LOCKSMITH® 8

Version 6.0

USER'S MANUAL

for the

apple II, II+, //e, //c, //gs

DISCLAIMER

Alpha Logic Business Systems, Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

Alpha Logic Business Systems, Inc. and the authors 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 Alpha Logic, 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 Alpha Logic 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

This manual is copyrighted. All rights are reserved. This document may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior consent, in writing, from Alpha Logic Business Systems, Inc.

This software is a fully copyrighted work and as such is protected under the copyright laws of the United States of America. According to these laws, consumers of copywritten material may make copies for their personal use only. Duplication for any other purposes whatsoever would constitute infringement of copyright.

Copyright 1980-1986 by:
Alpha Logic Business Systems, Inc.
4119 North Union Road
Woodstock, Illinois 60098
(815) 568-5166

The word LOCKSMITH and the Locksmith logo are registered trademarks of Progressive Business Systems, Inc.

HARDWARE REQUIREMENTS

Apple II, Apple II Plus, Apple //e, Apple //c, or compatible computer. 48K required. One or two disk drives.

Optional hardware supported: 16K, 32K, 64K, 128K, 256K slot RAM boards or auxiliary memory boards; optional printer.

Locksmith 6.0 - manual revision: 6.0.00

TABLE OF CONTENTS

LPL routine, LPL variable, manual errata and nibble count notes are
located in ∧ files at the end of the Locksmith 6.0 library disc.

INTRODUCTION

ABOUT YOUR NEW LOCKSMITH 6.0

Locksmith was first released in December, 1980. It was the first bit-copy
(or nibble-copy) program available for the Apple. Since that time,
Locksmith has evolved from a bit-copy program into a powerful disk and
memory utility.

While other bit-copiers made minor enhancements and charged update fees
for the "new" software, Locksmith users have been able to install
enhancements to their own copies of Locksmith by applying patches which
are supplied free of charge.

As new enhancements to Locksmith become available in the form of updates,
user applied patches, and new parameter disks, registered users are
notified by mail or the Locksmith newsletter. If you have not already
done so, be sure to complete the registration card included with your
Locksmith.

Locksmith programming language (LPL) has been greatly improved. Many
users found the LPL section in the version 5.0 manual to be difficult to
read. A large part of this manual is dedicated to explaining the details
of LPL, using many examples. While a great deal of effort has been made
to keep LPL in version 6.0 compatible with LPL in version 5.0, the code
itself has been completely rewritten, and language definitions have been
extended to allow for easy future expansion. Almost all Locksmith
functions can now be invoked from LPL (including Fast Disk Backup and
sector editing).

The Locksmith philosophy has always been to allow the user to backup his
software, but not to promote software piracy in any way. Locksmith always
made an identical copy of the original diskette, including
copy-protection, copyright notices, and serial numbers. The copy made by
Locksmith was, like the original diskette, copy-protected. New protection
techniques were introduced and Locksmith handled them. However, recently
some manufacturers have introduced copy protection techniques which are
based on the fact that special hardware can create a special "signature"
which can be read by a standard Apple disk drive, but not rewritten.
Because of the introduction of these truly uncopyable diskettes, we have
found it necessary to allow the user to "unprotect" or "break" the
software in order to make a backup. It is important to understand that by
providing tools to allow the user to remove the copy protection, that we
in no way condone or promote software piracy. We will not knowingly
accept or publish parameters which remove copyright notices, serial
numbers, or other identifying information along with the copy protection.

Locksmith 6.0 has many completely new functions, including:

Automatic Boot Tracer
Sector and Track Editor
Sector Editor for Protected diskettes
RAM Card Utilities including RAMtest.

Advanced Disk Recovery:
   Data written off-center
   Data written on misaligned drive
   Partially overwritten sectors

DOS File Utilities:
   Alphabetize CATALOG
   Un-Delete a file
   Remove DOS from diskette
   Fix sector counts
   Verify VTOC and fix errors
   Disk space map
   File Editor
   Encrypt/Decrypt a file

BACKING UP YOUR LOCKSMITH DISK

Your Locksmith 6.0 disk and the included parameter diskette (parmdisk) are
not copy-protected. You should immediately make a backup copy of your
original diskettes using the FAST DISK BACKUP function of the main menu
and place the originals in a safe place. You may need the original
Locksmith diskette for program updates, as they are made available.

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 $250.00 to purchase.

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

The original Locksmith program was written in 1979 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
excercise -- "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 December 1980, Locksmith version 2.0 was released. It 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 -- PIRATING. 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 it copied. This fact, by the way, was never made known until

now.

Encoding the Locksmith serial number on the copied disk provided the
distributor with the capability of identifying the owner of any Locksmith
used for pirating any manufacturers software, and the distributor offered
to assist software manufacturers in the prosecution of software pirates.
No software manufacturer however, has ever requested this assistance.
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 in early (version 4)
releases of Locksmith. Because of this, Locksmith, beginning with version
5.0 no longer encoded 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, 5.0, 5.1, and now 6.0.
No longer the work of a single programmer, Locksmith 6.0 is now maintained
by a team of Apple copy-protection experts.
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, Alpha Logic 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 to appear 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 Micro-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 was capable of determining self-sync nibbles while reading them without regard to context nibbles, and it was 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 6.0 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.

If, while using Locksmith, you find a particularly useful suggestion or technique that you would like to see included in a future release of Locksmith, please let us know.

COMMON LOCKSMITH CONTROL KEYS

Three keystrokes listed in the Locksmith main menu can be typed at any
time, whether in the main menu or not. These keys are [control-Z], [ESC],
and the [RESET] key.

[control-Z]

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 parameter name 'PRT.SLOT'.

[ESC]

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

[RESET]

To exit Locksmith and re-boot your system, press the RESET key while
holding down the CTRL key.

IMPORTANT LOCKSMITH INFORMATION

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 entire Locksmith program is too large to fit in the memory of your
Apple at one time. It is loaded from disk into memory in sections called
"overlays". It is important to keep your Locksmith disk in the boot drive
until instructed to insert other disks to process. If an overlay is
required to be loaded, and the Locksmith diskette is not inserted, you
will be prompted to re-insert the Locksmith disk.

When a Locksmith function is invoked which requires you to insert
diskettes either for input or output, you will be prompted to insert them
before the function begins processing. Signal that the diskettes have
been inserted by pressing the space bar. If at any time you wish to
cancel or abort a function, press the ESC key.

THE TRACK STATUS DISPLAY

The top 7 lines of the Locksmith screen are reserved for the track status
display.

This display shows the status of the last operation on each track of the
diskette. Tracks are numbered from 00 to 22 (in hexadecimal, which is 0
to 34 in decimal). A position for track 23 is provided because some early

protection techniques made use of this accessable but normally unused
track.

