several times to anticipate new methods of protection.

In this chapter, we will discuss the origin of Locksmith, and some of the different methods that have been used for protection over the past several years.

Locksmith was written by an Apple programmer with 18 years of computer experience, including systems programming on large IBM mainframes at several large corporations. His interest in computers dates back to grammar school, when he would spend his Saturdays taking computer courses at the IIT computation center in Chicago.

The first version of Locksmith, which was never released, was a primitive nibble copy program known as 'NIBY'. It was written as an educational exercise — "because it was a challenge". When it was shown to some of his close friends from the local Apple users group, it gathered much interest, and because no program like it was available, several Apple users suggested that the program might be marketable.

In January, 1981, Omega MicroWare (then called Omega Software Products) released its first product. Locksmith version 2.0 was the first program ever introduced to allow the Apple user to backup his copy-protected software.

Like any tool in the wrong hands, it was feared that Locksmith might be used for tasks it was not intended to perform. Because of this fear, each Locksmith was uniquely serial numbered and registered, and every copy of Locksmith also placed this unique serial number on EVERY disk that copied. This fact, by the way, was never made known until now. Encoding the Locksmith serial number on the copied disk provided Omega with the capability of identifying the owner of any Locksmith used for pirating any manufacturers software, and Omega offered to assist software manufacturers in the prosecution of software pirates. No software

manufacturer however, has ever requested this assistance from Omega. Placing the Locksmith serial number in an inconspicuous place on every disk that is copied is not an easy task. In fact, this practice has caused several problems with copying some disks. Because of this, the Locksmith version 5.0 no longer encodes its serial number on the copy disk.

Locksmith has evolved from version 1.0 (the unreleased 'NIBY'), to versions 2.0, 2.1, 2.2, 3.0, 3.1, 4.0, 4.1, 4.1a, and now 5.0. No longer just a nibble-copier, Locksmith is now a full-featured utility and diagnostic tool for the Apple II computer. Always searching for improvements, Omega welcomes suggestions, comments, and any questions you may have about Locksmith.

We will now discuss some of the different methods that have been used for protection over the past several years. Some of the descriptions are of a technical nature, and are intended for the more advanced user.

The very first types of copy protection were very simple in nature. The first protected disks used nothing more complicated than erasing an unused track on the disk. This was usually track 3. This method is not very complicated, but initially it was quite effective. All of the copy programs at that time copied the disk one track at a time. When it tried to read the erased track, it would get an I/O error, causing the copy program to stop. By doing this, none of the tracks beyond the erased track would copy.

Some of the copy programs that came out a little later would copy only those sectors that were marked on the catalog track (track $11) as being used. This got past the erased track problem. To combat those copy programs, companies started to move the catalog to a different track. When the copy program went out to track $11 to read the information, the information would not be there. This also prevented a normal Disk Operating System (DOS) from reading and writing to the protected disk.

Shortly after that time, a new method was introduced. It was a little known fact that while the disk normally used only tracks $00-$22, it was actually capable of reaching track $23. Some of the software began using this track for program information. All copy programs at that time were incapable of copying track $23, so that when a copy was made, some information was lost. This method proved to be very dangerous, because some disk drives could not reliably read or write to track $23. This means that you could not even use the original protected disk on that drive, since it could not read that track.

At this point, the protection methods started to become more sophisticated. State of the art had progressed to the point where the manufacturers were actually changing the format of information on a disk. At first, this was done by changing the checksum for the address field on the disk. This would cause I/O errors which would halt the copy process. Disks which were protected by changing the format of information required their own Disk Operating System.

At approximately the same time, some manufacturers started changing the format of the address field on a disk sector. Normally, the format is to have an address header, followed by information concerning the volume, track, sector and checksum. This was followed by an address trailer. The order of the volume, track and sector was changed around, or put in a different format. For example, one company changed all sector numbers to be even numbers. Instead of sectors 0, 1, 2, 3 etc., they used numbers of 0, 2, 4, 6 etc. Normal DOS could not understand these formats.

The headers and trailers for both the address fields and data fields were changed as the next type of protection. Since DOS looks for a specific header or trailer to read a sector, it will never find a sector on this type of disk. On these disks, it would impossible to read any information with a normal DOS.

Once again, an entirely new technology appeared for protection. Up until this time, all information was stored on a disk in Track/Sector format. Now, tracks started to appear using pseudo-sectors. A pseudo-sector is a long string of data, with only a data header of some type. Some of these pseudo-sectors were an entire track in length. None of the programs for reading Track/Sector format could decode this type of track. With the advent of pseudo-sectors, nibble copy programs became necessary. Until this time, it was usually possible to modify normal DOS, to copy these disks. This was no longer possible. Shortly after pseudo-sectors appeared, Locksmith was first introduced. It was capable of copying tracks which were in a non-standard format.

Synchronized tracks were the next method of disk protection. Synchronized tracks are tracks that are written in a specific timing relationship to each other. For example, after reading track $00, the disk drive would then seek to track $01. Upon arriving at track $01, data would be read in from that track. The program that was booting would look for specific data to be present when it arrived at the new track. If this data was not at the beginning of the track it read, it would cause the program to fail. This meant that copying tracks without preserving this timing relationship would result in a bad copy, even though all of the information was transfered.

Another type of protection which was concerned not only with the actual data that was copied, was nibble counting. After writing a track when generating a disk, the track would be read back, and a count of the nibbles on a track would be stored on the disk. Upon booting, the disk would look for the track to be that specific length. Since very few disk drives run at exactly the same speed, the chances were very unlikely that the track length would be the same on a copied disk.

Software manufacturers next started to take advantage of a little known fact concerning the disk drive. While disk drives were normally used on tracks $00 through $22, they were capable of reaching between tracks. This area between tracks is known as a half-track. Due to the width of the read/write head, it is not possible to write data on adjacent tracks and half tracks without experiencing cross-talk problems. However, it is possible to write data on half-tracks, providing that the adjacent tracks are not used. It then became necessary to use copy programs that were capable of reaching these half-tracks. One major problem with this type of protection scheme is that not all disk drives are capable of reaching half-tracks. Some very popular drives can only reach integral tracks, and disks using this type of protection can not boot on these drives. If you are using Micr-Sci type A40 drives with your Apple, keep in mind that half-tracks can not be accessed, although all other Locksmith functions work as documented.

There was one type of protection which appeared and shortly thereafter, disappeared from the market. This type of protection actually physically damaged the disk. A scratch was made on the disk with a sharp instrument. When booting, the disk would attempt to write and then re-read data on the track with the scratch. If the test passed, it meant that the disk was not damaged, and therefore, not an original disk. This was a very undesireable method, since the damaged portion of the disk would need to come in contact with the read/write head on the disk drive. When the head was over the damaged track, you could actually hear a 'tick-tick' as the scratch hit the head. This could cause damage to the

head, and because of that, the method was quickly abandoned.

The chief difficulty in copying protected disks was identifying which nibbles on a track were normal, and which nibbles were self-sync. Locksmith versions 2, 3, and 4 attempted to identify self-sync nibbles by context, that is by the surrounding nibble patterns. Up until this time, disks used nibbles with a value of $FF for self-sync nibbles. It was a fair assumption that a string of $FF nibbles represented fields of self-sync. To combat this, manufacturers started to use different values for self-sync. This made identifying self-sync nibbles more difficult. In some cases, multiple nibbles were used, for example $D5 $AB $D5 $AB, etc. Because Locksmith identified self-sync nibbles contextually, parameter changes were required for copying disks of this type with Locksmith versions 4.1 and earlier.

One very sophisticated method of protection appeared on the market shortly thereafter. This method required that specific nibbles in the middle of normal data be special self-sync nibbles. By using timing routines, it was possible to determine if this nibble was normal, or special. This special nibble is called a data-latched nibble. When reading a track of nibbles normally, the data-latched nibble was indistinguishable from a normal nibble. Copying these tracks was very difficult, since it required actually breaking or deciphering the code to determine which nibbles had to be data-latched. This method was very effective, and has been in use for quite a while. Locksmith 5.0 is capable of determining self-sync nibbles while it is reading them without regard to context nibbles, and it is able to detect data-latched nibbles without any user-supplied parameters.

As mentioned before, it is not possible to write adjacent tracks and half-tracks. This is due to the fact that the read/write head is wide enough to overlap onto the adjacent track or half-track, effectively erasing information. To alleviate this problem, the concept of spiral tracks was invented. This is simply writing approximately 1/3 of a track, jump out a half track, write another 1/3 of a track, etc. By using this method, adjacent tracks and half-tracks may be used without actually writing any data closer than one full track apart. The data on the disk actually seems to spiral in toward the center of the disk, hence the name 'spiral track'.

The most recent type of protection is much the same as half-tracks. It is the use of quarter tracks. While it is true that the disk drive is not normally capable of reaching quarter tracks, it is possible to drive the stepper motor on the drive so that it will stop on the quarter track. This requires some very special timing routines. It works basically the same as half-tracks, and the same restrictions about adjacent data apply.

Locksmith 5.0 is capable of handling all of these types of protection methods, along with many others. Due to its extreme flexibility, it will also support many protection methods which have not yet appeared. In the meantime, the conflict between software manufacturers and software users will continue.

IMPORTANT LOCKSMITH INFORMATION

This chapter includes general information important to the Locksmith 5.0 user. This information concerns many of the Locksmith functions.

The CTRL-Z key may be pressed at any time that you wish the screen contents to be dumped to a printer. See the chapter in this manual on the 'CTRL-Z' key.

The ESC key may be pressed at any time to abort a function. Pressing this key will eventually bring you back to the Locksmith main menu.

When the prompt 'PRESS SPACE TO CONTINUE' appears in flashing characters at the bottom of the screen, you may press the space bar to continue, or press the ESC key to abort the function.

The status display at the top of the Locksmith screen shows tracks from $00 to $23 (decimal 0 to 35). Normally, only tracks $00 to $22 are used, but Locksmith supports track $23, in case it is used on the disk to be copied. The status display has four rows of status information. The top one is for integral tracks (00, 01, 02, etc.). The 3rd one is for half-tracks. The 2nd and 4th rows are for 1/4 and 3/4 tracks, respectively. The status display is not cleared by Locksmith functions, with the exception of the 'CLEAR STATUS' command. The status indicators that may appear in the status display are discussed in the individual chapters describing the Locksmith functions and are summarized in the appendix.

The RESET key will cause Locksmith to reboot your system. Do not use the RESET key unless you wish to reboot.

The message that appears below indicates that the function you have selected destroys data on the target disk. Make sure that your Locksmith disk or any other disk you wish to keep is out of the drive at the time the function begins.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
             DISK SPEED



                    WARNING:


  THIS LOCKSMITH FUNCTION DESTROYS DATA

          ON THE TARGET DISK.

          PRESS ESC TO ABORT




        PRESS SPACE TO CONTINUE
```

The prompt 'INSERT DISK(S)' will appear immediately before a Locksmith function is to start, giving you a last chance to insure that the drives contain the proper disks, before you press the space bar to begin the function.

The prompt 'INSERT LOCKSMITH DISK' indicates that to continue, Locksmith needs information that is on the Locksmith disk. Place the Locksmith disk in the drive

requested by the prompt, and press the space bar. The slot and drive requested will always be the one booted from, regardless of which drives are used for Locksmith functions. If you have more than two drives on your system, you can boot Locksmith in one slot, use the 'PARAMETER CHANGES' function to change the SLOT that Locksmith is to use, and keep the Locksmith disk in the boot drive to eliminate the need to switch disks when Locksmith needs information from the Locksmith disk.

GETTING STARTED

A step by step example of how to copy a disk that requires no parameter changes.

1. First insert your Locksmith disk in drive one.

2. Turn on the computer to load the Locksmith program into memory.

3. When the Locksmith title page appears, press the space bar.

4. You should now be at the main menu.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
THE LOCKSMITH - VERSION 5.0 - SER#10000

B BACKUP/COPY DISK    I INSPECTOR/WATSON

P PARAMETER CHANGES E ERASE DISKETTE

/ CLEAR STATUS        N NIBBLE EDITOR

T TEXT EDITOR         S DISK SPEED

Q QUICK SCAN DISK     C CERTIFY DISK

U 16 SCTR UTILITIES X EXIT / REBOOT


CTRL-Z SCREEN PRINT ESC ABORT/RESTART
```

5. Press the letter 'B' for Backup/Copy disk.

6. You will see the following screen display.

```
HEX 0000000000000000111111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
              BACKUP/COPY DISK


              PRESS ESC TO ABORT

DRIVE-   ORIGINAL:1
```

7. NOW REMOVE YOUR LOCKSMITH DISK FROM THE DRIVE.

8. When prompted for the drive of the ORIGINAL it will show the default '1'. To accept the default press the 'RETURN' key.

9. When prompted for the drive of the COPY it will show the default '2'. If you have two drives accept the default by pressing the 'RETURN' key. If you only have one drive enter a '1'.

10. Now you will be prompted for the starting TRACK number. It will show the default starting track as '00'. Press the 'RETURN' key to accept the default.

11. Now you will be prompted for the ending TRACK number. It will show the default '22'. Press the 'RETURN' key to accept the default.

12. Now you will be prompted for the INCREMENT. It will show the default increment as '1'. Press the 'RETURN' key to accept the default.

13. Now you will be asked if you wish to SYNCHRONIZE tracks. It will show the default as 'N'. Press the 'RETURN' key to accept the default.

14. Now you will be asked if you wish to PRESERVE TRACK LENGTH. It will show the default as 'N'. Press the 'RETURN' key to accept the default.

JUST PRIOR TO YOUR PRESSING THE 'RETURN' KEY IN ANSWER TO THE AUTO ERROR RETRY QUESTION, THE LOCKSMITH SCREEN SHOULD LOOK LIKE THIS:

```
HEX 0000000000000000111111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
              BACKUP/COPY DISK


              PRESS ESC TO ABORT

DRIVE-   ORIGINAL:1   COPY:2

TRACK-   START:00     END:22      INC:01

SYNCHRONIZE (Y/N):N

PRESERVE TRACK LENGTH (N/A/M):N

AUTO ERROR RETRY (Y/N):Y
```

15. Finally, you will be asked if you wish AUTO ERROR RETRY. It will show the default as 'Y'. Press the 'RETURN' key to accept the default.

16. After you have pressed the 'RETURN' key you will be asked to INSERT DISK(S) and press the space bar to continue.

17. Your Locksmith program will now attempt to copy the disk using the default parameters built into the program.

18. While copying the disk Locksmith will provide you with a dynamic screen display showing the track it is copying and certain information about the track. You will be using this information as you become more familiar with Locksmith.

19. When Locksmith is finished analyzing and copying the disk it will display a flashing 'PRESS SPACE TO CONTINUE' at the bottom of the screen. When you press the space bar, the main menu will reappear.

[B] BACKUP/COPY DISK

This command is used when you are ready to copy a disk. All parameter changes should have been made either through the parameter change mode, or loaded into Locksmith from the Text Editor.

Locksmith will prompt you for the DRIVE of the ORIGINAL disk. The default drive number is being displayed on the screen. If this is the drive you wish to use for the original, press RETURN. If you wish a different drive for the original disk, enter the number of the drive you wish to use. Locksmith will prompt for the drive number for the COPY. Again the default drive is shown. If you wish to use the default drive press RETURN, if not enter the drive number you wish to use.

You will now be prompted for the starting track number you wish to use to make this copy. Again a default value is shown. If you wish to use the default value, just press RETURN. If not, enter in the starting track number you wish to use and press RETURN. After you have made your selection for the starting track number, Locksmith will prompt you for the ending track number. Again the default value is shown. If you wish to use the default value, press RETURN, otherwise enter the value you wish to use and press RETURN. The next prompt from Locksmith will be for the increment you wish to use between tracks. The default value is shown. If you wish to use the default press RETURN, otherwise enter the increment you wish to use and press RETURN.

Now Locksmith has the information about what drives you wish to use and the tracks you want to copy. It now needs to know some other information about how you wish to copy the disk.

You will now be asked whether you wish to synchronize the tracks or not. The default value is No. If you wish the default just press RETURN. If not enter in a 'Y' for yes. Some disks are protected by having the tracks written to the disk in a certain time relationship to each other. If you select synchronization, this time relationship will be preserved.

You will now be asked whether you wish to preserve the track length.
The prompt is:

PRESERVE TRACK LENGTH (N/A/M):N

This is a technique called nibble counting. Some protection schemes count the number of nibbles on a track when the disk is made. When you copy the disk the odds are against you writing the same number of nibbles on the track. This is due to the variations in drive speed that occur all the time. If you require Locksmith to write the same number of nibbles on the copy as it read from the original, it could increase the copying time significantly.

If you wish the default value press RETURN. If you do not wish to use the default value you have two choices. If you select 'A', Locksmith will attempt to adjust the track length automatically for you.

If you choose 'M' then you must adjust the track length yourself. There are two ways to do this. One way is to adjust the drive speed, using the speed adjustment screw inside the disk drive. The other way is with the '<' and the '>' keys. While preserve track length is in effect Locksmith will continuously print either a '<' or a '>' followed by a four digit hex number. This number represents the difference in count between the original and copy disks. To adjust the track length with the '<' and the '>' keys, you should press the key that is shown in front of the four digit hex number. When you press this key, your computer will start to beep. Pressing any other

key will stop the beeps and show the difference in track length again. The object is to get this number to be '0000'. When this occurs the track length is the same as the original and Locksmith will move to the next track.

If you do adjust your drive speed with the set screw, be sure and use the Locksmith disk speed test to reset your drive to the proper speed before you leave Locksmith.

If you have selected 'M' manual mode and have gotten the track length fairly close, you may press 'A' and have Locksmith take over and finish the adjustments.

You will now be prompted for how you wish errors to be handled. You have two options. The first is to let Locksmith attempt to fix the error, and the second is to have Locksmith ask you to fix the error.

The prompt is:

AUTO ERROR RETRY (Y/N):Y

The default is to let Locksmith do the work. If you wish the default just press RETURN, if not enter 'N'.

Just prior to your answering the last prompt about error retry the screen would look like this if you accepted all the defaults to the prompts.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
                BACKUP/COPY DISK

                PRESS ESC TO ABORT

DRIVE-   ORIGINAL:1   COPY:2

TRACK-   START:00     END:22     INC:01

SYNCHRONIZE (Y/N):N

PRESERVE TRACK LENGTH (N/A/M):N

AUTO ERROR RETRY (Y/N):Y
```

After all the questions are answered you will be prompted to INSERT DISK(S). After the disks are in the appropriate drives, press the space bar to proceed with the actual copying process. When Locksmith is done with the copying process you will be prompted to PRESS SPACE TO CONTINUE. Pressing the space bar will return you to the main menu.

ERROR CODES FOR BACKUP FUNCTION

'o' indicates that no error occured.

'1' nothing inteligible found on track - track is garbage.

'2' can't find the repeat of track start.

'4' error compareing first and second track images.

'5' end of limit during automatic count preserve.

'8' verify field after write (perhaps track start overwritten).

NOTE — Because of the flexibility of Locksmith BACKUP function using LOCKSMITH PROGRAMMING LANGUAGE, other error codes (ranging from 0 to $F) may appear on the status display, and other error codes (ranging from 0 to $FF) may appear on the main screen during a backup function. The cause of these error codes can be determined by examining the current LPL file in use during the backup function, as they are entirely user-defined.

[Q] QUICKSCAN DISK

This utility will help you to determine what tracks are in use on a disk you are trying to copy.

[Q]

Pressing 'Q' from the main menu will put you into the quickscan utility. This utility is used to determine what tracks on a disk contain valid data. You will be prompted for the drive number you wish to use and then for the starting track, ending track and increment between tracks.

The display is a hi-res graphic display of the sync bytes on a disk. The graphic display of a track runs from the bottom to the top of the screen. The first time you run this utility you should do it on a normal DOS 3.3 disk. First try tracks 0 to 22 in whole track increments. This will show you what a good track will normally look like. The series of dots you see on the screen above each track number are the gaps of self sync bytes between each sector. Normally on a 16 sector disk there will be 16 or 17 of these dots. This is because Locksmith always reads long enough to read the whole track into memory and it may have read more than the whole track. Similarly a 13 sector disk would have 13 or 14 of the little dots. On a 16 sector disk one of the dots will be a little longer than the others, that is the self sync group in front of sector zero on that track. If you look closely you can see a definite pattern to the longer dots; they will either move up or down as you move from track to track on the disk. This is due to the time it takes to move the disk drive head from track to track.

Now using the same disk try quickscanning from track .5 to track 22.5 with an increment of 1. You will see very long lines of white in no particular pattern. This means there is no valid data on the track.

If the disk is using a protection scheme called spiral tracking you will see the long band of white but there will also be a pattern of black in between the white. The black sections will not be right next to each other but will be offset slightly as you move across the tracks. This is due to the time it takes the disk drive to move from track to track. The more that you use the quickscan feature the more valuable you will find it. This is because you will become more adept at interpreting the results it gives you.

[P] PARAMETERS

[ESC]
Pressing the 'ESC' key will always abort the current operation and return you to a menu.

[CTRL-Z]
This option is activated by pressing and holding the key marked 'CTRL' and pressing the 'Z' key. This option is a screen print. It will print whatever is showing on the text screen at the time it is pressed. This is assuming that you have a printer turned on and that Locksmith has been told the correct slot for the printer interface card.

Now we will cover cursor movement within the parameter buffer.

CURSOR MOVEMENT:

Locksmith supports the normal Apple II cursor movement keys.

```
              UP
              [I]
   LEFT [J] [K] RIGHT
              [M]
             DOWN
```

You may also move left or right with the left and right arrow keys.

If you have an Apple //e the up and down arrow keys are also operational.

If you move left past the beginning of the line, you will be placed on the last character of the previous line. Similarly, if you move past the end of the current line, you will be placed at the beginning of the next line.

[<]

Pressing the '<' key will move backwards through the parm buffer one screen page, unless you are already at the beginning of the buffer.

[>]

This key moves forward through the parm buffer one screen page, unless you are already at the beginning of the buffer.

[,]

This key scrolls continuously backwards until another key is pressed.

[.]

This key scrolls continuously forwards until another key is pressed.

Pressing 'C' will allow you to change the parameter you have the cursor on by typing in a 2 digit hex number. After typing the number you may press either the 'RETURN' key or the space bar. If you press RETURN, you will exit change mode. If you press the space bar you will move to the next position within the buffer and can continue to make changes.

[CTRL-P]

This command is used to change prameters from within the Parameter Editor. It has the following options available. After pressing 'CTRL-P' you will get the prompt PARM:.

PARM:

Enter 'CTRL-R'and press the RETURN key. This will restore all default parameters.

PARM:

Enter '?' and press RETURN. This will display the valid parm names.

PARM:

Enter <name> and press RETURN. This tells Locksmith you wish to change the named parameter. You will be shown the parameter number in parenthesis, and the current value of the parameter. You may type in a new value and press RETURN or simply press RETURN to accept the current value. Valid parameter names and their definitions can be found in the appendix.

PARM:

Enter <hex value> (values [0-$1FF) are valid) and press RETURN. This is an alternate way to specify a parameter. The results and options are the same as with <name> above.

PARM:

Enter '+' and press RETURN. This tells Locksmith you wish to change the next parameter in sequence. The current value will be displayed. The options and results are the same as the two previous commands.

PARM:

Press <return>. Exit parameter change mode.

AFTER PARM HAS BEEN ENTERED YOU HAVE THE FOLLOWING OPTIONS FOR VALUE.

VALUE:

Enter <return>. Accepts displayed current value.

VALUE:

Enter <hex> and press RETURN.

VALUE:

Enter <hex hex hex ...> and press RETURN. This enters the hex values into memory starting at PARM and continuing in sequence until the RETURN is encountered.

VALUE:

Enter <track number with decimal point> and press RETURN.

NOTE: NO CHECK IS MADE TO SEE IF THE PARM ACTUALLY REPRESENTS A TRACK VALUE. SINCE THE TRACK VALUES ARE STORED DIFFERENTLY THAN NORMAL PARM VALUES, USING A DECIMAL POINT WHEN ENTERING NORMAL PARMS WILL STORE INCORRECT VALUES.

[C] CERTIFY DISK

This is a utility you can use to check disks for flaws prior to use. It may also tell you if there is something wrong with your disk drive.

[C]

Pressing 'C' from the main Locksmith menu will take you to this utility. THIS UTILITY DESTROYS DATA ON THE TARGET DISK.

You should place the blank disk you wish to check in one of your drives. The program will prompt you for the drive number, the starting and ending track numbers and the increment you wish to use between them.

This program works by writing a specific pattern onto every track you specified. It then reads this pattern back to verify that it can read what was written. If Locksmith is unable to read what it wrote, it will flag the track as bad. There can be three different reasons for the track being flagged as bad. First it may not have been written correctly due to a disk drive malfunction. Second it may not read correctly due to a disk drive malfunction. Third and most likely the disk media may be flawed.

When Locksmith checks the disk it will write a period '.' in the status area for every track that checks good. If a track checks bad it will be flagged with an asterisk '*'.

When the program terminates it will take you back to the Locksmith main menu. The status area will not be cleared. The following print out is an example of a bad disk.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00 ..**...**..**..*..*......**.....*..
.25
.50
.75
THE LOCKSMITH - VERSION 5.0 - SER#10000

B BACKUP/COPY DISK   I INSPECTOR/WATSON

P PARAMETER CHANGES E ERASE DISKETTE

/ CLEAR STATUS       N NIBBLE EDITOR

T TEXT EDITOR        S DISK SPEED

Q QUICK SCAN DISK    C CERTIFY DISK

U 16 SCTR UTILITIES X EXIT / REBOOT


CTRL-Z SCREEN PRINT ESC ABORT/RESTART
```

[E] ERASE DISK

This utility is used to erase all the data from the tracks you specify on the target disk.

[E]

Pressing 'E' from the Locksmith main menu will take you to this utility. THIS UTILITY DESTROYS DATA ON THE TARGET DISK.

You should place the disk you wish to erase in the drive of your choice. Locksmith will prompt you for the drive of the target disk, the starting track number and the ending track number you wish erased, and the increment between tracks.

After you have specified the above information, Locksmith will prompt you to insert the disk to be erased. After pressing the spacebar, Locksmith will proceed to erase the specified tracks on the disk.

When the program is finished erasing the specified tracks, it will return you to the main menu. The status area is not cleared. Every track that was erased will have an 'E' in the status area.

Some disk protection schemes require that a track never have been used. The only way to accomplish this is to use either a new disk or erase the track on a used disk.

[S] DISK SPEED

This is a utility to allow you to set the speed of your disk drive. Normally disk drive speed changes from the optimum as it gets more use.

[S]

Pressing 'S' from Locksmith's main menu will put you into the set disk speed utility. Depending on which overlay you have in memory at the time, you may be asked to insert your Locksmith disk to load the program. The first question will be which drive number. Then you will be asked which of three types of speed check you wish. The first is to calibrate your drive to 300 RPM. This is the normal drive speed recommended by the manufacturer. The second choice is to calibrate your drive to the optimum drive speed. This speed is recommended for copying disks, since it runs slightly slower than normal. This helps to ensure that an entire track can be written, regardless of the drive speed of the original.

The third option is to calibrate your drive to the same speed as the drive on which the original disk was written. You will be prompted to insert your original disk in the drive. This is to allow Locksmith to determine the original drive speed. After inserting your Locksmith disk to load an overlay, you will then be given three choices for the graph scale. The screen you see is the one shown below.

```
HEX 0000000000000000111111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
                DISK SPEED

SELECT GRAPH SCALE:
1. FINE ADJUST (2.5%=70 UNITS)
2. MEDIUM ADJUST (5%=140 UNITS)
3. COARSE ADJUST (10%=280 UNITS)
```

You should normally choose fine adjust. The other two options are for disks that are so far out of adjustment that they can't be seen on the fine scale. After you choose the type of graph you wish, Locksmith will prompt for the number of samples you wish per plot. The screen you see below is the one you should now see.

```
HEX 0000000000000000111111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
                DISK SPEED

SELECT GRAPH SCALE:
1. FINE ADJUST (2.5%=70 UNITS)
2. MEDIUM ADJUST (5%=140 UNITS)
3. COARSE ADJUST (10%=280 UNITS)
1

ENTER NUMBER OF SAMPLES PER LINE
 (1-3) :
```

The choices you have on the above menu are used by Locksmith to determine how many times to check the speed prior to plotting a point on the hi-res screen. Normally you would select one sample per plot. This will check the drive speed once for every point it plots on the screen.

Next you will be asked to insert a blank disk in the drive you selected for the speed test. After you press the space bar Locksmith will write and read from track zero of the blank disk. **You must use a blank disk for this test since the data on track zero will be destroyed.**

[N] NIBBLE EDITOR

[ESC]

Pressing the 'ESC' key will always abort the current operation and return you to a menu.

[CTRL-Z]

This option is activated by pressing and holding the key marked 'CTRL' and pressing the 'Z' key. This option is a screen print. It will print whatever is showing on the text screen at the time it is pressed. This is assuming that you have a printer turned on and that Locksmith has been told the correct slot for the printer interface card.

CURSOR MOVEMENT:

Locksmith supports the normal Apple II cursor movement keys.

```
           UP
           [I]
LEFT [J] [K] RIGHT
           [M]
          DOWN
```

You may also move left or right with the left and right arrow keys.

If you have an Apple //e the up and down arrow keys are also operational.

If you move left past the beginning of the line, you will be placed on the last character of the previous line. Similarly, if you move past the end of the current line, you will be placed at the beginning of the next line.

[<]

Pressing the '<' key will move backwards through the nibble buffer one screen page, unless you are already at the beginning of the buffer.

[>]

Pressing the '>' key will move forward through the nibble buffer one screen page, unless you are already at the end of the buffer.

[,]

Pressing the ',' key will allow you to scroll continously back through the buffer until either a key is pressed or you reach the beginning of the buffer.

[.]

Pressing the '.' key will allow you to scroll continously forward through the buffer until either a key is pressed or you reach the end of the buffer.