Although most Apple software only uses integer tracks from 00 to 22, the
Apple disk drive is actually capable of moving the head between tracks.
For this reason, four lines are provided for each numbered track. The
topmost of these four lines contains the status codes for integer track
numbers such as 00, 01, 02, etc. The other 3 of the 4 lines contain track
status codes for 1/4 tracks, half-tracks, and 3/4 tracks, respectively.

Shown below is an example track status display.

```
            THE LOCKSMITH - VERSION 6.0 - REVISION A
    .00              0
    .25
    .50                                4              5
    .75                                                  5
    HEX  000000000000000001111111111111112222
    TRK  0123456789ABCDEF0123456789ABCDEF0123
```

The example shows status code 0 for track 0A, status code 4 for track
13.5, and status code 5 for track 1B.75.

Status codes are dependent on the individual Locksmith function currently
running, and are described separately.

GETTING STARTED:
BACKING UP A DISK

There are several functions within Locksmith which are capable of backing
up a diskette. Which one you will want to use depends on the diskette you
wish to copy.

If the diskette is not copy-protected, use the FAST DISK BACKUP function
by pressing 'F' from the main menu.

If the diskette is copy-protected, use the TEXT EDITOR Backup function
('T' from the main menu, 'B' from the text editor menu) and determine if
the name of the program is in the list of supplied parameters on the
parameter diskette (included with Locksmith). If the name is found,
select the name and the backup operation will begin automatically.

If the name is not found on the parameter diskette, Try using the standard
copy routines by selecting BACKUP/COPY (key 'B') from the main menu. Most
protected software can be copied using the standard backup/copy function.

If you used parameters on the parameter diskette to backup your disk, but
are still unable to successfully backup your disk, it is possible that a
different protection technique is being used for that particular piece of
software. Software companies frequently change protection techniques to
make it as confusing as possible for those trying to copy the software.
.Call or write our customer support group to find out if new parameters are
available for your particular software.

Finally, if you are technically experienced and enjoy challenges, you
might consider developing parameters to copy the software yourself.
Locksmith has many useful tools to help you determine the type of
copy-protection and help you either copy the software or disable the
protection.

THE LOCKSMITH MAIN MENU

Immediately after booting your Locksmith disk, the main menu appears.

```
              THE LOCKSMITH - VERSION 6.0 - REVISION A
              .00
              .25
              .50
              .75
              HEX  000000000000000011111111111111110000
              TRK  0123456789ABCDEF0123456789ABCDEF0123

              B BACKUP/COPY F FAST BACKUP / CLR STATUS

              N DISK EDITOR L LOAD RAM CD * PARAMETERS

              T TEXT EDITOR R RAMCD UTILS Q SCAN DISK

              A BOOT TRACER C CERTIFY DSK U 16-S UTILS

              D DOS3.3 UTIL X DSK RECOVER S DISK SPEED

                                          E ERASE DISK

                                          I INSPECTOR

              CTRL-Z PRT SC RESET EXIT LS ESC RESTART
```

The top 7 lines of the display are the track status display.

The body of the main menu displays the keystrokes (in inverse) which you
may enter at this time and a brief description of what function each key
performs. The functions available in the main menu are each discussed
separately in this manual. Briefly they are:

[B] BACKUP/COPY - This function allows you to backup protected software
using the Locksmith standard copy routines, which will backup most
protected software without parameters.

[F] FAST BACKUP - The Fast Backup function allows you to quickly create
backup copies of any unprotected software. Depending on the amount of
memory in your system, this utility can backup a disk in as little as 8
seconds!

[/] CLR STATUS - Pressing the "/" key will clear the track status display
at the top of the screen.

[N] DISK EDITOR - The Disk Editor (formerly the Nibble Editor in earlier
releases of Locksmith) can be used to manually read, search, change, and
rewrite data, either in sector format or nibble format. In addition, the
Disk Editor is used to edit DOS and PRODOS files, and can be used to edit
data on any RAM cards installed in the Apple.

[L] LOAD RAM CD - This function will load a slot 0 RAM card on an Apple II
or Apple II Plus, or load the built-in 16K RAM on an Apple //e or Apple
//c. This function loads 12K of data which is found on tracks 12, 13, and

14 (hex) of the Locksmith disk. If you own Inspector and Watson, you can write them to these tracks, and they will be loaded whenever you select this function. Once loaded, Inspector and Watson are available by pressing the 'I' key from the main menu.

[*] PARAMETERS - This function allows you to display or change the current Locksmith operating parameters by name. For a complete list of all of the Locksmith parameter names, please refer to the Locksmith Technical Reference Manual.

[T] TEXT EDITOR - The Locksmith Text Editor is used to edit files containing Locksmith Programming Language (LPL), which can be loaded and saved to your Locksmith parameter disk. The BACKUP function of the text editor will allow you to automatically backup protected software after specifying the name of the software.

[R] RAMCD UTILS - The Locksmith RAM card utilities can be used to test RAM cards in the Apple. RAM cards of any size can be tested, and two tests are provided: a basic test and an extensive test. The test can be performed once or continuously to help isolate intermittent errors. In addition, the contents of any 16K bank of the RAM card can be dumped into main memory, edited with the disk editor, and re-loaded to the RAM card.

[Q] SCAN DISK - Formerly known as "quick scan" in earlier versions of Locksmith, this utility allows you to examine on the hi-res graphics screen an overall "picture" of the diskette, track by track. This function is useful in determining how each track of an unknown disk is formatted.

[A] BOOT TRACER - The Automatic Boot Tracer is intended for use by the more experienced Apple programmer. It is actually a sophisticated debugger which can simulate the operation of the 6502 in the Apple. Because disk reading is simulated, it is possible to actually "boot" a disk (whether protected or not) under control of this debugger and trace the boot code of the program.

[C] CERTIFY DSK - The Certify Disk function will write a special pattern to each track of the diskette and immediately read it back to make sure that no scratches or other imperfections on the diskette surface will affect the reading and writing of data to the diskette.

[U] 16-S UTILS - This menu option will bring up a secondary menu containing useful 16-sector utilities. These utilities can be used to verify a diskette, compare two diskettes, format a diskette, or display the "sync signature" of a diskette.

[D] DOS3.3 UTIL - This menu option will bring up a secondary menu containing useful DOS 3.3 utilities. These utilities can be used to alphabetize a catalog, undelete a file, load a DOS file into memory for examination by the Disk Editor, show a disk free space map, fix incorrect sector counts in the catalog, identify and correct catalog/VTOC errors, and remove DOS from a disk to allow for the storage of more files.

[X] DSK RECOVER - The Advanced Disk Recovery utility will read a difficult or impossible to read diskette and write a good copy to another diskette. This is especially useful for recovering data from diskettes which were written in drives which were aligned improperly. Advanced Disk Recovery will actually recover data from a diskette which was written while

inserted in the disk drive off-center!