CONTROL KEY COMMANDS:

[CTRL-R]

Pressing 'CTRL-R' will allow you to read a track into the buffer. You will be prompted with TRACK:. If you have previously read a track into the buffer, that track number will also be displayed. The current default drive for the track read will also be displayed. If you wish to reread the same track just press the RETURN key. If not then enter the number of the track you wish to examine. You may enter a decimal point in the track number. The track number you enter will be multiplied by four before it is stored internally. This is necessary due to the way Locksmith finds the tracks specified. After you have entered the track number you wish, press the RETURN key to tell Locksmith you are finished with the entry. The cursor will move to the drive entry to allow you to change the default if you wish. If you want the default drive, press RETURN. The first time you read a track into the nibble buffer Locksmith will recalibrate. If you wish to recalibrate at any other time enter CTRL-R and when the prompt TRACK: appears enter the track number, followed by 'R' and press the return key. This will force

Locksmith to recalibrate. The following screen is an example of the display after reading a track.

```
HEX 0000000000000000111111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00 N
.25
.50
.75
              NIBBLE EDITOR

0000-  (A7]  AC   AC   96   ED   D9   BF   BF
0008-   BE   CB   B4   B4   EE   B5   BB   BB
0010-   BE   BE   BD   E7   F2   F9   E5   FC
0018-   DC   96   D9   9E   EA   E5   E6   D6
0020-   D6   96   9B   A6   F6   F9   AC   B2
0028-   EB   B6   D6   FA   DA   CB   FC   AF
0030-   F9   B6   9F   F6   A7   FF   FB   DF
0038-   D3   A6   CD   AD   96   DC   AF   D6
0040-   B6   BF   D6   E6   B2   EC   AF   CD
0048-   AC   D9   E9   B6   9A   ED   FA   EF
0050-   D9   DC   E5   E7   DE   F5   CD   D7
0058-   AF   B6   DF   EF   E6   D7   EB   DE
0060-   A7   B9   F3   E5   F7   BE   CE   BF
0068-   DA   F5   B6   9A   9D   D9   F4   D6
0070-   F5   DD   DD   FD   F5   D6   D3   B7
0078-   E7   D9   EB   DC   DD   EC   FD   D3
```

[CTRL-A]

After you have a track in the buffer you will probably wish to perform some analysis on the track. Entering 'CTRL-A' will allow you to perform analysis with one algorithm at a time and see the results. See the chapter refering to the Text Editor for an explanation of algorithms. After you enter 'CTRL-A' you will be prompted with ALG:. Enter the number of the algorithm you wish to perform and press RETURN. Next you will be prompted with PASS:. Locksmith needs to know the value you wish the algorithm to use. The default value will be displayed. If you wish to use the default value just press the RETURN key. If you do not wish to use the default value, type in the hex value you wish to use and press the RETURN key. If the algorithm is unsuccessful, Locksmith will beep and display an inverse 'FAILED' at the upper right corner of the nibble buffer. You may repeat this procedure for as many algorithms as you wish.

[CTRL-S]

Entering a 'CTRL-S' tells Locksmith you wish to perform the current set of analysis algorithms on the track in the nibble buffer. All the current algorithms will be displayed as they are performed by Locksmith.

[CTRL-W]

Entering a 'CTRL-W' tells Locksmith to write the current track back to disk. You will be prompted with TRACK:. Enter the number of the track you wish to be written to the disk followed by RETURN. Pressing return will write the data to the current track that was read. Next the cursor will be placed on the default drive number for the write. If you wish to use the default drive just press RETURN. If you wish to change the default, enter the number of the drive you wish to write to.

WARNING! IF NO ANALYSIS HAS BEEN DONE ON THE TRACK TO SET THE TRACK START AND TRACK END, LOCKSMITH WILL ATTEMPT TO WRITE THE ENTIRE BUFFER.

[CTRL-V]

This command is used to tell Locksmith where to start verifying the track start after it writes the track to the disk. The series of bytes that follow the verify start are the ones that are checked when the track is written to disk. This is done to make sure that the beginning of the track was not overwritten and destroyed by the end of the track. Normally, if the verify bytes are overwritten the track will be shortened and rewritten until they are not overwritten or until the track can no longer be shortened.

If the track can no longer be shortened you will get a verify error. This error may possibly be corrected by adjusting the copy drive to a slower speed prior to writing the track.

To set verify start, place the cursor on the nibble you wish to start verifying, and press CTRL-V. This will set the verify start to this location. There will be a 'V' displayed in front of the nibble you selected for verify start.

[CTRL-I]

This command is used to add nibbles to the current buffer. When you enter 'CTRL-I' the nibble that is at the current cursor location is duplicated and all the nibbles to the right are moved one position to the right.

[CTRL-D]

This command is used to delete nibbles from the current buffer. When you enter 'CTRL-D' the nibble that is at the current cursor location is deleted from the buffer and all the nibbles to the right of the cursor are moved one position to the left.

[CTRL-F]

This command is used to find different patterns of nibbles within the buffer. Enter CTRL-F, and you will see the prompt FIND:. The Find Command has four options.

> **[D]**
>
> Enter 'D' followed by a series of hex numbers seperated by spaces and followed by the RETURN key. Locksmith will start searching forward through the buffer from the current cursor position for this sequence of nibbles. When the sequence is found the cursor will be moved to the first nibble in the pattern within the buffer. If the pattern is not found Locksmith will print in inverse at the top right of the buffer 'NOT FOUND' and beep. If the pattern is found and you wish to repeat the

search then you would press 'CTRL-F' followed by the RETURN key. This tells Locksmith to repeat the last 'CTRL-F' search.

[L]

Entering 'L' will give you the prompt LENGTH:. You may now enter in a length from (1-F). This instruction tells Locksmith to start looking forward through the buffer for a pattern that matches the one that starts at the current cursor position and is LENGTH nibbles long. When the pattern is found, the cursor will be moved forward in the buffer to the first nibble of the matching pattern. If you wish to repeat the search from your present cursor position, type 'CTRL-F' and press the RETURN key. This repeats the last search again. If Locksmith is unsuccessful in its search for the pattern, the cursor will not move and Locksmith will print in inverse at the top right of the buffer 'NOT FOUND' and beep.

[P]

After you press 'P' Locksmith will prompt with PAT:. There are seven general purpose patterns. When you enter a number between 1 and 7 in answer to the prompt, Locksmith will take the appropiate pattern and use that as the search pattern. The results are the same as described for the two previous commands. The 'CTRL-F' RETURN command to repeat the search is not available.

[O]

Entering 'O' tells Locksmith to search for the first nibble in the buffer that is different from the one the cursor is presently on. When a different nibble is found, the cursor will be placed on it. This command would be used if you

had a track that was almost all the same value. Instead of searching slowly through the buffer for the different nibbles, you could use this command to quickly locate them.

[CTRL-P]

This command is used to change parameters from within the Nibble Editor. It has the following options available. After pressing 'CTRL-P' you will get the prompt PARM:.

PARM:
Enter 'CTRL-R'and press the RETURN key. This will restore all default parameters.

PARM:

Enter '?' and press RETURN. This will display the valid parm names.

PARM:

Enter <name> and press RETURN. This tells Locksmith you wish to change the named parameter. You will be shown the current value of the parameter. You may type in a new value and press RETURN or simply press RETURN to accept the current value.

PARM:

Enter <hex value>, values (0-$1FF) are valid, and press RETURN. This is an alternate way to specify a parameter. The results and options are the same as with <name> above.

PARM:

Enter '+' and press RETURN. This tells Locksmith you wish to change the next parameter in sequence. The current value will be displayed. The options and results are the same as the two previous commands.

PARM:
Enter <return>. Exit parameter change mode.

AFTER PARM HAS BEEN ENTERED YOU HAVE THE FOLLOWING OPTIONS FOR VALUE.

VALUE:
Enter <return>. Accepts displayed current value.

VALUE:
Enter <hex> and press RETURN.

VALUE:
Enter <hex hex hex ... > and press RETURN. This enters the hex values into memory starting at PARM and continuing in sequence until the RETURN is encountered.

VALUE:
Enter <track number with decimal point> and press RETURN.

NOTE: NO CHECK IS MADE TO SEE IF THE PARM ACTUALLY REPRESENTS A TRACK VALUE. SINCE THE TRACK VALUES ARE STORED DIFFERENTLY THAN NORMAL PARM VALUES, USING A DECIMAL POINT WHEN ENTERING NORMAL PARMS WILL STORE INCORRECT VALUES.

MISCELLANEOUS COMMANDS:

[CTRL-B] Moves the cursor to track start. If the cursor is at track start, the cursor is moved to beginning of buffer.

[CTRL-E] Moves the cursor to track end. If the cursor is at track end, the cursor is moved to the end of buffer.

[(]
Sets track start to current cursor position.

[)]
Sets track end to current cursor position.

[S]
Sets the nibble under the cursor to self sync.

[N]
Sets the nibble under the cursor to normal.

[C]
Change mode. Enter <hex hex ...> and RETURN to exit change mode. Changes nibbles under the cursor to the hex values entered. Pressing the space bar moves the cursor to the next position. The commands 'S' and 'N' also work in change mode.

[H]
Entering 'H' will display the current buffer on the Hi-Res screen.

[HG]
Entering a 'G' while in Hi-Res mode will print the Hi-Res screen if you have a printer capable of Hi-Res graphics and a graphics printer interface card. The printer string required by your printer may be defined by the parm 'GRCHARS'. The default is set to CTRL-I G <CR> CTRL-Q <CR>. This string works for both Silentype

and Epson printers with interfaces that support graphic screen dumps.

[G]

Entering a 'G' from the text mode of the Nibble Editor will display a picture of the buffer using text characters. Each location on the screen represents a string of nibbles in the buffer. The length of the string (sample interval) is defined by the parm 'TSAMP', and defaults to $0A. (Note: for 13 sector disks, a value of $0C works well). On the graphic display the following symbols are used. A period '.' means that all the nibbles in the string are normal (non-self sync). An inverse '#' means the nibbles are all self sync. The '+' means the nibbles are a combination self sync and normal.

The cursor may be moved within the screen area using the I,J,K,M keys or the arrow keys. The cursor may be moved to the location on the screen corresponding to the area in the nibble buffer that you wish to examine. Pressing any other key at this point will return to the nibble display with the cursor set to the area you selected.

HINT: The display starts at either buffer start or track start and the display may not be centered. Move the cursor to the center of the screen and press the RETURN key. It will take you back into the Nibble Editor. Set track start '(' and press 'G' again. The display will now be centered. If the disk you are examining is a 16 sector disk, you will see one pattern of '#'s that is larger than the others. This is the field of self sync in front of sector zero. Move the cursor to the first period '.' following the large number of '#'s and press RETURN. You will now be back in the Nibble Editor and the cursor should be near the first nibble in the address header for sector zero. The following is an example of the display you will see when this command is used.

```
HEX 0000000000000000111111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00 N
.25
.50
.75
          NIBBLE EDITOR
.................+#+++..................
................+#+++...................
...............++.+.....................
..............###########+.++...........
.......................#+.+.............
.....................+#+++..............
....................+#.++...............
...................++.+.................
.................+#+++..................
................+#+++...................
...............++.+.....................
.............+#+++......................
............+#+++.......................
...........++.++........................
..........#+.+..........................
........+#+++...........................
```

[D]

This is a 16 sector address decode command. You will see two columns displayed on the screen. They are decoded in the following manner.

The first four numbers in inverse are the buffer address. Next is the letter 'V' followed by a hex number. This is the volume number of the disk. Next is a two digit hex number followed by a '/' followed by another two digit hex number. This is the track number/sector number. This field may be followed by any of these three symbols, '?','CS','**', or if nothing is wrong it will be followed by a blank space. They have the following meanings:

The '?' means that either the check sum or the trailer was incorrect in the address field.

The 'CS' means the data field checksum is bad.

The '**' means there is something wrong with the data field information. It is either a bad data field header or trailer. If the disk is a 13 sector format, '**' will appear for all sectors.

As mentioned above, '**' indicates that either the data field header (D5 AA AD) or data field trailer (DE AA) is incorrect. If they both appear correct, but are still marked with '**', the trailer is probably in the wrong location. Exactly 343 nibbles should occur between the header and the trailer. (342 data nibbles and one checksum nibble).

A simple way to test this is to perform the following.

Place the cursor on the D5 of the header field (D5 AA AD). Now, press '>' (shifted) 3 times, 'I' 5 times, and 'K' twice. The cursor is now where the trailer should start. If the cursor is not on the DE of the trailer (DE AA), the trailer is not in the correct location.

The address field nibbles occur in double-nibble format after the address field header (D5 AA 96) and represent the volume number, track number, sector number, and checksum. A chart to decode the address field double-nibbles is located in the appendix.

The data field nibbles consist of 342 data nibbles after the data field header (D5 AA AD) plus an additional 343rd nibble, which is used for the data field checksum. This checksum is calculated by taking each of the 342 data nibbles, translating them according to a chart (which is supplied in the appendix), and exclusive-or'ing them together to form a checksum. The resulting checksum is then reverse translated using the same table and becomes the 343rd nibble. Note that only 64 different nibbles are present in this table. Data fields are validated

only by this 6-bit checksum, and data nibbles each contain only 6-bits of information each.

If the disk is using a non standard address or data header you will not receive this information unless you set PARM:SECAF to the correct address field pattern and PARM:SECDF to the correct data field pattern.

The following is an example of the display you should see.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
              NIBBLE EDITOR
0652  VFE  00/00        07D5  VFE  00/07
0958  VFE  00/0E        0ADB  VFE  00/06
0C5E  VFE  00/0D        0DE1  VFE  00/05
0F64  VFE  00/0C        10E7  VFE  00/04
126A  VFE  00/0B        13ED  VFE  00/03
1570  VFE  00/0A        16F3  VFE  00/02
1876  VFE  00/09        19F9  VFE  00/01
1B7C  VFE  00/08        1CFF  VFE  00/0F
1EEA  VFE  00/00        206D  VFE  00/07
21F0  VFE  00/0E        2373  VFE  00/06
24F6  VFE  00/0D        2679  VFE  00/05
27FC  VFE  00/0C        297F  VFE  00/04
2B02  VFE  00/0B        2C85  VFE  00/03
2E08  VFE  00/0A        2F8B  VFE  00/02
310E  VFE  00/09        3291  VFE  00/01
3414  VFE  00/08        3597  VFE  00/0F
           PRESS SPACE TO CONTINUE
```

If you look closely at the above display you will see that the sector numbers 0-F are present. If that is not the case on your display, then there is a sector missing.

[#]

Pressing the '#' key from within the Nibble Editor prints the current track in the buffer from '(' track start to ')' track end to your printer. The self sync nibbles will have '*' on either side of them in the printout. The track verify start will have the letter 'V' in front of the verify start nibble sequence.

```
0630-  *FF**FF*(FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF*
0640-  *FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF*
0650-  *FF**FF*VD5  AA  96  FF  FE  AA  AA  AA  AA  FF  FE  DE  AA  EB
0660-  *FF**FF**FF**FF**FF**FF**FF**FF* D5  AA  AD  B6  DB  DC  F4  F3
0670-   BB  BD  CF  97  9A  AE  AE  96  AD  AC  9A  AB  97  B2  B2  AD
0680-   AB  9A  9B  AB  9F  97  B3  9A  B3  9E  97  9F  B3  96  AC  AE
0690-   A7  9A  EB  ED  CB  E6  9B  B2  A6  DC  CF  EE  AC  A7  E5  D7
06A0-   9F  A6  DC  97  BA  F7  EA  9E  AE  9E  AE  9F  9B  97  9B  B2
06B0-   AF  B3  AE  AC  9A  B3  B2  AC  A7  AC  97  AB  AB  BA  F4  97
06C0-   A6  DC  E5  D6  FB  ED  F5  EF  EC  DA  BD  96  96  96  B4  EF
06D0-   B5  EB  DC  FD  F5  EB  AB  EA  B9  FD  BE  DB  FD  D7  CD  FD
06E0-   E5  B9  FD  B2  AB  E6  FC  B5  DA  EB  FC  AE  FD  E5  B9  FD
06F0-   DA  DF  FA  AE  FD  E7  DA  B5  B9  B3  FB  9D  FD  F9  9D  FD
0700-   AC  E6  CE  F6  E9  CB  F6  9B  F4  BC  DA  B5  DB  FD  9A  9B
0710-   97  96  9B  96  97  96  9B  96  97  96  9B  96  97  9B  9B  A7
0720-   B5  B4  DC  EB  DF  E6  E6  AB  F3  BF  E5  F3  FB  BF  FC  CF
0730-   DE  FD  D9  FC  BF  EE  FE  F9  B9  D9  F9  F3  FB  B9  D9  DE
0740-   E6  DA  F3  BF  CF  9A  BD  E6  AE  D9  9E  D9  DA  97  EF  EF
0750-   CB  AE  BC  D6  E9  DF  FC  DB  EE  F6  9A  F4  EE  AB  BA  EB
0760-   E6  FC  F5  B3  DB  FF  9A  F6  F2  AB  EB  FC  DD  D7  BC  CD
0770-   F3  BF  F9  AB  FD  96  96  96  96  96  96  96  96  96  96  96
0780-   96  96  96  96  96  96  96  96  96  96  96  96  96  96  96  96
0790-   96  A7  CF  E5  BA  BF  F7  F2  9D  BA  E6  A7  FE  AB  BA  97
07A0-   DA  F2  9D  BA  E6  A7  FE  AB  DE  BD  96  96  96  96  96  96
07B0-   96  96  96  96  96  96  96  96  96  96  96  96  96  96  96  AF
07C0-   B3  9A  DE  AA  EB)*FF**FF**FF**FF**FF**FF**FF**FF**FF**FF**FF*
```

TEXT EDITOR MENU

The text editor is used to enter a series of Locksmith Programming Language commands to allow custom tailoring of the copy program for specific needs. The files created by the text editor can be saved for later use in the form of standard DOS text files. In addition, patches may be applied to the Locksmith disk using the text editor.

The Locksmith text editor is not a general purpose text editor and is not designed for word processing use. Files used by the Locksmith text editor are limited to 39 characters per line, and 255 lines. If a file which was created by another text editor is loaded which exceeds the above limits, the lines or file will be truncated to 39 characters and 255 lines.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
              TEXT EDITOR

L LOAD FILE           A APPEND FILE

S SAVE FILE           N NEW FILE

C CATALOG             X SYNTAX CHECK

1 DRIVE 1             D DELETE FILE

2 DRIVE 2             E ENTER EDITOR

B BACKUP/COPY DISK    P DISK PATCH


ESC RETURN TO MAIN MENU
```

The screen shown above is the text editor main menu. The commands are:

'L' loads a file from the selected disk.

'S' saves a file from the selected disk.

'C' displays the catalog from the selected disk.

'1' selects drive 1.

'2' selects drive 2.

'A' appends a file from disk to the end of the current file in the text editor memory.

'D' deletes a file from the selected disk.

'N' clears the text editor memory.

'E' enters the text editor. (see the chapter "Text Editor")

'B' invokes the BACKUP/COPY function after compiling the LPL commands in the text editor buffer.

'X' compiles and syntax checks the LPL commands in the text editor buffer, but does not actually invoke the BACKUP/COPY function.

[T] TEXT EDITOR

The Text Editor is used to enter a series of commands to Locksmith that may be saved to disk and recalled from disk for reuse. This will preclude having to reenter the commands manually when you wish to make another copy of a specific disk. You may also use any other text editor that stores its files as standard DOS text files. The Text Editor should normally be used because it will not allow command lines to be longer than 39 characters. The maximum number of lines allowed in the Text Editor is 255.

TEXT EDITOR COMMANDS

[ESC]

Pressing the 'ESC' key at any time will take you back to the Text Editor main menu. If you press the 'ESC' key while on a line that has not been entered into memory by your pressing the 'RETURN' key, the line will be lost.

[RETURN]

Pressing 'RETURN' enters the present line into memory and moves the cursor to the beginning of the next line. The 'RETURN' key may be pressed at any position on the line. You do not need to be at the end of the line, it will still be entered into memory.

[left arrow]←

Pressing the left arrow key will move the cursor one character to the left without erasing the character it passes over. If you are at the beginning of a line, the cursor will move to the first character on the line directly above the one you were on.

[right arrow]→

Pressing the right arrow will move the cursor one character to the right without disturbing the characters it passes over.

[up arrow (APPLE IIE only)]↑

Pressing the up arrow will move the cursor up one line. If you had not entered the line you were on by pressing the 'RETURN' key, this will enter it into memory.

[down arrow (APPLE IIE only)]↓

Pressing the down arrow will move the cursor down one line. If you had not entered the line you were on by pressing the 'RETURN' key, this will enter it into memory.

[CTRL-K]

Pressing and holding the key marked 'CTRL' and pressing the 'K' key will move the cursor up one line. If the line you were on had not been entered into memory by pressing the 'RETURN' key, this will enter it.

[CTRL-J]

Pressing and holding the key marked 'CTRL' and pressing the 'J' key will move the cursor down one line. As above, the line you were on will be entered into memory.

[CTRL-I]

Pressing and holding the key marked 'CTRL' and pressing the 'I' key will allow you to insert a character or a line. If the cursor is on the first character of a line this will insert a line. If the cursor is on any character except the first one, you will be allowed to insert one (1) character.

[CTRL-D]

Pressing and holding the key marked 'CTRL' and pressing the 'D' key will allow you to delete a character or a line. If the cursor is on the first character of a line when you do this, the line will be deleted. If the cursor is on any character but the first one, the character will be deleted.

LOCKSMITH PROGRAMMING LANGUAGE

Locksmith Programming Language (LPL) can be used to write custom-tailored backup/copy procedures for disks that are difficult to copy using the standard Locksmith defaults. LPL provides commands to change parameters by name, and to invoke named algorithms. The algorithms perform buffer analysis functions, as well as some miscellaneous functions, and provision is made for error handling and looping.

LPL commands are entered in a normal text file. Each line in the text file starts a new command, but multiple commands may be entered on the same line by seperating them with a colon. The maximum line length for a command line is 39 characters. A command may not be continued on a second line.

The commands are of two types:

1. Parameter setting commands.
2. Algorithm or Processing commands.

Comments may be entered at any time. Comments begin with '*'. They may be added to the end of a line by preceding the comment with a '*'.

There exist within Locksmith, seven (7) general purpose status registers for passing status from one algorithm to another. The value of the status registers can be set to indicate FAIL, SUCCEED, or CLEAR. If, on an algorithm command line, the keyword 'STATUS' is found followed by a number from 1 to 7, the specified status register will be set to the status of the algorithm, either SUCCEED or FAIL, after the algorithm is performed. The STATUS keyword and its associated status register number must be the last non-comment keywords in the statement. All status registers are initially CLEAR (neither SUCCEED or FAIL), and can be cleared with the CLEAR STATUS command.

Conditional execution of parameter setting or algorithm processing can be done by starting the statement with the keywords IF FAIL x, or IF SUCCEED x, where 'x' is the status register to test. If neither keyword is found, the statement is an unconditional one.

In addition to testing the status registers, you can test the value of a parameter, and conditionally execute the statement in that way.

As mentioned earlier, statements can be parameter setting or algorithm processing. A parameter setting statement always contains the keyword 'SET', followed by the parameter name and one or more parameter values. For example, the following are valid parameter setting statements:

```
SET MINGAP 5
IF FAIL 3 THEN SET MINGAP 5
SET DISPL 0157
```

To allow for pattern matching within Locksmith, seven (7) general purpose pattern parameters, each 16 bytes long, are provided. They are PAT1 through PAT7. When setting these or any other 16 byte parameters, several values may appear on the same statement. For example:

```
SET PAT4 D5 AA 96
```

The above example would change the first 3 bytes of PAT4, and the remaining 13 bytes would remain unchanged. To set the remaining values to zero ("don't care" values in a pattern), follow the last value with a

slash '/' character. For example:

```
SET PAT4 D5 AA 96 /
```

To indicate "don't care" values in the middle of a pattern, code a question mark for each "don't care" position. For example:

```
SET PAT4 D5 AA 96 ? ? ? ? AA AA /
```

To replicate the last value given in the statement through the end of the pattern, code 3 periods. For example:

```
SET PAT4 D5 AA 96 AA ...
```

In the above example, the rest of the pattern would be filled with AA's.

Some algorithms require that certain values in the specified pattern be "flagged", to set them aside from other values in the pattern. To "flag" a value of the parameter, enclose the value in parentheses. For example:

```
SET PAT1 D5 (AA) AD /
```

The above example "flags" the AA value. If this SET command were to be used before the algorithm command

```
CHANGE PAT1 TO SS
```

the flagged value (AA) would be changed to self-sync.

When setting the value of parameters which represent track values, code a decimal point with the number. For example:

```
SET SYNCTRK 3.0
```

Note that the statement
SET SYNCTRK 3
is NOT equivalent!

To leave the start of a pattern unchanged, and begin setting values in the middle of the pattern, code '+', followed by a hex value from 1 to F. For example:

```
SET PAT4 + 2 B5
```

The above example would change the value of the 3rd byte (displacement +2) in the pattern.

This technique may be necessary to change the entire 16 bytes of a pattern to specific values, because the statement can have a maximum length of 39 characters. For example:

```
SET PAT7 D5 AA AD 97 AD 96 DD FF
SET PAT7 + 8 FD FE FF AD DD FF ED FF
```

Some parameters are 16-bit (2-byte) values. These can be coded in two ways. For example, to set the 2 byte parameter DISPL to the value 157 (hex), either of the following statements would work:

```
SET DISPL 0157
SET DISPL 57 01
```

Note that in the second example, that the numbers are reversed.

The format for algorithm commands is dependent upon the individual algorithm being used. See the chapter "Algorithms" for a complete description of each algorithm.

PATCHING THE LOCKSMITH DISK

Locksmith Programming Language makes Locksmith very flexible. It is designed to have the capability to copy virtually every protection technique in current use, and many that have not yet been introduced. However, new techniques may be developed which the current version of Locksmith cannot handle. In addition, software bugs, not discovered during extensive testing procedures, may appear. For these reasons, Locksmith has a built-in routine to allow the user to apply patches to the Locksmith program. These patches, if required, will be distributed by Omega MicroWare in one of the following forms:

1. update diskette

2. printed material

3. modem files

In any case, to apply the distributed patch, the user simply needs to LOAD the file containing the patches into the text editor, and press 'P' from the text editor menu to apply the patch to the Locksmith disk. The patches are verified to make sure they are entered correctly, and that the patches are not already applied. Then the Locksmith master disk is updated to reflect the patches. This means that small revisions can be provided to the registered Locksmith user without the need for returning the Locksmith diskette to Omega for updating.

No provision is made for Locksmith to download files from modem, but after using any popular modem file download program, Locksmith can use the resulting text file without the need for entering the data manually. It is expected that all Locksmith patch files will be made available on the Source information utility, and perhaps other major systems.

After pressing 'P' from the text editor menu, if the bell sounds and the editor is entered, an error was encountered by the patcher. The cursor will be on the line with the error. No patches are applied to disk until all of the patcher commands are verified.

16 SECTOR UTILITIES

This option gives you access to five utilities designed to work with normal 16 sector (generally unprotected) disks.

[U]

Pressing 'U' from the main Locksmith menu takes you to the 16 sector utility submenu. If the correct overlay is not in memory, you will be asked to insert your Locksmith disk and press the space bar to load the utilities into memory.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00
.25
.50
.75
            16 SCTR UTILITIES


      SELECT FUNCTION:

         V 16 SECTOR FAST DISK VERIFY

         B 16 SECTOR FAST DISK BACKUP

         F 16 SECTOR FORMAT

         C 16 SECTOR COMPARE

         S 16 SECTOR SYNC SIGNATURE
```

[V] 16 SECTOR FAST DISK VERIFY

Pressing 'V' from this submenu will take you to the FAST DISK VERIFY utility. You will be prompted for the drive number of the disk you wish to verify.

Locksmith will then proceed to read the disk from track 0 to track $22. You will see the following display.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00 ....................................
.25
.50
.75
             16 SECTOR FAST DISK VERIFY
    0123456789ABCDEF0123456789ABCDEF0123
 S0 ....................................
  1 ....................................
  2 ....................................
  3 ....................................
  4 ....................................
  5 ....................................
  6 ....................................
  7 ....................................
  8 ....................................
  9 ....................................
  A ....................................
  B ....................................
  C ....................................
  D ....................................
  E ....................................
  F ....................................
```

On the status display at the top of the screen, a period '.' means the track read correctly. An asterisk '*' means the track did not read correctly. A number represents the number of extra times the disk had to spin to read all sectors correctly. A number '1' for example means the track had to be reread once in order to read correctly.

Below the status display is the track/sector display. On the track/sector display the symbols have the following meaning.

A period '.' means the sector was read correctly on the first disk revolution. An inverse character 'A' means there was something wrong with the address field or the address field was missing. An inverse character 'D' means there was something wrong with the data field. Again, a number indicates that the sector was read correctly, but that it took several rereads to read it correctly.

```
HEX 000000000000000011111111111111112222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00 ...*...1.....11...**......1..1......
.25
.50
.75
             16 SECTOR FAST DISK VERIFY
    0123456789ABCDEF0123456789ABCDEF0123
 S0 .......1.....1.....A.........1......
  1 ..................AA................
  2 ..................AA................
  3 ..................AA................
  4 ...A..............AA......1.........
  5 ...A..............AA......1.........
  6 ...A...1......1....A................
  7 ...D...1...........A.........1......
  8 ..................AA................
  9 ..................AA................
  A ..................AA................
  B ...A..............AA................
  C ...A..............AA......1.........
  D ...A..............DA................
  E ...A...1......1....A................
  F ..................AA.........1......
```

[B] 16 SECTOR FAST DISK BACKUP

Pressing a 'B' from this submenu will take you to the 16 SECTOR FAST DISK BACKUP. This utility is used to copy standard disks very quickly. If you have two drives and copy from one to the other, it takes approximately 19 seconds to copy the disk. This time can be be even shorter if you have RAM boards (language cards) in your Apple.

Locksmith will automatically search for and use any combination of RAM cards in your Apple. The screen display will show the slot and the amount of memory in each ram card it finds during its search. A complete disk requires 140K of memory (4K per track). 40K of main memory is used to allow storage for 10 ($0A) tracks. Each 16K card found will allow 4 additional tracks to be stored. If 100K or more in total RAM board space is found in addition to the 40K used of main memory, the program will allow one-pass disk copies. That is, the entire disk will be read into memory and can be written many times without reading the original disk again. This is very useful for mass duplication of disks for clubs or software manufacturers.

If an Apple //e is used, 16K RAM is built-in, in addition to the 48K main memory. If the Apple //e contains an 80-column auxiliary card with 64K RAM, only 56K of the 64K auxiliary card will be used. If an Apple //e is used with 64K auxiliary card, an additional 32K card in any other slot would allow the user to make single-pass copies. (16K + 56K + 32K — 104K) Note that ALL slots are searched for RAM cards, including SLOT3, which is not normally available on an Apple //e with auxiliary card installed. If a RAM card is installed in slot 3 on an Apple //e with auxiliary card, it will be used by Locksmith FAST DISK BACKUP, even though other software may not be able to access it.