[S] DISK SPEED - This utility will allow you to check the speed of your disk drives so that you may adjust the speed to the standard 300 R.P.M., a recommended speed which is slightly slower, or set the speed of the drive to the same speed that created a given diskette.

[E] ERASE DISK - This function is used to entirely or partially erase a diskette.

[I] INSPECTOR - Invokes Inspector and Watson if the utility programs are available or have been previously loaded with the 'L' menu item.

[ESC] RESTART - At any time during the operation of Locksmith the 'ESC' key may be pressed to abort the current function and return to the previous menu or main menu.

[CTRL-Z] PRT SC - To print the contents of the text screen to the printer at any time during the operation of Locksmith you should hold down the 'CTRL' key and press 'Z'.

[RESET] EXIT LS - To exit Locksmith hold down the 'CTRL' key and press the 'RESET' key. This will cause an immediate reboot.

BACKUP / COPY DISK

[B] BACKUP/COPY

Pressing 'B' from the main Locksmith menu will allow you to make a backup copy of most protected software. If the disk you wish to backup is not protected, you will want to use the much faster FAST BACKUP utility (key 'F' from the main menu).

The 'B' (backup/copy) function uses the standard copy routines, which work well for most protected software. To use copy routines which are already tailored for a specific protected diskette, use the main menu 'T' (text editor) function to select the name of the program you wish to back up. See the text-editor section of this manual for more information.

After pressing 'B' from the main menu, you are prompted for input and output drives. If you specify the same drive for both (if you have only one drive), you will need to swap disks for each track to be copied. You are then prompted for the range of tracks for the copy operation and the track increment. To use the default values displayed for track start, track end, and track increment, you may simply press the RETURN key. After specifying whether the tracks should be synchronized and whether the track length should be preserved (a technique known as "nibble counting"), you will be prompted to insert the diskettes for the copy operation, and the backup will begin.

The track status display at the top of the screen will contain the status codes for each track copied. A zero indicates that the track copied with no errors. Other codes indicate errors which have occurred while copying the track. Since the status codes are dependent on the LPL (Locksmith Programming Language) being executed, refer to the individual LPL file for a description of status codes and error messages.

FAST DISK BACKUP

[F] FAST DISK BACKUP

Pressing 'F' from the Locksmith main menu will enter the Fast Disk Backup utility. Fast Disk Backup (FDB) is intended to be used to backup unprotected diskettes.

FDB is very fast. It reads a disk in as little as 8 seconds, and writes a disk in about the same amount of time. Optionally verifying the written data by reading it back and comparing it requires another 8 seconds, if it's desired.

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

Locksmith 6.0 FDB can automatically switch between two drives and write disks from memory, producing a new disk every 8 seconds. This is very useful for mass-production of software for clubs or software manufacturers.

FDB will automatically recognize any RAM cards in your Apple, whether they are in slots 0-7 or in the auxiliary slot of the Apple //e. The auxiliary 64K present on the Apple //c is also used. Both the Apple //e and Apple //c have a built-in 16K RAM which appears as slot 0. RAM boards from several manufacturers, including Checkmate Technology, Applied Engineering, and Titan Technology have been tested with Locksmith 6.0 and work well. Locksmith FDB will recognize 16K, 32K, 64K, 128K, and 256K RAM cards. To copy an entire disk into memory requires 140K (4K per track). FDB uses 40K of main memory for track storage, and therefore needs 100K in RAM cards to make one-pass copies or copies entirely from memory. All slots, including slot 3, are searched for RAM cards.

Fast Disk Backup Commands

Simple commands are entered from the keyboard. After pressing the return key, the current command can be seen in the current command display area in the lower left of the screen. The default command is "12", which means to copy drive 1 to drive 2.