Locksmith FAST DISK BACKUP is the FASTEST Apple copy program with or without the use of RAM boards.

Locksmith will read and write a disk without RAM boards in 19 seconds, copying 10 tracks per pass. If verifying after each write, the disk is copied in 26 seconds.

The following table summarizes timing tests done with some popular copy programs without the use of RAM boards:

Program	trks/ pass	time to copy	time to copy & verify
Locksmith 5.0	10	19	26
Penulta Copy	5		38
Disk Muncher	7	26	
Pack Rat	4	35	
Apple COPYA	8	88	

Also note that Disk Muncher and Pack Rat do NOT validate checksums during read, and are thus extremely unreliable.

If RAM boards are found to total at least 100K (128K RAM boards work fine), the disk can be read in 8 seconds, and a copy disk written in 8 seconds. If verify-after-write is desired, the disk is written in 15 seconds.

The following table summarizes timing tests done with some one-pass copy programs with the use of 128K RAM boards:

Program	time to read	time to write	time to write & verify
Locksmith 5.0	8	8	15
CopyWriter	24	16	23
Copy Cruiser	9	16	23

Note that CopyWriter also has a 'read-twice' mode which takes 45 seconds to perform instead of 24 seconds, but can be more reliable on original disks recorded on questionable media.

When entering this routine, you will be prompted for the original and copy drives. Enter a '1', a '2' or RETURN (to accept default) in answer to these prompts. After inserting your disks, you will enter the backup routine. Once the FAST DISK BACKUP routine is entered, it will be necessary to reboot if you wish other Locksmith functions, as the FAST DISK BACKUP function uses all available memory for disk buffers.

From within the backup routine, the following commands are available:

Function selection commands:

```
[12]   Copy drive 1 to drive 2
[21]   Copy drive 2 to drive 1
[11]   Copy drive 1 to drive 1
[22]   Copy drive 2 to drive 2
[10]   Copy drive 1 to memory (if enough RAM is  available)
[20]   Copy drive 2 to memory (if enough RAM is available)
[01]   Copy memory to drive 1 (if memory previously loaded)
[02]   Copy memory to drive 2 (if memory previously loaded)
[1]    Verify drive 1
[2]    Verify drive 2
[V]    Toggles verify-after-write mode
```

Other commands:

[ESC] exit FAST DISK BACKUP

[*] Allows a comment statement on screen (useful with screen print)

[RETURN] OR (SPACE) Start current copy or verify function

While the copy operation is in progrss, you may press [V] to toggle the verify-after-write mode. Verifying after writing will increase the time it takes to write an entire disk from 8 seconds to 15 seconds.

[F] 16 SECTOR FORMAT

Pressing 'F' will allow you to format a disk. This utility will format a disk or a range of tracks with the volume number you specify. This could be very useful if a track had been destroyed accidently. In this case the disk would normally be unusable. However with this utility you could simply reformat the one track and use the disk.

NOTE: IT WILL NOT RECOVER DATA THAT WAS ON THE DESTROYED TRACK.

You will first be prompted for the disk drive you wish to use. Press either '1' or '2', depending on which drive you wish to use. You will then be prompted for track start, end and increment. Specify the tracks you wish reformatted. You will next be prompted for Volume Number. Specify the volume number that you wish to format in the Address Field. Following the Volume Number, you will be asked which Track Numbers to use. Normally, the defaults will be used. Since it is possible that you may at some time wish to use a non-standard format, this is left up to the user. For example, some disks currently on the market use tracks $06.5 through $22.5. These tracks are formatted with track numbers $06 through $22.

[C] 16 SECTOR COMPARE

Pressing a 'C' from this submenu will allow you to compare two disks for differences. When you select this option you will be asked for the drive number of the disk you wish to compare. This routine stores a double-byte (16-bit) checksum for each sector in memory and compares it to the one already there. If they don't match you will get a 'C' on the sector number display. When reading the first disk to compare, it is normal to get many 'compare errors' because incorrect sector checksums are initially in memory.

After the disk you wish to compare is read into memory, take it out of the drive and replace it with the disk you wish to compare to. Press the space bar to begin compare, and any sector that matches will have a period '.' on the sector display for that sector. If a sector doesn't match there will be a letter 'C' on the sector display. The checksums that were there from the original disk have now been replaced by those of the disk you just compared, so if you pressed the space bar again without removing the disk just compared you should get all periods '.' on the sector display. In addition to the '.' and 'C', you may also find inverse 'A' and inverse 'D'. These indicate address field and data field errors, respectively.

The following is a display of two disks that were not the same.

```
HEX 00000000000000001111111111111111 2222
TRK 0123456789ABCDEF0123456789ABCDEF0123
.00 ...................................
.25
.50
.75
           16 SECTOR COMPARE
    0123456789ABCDEF0123456789ABCDEF0123
 S0 ...CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  1 ...CCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  2 ...CCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  3 ...CCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  4 ...CCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  5 ..CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  6 ..CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  7 ..CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  8 ..CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  9 C.CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  A ..CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  B ..CCCCCCCCCCCCCCC.CCCCCCCCCCCCCCCCC
  C ..CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  D ..CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  E ..CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
  F ..CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

[S] 16 SECTOR SYNC SIGNATURE

This utility is used to obtain a signature of the sync pattern on a normal 16 sector disk. This can be useful when working on synchronized disks. You will be prompted for the disk drive you wish to use. Enter either '1' or '2'. This routine starts at track $00 sector $00. After reading this sector, it moves to track 1, and displays the first sector number that it encounters. This continues until all $22 tracks have been checked.

The sync signature is displayed again and again. It may be slightly different due to disk speed variations from time to time. To terminate the sync signature routine, press ESC.

In addition to checking synchronization, this routine can be used to determine what copy program created a given 16-sector diskette.

A sync signature on a 16-sector diskette will normally show a progression of hex numbers, either ascending or descending. For example, the following progression shows hex numbers descending by one:

0FEDCBA9876543210FEDCBA9876543210FE

We will identify the above progression as (-1), to indicate that each hex digit is one less than the one before. The following table shows several copy programs, and the progression identifier that identifies the sync signature for a disk which was created by that specific copy program. Note that simply writing data to a disk will not change its sync signature - the disk must actually be formatted or generated by a copy program that formats the disk.

Program signature identifier

Program	signature	identifier
DOS INIT	0DA741EB85...	(-3)
Locksmith 'U' 'F'	048C048C...	(+4)
Locksmith 'U' 'B'	0FEDCBA987...	(-1)
Penulta Copy	0ECA8...	(-2)
CopyWriter (no verify)	0000000....	(+0)
CopyWriter (verify)	0FEDCBA987...	(-1)
Disk Muncher	0D852FC964...	(**)
Pack Rat	0D85630DA5...	(**)

Note that the signatures will have the same progression throughout the signature only if the disk was recorded in a single pass without turning the drive off between tracks. For example, the following signature was generated by Locksmith FAST DISK BACKUP in a single pass copy (with 128K RAM board):

```
0FEDCBA9876543210FEDCBA9876543210FE
```

Note that each hex digit is one less than the previous digit. This is in agreement with the table shown above. Now compare the following signature:

```
0FEDCBA9873210FEDCBA6543210FED98765
          ^         ^         ^
```

Note that the progression is the same (-1), except that at 3 places (identified by arrows below the signature), where the progression is instead (-4). This is because this signature is from a disk that was created by Locksmith FAST DISK BACKUP running without RAM cards. Notice that every 10 tracks, a break in the progression occurs. This is because without RAM cards, FAST DISK BACKUP copies 10 tracks at a time, and then reads the original disk again. So only groups of 10 tracks remain in sync with each other. Incidentally, the break in the progression was (-4). If the disk was created by Locksmith FAST DISK COPY with verify after write, the break in the progression would have been (-3). As you can see, a lot can be determined about a disk using sync signature.

[I] INSPECTOR/WATSON

[I]

If you have previously booted with a DOS disk that loads the Inspector/Watson onto a ram card or if you have a firmware card with these programs on it then pressing 'I' from the main Locksmith menu will place you in either the Inspector or Watson.

The Inspector/Watson program works exactly as documented in the respective manuals, with the following exception. When Inspector/Watson is given control from Locksmith, the default buffer address will be $4000 instead of $0800. This is because $0800 is reserved for Locksmith use.

If you only have the Inspector, control will be passed to the Inspector. If you have both the Inspector and Watson then control will be passed to Watson. All the normal program commands are useable. To exit Inspector/Watson press 'CTRL-C' and you will be back in Locksmith.

If your Inspector/Watson resides on a RAM board in slot 0, the ESC key will also return you to Locksmith. If your Inspector/Watson is in ROM, you must use 'CTRL-C' to exit.

For instructions on how to use Inspector and Watson refer to their respective documentation. These programs are available from Omega Microware, Inc.

Some notes you might find useful, while using INSPECTOR/WATSON from within Locksmith follow.

When reading a disk with many soft errors (errors which are temporary, and if read again, may succeed), it may be desirable to inhibit the recalibration of the seek mechanism. This will eliminate the constant 'burping' sound the disk drive makes before an I/O error occurs. Inhibiting the recalibration prevents the read operation from terminating with an I/O error. The following patch can be applied while in INSPECTOR, and should be removed before exiting INSPECTOR. Note that after applying this patch, that a read operation to a sector with a permanent I/O error (hard error), will never terminate. For this reason, this patch should only be applied if soft errors occur while reading a disk using INSPECTOR. If a read operation fails to terminate because of a hard error on the disk, the operation can be terminated by removing the disk and inserting a known good disk in the drive.

Also note, that unlike 'patches' applied to the Locksmith disk using the patcher function of the TEXT EDITOR, the patch given here is not applied to the Locksmith disk, and will automatically be removed upon reboot.

To apply the patch, first read a sector using INSPECTOR. Any sector will do. The purpose of this is to insure that the seek mechanism is on the correct track. Then, change the buffer to $BD00 using the 'B' command. (press 'B','B','D',RETURN) Then, change data within the buffer using the following keystrokes:

E C C space 4 C space C 1 space B D return

Then change the buffer back to $4000 (press 'B','4','0',RETURN)

To remove the patch, set the buffer to $BD00 again and restore the data using the following keystrokes:

E C C space 1 0 space F 3 space A D return

If you are not sure that you did this correctly, simply press CTRL-RESET, and reboot Locksmith.

[/] CLEAR STATUS

[/]

The status display at the top of the screen is not cleared after each Locksmith function, so that the user can use the status display with other functions. To clear the status display, press the '/' key from the main menu.

[CTRLZ]

[CTRLZ]

Pressing 'CTRL-Z' at any time will print the text screen to a printer. The printer should be turned on and enabled. Locksmith assumes that the printer interface is installed in Slot 1, but may be changed to any slot by changing the parm 'PRSLOT'. The parm 'PAUTOCR' is used to select whether or not Locksmith will send a carriage return at the end of each line. The default is $00. Any other value will cause a carriage return to be sent.

[X] EXIT/REBOOT

[X]

Pressing the 'X' key from the main menu will exit Locksmith.

You will be prompted to insert the disk you wish to boot and to press the space bar. Pressing the spacebar will exit Locksmith and perform a normal boot.

STATUS DISPLAY CODES

Codes which may appear on the 4-line status display at the top of the screen are of two types: inverse characters, which represent activity codes, and normal characters, which represent status codes. Inverse characters are displayed to indicate that a Locksmith function is active on the track displayed with the activity code. Normal characters are displayed to indicate that a Locksmith function has processed that track, and the status code indicates the status of the last Locksmith function that processed that track.

STATUS CODES

Status codes (normal characters) which you may see in the status display are:

'.' indicates normal completion of a Locksmith function on a track. This means that the certify function, verify function, or compare function found no errors on the track.

'*' indicates that an error occurred on the track. The certify function was unable to certify the track, or the verify or compare function found one or more unreadable sectors on the track.

'1' to '9', when displayed during a verify function, indicate that the track read correctly, but only after the indicated number of rereads.

'1' to 'F', when displayed during the backup function, indicate that the displayed error code was generated while processing the track. See chapter concerning error codes.

'0' indicates that the track was copied with no errors during a backup function.

'E', when displayed during the erase disk function, indicates that the track was successfully erased.

'F', when displayed during the 16-sector utility format function, indicates that the track was successfully formatted.

ACTIVITY CODES

Activity codes (inverse characters) which you may see in the status display are:

'R' indicates the track is being read.

'W' indicates the track is being written.

'V' indicates the track is being verified or compared.

'S' indicates the track is currently being synchronized before read or write.

'F' indicates the track is being formatted, or during copy, nibbles are being fixed to shorten the track or lengthen the track.

'A' indicates the track is being analyzed after read.

'E' indicates the track is being erased.

'C' indicates the track is being certified or, during copy, nibbles are being counted.

'O' indicates that FAST DISK BACKUP is waiting for the motor to come up to speed.

'N' indicates the nibble editor is processing the track.

DOS ERROR CODES

The following DOS error codes may appear during use of the TEXT EDITOR file management functions. They are described in detail in the DOS manual.

'04'	indicates that the disk is write-protected.
'06'	indicates that the specified filename was not found.
'08'	indicates that a DISK I/O error occurred. The disk drive door may have been left open.
'09'	indicates that the disk is full.
'0A'	indicates that the file specified is locked. Save with a diffeent filename, or unlock the file using standard DOS.

PARM KEYWORDS AND THEIR DEFAULT VALUES

The valid parameter keywords and their meaning are described in this section of the manual. Note that parameters may be changed in 3 ways:

1. Nibble editor

2. Parameter editor

3. Locksmith Programming Language

Parm numbers, because they may change in the future, are not listed here, but can be determined by using the parameter modifier or nibble editor, and selecting a parameter by name. The associated parameter number will be displayed.

Because future versions of Locksmith may have different default parameter values, the defaults will not be listed here, but may be determined by using the parameter modifier or nibble editor.

Some parameters may be defined within Locksmith and are not described in this manual. These are parameters which are used for test purposes, or ones which are not fully operational. Use these at your own risk.

PAT1
PAT2
PAT3
PAT4
PAT5
PAT6
PAT7 are 7 general purpose pattern parameters. Each is 16 bytes in length. These are used by pattern-matching algorithms.

BITTAB is a 16-byte parameter which is used by algorithm 18 (bit lookup). For more information, see the section discussing algorithms in detail.

INVTAB is a parameter which consists of a 128-byte lookup table to determine invalid nibbles. If the corresponding value in this table is $00, the nibble is 'invalid'.

MAXERR is a 16-byte parameter which contains the maximum error counts for error codes 0 to F.

GRCHARS is a 7-byte parameter which contains the character string sent to the printer when it is desired to print the graphics screen.

SECAF is a 6-byte parameter which contains standard sector address field header and trailer nibble values. This parameter is used by the nibble editor 'D' command.

SECDF is a 5-byte parameter which contains standard sector data field header and trailer nibble values. This parameter is used by the nibble editor 'D' command.

PATMBK is a parameter which is used by the track image compare algorithm. See the description in the section of this manual describing algorithms.

SSLAHD is a parameter which contains the number of nibbles ahead of the current nibble being compared, for which self-sync is searched.

DSPCMP can be set to 00 if compare failures are not to be displayed.

MAXCOR is the number of 3rd image corrections allowed per track.

BIGTRK is the high-order byte of the maximum track size, in nibbles. Track sizes greater than this value are assumed to be in error.

TSIZMIH is the high-order byte of the number of nibbles to skip when searching for the 2nd track image in the buffer.

TSTLEN is the number of nibbles to compare with the 1st image, to determine that the 2nd image has been found.

SYNCPAT is the pattern number for the sync pattern.

LEADSS is the value of the lead-in self-sync nibble.

LEADFB is the number of framing bits (1 or 2) in lead-in self-sync.

DATAFB is the number of framing bits (1 or 2) in self-sync within the data buffer.

SLOT is the slot (1 through 7) for all Locksmith functions. Note that overlays will always be loaded from the boot slot and drive.

DRIVE is a 2-byte parameter which contains the from-drive and to-drive for copy operations.

SYNCTYP controls track-syncing. If it is set to $01, the previous track will be sync'ed to. If it is set to $00, a specific track will be sync'ed to. If it is set to $02, the current track will be sync'ed to.

SYNCTRK is the track number to sync to, if SYNCTYP is set to the value $00.

BDISPL is the buffer displacement for SYNCTRK.

MINSS is the minimum number of self-sync to be effective.

PRSLOT is the slot number of the printer to be used, if a control-Z (screen print) command is given.

PAUTOCR is set to 0 if no automatic carraige return is desired at the end of each line to the printer.

DLINSS is a 2-byte parameter which represents the number of lead-in self-sync for a non-sync'ed track.

SLINSS is a 2-byte parameter which represents the number of lead-in self-sync for a sync'ed track.

PATTOL is the tolerance value for finding the start of the 3rd image based on the 1st and 2nd images.

PATPFX is the length of the 3rd image pattern-match prefix.

CNTERR indicates how close to get during nibble counting. If set to $00, the count must match exactly.

VERLEN is the number of nibbles to verify at the start of track after writing to insure that the track was not overwritten.

DDH1,DDH2,DDH3 (default D5 AA AD) are 3 parameters which define the start of data field for the DOS data-field fix algorithm.

DISPL is a 2-byte parameter representing the displacement for pattern match.

SPAN is the range of values used by the CHANGE RANGE algorithm. This parameter is 2-bytes. The first is the minimum value, the 2nd is the maximum value.

PREMIN,POSTMIN,MINLEN,MAXLEN are described in the section on algorithms.

VALUE is a general purpose parameter which is used to pass a value to the CHANGE pattern to VALUE algorithm.

CHOPS is the number of times to shorten for each iteration of shortening.

TSAMP is a parameter which represents the length of the sample size for the single track status map produced by the 'G' command in the nibble editor.

QSAMP is a parameter which represents the length of the sample size for one pixel on the display created by the quickscan function.

MINGAP is the minimum gap size allowed by the shortening algorithms.

NOTIFY can be set non-zero to ring the bell continuously on completion of a backup/copy operation, to notify the user that the operation has completed.

ATRACE can be set to zero to inhibit the screen display of algorithm number tracing.

ALGORITHMS

Algorithms are the basic building blocks for the Locksmith backup routine. Each algorithm is a function that operates on the nibble buffer or performs some other control function. Which algorithms to use, and the order they are to be used, is defined by the Locksmith Programming Language (LPL) commands, which are specified in a text file, and 'compiled' by the user. Because of this, Locksmith backup is very flexible and can be adapted easily to almost any protection scheme.

The algorithms fall into several categories. They are listed in this section by the algorithm number assigned to them. Algorithm numbers range from \$01 to \$7F. They are categorized as follows:

General purpose	01-0F
nibble changing	10-1F
track start setting	20-2F
track end setting	30-3F
verify start setting	40-4F
track shortening	50-5F
reserved	60-6F
special purpose	70-7F

ALGORITHM NAME: ERROR

PURPOSE: ERROR HANDLER

ALG # [1]

SYNTAX: ERROR n

PARMS USED: MAXERR

PASS VALUE: error code

FUNCTIONAL DESCRIPTION:

MAXERR is tested if error code is \$0-\$F. If MAXERR is exceeded, then fail flag is set, otherwise succeed flag is set.

ALGORITHM NAME: TEXT

PURPOSE: SET TEXT MODE

ALG # [2]

SYNTAX: TEXT

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Sets page 1 text mode

ALGORITHM NAME: GRAPHICS

PURPOSE: SETS GRAPHICS MODE

ALG # [3]

SYNTAX: GRAPHICS

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Sets full screen page 1 hi-res graphics mode.

ALGORITHM NAME: CHANGE PATTERN TO PATTERN

PURPOSE: CHANGE 1 PATTERN TO ANOTHER PATTERN

ALG # [4]

SYNTAX: SET PATx to PATx

PARMS USED: PAT1 TO PAT7

PASS VALUE: PATTERN NUMBER TO CHANGE AND THE PATTERN NUMBER TO CHANGE TO.

FUNCTIONAL DESCRIPTION:

Changes all occurrences of the first specified pattern to the second specified pattern. I.E. CHANGE PAT3 TO PAT7.

ALGORITHM NAME: INSERT

PURPOSE: INSERT A 00 NIBBLE

ALG # [5]

SYNTAX: INSERT PATx

PARMS USED: PAT1 THRU PAT7

PASS VALUE: PATTERN #

FUNCTIONAL DESCRIPTION:

Inserts a 00 nibble at all occurrences of the specified pattern. The 00 nibble is inserted after the specified nibble.

ALGORITHM NAME: INSERT SS LONER

PURPOSE: INSERT A 00 NIBBLE

ALG # [6]

```
SYNTAX: INSERT SS LONER
        INSERT SS LONER target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync
```

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Insert a zero 00 nibble whenever it finds a self-sync loner. A self-sync loner is any self-sync nibble with a normal nibble on either side of it. The nibble is inserted after the self-sync nibble. If 'target' is specified, the 00 nibble is inserted only if the loner self-sync found matches 'target'.

ALGORITHM NAME: ANCHOR

PURPOSE: SET ANCHOR

ALG # [7]

SYNTAX: ANCHOR

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Sets a pointer to the current buffer location. This is used for further processing. I.E. Moving the track end or the track start forwards or backwards in the buffer will move from this point.

ALGORITHM NAME: NOP

PURPOSE: NO OPERATION

ALG # [8]

SYNTAX: NOP

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Always succeeds in case you wish to set a STATUS FLAG to succeed.

ALGORITHM NAME: COMPARE

PURPOSE: COMPARE TRACK IMAGES IN MEMORY

ALG # [9]

SYNTAX: COMPARE

PARMS USED: SSLAHD
PATTOL
PATMBK
MAXCOR
PATPFX
DSPCMP

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm compares the first track image in the nibble buffer with the second image, one nibble at a time. If the nibbles match, the compare continues. If they do not match (a soft-error has occurred), the third image of the track is checked, and used as a tie breaker. The nibble that is wrong is corrected, assuming the third image matches one of the other two. If the third image does not match either of the other two, then a 4 error will occur.

ALGORITHM NAME: NIBED

PURPOSE: invoke NIBBLE EDITOR.

ALG # [0A]

SYNTAX: NIBED

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

invokes NIBBLE EDITOR. When ESC is used to exit nibble editor, control is returned to backup routine.

ALGORITHM NAME: TEST1FB

PURPOSE: TEST SELF-SYNC FRAMING BITS

ALG # [OB]

SYNTAX: TEST1FB

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Tests to see if a track has one or two framing bits in the self-sync fields.
Suceeds if it finds one framing bit in the self-sync nibbles.

COMMENTS: This algorithm works only if RDTYP is set to 2.

ALGORITHM NAME: FIX SS DHDR

PURPOSE: FIX DATA HEADER.

ALG # [OD]

SYNTAX: FIX SS DHDR

```
PARMS USED: DDH2 (DEFAULT AA)
            DDH3 (DEFAULT AD)
```

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Changes the AD in a standard data header to a normal nibble. Sometimes DOS writes the AD as a self-sync nibble. This algorithm looks for self-sync, then any normal nibble, then the values in DDH2 and DDH3. If this test is passed, then the value that matches DDH3 is set to normal.

ALGORITHM NAME: DISPLAY

PURPOSE: DISPLAYS TRACK START AND LENGTH

ALG # [OE]

SYNTAX: DISPLAY

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Displays track start and length.

ALGORITHM NAME: TRKMAP

PURPOSE: DISPLAYS TRACK MAP

ALG # [OF]

SYNTAX: TRKMAP

PARMS USED: TSAMP

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Displays track map from within the NIBBLE EDITOR. Used by the [D] command in the NIBBLE EDITOR. Each character on the screen represents a string of TSAMP length.

ALGORITHM NAME: CHANGE SS

PURPOSE: CHANGE SELF-SYNC NIBBLES

ALG # [10]

```
SYNTAX: CHANGE SS TO target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync
```

```
PARMS USED: PREMIN
            POSTMIN
            MINLEN
            MAXLEN

PASS VALUE: target
```

FUNCTIONAL DESCRIPTION:

Find PREMIN number of normal nibbles followed by at least MINLEN self-sync nibbles. The number of self-sync nibbles may not exceed MAXLEN. Then the algorithm looks for the minimum number of normal nibbles that are specified by POSTMIN.

```
I.E  (1) nnnnnsssssnnnnn
     (2) nnnnnnnssnnnnnnn
```

If the values for the parms are:
PREMIN=5 MINLEN=3 MAXLEN=6 POSTMIN=4

Then this algorithm will find pattern (1) but not pattern (2). After finding pattern (1) it would change the self-sync nibbles to the requested value.

COMMENTS: THIS IS A SELF-SYNC CLEANUP ROUTINE.

<u>ALGORITHM</u> <u>NAME:</u> CHANGE NORM

PURPOSE: CHANGE NORMAL NIBBLES

ALG # [11]

```
SYNTAX: CHANGE NORM TO target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync

PARMS USED: PREMIN
            POSTMIN
            MINLEN
            MAXLEN

PASS VALUE: target
```

FUNCTIONAL DESCRIPTION:

This algorithm functions exactly like algorithm 10 except that it is reversed. PREMIN and POSTMIN represent fields of self-sync nibbles, and MINLEN and MAXLEN represent normal nibbles.

COMMENTS: CLEANUP NORMAL NIBBLES IN THE MIDDLE OF A SELF-SYNC STRING.

<u>ALGORITHM</u> <u>NAME:</u> CHANGE PATTERN TO SS

PURPOSE: CHANGE A PATTERN

ALG # [12]

SYNTAX: CHANGE PATx TO SS

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE.

FUNCTIONAL DESCRIPTION:

Change any occurrance of PATx to self-sync.

<u>ALGORITHM</u> <u>NAME:</u> CHANGE PATTERN TO NORM

PURPOSE: CHANGE A PATTERN

ALG # [13]

SYNTAX: CHANGE PATx TO NORM

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE.

FUNCTIONAL DESCRIPTION:

Changes any occurrance of PATx to normal nibbles.

ALGORITHM NAME: CHANGE ALL

PURPOSE: CHANGE ENTIRE BUFFER

ALG # [14]

```
SYNTAX: CHANGE ALL TO target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync
```

PARMS USED: NONE

PASS VALUE: target

FUNCTIONAL DESCRIPTION:

Changes the entire nibble buffer to the target value.

COMMENTS: THIS MAY BE USED TO DO TESTING ON A DISK. IT IS ALSO A CONVENIENT WAY TO CLEANUP THE BUFFER AND WRITE A TRACK WITH A KNOWN VALUE.

ALGORITHM NAME: CHANGE RANGE

PURPOSE: CHANGE A RANGE OF NIBBLES

ALG # [15]

```
SYNTAX: CHANGE RANGE TO target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync
```

PARMS USED: RANGE (THESE ARE 2 BYTE WHICH DEFINE THE LOW AND HIGH VALUES FOR THE RANGE. I.E. FE AND FF (THESE ARE THE DEFAULT VALUES

PASS VALUE: target

FUNCTIONAL DESCRIPTION:

Changes all nibbles greater than or equal to the lower boundary and less than or equal to the higher boundary to the specified value.

COMMENTS: CLEANUP SELF-SYNC.

ALGORITHM NAME: CHANGE INVALIDS

PURPOSE: CHANGE INVALID NIBBLES

ALG # [16]

```
SYNTAX: CHANGE INVALIDS TO target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync
```

PARMS USED: NONE

PASS VALUE: target

FUNCTIONAL DESCRIPTION:

Changes all invalid nibbles to the specified value. DEFINITION: An invalid nibble is any nibble that contains more than two consecutive zero bits. A nibble with more than two consecutive zero bits will not read reliably.

COMMENTS: THIS ROUTINE IS USED TO CLEANUP GLITCHES WHICH ARE LEFT AFTER A WRITE HEAD IS TURNED OFF. IT MAY ALSO BE USED TO CLEANUP PORTIONS OF A TRACK THAT HAVE NOT BEEN FORMATTED.

ALGORITHM NAME: CHANGE SHIFTED

PURPOSE: CHANGE SHIFTED SELF-SYNC NIBBLES

ALG # [17]

SYNTAX: CHANGE SHIFTED

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm looks for patterns of FF self-sync that were read in before the read head was synchronized. It then changes these nibbles to self-sync FF. This routine only works for self-sync value FF.

COMMENTS: CLEANUP SELF-SYNC ROUTINE.

<u>ALGORITHM</u> <u>NAME:</u> CHANGE BITLOOKUP

PURPOSE: CHANGE NIBBLES

ALG # [18]

```
SYNTAX: CHANGE BITLOOKUP TO target
        Where target may be: SS
                             NORM
                             value     normal
                             (value)   self-sync
```

PARMS USED: BITTAB

PASS VALUE: target

FUNCTIONAL DESCRIPTION:

Every nibble in the nibble buffer is translated according to a 16 byte parameter called BITTAB. Each of the 128 bits in the 16 byte parameter BITTAB represents a nibble value from $80 to $FF. If the bit corresponding to the nibble being translated is a 1-bit, the nibble is changed to the target value. If the corresponding bit is a 0-bit, no change takes place.

EXAMPLE:
If the nibble being translated is $96, BITTAB+2 is examined, and the $02 bit determines whether the nibble being translated will be changed or not.

COMMENTS: A fast way to change many different nibbles to the same target value, or to SS or to NORM.

<u>ALGORITHM</u> <u>NAME:</u> CHANGE EXTEND

PURPOSE: EXTEND SELF-SYNC NIBBLES

ALG # [19]

SYNTAX: CHANGE EXTEND BY x
Where x is the number of nibbles you wish to extend to the right of a self-sync field.

PARMS USED: NONE

PASS VALUE: x

FUNCTIONAL DESCRIPTION:

If the nibble following a self-sync (SS) string is the same value as the self-sync in the string then change it to self-sync as long as the length doesn't exceed the number of the pass value. In the following example the FF prior to the AA would be changed to self-sync but the AA would not. If however there were three normal FF's following the self-sync string, only the first two would be changed to self-sync.

EXAMPLE:
PASS=2