A complete list of 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
1     read verify drive 1
2     read verify drive 2
10    copy drive 1 into memory
20    copy drive 2 into memory
01    copy memory to drive 1
02    copy memory to drive 2
V     turn verify-after-write flag on or off
[space] or [return] begins the copy or verify operation
[reset] exits Fast Disk Backup
[control-Z] prints the screen to a printer
[control-X] cancels the key entry
```

In addition to the commands listed above, there are several parameters
which can be modified from within FDB. These parameters can be used to
allow FDB to read and write sector oriented protected disks, and set
internal FDB operating parameters. Here is a list of some of the more
useful parameters, along with their initial default values, and a
description.

    0007=00 Requested output volume number or zero, if same as input
            volume.

    0008=00 Begin track to process.

    0009=22 End track to process.

    0010=08 Maximum read retry count. The retry number is displayed as
            1-9 for the first nine revolutions of the disk and A-Z for
            the next twenty-six. After that the display will not
            change.

    0011=03 Maximum verify after write retry count.

    0012=10 Motor On delay for read. Maximum value is 7F.

    0013=10 Motor On delay for write. Maximum value is 7F.

    0014=80 Seek Off delay for read. DOS uses FF, the longest value.
            A value of 00 is satisfactory and will reduce the overall
            copy time by about a second.

    0015=80 Seek Off delay for write. It is not recommended to change
            this value below 80 because the drive may begin writing
            before the seek mechanism has settled.

    0016=0B Number of self-sync before address field. This number is
            expressed in excess-5 notation. For example, the default
            of 0B will write sixteen self-sync nibbles and the value of
            01 will write 6 self-sync nibbles. Setting this value very
            low will create a disk which cannot be written to because
            writing a data field would over-write the subsequent
            address field. Setting this value too high will cause a
            problem fitting all of the sectors on the track.

    0017=08 Number of self-sync before data field. If this value is
            too low, neither DOS nor FDB will be able to read the data.
            Setting this value too high will cause a problem fitting
            all of the sectors on the track.

    0018=00 Alternate writing to drive 1 and 2. Setting this flag to
            FF will cause FDB to alternate writing between two drives,
            allowing for very fast and efficient disk copying from
            memory.

To change parameters within FDB, enter the 4-digit address and press
return. The current value will be displayed and you can key over the
value to change it.

While the copy operation is in progress, the ESC key can be used to
cancel/abort the copy operation. The 'V' key can be used to turn on or

turn off the verify-after-write flag.

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 6.0	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 tc write & verify
Locksmith 6.0	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.

CLEAR TRACK STATUS DISPLAY

[/] CLR STATUS

The track 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.

DISK, NIBBLE, AND MEMORY EDITOR

[N] DISK EDIT

Pressing 'N' from the main menu will select the Disk Editor. This function was referred to as the Nibble Editor in previous versions of Locksmith, but it has been expanded to be able to edit sectors, entire tracks of sectors, DOS files, and RAM card data in addition to disk nibbles. The keystroke 'N' to invoke it has been kept for compatibility with earlier versions of Locksmith. Some text in this manual may still refer to it as the Nibble Editor.

The following single keystroke commands are supported by the disk 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.

Now we will cover how to move around within the buffer.

CURSOR MOVEMENT:

The disk editor cursor consists of a flashing box on both sides of the nibble or byte of data.

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 buffer one screen page, unless you are already at the beginning of the buffer.

[>]
        Pressing the '>' key will move forward through the 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.

DISPLAY CONTROL COMMANDS:

[A]
        This key toggles the display of ASCII data to the right of the hex display. The ASCII display is normally meaningless for nibble data, so it is normally turned off.

[B]
        This key toggles the display between nibble mode and byte mode.
In nibble mode, self-sync nibbles are displayed in inverse text, while normal nibbles (not self-sync) are displayed in normal text. In self-sync mode, all data is displayed with the high order bit turned on, because all nibble data by definition has the high order bit on. In byte mode, all data is displayed in normal text with the high order bit of each byte left intact.

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.

[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 two options:

    [RETURN]
        Pressing the return key will cause find to use the pattern in
    the string variable PAT0. You can set this variable with the LPL
    command processor described elsewhere in this section of the manual.

    [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.

Earlier versions of the Nibble Editor had three other options to the Find
command. The O (find other), P (find pattern name), and D (find data).
Because of the capability of using LPL directly from within the Disk
Editor of version 6.0, these options are no longer needed within the Disk
Editor. The equivalent LPL commands are:

    NEXT.DIFF finds the next nibble in the buffer different than the one
    at the cursor. This is equivalent to the find "O" (other) option.

    FIND PAT4 finds the next occurrence of the data contained in the
    variable PAT4. This is equivalent to the find "P" (pattern name)
    option.

    FIND D5 AA 96 finds the next occurrence of the data supplied by the
    user. This is equivalent to the find "D" (data) option.

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 at 2000.

[CTRL-E]   Moves the cursor to track end. If the cursor is at track end,
the cursor is moved to the end of buffer at 7FFF.

[(]   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 data and press RETURN to exit change mode.
You may also use the left and right arrow keys and the space bar in change
mode. This changes the data 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 locaton 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.

[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,

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.

[#]
        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.

[RETURN]
        Pressing the RETURN key places the prompt "LPL:" above the data
display. You may enter an LPL statement at this time. The LPL statement
may be a SHOW command to display the value of a named variable, a variable
name assignment statement, or a processing routine to invoke. See the
chapter on LPL for more information.

[CTRL-S]
        Pressing control-S will process the analysis portion of the
current track procedure. This is used after reading to analyze the nibble
data and set pointers for later writing.

FRAMING BIT ANALYZER

The framing bit analyzer is intended as a tool for the more advanced Locksmith user.

Framing bits (sometimes called "sync bits" or "timing bits") are the zero bits that occur between some nibbles of data on the disk. These nibbles with framing bits are called self-sync nibbles. Each bit of nibble data requires 4 microseconds to read or write. An 8-bit nibble of raw disk data with no framing bits therefore requires 32 microseconds. For each framing bit occurring after the 8-bit nibble, 4 microseconds are added to this time. A 10-bit (2 framing bit) self-sync nibble therefore requires 40 microseconds to read or write.

13-sector DOS (DOS 3.2 and earlier) used 9-bit self-sync (1 framing bit) while 16-sector DOS (DOS 3.3 and PRODOS) use 10-bit self-sync (2 framing bits). It is also possible to write 3 or more framing bits after a nibble of data although if 3 or more framing bits occur, it is more likely that sync may be lost by the disk controller while reading.

Some copy protection techniques rely on self-sync data with a specific number of framing bits on a specific nibble of data on the track. This can be detected in a number of ways. One way to determine framing bit information is to cause the disk controller shift register to deliberately lose sync. This causes the normally invisible framing bits to be shifted into the disk controller shift register. This method doesn't work with some models of disk drives. Another method to determine the number of framing bits is to perform a statistical analysis of the timing of the disk data using several reads of the same data.

The framing bit analyzer performs a statistical analysis using precise timing loops and reports the timing relationship of the nibble data. The framing bit analyzer reads the track data repeatedly, each time updating cummulative timing statistics for each nibble of data. After several read passes, this timing information can precisely determine the number of timing bits used to write any nibble of data on the disk.

The framing bit analyzer is entered by pressing '$' from the disk editor after reading in a track from disk and manually setting certain pointers in the track buffer.

Let's use an example to describe the use of the framing bit analyzer. In the disk editor, use the control-R command to read a track of nibbles. Place the "(" start pointer on the start of an address field (D5 AA 96). Move the cursor down about 10 lines and set the ")" end pointer. Now move the cursor back up to the "DE AA" after the start of the address field (D5 AA 96).

The framing bit analyzer uses the data from the start pointer "(" to the nibble immediately before the cursor as a "key". If you placed "(" before the D5 nibble and the cursor on the DE nibble, the key length would be $0B (decimal 11) nibbles.

Enter the framing bit analyzer by pressing the "$" key. The display shows the data being analyzed on the left side of the screen with statistical information on the right.

The following command keys control the framing bit analyzer:

The space bar temporarily stops the analyzer so you may examine the statistics. Pressing the space bar again will cause a single read to occur.

The RETURN key starts the analysis again.

ESC exits the framing bit analyzer and returns to the disk editor.

"<" and ">" allow you to scroll through the buffer if it is more than one screen in length.

"T" switches the display from nibble data to the timing statistics information associated with each nibble.

The timing statistics information associated with each nibble would be 10 for an ideal normal 8-bit (no framing bit) nibble, 20 for an ideal 9-bit (1 framing bit) nibble, 30 for an ideal 10-bit (2 framing bit) nibble, and so on. Timing statistics between these "ideal" values indicate differences between the writing and reading drive speeds and statistical error due to the number of samples read. This error is reduced when more samples are taken.

The information on the right side of the screen is a statistical summary of the analysis and might look like the following:

```
R=0007
F=07
      ??
     --  ++
0    02
1 00  25
2 25  03
3 02  00
4 01  01
5 00  18
6 0B  00
7 00  07
8    03
```

The "R=" value is the number of reads which occurred. The "F=" value is the number of reads in which the "key" of the data was found on the track. Because of limited buffer space, if a large data length is specified, it is possible that the buffer may not fully contain the data specified.

The table of numbers represents the count of the differences of the data read with the "ideal" values of 10, 20, 30, etc. These error differences range from -7 to +7, with 00 being the ideal. For example, in the table shown above, the count for +1 is 25 (hex). This means that there are 25 occurrences of nibbles with timing values of one greater than the ideal value (11, 21, 31, etc.). The "8" count of 03 indicates 3 occurrences of timing values exactly halfway between ideal values. This indicates that more samples are needed. The "??" is displayed when the values for the +/- 6,7, and 8 counts are non-zero, and indicates that more samples are needed.

When nibble data is displayed on the left side, the inverse/normal mode of

each nibble indicates the number of framing bits after the nibble. Normal
text indicates no framing bits. Both digits are inverse if 2 framing bits
while only one digit is inverse if 1 framing bit. A flashing value
indicates 3 or more framing bits.

If the nibble data has the high order bit turned off, this indicates that
at least one read occurred where the data after the key did not match.
This could have been caused by a loss of sync while reading or by the key
being specified as too short which caused a non-unique part of the track
to be analyzed instead of the desired part.

LOAD RAM CARD

[L] LOAD RAM CD

Pressing the 'L' key from the main menu will cause tracks $12,$13, and $14
to be read into memory addresses $D000 through $FFFF of a RAM card in SLOT
0. If you have an Apple //e or //c, there is a "built-in" slot 0 16K RAM
card already in your system.

If you own Inspector / Watson, you can place them on your Locksmith disk
by using the following procedure and then quickly load the RAM card when
Locksmith is booted by pressing 'L' from the main menu.

Your Locksmith diskette already contains a RAM image of INTEGER BASIC and
the monitor on tracks $13 and $14. To install Inspector / Watson on track
$12, follow these instructions:

    Boot a copy of a disk that contains your copy of Inspector and
    Watson. Then enter Inspector or Watson, insert a copy of your
    Locksmith diskette and press the following keys:

    B D 0 <return>
    T 1 2 <return>
    control-W

    Then press the control-I key 15 times. Inspector and Watson are now
    placed on your Locksmith diskette.

PARAMETERS

[*] PARAMETERS

Pressing the '*' key from the main menu will allow you to display and set Locksmith parameters. Parameters are displayed and modified using their parameter name (LPL variable name).

To exit from the parameter function press the ESC key.

To display the value of a parameter, enter the keyword "SHOW" followed by the name of the parameter. For example, to display the current value of the parameter SLOT, enter:

    SHOW SLOT

To change the value of a parameter, simply enter its name followed by the value you wish to assign to it. For example, to change the value of the parameter SLOT to 4, enter:

    SLOT 4

If an error was made in entering the command, the speaker will beep, and the command will be ignored. Simply type it again correctly.

To display the addresses of the parameter names as they are being displayed, set the parameter SHOW.ADDR to the value YES, as follows:

    SHOW.ADDR YES

Now if you enter the command:

    SHOW SLOT

the address of the parameter SLOT is displayed in addition to its value.

Any named Locksmith parameters can be displayed and changed in this manner.

Here are a few useful parameters and a brief description of what they are used for:

    SLOT is the slot number of the disk drive Locksmith is to use for
    disk functions. For example:

        SLOT 4

    PRT.SLOT is the slot number which contains the interface for the
    printer to be used for print, print screen and graphics print
    commands. For example:

        PRT.SLOT 1

    GR.CHARS is a string variable (maximum length of 15) which contains
    the printer setup string which will cause the interface to print the
    graphics screen. Locksmith sends this string to the printer when the

'G' key is pressed while a graphics screen is displayed in the Disk Scan or Disk Speed utilities. The following statement sets the setup string to control-I 'GRD', which will cause Grappler and GrafStar printer interfaces to print the graphics screen rotated and double-size:

    GR.CHARS 89 'GRD' 8D

LS.SLOT is the name of the parameter to change the drive slot from which Locksmith will load overlays from. For example:

    LS.SLOT 5

The following statement, while not really a parameter, can be useful to assist in reading sectors from a disk that reads unreliably. This patch to RWTS inhibits the recalibrate of the disk head after an I/O error occurs and instead will retry the read operation until it succeeds. The second statement listed below removes the patch and allows normal reading of sectors. Note that with this patch installed that sector reads for a sector that is permanently bad will never terminate. If this occurs, simply remove the bad diskette and replace it with a good one until the read terminates normally.

    BDCC 4C C1 BD
    BDCC 10 F3 AD

TEXT EDITOR / LPL

[T] TEXT EDITOR

The Locksmith text editor is entered from the main menu by pressing the
'T' key.   The text editor is a special purpose file editor which is used
to edit Locksmith Programming Language (LPL) files.   These files can be
loaded from and saved to a Locksmith Parameter Diskette, which is included
with the distribution of Locksmith 6.0.   The parameter diskette (sometimes
referred to as the "parmdisk") is not copy protected, yet is formatted in
a special way to maximize use for saving many parameter files.   The free
space and alphabetical directory on the parmdisk is automatically managed
by the text editor.

After entering the text editor menu, the display screen will show the
following choices:

[L] LOAD FILE

[S] SAVE FILE

[D] DELETE FILE

[X] SYNTAX CHECK

[C] CHANGE SLOT AND DRIVE

[I] INITIALIZE PARMDISK

[Z] PARMDISK STATISTICS

[E] ENTER EDITOR

[N] NEW FILE

[P] PRINT FILE

[A] PRINT ALL FILES

[B] BACKUP / COPY DISK

By pressing a single key from the text editor menu, any text editor
function can be selected.

BACKUP USING CUSTOM PARAMETERS

[B] BACKUP / COPY DISK

This text editor function is used to automatically select a parameter file
to be used to backup a disk with a minimum of keystrokes from the user.
To enter this function from the text editor menu, press the 'N' key ("NEW"
clears the work buffer if any file is currently loaded) followed by the
'B' key (backup / copy).

If the parmdisk is not in the selected drive you will be prompted to
insert it.   After the parmdisk directory is read into memory, a list of

parameter files contained on the parmdisk is displayed.   The "light bar"
will be positioned on one of the names, probably the first one if no
parmdisk operations have been previously performed.   To move the light bar
to select a different file, you can press one of the following keys:

[up-arrow] and [down-arrow] will move the light bar up and down.   If you
have an Apple II or II PLUS you can use [control-K] to move up and
[control-J] to move down.

[control-N] (next page) and [control-P] (previous page) can be used to
scroll ahead or backward one complete page of names.

If you know the name of what you are looking for, you may enter the first
character of the name.   The light bar will be immediately positioned on
the first name beginning with the character you entered.   You may furthur
qualify your suggestion by entering more characters of the name or may use
the up and down arrow keys to move the light bar.   The left and right
arrow keys move the cursor in the name key-in field.   The name key-in
field always shows the current name selected.

If while entering a name to select, you enter a character that causes the
name key-in field to contain a file name that is not in the directory, the
speaker will "beep" and the light bar will be positioned on the name
closest to the one you have entered so far.

To select a file name for backup, press the RETURN key.   The file will be
loaded from the parmdisk, automatically syntax checked, and control will
be transferred to the Locksmith backup/copy routines after suitable
prompts for you to insert the Locksmith diskette and finally, the
diskette(s) to copy.   The backup/copy routines will automatically copy the
proper tracks with the proper LPL parameters.

LOAD A PARAMETER FILE

[L] LOAD FILE

Press 'L' from the text editor menu.   If the parmdisk is not in the
selected drive you will be prompted to insert it.   After the parmdisk
directory is read into memory, a list of parameter files contained on the
parmdisk is displayed.   The "light bar" will be positioned on one of the
names, probably the first one if no parmdisk operations have been
previously performed.   To move the light bar to select a different file,
you can press one of the following keys:

[up-arrow] and [down-arrow] will move the light bar up and down.   If you
have an Apple II or II PLUS you can use [control-K] to move up and
[control-J] to move down.

[control-N] (next page) and [control-P] (previous page) can be used to
scroll ahead or backward one complete page of names.

If you know the name of what you are looking for, you may enter the first
character of the name.   The light bar will be immediately positioned on
the first name beginning with the character you entered.   You may furthur
qualify your suggestion by entering more characters of the name or may use
the up and down arrow keys to move the light bar.   The left and right
arrow keys move the cursor in the name key-in field.   The name key-in

field always shows the current name selected.

If while entering a name to select, you enter a character that causes the
name key-in field to contain a file name that is not in the directory, the
speaker will "beep" and the light bar will be positioned on the name
closest to the one you have entered so far.

When the light bar is on the name you wish to select, press the RETURN
key. The file will be loaded into the work area.

SAVE A PARAMETER FILE

[S] SAVE FILE

Press the 'S' key. If the parmdisk is not in the selected drive you will
be prompted to insert it. After the parmdisk directory is read into
memory, a list of parameter files contained on the parmdisk is displayed.
The "light bar" will be positioned on one of the names, probably the last
name you previously used. To use the same name (replace file), simply
press the RETURN key. To move the light bar to select a different file,
you can press one of the following keys:

[up-arrow] and [down-arrow] will move the light bar up and down. If you
have an Apple II or II PLUS you can use [control-K] to move up and
[control-J] to move down.

[control-N] (next page) and [control-P] (previous page) can be used to
scroll ahead or backward one complete page of names.

If you know the name of an existing file you wish to replace, or want to
name your file with a similar name to an existing file, you may enter the
first character of the name. The light bar will be immediately positioned
on the first name beginning with the character you entered. You may
furthur qualify your suggestion by entering more characters of the name or
may use the up and down arrow keys to move the light bar. The left and
right arrow keys move the cursor in the name key-in field. The name
key-in field always shows the current name selected.

If while entering a name to select for saving, you enter a character that
causes the name key-in field to contain a file name that is not in the
directory, a new name will be displayed in the key-in area with the words
"ADD FILE:" displayed. Key over using the space bar, any unwanted
characters and press the RETURN key when you are finished specifying the
name you wish to save. The name will either be added or replaced,
depending on whether the name already exists.

NEW — CLEAR EDIT WORK AREA

[N] NEW FILE

Pressing the 'N' key from the text editor menu causes the text editor work
area to be cleared. Any file contained in the work area not already saved
on the parmdisk is lost. If you have made changes that you wish to save,
be sure to save the file before you clear the work area.

CHANGE SLOT AND DRIVE

[C] CHANGE SLOT AND DRIVE

Pressing the 'C' key will prompt you for a new slot number and drive
number of the disk to use for the parameter disk. Either enter a new slot
and drive number or press the RETURN key to use the default values
displayed.

PARAMETER DISK STATISTICS

[Z] PARMDISK STATISTICS

Press the 'Z' key to display statistics about the parameter disk. The
information displayed is in two parts. Information about the "CATALOG" or
directory is displayed first, followed by information about the "DATA"
portion of the parameter diskette. Shown below is an example parameter
disk statistics display.