```
SS SS SS NN NN
FF FF FF FF AA
```

<u>ALGORITHM</u> <u>NAME:</u> CHANGE PATTERN TO VALUE

PURPOSE: CHANGE A PATTERN
ALG # [1A]

SYNTAX: CHANGE PATx TO VALUE
Where VALUE is any value between $0 and $FF.

PARMS USED: VALUE

PASS VALUE: PATTERN NUMBER TO USE (1-7).

FUNCTIONAL DESCRIPTION:

Change any occurrances of the pattern to the specified value.

ALGORITHM NAME: CHANGE FRAME1

PURPOSE: CHANGE NORMAL NIBBLES

ALG # [1B]

SYNTAX: CHANGE FRAME1

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

Changes a maximum of two normal nibbles in a string of self-sync nibbles to self-sync.

COMMENTS: PROVIDES A FAST METHOD OF CLEANING UP NORMAL GLITCHES IN A SELF-SYNC FIELD. THIS COULD ALSO BE DONE WITH ALGORITHM 11.

ALGORITHM NAME: CHANGE SS INVALIDS

PURPOSE: CHANGE SELF-SYNC INVALIDS

ALG # [1E]

```
SYNTAX: CHANGE SS INVALIDS TO target
        Where target may be: SS
                             NORM
                             value    normal
                             (value)  self-sync
```

PARMS USED: NONE

PASS VALUE: target

FUNCTIONAL DESCRIPTION:

Changes invalid self-sync to specified value.
DEFINITION: Invalid self-sync is any self-sync nibble with the two low order bits equal to zero.
I.E. FC.

Normally this is not an invalid nibble but since it is self-sync and therefore followed by more zero bits it may not read reliably.

ALGORITHM NAME: TSTART PATTERN

PURPOSE: SET TRACK START

ALG # [20]

SYNTAX: TSTART PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE.

FUNCTIONAL DESCRIPTION:

Set track start to first occurrance of the specified pattern in the buffer.

ALGORITHM NAME: TSTART FSPACE

PURPOSE: SET TRACK START

ALG # [21]

```
SYNTAX: TSTART FSPACE bytes
        Where bytes may be: 1 field         $00
                                  # of bytes $01-$FF
```

PARMS USED: NONE

PASS VALUE: NUMBER OF NIBBLES TO MOVE THE TRACK START FORWARD.

FUNCTIONAL DESCRIPTION:

This algorithm takes the current track start and moves it forward the specified number of nibbles in PASS value.
EXCEPTION: If PASS value is zero (0) then track start is moved forward one field.
I.E. If track start is on a normal nibble then it will move forward to the first self-sync nibble if finds. If track start is on self-sync then it will move forward to the first normal nibble it finds.

ALGORITHM NAME: TSTART BSPACE

PURPOSE: SET TRACK START

ALG # [22]

SYNTAX: TSTART BSPACE bytes

```
        Where bytes may be: 1 field         $00
                                # of bytes  $01-$FF
```

PARMS USED: NONE

PARMS USED: bytes

FUNCTIONAL DESCRIPTION:

Same as algorithm number 21 except it moves backwards through the buffer instead of forwards.

COMMENTS: AFTER SELECTING TRACK START THIS ALGORITHM MAY BE USED TO MOVE THE TRACK START TO THE BEGINNING OF THE LEADIN SELF-SYNC FIELD.

ALGORITHM NAME: TSTART FIRST NORM

PURPOSE: SET TRACK START

ALG # [23]

SYNTAX: TSTART FIRST NORM

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm sets the track start to the first normal nibble occurring after a self-sync nibble. A self-sync nibble is looked for to make sure the entire normal field is present.

COMMENTS: THIS IS OFTEN USED IN SYNCHRONIZING TRACKS.

ALGORITHM NAME: TSTART FIRST SS

PURPOSE: SET TRACK START

ALG # [24]

SYNTAX: TSTART FIRST SS

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm set the track start to the first self-sync nibble following a normal nibble.

ALGORITHM NAME: TSTART LONG NORM

PURPOSE: SET TRACK START

ALG # [25]

SYNTAX: TSTART LONG NORM

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm sets the track start to the beginning of the longest normal nibble field that occurs in the first $2000 nibbles.

COMMENTS: THIS MAY BE USEFUL FOR DETERMINING DATA AREAS ON SPIRAL TRACKS.

ALGORITHM NAME: TSTART LONG SS

PURPOSE: SET TRACK START

ALG # [26]

SYNTAX: TSTART LONG SS

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm sets track start to the first normal nibble following the longest self-sync field in the first $2000 bytes of the nibble buffer.

COMMENTS: THIS NORMALLY FINDS A GOOD TRACK START. THIS IS THE DEFAULT VALUE.

ALGORITHM NAME: TSTART DOS

PURPOSE: SET TRACK START

ALG # [27]

SYNTAX: TSTART DOS PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE. DEFAULT=1.

FUNCTIONAL DESCRIPTION:

This algorithm uses a PATTERN (1) to determine a normal 13 or 16 sector track start. This will normally be sector 0.

COMMENTS: THIS HELPS FIND THE TRACK START ON NORMAL DOS DISKS.

ALGORITHM NAME: TSTART ASSIGN

PURPOSE: SET A PATTERN

ALG # [28]

SYNTAX: TSTART ASSIGN PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO STORE TRACK START INFORMATION

FUNCTIONAL DESCRIPTION:

This algorithm finds the current track start location and sets the PATTERN= to the first 16 nibbles following the track start.

ALGORITHM NAME: TSTART FSPACE EQUAL

PURPOSE: SET TRACK START

ALG # [29]

SYNTAX: TSTART FSPACE EQUAL

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm takes the current track start and moves it forward to the first nibble of a different value.

ALGORITHM NAME: TSTART BSPACE EQUAL

PURPOSE: SET TRACK START

ALG # [2A]

SYNTAX: TSTART BSPACE EQUAL

PARMS USED: NONE

PARMS USED: NONE

FUNCTIONAL DESCRIPTION:

This algorithm moves the track start back through the buffer to the first nibble of a different value.

ALGORITHM NAME: TEND PATTERN

PURPOSE: SET TRACK END

ALG # [30]

SYNTAX: TEND PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE.

FUNCTIONAL DESCRIPTION:

This algorithm sets the track end to the first occurrance of the specified pattern.

ALGORITHM NAME: TEND FSPACE

PURPOSE: SET TRACK END

ALG # [31]

SYNTAX: TEND FSPACE bytes
Where bytes may be: 1 field $00 # of bytes $01-$FF

PARMS USED: NONE

PASS VALUE: bytes

FUNCTIONAL DESCRIPTION:

This algorithm takes track end and moves it forward the specified number of nibbles in PASS value.
EXCEPTION: If PASS value=0 then track end is moved forward one field.
I.E. If track end is on a normal nibble then it will be moved forward to the first self-sync nibble. If track end is on a self-sync nibble then it will be moved forward to the first normal nibble.

ALGORITHM NAME: TEND BSPACE

PURPOSE: SET TRACK END

ALG # [32]

SYNTAX: TEND BSPACE bytes
Where bytes may be: 1 field $00 # of bytes $01-$FF

PARMS USED: NONE

PASS VALUE: bytes

FUNCTIONAL DESCRIPTION:

This algorithm is the same as algorithm 32 except it moves the track end back through the buffer instead of forward.

ALGORITHM NAME: TEND REPEAT

PURPOSE: SET TRACK END

ALG # [33]

SYNTAX: TEND REPEAT

PARMS USED: TSIZMIH
TSTLEN
BIGTRK

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm sets the track end - to track start. It takes track start and adds TSIZMIH * $100 nibbles and searches forward through the buffer for a repeat of track start. It searches for TSTLEN number of nibbles for a match. It then calculates track length. Track length must be less than BIGTRK * $100. After the pattern is found, track end is set to this location and then moved back through the buffer to the beginning of the previous self-sync field.

COMMENTS: DEFAULT ALGORITHM

ALGORITHM NAME: TEND TSTART

PURPOSE: SET TRACK END

ALG # [34]

SYNTAX: TEND TSTART pages

PARMS USED: NONE

PASS VALUE: pages

FUNCTIONAL DESCRIPTION:

This algorithm takes track start and adds the specified number of pages and sets track end.

ALGORITHM NAME: TEND ASSIGN

PURPOSE: SETS A PATTERN

ALG # [38]

SYNTAX: TEND ASSIGN PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE.

FUNCTIONAL DESCRIPTION:

This algorithm sets a specified PATTERN to the 16 nibbles that preceed track end.

ALGORITHM NAME: TEND FSPACE EQUAL

PURPOSE: SET TRACK END

ALG # [39]

SYNTAX: TEND FSPACE EQUAL

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm takes track end and moves it forward to the first nibble of a different value.

<u>ALGORITHM</u> <u>NAME:</u> TEND BSPACE EQUAL

PURPOSE: SET TRACK END

ALG # [3A]

SYNTAX: TEND BSPACE EQUAL

PARMS USED: NONE

PARMS USED: NONE

FUNCTIONAL DESCRIPTION:

This algorithm moves the track end backwards through the buffer to the first nibble of a different value.

<u>ALGORITHM</u> <u>NAME:</u> VSTART PATTERN

PURPOSE: SET VERIFY START

ALG # [40]

SYNTAX: VSTART PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO USE.

FUNCTIONAL DESCRIPTION:

This algorithm sets the verify start to the first occurrance of the specified pattern in the buffer.

<u>ALGORITHM</u> <u>NAME:</u> VSTART FSPACE

PURPOSE: SET VERIFY START

ALG # [41]

SYNTAX: VSTART FSPACE bytes
Where bytes may be: 1 field $00 # of bytes $01-$FF

PARMS USED: NONE

PASS VALUE: bytes

FUNCTIONAL DESCRIPTION:

This algorithm takes the verify start and moves it forward the number of nibbles in PASS value.
EXCEPTION: If PASS value=0 then verify start is moved forward one field.
I.E. If verify start is on a normal nibble then it will move forward to the first self-sync nibble. If verify start is on a self-sync nibble it will move forward to the first normal nibble.

<u>ALGORITHM</u> <u>NAME:</u> VSTART BSPACE

PURPOSE: SET VERIFY START

ALG # [42]

SYNTAX: VSTART BSPACE bytes
Where bytes may be: 1 field $00 # of bytes $01-$FF

PARMS USED: NONE

PASS VALUE: bytes

FUNCTIONAL DESCRIPTION:

This algorithm takes the verify start and moves it exactly like algorithm 41 except it is moved backwards through the buffer. All the information in algorithm 42 is the same as that in algorithm 41.

ALGORITHM NAME: VSTART NORM

PURPOSE: SET VERIFY START

ALG # [43]

SYNTAX: VSTART NORM

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm moves the verify start forward to the first normal nibble from track start.

COMMENTS: SINCE TRACK START IS NORMALLY SET TO THE BEGINNING OF SELF-SYNC THIS IS USED TO MOVE THE VERIFY START FORWARD TO THE FIRST NORMAL (DEFAULT).

ALGORITHM NAME: VSTART TSTART

PURPOSE: SET VERIFY START

ALG # [44]

SYNTAX: VSTART TSTART pages

PARMS USED: NONE

PASS VALUE: pages

FUNCTIONAL DESCRIPTION:

This algorithm takes track start and adds the specified number of pages to it and uses that position for the verify start.

ALGORITHM NAME: VSTART ASSIGN

PURPOSE: SET A PATTERN

ALG # [48]

SYNTAX: VSTART ASSIGN PATx

PARMS USED: NONE

PASS VALUE: PATTERN NUMBER TO ASSIGN VERIFY START TO.

FUNCTIONAL DESCRIPTION:

This algorithm finds the verify start and sets the assigned PATTERN equal to the following 16 bytes.

ALGORITHM NAME: VSTART FSPACE EQUAL

PURPOSE: SET VERIFY START

ALG # [49]

SYNTAX: VSTART FSPACE EQUAL

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm takes the current verify start and moves it forward to the first nibble of a different value.

ALGORITHM NAME: VSTART BSPACE EQUAL

PURPOSE: SET VERIFY START

ALG # [4A]

SYNTAX: VSTART BSPACE EQUAL

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm moves the verify start backwards through the buffer to the first nibble of a different value.

ALGORITHM NAME: SHORTEN ALL EQUAL

PURPOSE: SHORTEN TRACK

ALG # [50]

SYNTAX: SHORTEN ALL EQUAL BY n

PARMS USED: MINGAP

PASS VALUE: n where n is the number of self-sync nibbles to remove each time

FUNCTIONAL DESCRIPTION:

This algorithm goes through the buffer removing the specified number of self-sync nibbles from each self-sync field, leaving at least MINGAP self-sync nibbles in each field.
Note: This removes only self-sync values that are the same as other self-sync values in the self-sync string. It will not remove self-sync nibbles with differing values. The self-sync nibbles are removed from the beginning of the self-sync string.

ALGORITHM NAME: SHORTEN ALL CENTER

PURPOSE: SHORTEN TRACK

ALG # [51]

SYNTAX: SHORTEN ALL CENTER BY n

PARMS USED: MINGAP

PASS VALUE: n where n is the number of self-sync nibbles to remove each time

FUNCTIONAL DESCRIPTION:

This algorithm is the same as algorithm 50 except it doesn't require the self-sync nibbles to be of the same value. It removes the self-sync nibbles from the middle of the self-sync string.

ALGORITHM NAME: SHORTEN LONGEST EQUAL

PURPOSE: SHORTEN TRACK

ALG # [52]

SYNTAX: SHORTEN LONGEST EQUAL BY n

PARMS USED: MINGAP
CHOPS

PASS VALUE: n where n is the number of self-sync nibbles to remove each time

FUNCTIONAL DESCRIPTION:

This algorithm searches the buffer for the longest self-sync field and removes the specified number of self-sync nibbles. This sequence is repeated the number of times specified in CHOPS. The self-sync nibbles value must be the same and they are removed from the beginning of the self-sync string.

ALGORITHM NAME: SHORTEN LONGEST CENTER

PURPOSE: SHORTEN TRACK

ALG # [53]

SYNTAX: SHORTEN LONGEST CENTER BY n

PARMS USED: MINGAP
            CHOPS

PASS VALUE: n where n is the number of self-sync nibbles to remove each time

FUNCTIONAL DESCRIPTION:

This algorithm performs the same as algorithm 52. The differences are that it takes the self-sync nibbles from the middle of the self-sync string and doesn't check to see if the self-sync values are the same.

ALGORITHM NAME: SHORTEN MANUAL

PURPOSE: SHORTEN TRACK

ALG # [54]

SYNTAX: SHORTEN MANUAL

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm prints SHORTEN on the screen. In the manual mode you must shorten the track yourself.

ALGORITHM NAME: GOTO

PURPOSE: GOTO A LOCATION IN DYNAMIC STACK

ALG # [70]

SYNTAX: GOTO label

PARMS USED: NONE

PASS VALUE: label where label is a value between $0 and $FF

FUNCTIONAL DESCRIPTION:

This algorithm is used to goto a label at a location in the algorithm processing stack. It is very useful if you wish to return and do further processing after encountering an error while analyzing.

ALGORITHM NAME: LABEL

PURPOSE: SET A LABEL IN DYNAMIC STACK

ALG # [71]

SYNTAX: LABEL label

PARMS USED: NONE

PASS VALUE: label where label is a value between $0 and $FF.

FUNCTIONAL DESCRIPTION:

This algorithm is used to set a label which may be used by the GOTO label algorithm (70)

ALGORITHM NAME: RESTORE

PURPOSE: RESTORE PARAMETERS

ALG # [72]

SYNTAX: RESTORE

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm restores all default parameters in Locksmith's parameter tables.

ALGORITHM NAME: SCRNPRT

PURPOSE: SCREENPRINT

ALG # [73]

SYNTAX: SCRNPRT

PARMS USED: PRSLOT (default 1)
PAUTOCR (default 1)

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm prints the current text screen on the printer.

COMMENTS: THIS ALGORITHM ASSUMES THE PRINTER IS TURNED ON AND ON LINE WHEN IT IS INVOKED.

ALGORITHM NAME: CLEAR ANCHOR

PURPOSE: CLEARS ANCHOR

ALG # [74]

SYNTAX: CLEAR ANCHOR

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm clears the ANCHOR to the beginning of the buffer.

ALGORITHM NAME: ABORT

PURPOSE: ABORTS CURRENT OPERATION

ALG # [75]

SYNTAX: ABORT

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm aborts the current operation and returns you to the previous menu.

ALGORITHM NAME: TSTLONG

PURPOSE: TEST TRACK LENGTH

ALG # [76]

SYNTAX: TSTLONG

PARMS USED: BIGTRK

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm checks to make sure that track length is less than BIGTRACK * $100.

ALGORITHM NAME: SKIP

PURPOSE: SKIP A TRACK

ALG # [78]

SYNTAX: SKIP

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm skips the current track, and continues with the next track in the copy process.

COMMENTS: This is often used after encountering errors in processing, to continue the copy.

ALGORITHM NAME: CLEAR STATUS

PURPOSE: CLEAR STATUS REGISTERS

ALG # [7A]

SYNTAX: CLEAR STATUS n

PARMS USED: NONE

PASS VALUE: n where n is the number of the status register to clear.

FUNCTIONAL DESCRIPTION:

This algorithm clears the STATUS register with the value n.

ALGORITHM NAME: READ

PURPOSE: READS A TRACK

ALG # [7B]

SYNTAX: READ

PARMS USED: REREAD
ACTREAD
ACTERR

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm reads a track. If it encounters an error it checks to make sure that the number of times the track has been read is less than the value of REREAD. If so it increments ACTREAD, clears all error flags and rereads the track. If the value is equal or greater than REREAD it exits with the appropiate error flags set.

```
---------------------------------------------------------------
ALGORITHM NAME: COPY

PURPOSE: COPY TRACKS

ALG # [7F]

SYNTAX :COPY TRACK nn TO nn BY nn

PARMS USED: NONE

PASS VALUE: NONE

FUNCTIONAL DESCRIPTION:

This algorithm sets up the copy process after all
analysis algorithms have been entered into the text
file.  The starting track, ending track and track
increment are defined.
```

NIBBLE EDITOR COMMANDS

CURSOR MOVEMENT

[I] Move up one line.
[J] Move left one character.
[K] Move right one character.
[M] Move down one line.
[up] Up arrow Move up one line Apple //e only.
[dn] Up arrow Move down one line Apple //e only.
[lt] Left arrow Move left one character.
[rt] Right arrow Move right one character.
[<] Move back one screen page.
[>] Move forward one screen page.
[,] Continous scroll backwards.
[.] Continous scroll forward.
[CTRL-B] Move to beginning of buffer or to track start.
[CTRL-E] Move to end of track or end of buffer.

CONTROL KEY COMMANDS

[CTRL-R] Read a track into the buffer.
[CTRL-W] Write the current buffer to disk.
[CTRL-B] Move to beginning of buffer or to track start.
[CTRL-E] Move to end of track or end of buffer.
[CTRL-A] Perform an Algorithm. It will ask for PASS: value. RETURN defaults.
[CTRL-S] Perform current set of algorithms on track.

[CTRL-V] Set track verify start. Should be used after '('. Check '(' below.
[CTRL-I] Insert nibble. Replicates nibble under cursor.
[CTRL-D] Delete nibble under cursor.
[CTRL-F] Find has four options.
[CTRL-F] Enter data as hex numbers separated by spaces, end with RETURN key.
[CTRL-F] RETURN repeats the last CTRL-F command.
[CTRL-FL] Enter length (0-F). Look for pattern starting at cursor of (length).
[CTRL-FL] RETURN repeats the last cFL command.
[CTRL-FP] Enter a number (1-7) uses parm (pattern) for the search.
[CTRL-FO] Finds the first nibble that is different from the one the cursor is on.
[CTRL-P] Change parameters from within Nibble Editor.

CTRLP HAS THE FOLLOWING OPTIONS.

PARM: Enter <ctrlR> + RETURN. Restores all default parameters.
PARM: Enter <name> + RETURN.
PARM: Enter <hex value> (0-1FF valid) + RETURN.
PARM: Enter <?> + RETURN. Displays all valid parm names.
PARM: Enter <+> + RETURN. Moves to next parameter in sequence.

PARM: Enter <RETURN>. Exit parameter change mode.
AFTER PARM HAS BEEN ENTERED YOU HAVE THE FOLLOWING OPTIONS FOR VALUE.
VALUE: Enter <RETURN>. Accepts displayed default value.
VALUE: Enter <hex> + RETURN.
VALUE: Enter <hex hex hex ...> + RETURN.
VALUE: Enter <track number with decimal point.> + RETURN.
WARNING! NO CHECK IS MADE TO SEE IF THE PARM IS TRACK.

MISCELLANEOUS COMMANDS.

[(] Sets track and verify start at cursor position.
[)] Sets track end.
[S] Sets nibble under cursor to self synch.
[N] Sets nibble under cursor to normal (non-self synch).
[C] Enter change mode. Enter <hex hex ...> + RETURN to exit change mode.
[C] The commands (S and N) also work in change mode.
[H] Entering 'H' will display current buffer on the hi-res screen.
[HG] Entering a 'G' while in hi-res mode will print the hi-res screen if:
You have a printer capable of graphics and
a graphics printer card.
[G] Entering a 'G' shows you a text graphic display of the buffer.

[D] Entering a 'D' will give you a display of the 16 sector addresses and the data checksum. For 13 sector it will display address fields only.
[CTRL-Z] Prints current screen to printer.
[#] Prints from '(' <track start> to ')' <track end> on the printer.
[ESC] Pressing the 'ESC' key exits the Nibble Editor.

16 SECTOR 6-BIT NIBBLE TRANSLATE TABLE

The following translate table is used for calculating data field checksums. It is described in the chapter on the nibble editor describing the 'D' command.

00:96	01:97	02:9A	03:9B
04:9D	05:9E	06:9F	07:A6
08:A7	09:AB	0A:AC	0B:AD
0C:AE	0D:AF	0E:B2	0F:B3
10:B4	11:B5	12:B6	13:B7
14:B9	15:BA	16:BB	17:BC
18:BD	19:BE	1A:BF	1B:CB
1C:CD	1D:CE	1E:CF	1F:D3
20:D6	21:D7	22:D9	23:DA
24:DB	25:DC	26:DD	27:DE
28:DF	29:E5	2A:E6	2B:E7
2C:E9	2D:EA	2E:EB	2F:EC
30:ED	31:EE	32:EF	33:F2
34:F3	35:F4	36:F5	37:F6
38:F7	39:F9	3A:FA	3B:FB
3C:FC	3D:FD	3E:FE	3F:FF

NIBBLE DECODE TABLE

```
00:AA AA   20:BA AA   40:AA EA   60:BA EA   80:EA AA   A0:FA AA   C0:EA EA   E0:FA EA
01:AA AB   21:BA AB   41:AA EB   61:BA EB   81:EA AB   A1:FA AB   C1:EA EB   E1:FA EB
02:AB AA   22:BB AA   42:AB EA   62:BB EA   82:EB AA   A2:FB AA   C2:EB EA   E2:FB EA
03:AB AB   23:BB AB   43:AB EB   63:BB EB   83:EB AB   A3:FB AB   C3:EB EB   E3:FB EB
04:AA AE   24:BA AE   44:AA EE   64:BA EE   84:EA AF   A4:FA AE   C4:EA EE   E4:FA EE
05:AA AF   25:BA AF   45:AA EF   65:BA EF   85:EA AF   A5:FA AF   C5:EA EF   E5:FA EF
06:AB AE   26:BB AE   46:AB EE   66:BB EE   86:EB AE   A6:FB AE   C6:EB EE   E6:FB EE
07:AB AF   27:BB AF   47:AB EF   67:BB EF   87:EB AF   A7:FB AF   C7:EB EF   E7:FB EF
08:AE AA   28:BE AA   48:AE EA   68:BE EA   88:EE AA   A8:FE AA   C8:EE EA   E8:FE EA
09:AE AB   29:BE AB   49:AE EB   69:BE EB   89:EE AB   A9:FE AB   C9:EE EB   E9:FE EB
0A:AF AA   2A:BF AA   4A:AF EA   6A:BF EA   8A:EF AA   AA:FF AA   CA:EF EA   EA:FF EA
0B:AF AB   2B:BF AB   4B:AF EB   6B:BF EB   8B:EF AB   AB:FF AB   CB:EF EB   EB:FF EB
0C:AE AE   2C:BE AE   4C:AE EE   6C:BE EE   8C:EE AE   AC:FE AE   CC:EE EE   EC:FE EE
0D:AE AF   2D:BE AF   4D:AE EF   6D:BE EF   8D:EE AF   AD:FE AF   CD:EE EF   ED:FE EF
0E:AF AE   2E:BF AE   4E:AF EE   6E:BF EE   8E:EE AF   AE:FF AE   CE:EF EE   EE:FF EE
0F:AF AF   2F:BF AF   4F:AF EF   6F:BF EF   8F:EF AF   AF:FF AF   CF:EF EF   EF:FF EF
10:AA BA   30:BA BA   50:AA FA   70:BA FA   90:EA BA   B0:FA BA   D0:EA FA   F0:FA FA
11:AA BB   31:BA BB   51:AA FB   71:BA FB   91:EA BB   B1:FA BB   D1:EA FB   F1:FA FB
12:AB BA   32:BB BA   52:AB FA   72:BB FA   92:EB BA   B2:FB BA   D2:EB FA   F2:FB FA
13:AB BB   33:BB BA   53:AB FB   73:BB FB   93:EB BB   B3:FB BB   D3:EB FB   F3:FB FB
14:AA BE   34:BA BE   54:AA FE   74:BA FE   94:EA BE   B4:FA BE   D4:EA FE   F4:FA FE
15:AA BF   35:BA BF   55:AA FF   75:BA FF   95:EA BF   B5:FA BF   D5:EA FF   F5:FA FF
16:AB BE   36:BB BE   56:AB FE   76:BB FE   96:EB BE   B6:FB BE   D6:EB FE   F6:FB FE
17:AB BF   37:BB BF   57:AB FF   77:BB FF   97:EB BF   B7:FB BF   D7:EB FF   F7:FB FF
18:AE BA   38:BE BA   58:AE FA   78:BE FA   98:EE BA   B8:FE BA   D8:EE FA   F8:FE FA
19:AE BB   39:BE BB   59:AE FB   79:BE FB   99:EE BB   B9:FE BB   D9:EE FB   F9:FE FB
```

```
1A:AF BA   3A:BF BA   5A:AF FA   7A:BF FA   9A:EF BA   BA:FF BA   DA:EF FA   FA:FF FA
1B:AF BB   3B:BF BB   5B:AF FB   7B:BF FB   9B:EF BB   BB:FF BB   DB:EF FB   FB:FF FB
1C:AE BE   3C:BE BE   5C:AE FE   7C:BE FE   9C:EE BE   BC:FE BE   DC:EE FE   FC:FE FE
1D:AE BF   3D:BE BF   5D:AE FF   7D:BE FF   9D:EE BF   BD:FE BF   DD:EE FF   FD:FE FF
1E:AF BE   3E:BF BE   5E:AF FE   7E:BF FE   9E:EF BE   BE:FF BE   DE:EF FE   FE:FF FE
1F:AF BF   3F:BF BF   5F:AF FF   7F:BF FF   9F:EF BF   BF:FF BF   DF:EF FF   FF:FF FF
```

TRACK NUMBER DECODE TABLE

```
00:AA AA  01:AA AB  02:AB AA  03:AB AB  04:AA AE  05:AA AF  06:AB AE  07:AB AF
08:AE AA  09:AE AB  0A:AF AA  0B:AF AB  0C:AE AE  0D:AE AF  0E:AF AE  0F:AF AF
10:AA BA  11:AA BB  12:AB BA  13:AB BB  14:AA BE  15:AA BF 16:AB BE  17:AB BF
18:AE BA  19:AE BB  1A:AF BA  1B:AF BB  1C:AE BE  1D:AE BF  1E:AF BE  1F:AF BF
20:BA AA  21:BA AB  22:BB AA  23:BB AB
```

SECTOR NUMBER DECODE TABLE

NIBBLES	VALUE	NIBBLES	VALUE
AA AA	00	AE AA	08
AA AB	01	AE AB	09
AB AA	02	AF AA	0A
AB AB	03	AF AB	0B
AA AE	04	AE AE	0C
AA AF	05	AE AF	0D
AB AE	06	AF AE	0E
AB AF	07	AF AF	0F

PHYSICAL TO LOGICAL TRANSLATION TABLE

The sector numbers contained in the address fields of a 16-sector formatted diskette appear in ascending order ($0 to $F) on successive sectors. These physical sector numbers are converted by the disk operating system to logical sector numbers, to allow for faster read/write of multiple sectors. The following table shows the relationship between physical sector number and logical sector number.

PHYSICAL	LOGICAL
0	0
1	7
2	E
3	6
4	D
5	5
6	C
7	4
8	B
9	3
A	A
B	2
C	9
D	1
E	8
F	F

TRACK LAYOUTS (13 and 16 sector)

There have been two different track formats in common use for the Apple II. One of them recorded 13 sectors on each of the 35 tracks. The other, by employing a more efficient data packing algorithm and slightly modified hardware, is capable of recording 16 sectors per track.

Both formats are basically the same, with the exception of the method of packing the data field. In addition, the address field header is slightly different to allow the two different formats to be identified easily.

Since the 13 sector format is no longer in common use, we will discuss the 16 sector format, and will identify where the two formats differ.

The track is recorded with 16 (or 13) sectors, each consisting of an address field and a data field. The address field contains information about the data field which immediately follows it. The fields are separated by gaps, which contain 'self-sync' nibbles. These self-sync nibbles are specially recorded nibbles which cause the disk controller hardware to synchronize, so that the field following the self-sync can be read.

The address and data fields each contain a header, information nibbles, and a trailer.

The address field contains header nibbles of D5 AA 96 (or D5 AA B5, if 13 sector), followed by 4 items of information, encoded in double-nibble format. Two consecutive nibbles are used to represent the volume number, track number, sector number, and checksum. The checksum is simply an exclusive-or of the other 3 items of information. A table is included in this manual to allow you to convert these double-nibbles to the values they represent. Following these 4 items of information, is the address field trailer, which consists of DE AA.

After a gap of self-sync nibbles, the data field appears. The data field consists of a header, D5 AA AD, followed by 342 nibbles (or 410, if 13 sector format) which represent the actual sector data. These nibbles are encoded using a 6-bit table shown in the section of this manual titled "Data Field Nibble Encoding". (If 13 sector format, a 5-bit table is used.) After the data nibbles, a single nibble is provided for checksum, followed immediately by the data field trailer, DE AA.

In some early protection schemes, the header and trailer nibbles in the address and data fields was changed to some other value. (see "History of Locksmith and Copy Protection" chapter of this manual)