```
VOLNAME:LS 6.0 P1

        DISKETTE STATISTICS

        ***** CATALOG *****

    364 FILES
   6071 BYTES FREE
   5705 BYTES USED
     13 AVERAGE FILENAME LENGTH
        ENTRIES WHICH CAN BE ADDED:
    496 AVERAGE FILENAME LENGTH OF 10
    181 AVERAGE FILENAME LENGTH OF 30

        *****  DATA   *****

 103744 BYTES FREE
  27329 BYTES USED
     75 AVERAGE FILENAME LENGTH
        FILES WHICH CAN BE ADDED:
   1621 AVERAGE FILE LENGTH OF 64
    405 AVERAGE FILE LENGTH OF 256
```

DELETE A PARAMETER FILE

[D] DELETE FILE

Pressing the 'D' key will allow you to delete a single file from the parameter disk. Select a file to delete in the same way that you would select a file to load (described earlier) and press the RETURN key. The file will be removed from the parameter disk and the space it used reclaimed.

INITIALIZE PARAMETER DISKETTE

[I] INITIALIZE PARMDISK

Pressing the 'I' key from the text editor menu will allow you to format and initialize an empty parameter diskette to store additional files. WARNING: This function erases all data on the target diskette. Enter a volume name for the diskette. When prompted with "FORMAT?", enter "Y" if you wish the diskette to be formatted or "N" if the diskette is already formatted with standard 16 sectors per track. Press the space bar when prompted, to start the format and initialization process.

After the process is complete, you may use the disk to save additional parameters.

ENTER THE TEXT EDITOR

[E] ENTER EDITOR

Pressing the 'E' key will enter the text editor. To exit from the text editor and return to the text editor menu, press the ESC key.

If residue exists in the work area, you can clear it by pressing control-N (NEW) from the text editor or exit the text editor (press ESC) and enter 'N' (NEW) from the text editor menu.

USING THE TEXT EDITOR

The text editor is a line numbered editor. All lines entered are given sequential line numbers, which are displayed in inverse text on the leftmost two positions of each line. The line numbers are displayed in hex. Line numbers are used only for referring to a portion of a file with the ".I" (include) directive of LPL.

Lines in the text editor are displayed one line of text per one line of the Apple 40-column display. With 2 positions used for line number, 38 positions are allowed for the maximum length of a text line. If it is necessary to continue an LPL statement on the next line, key a minus sign ("-") as the last character entered on the line to be continued. Any number of lines may therefore be used to contain an LPL statement.

There are two cursors used in the text editor. If the line number is flashing, the cursor is on the line number. If a single character on the line is flashing, the cursor is within the line of text. This is important for the insert and delete functions which are described later.

To enter text, simply type the statements you wish followed by the RETURN key. The return key causes the cursor to go to the next line. The backspace key (left arrow) is used to backup the cursor one character position if it is within a line of text and move up one line if the cursor is on a line number.

To edit an existing line, use the RETURN key or the left arrow to position the cursor on the line number of the line to be edited, and use the right-arrow to move the cursor onto the text line. Simply key over data on the line to change the data.

To insert or delete characters on the line, position the cursor within the line over the character you wish to delete or insert before, and press the control-I key to insert blanks or the control-D key to delete characters.

To insert or delete entire lines of text, position the cursor on the number of the line and press the control-I key to insert blank lines or the control-D key to delete lines.

To exit from the text editor and return to the text editor menu, press the ESC key.

A summary of the text editor commands:

ESC   exit the text editor

ctrl-N NEW - clear workarea

RETURN moves cursor to the line number of the next line.

<---   if the cursor is on a line number, moves it up one line.
        If the cursor is within a line of text, moves it left one
        character position.

--->   moves the cursor one position to the right.

ctrl-I if the cursor is on a line number, inserts a blank line.
        If the cursor is within a line of text, inserts a blank
        character.

ctrl-D if the cursor is on a line number, deletes the line. If
        the cursor is within a line of text, deletes the character.

[up]   this arrow key or ctrl-K moves the cursor up one line.

[down] this arrow key or ctrl-J move the cursor down one line.

SYNTAX CHECK

[X] SYNTAX CHECK

Pressing the 'X' key will cause the text file in the work area to be
scanned for syntax errors. In doing so, all ".I" include directives are
expanded, using the files which they reference. If you use .I commands in
your text, be sure to save the file before you syntax check, because
syntax check will change the file in memory.

After expanding any .I commands and syntax checking the resulting LPL
statements, the message "SYNTAX CHECK SUCCESSFUL" will appear if no errors
were found.

If any errors were found, the text editor is entered with the cursor on
the statement that caused the error and the speaker will beep. No
checking is done beyond the first error that occurs.

PRINT A PARAMETER FILE

[P] PRINT FILE

Pressing the 'P' key from the text editor menu will print the current file
in the workarea. The standard printer slot is used (default of slot 1),
but may be changed by setting the parameter for the printer slot to a
different value.

PRINT ALL PARAMETER FILES

[A] PRINT ALL FILES

Pressing the 'A' key will print the entire contents of the parameter disk,
one file at a time, in alphabetical order.

RAM CARD UTILITIES

[R] RAMCRD UTIL

Pressing 'R' from the Locksmith main menu will display the RAM card
utilities menu.

All RAM cards found in slots 0 through 7 on the Apple are displayed along
with the amount of memory each contains. Apple //e and Apple //c have an
internal 16K of memory which appears as slot 0 although no physical slot 0
actually exists. Auxiliary memory, found on the Apple //e and built-in on
the Apple //c, is not tested by this utility at this time. An upgraded
RAM card test utility will include tests for auxiliary memory for the
Apple //e and Apple //c.

Beneath the SLOT RAM display are several menu item choices.

RAM CARD TEST

Selecting 'T' will perform a RAM card read/write test. If more than one
slot contains a RAM card you are prompted for the slot number. Enter a
slot number which contains a RAM card. Then select 'B' or 'E' to indicate
whether a basic or an extensive test should be performed. The basic test
runs 128 times faster than the extensive test but it is not as thorough.
After selecting the type of test, specify 'S' or 'C' for single test or
continuous test. Continuous mode should be selected for detecting
intermittent errors on the RAM card.

While the test is running, a message similar to the following one is
displayed on the screen:

TEST 0008 BANK 5 PAGE D6 ALT

This message shows the sequential test number (if in continuous mode), the
bank number, and page number currently being tested. Memory pages from D0
through DF have alternate memory pages, which, when selected, display
"ALT" after the page number.

The test can be terminated by pressing the 'ESC' key, suspended by
pressing the space bar, and continued by pressing the 'RETURN' key. Like
all Locksmith functions, the results can be printed by pressing control-Z.

If any errors were found, an error message similar to the following one is
displayed.

+ERROR WRITING AA(2A) AT FEB1 ON BANK 1

The message shows the test sequence number, the bank number, the address
of the error, what was written and what was read back in parenthesis. If
any other addresses in the bank of memory changed values spontaneously
(while other locations of the bank were being written), they will be
listed also, with the prefix "+" on the error message.

The extensive test writes each of the 256 possible values to each of the
locations of the RAM card, and then reads back the same location to insure
that the value was indeed stored correctly. It also checks all other

locations of the RAM card to make sure that no other locations were
modified. The basic test is the same as the extensive test, except that
instead of all possible 256 values, it uses two test values (55 and AA).

If any errors were found, you may want to consult the RAM card manual to
determine the location of the faulty RAM chips, or take the RAM card in
for servicing.

DUMP RAM CARD TO MEMORY

Selecting 'D' from the RAM card menu will allow you to dump the contents
of the RAM card, 16K at a time, into main memory, so that it can be
examined. The Locksmith disk and memory editor ('N' from the main menu)
can be used to edit the memory. 16K at a time is dumped to main memory
locations 2000 to 5FFF as follows:

2000-2FFF RAM card pages D0-DF ALT
3000-3FFF RAM card pages D0-DF
4000-4FFF RAM card pages E0-EF
5000-5FFF RAM card pages F0-FF

If the RAM card you have selected is larger than 16K, you will be prompted
for the bank number. For a 128K card, banks from 0 to 7 are valid.

LOAD RAM CARD FROM MEMORY

Pressing 'L' will load the RAM card from main memory. This function is
the reverse of the 'D' (dump) function. 16K at a time is loaded from main
memory to the selected bank of the RAM card.

SCAN DISK

[Q] SCAN DISK

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

Pressing 'Q' from the main menu will put you into the scan disk 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. Similiarly 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 scanning 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 you see some
evidence of sectors (short bursts of white), the read head experienced
"cross-talk" from adjacent half-tracks perhaps (but not necessarily) due
to a misalignment in either the reading drive or the drive that wrote the
diskette.

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 scan disk feature the more valuable you will find
it. This is because you will become more adept at interpreting the
results it gives you.

AUTOMATIC BOOT TRACER

Introduction to ABT

[A] BOOT TRACER

The Automatic Boot Tracer is intended for use by the more experienced
Apple programmer. It is actually a sophisticated debugger which can
simulate the operation of the 6502 in the Apple. Because disk reading is
simulated, it is possible to actually "boot" a disk (whether protected or
not) under control of this debugger, and trace the boot code of the
program.

Boot tracing, a normally manual and very tedious technique which is used
by the most sophisticated "hackers", can be performed automatically under
control of the Locksmith Automatic Boot Tracer.

To invoke the boot tracer, key 'A' from the main menu. You must have a
RAM card of at least 16K on your system for ABT to work. If you have an
Apple //e or Apple //c, the "built-in" 16K RAM will work.

Locksmith ABT will prompt you for the slot number of the RAM card. Key in
a digit from 0 to 7.

The ABT will be installed on the RAM card you choose, and the ABT will be
entered.

Note that in this manual, the ABT (automatic boot tracer) is also referred
to as the debugger and the simulator, since it actually simulates the
operation of the 6502, and can be used as a powerful debugger.

The screen will clear and a line of inverse text will appear on the top
line of the display. The ABT is now operating.

If you press reset at any time, you will be placed in the Apple monitor
and can reboot another disk by entering the slot number followed by
control-P. Be careful not to reboot a disk which will automatically load
over the ABT on the RAM card you selected.

Information Line

The top line of the screen which appears in inverse text is a one-line
status display which appears initially as follows:

FA62 CLD   A=00 X=00 Y=00 P=34 S=FD

The first 4 characters are the program counter (FA62 in this example).
The 6502 opcode at the program counter is also displayed (CLD in this
example). Next, the values of the A,X, and Y registers are displayed.
The "P=" value is the processor status register contents, and the "S="
value is the stack pointer.

At this time, press the R key followed by a key from A through X. Notice
that the information line disappeared and moved to another line of the
screen. You can put the information line on any line of the screen that
is convenient for the software you will be debugging / tracing. If you
don't want the information line displayed, you can place it on row Y or Z

(which are off the screen).

Idle Mode

The simulator is in "idle" mode at this time. That is, the program to be
simulated is not currently running, but is stopped at the address
displayed by the program counter.

Press control-C at this time to enable the processing of 65C02
instructions. This is necessary if you are running on an Apple //c or an
enhanced Apple //e.

Press the S key to start exectution under control of the simulator. The
ABT is now running simulated 6502 code. The simulator is now in "running"
mode. Note the rapidly changing program counter. The "beep" you hear
from the speaker may sound a bit different than the Apple "beep" which you
are used to, but that is only because under control of the simulator it is
slowed down considerably and sounds lower.

To stop the program being executed, press [control-Z]. You are now again
in "idle" mode. Control-Z is the default character used to stop execution
of the simulated program, but it can be set to a different "stop" key if
you need to be able to use control-Z with the software you are tracing.
To change the "stop" key, first stop the program being executed and return
to "idle" mode by pressing control-Z. Then press control-X followed by
any other key, and the other key will be used for the "stop" key.

To reset the "stop" key to control-Z, enter idle mode and press control-X,
control-Z.

Enter idle mode. Now press the space bar and watch the information line.
The space bar is used in idle mode to single-step one instruction. A "+"
or "-" will appear after each conditional branch instruction, depending on
whether the branch will be taken ("+") or will not be taken ("-").

While in idle mode, enter control-Y. You are placed in the system
monitor, and can enter any monitor commands such as "L" (to disassemble
6502 code). To re-enter the simulator, press control-Y, RETURN. Before
placing you in the system monitor, the simulator saved low memory pages 00
to 07 on its RAM card. After re-entering the simulator, this memory was
"refreshed", insuring that no memory was inadvertently changed while in
the system monitor.

To review the idle mode commands we have already learned:

Space-bar single steps one instruction. It can also be used to
"single-cycle" (see below).

"R" moves the information line to rows A through X.

"S" starts the simulated program running and enters "running" mode.

Control-Y enters the system monitor. To re-enter the simulator, press
control-Y again, followed by the Return key.

Control-X is used to change the program "stop" key, which stops the
program and enters idle mode.

Other idle mode commands:

"T" (trace subroutine) executes the simulated program until a JSR or RTS instruction is fetched.

Control-R causes a "simulated" reset to occur. The program counter is fetched from $FFFC.

Control-I causes a "simulated" IRQ interrupt.

Control-F turns off the "simulated" IRQ pending flag.

Control-N causes a "simulated" NMI interrupt.

Control-Q quits the simulator and returns to the system monitor.

Control-RESET also exits the simulator.

"1" is used to set single-cycle mode. In single-cycle mode, the space bar cycles one 6502 processor cycle at a time, instead of an entire instruction step.

"0" is used to set instruction-step mode. It is valid only when on an instruction boundary (not on a cycle in the middle of an instruction).

"B" turns the "beep" flag on and off. The beep is sounded when idle mode is entered.

"C" turns the "click" flag on and off. The click is sounded for every keystroke when not in "running" mode.

Control-C turns the 65C02 flag on and off. The default value for this switch is "off". If 65C02 instructions are to be simulated, this flag must be on. The Apple //e (enhanced version) and Apple //c both contain 65C02 processors and use 65C02 instructions in their resident ROM code. Note that the simulator itself does not use 65C02 instructions. You can therefore run 65C02 instructions on a normal 6502 processor.

"K" will take the next key pressed and place it in the keyboard character register. When instruction stepping through code that reads the keyboard, this key allows a convenient way to enter a keystroke to the program being traced, without entering the keystroke in "running" mode.

"ESC" is used to enter the simulator control menu. The simulator control menu is used to display and change internal simulator control information.

Simulator Control Window

Press the "ESC" key while in idle mode. The simulator control window is displayed, and the cursor appears in the upper left of the window.

Use the RETURN key, and the left and right arrow keys to move the cursor around the simulator control window. These keys only move the cursor and do not change any information in the window. To change data anywhere in the window, simply position the cursor over the value to change and

re-enter the desired value. To exit from the simulator control window and return to idle mode, press the ESC key again. If you wish to cancel any changes made in the simulator control window, you may press control-C instead.

Let's look at the simulator control window in detail.

The top line looks very much like the information line in idle mode, except that the program counter appears further to the right and no instruction is disassembled on the line. The number on the left of the line is used for single-byte reading, single-byte writing, and memory editing. Enter an address value followed by 'R' to read, 'W' to write (also specify value to write), and 'E' to edit, using the memory edit window.

To change display modes for the simulated program (text, graphics, hi-res, low-res, page1, page2, fullscreen, mixed), key in the address to toggle ($C050-$C057) and enter 'R'. When tracing a program in graphics mode, it is useful to put the information line on rows U,V,W, or X, and toggle mixed mode graphics. The simulator will display the information line on either text page 1 or 2, whichever is selected by the program being simulated.

Enter an address, followed by 'E' to enter the memory edit window.

While in the memory edit window, the memory is displayed in both hex and ASCII text. The cursor can be moved with the RETURN key and arrow keys. To change data, simply key in a new value in the appropriate address. The ESC key returns to the simulator control window, saving all changes to memory. If the changes made in the memory edit window are not to be made, enter control-C.

The second line of the simulator control window contains:

RU=65 0=I 1=I 2=I 3=S 4=I 5=I 6=D 7=I

"RU=65" This value (decimal 101), the "register update" value, represents the number of instructions that are simulated before the registers and program counter are updated on the screen, when in "running" mode. If this number is set small (01 for example), the registers will be updated after every instruction. This however causes the simulator to run less efficiently, because of the overhead involved in updating the information line.

Slot Specifications

The rest of the second line displays the slot numbers and how they are to be used. Because the simulator resides on a RAM board (indicated by 'S' in the slot display, for "SYSTEM"), it must know about all other RAM boards and firmware boards if it is to correctly simulate their operation. Initially, the slots will be set to 'I' (invalid). Any reference by the simulated program to these invalid slots will cause the simulated program to stop and control is passed to idle mode. Valid slot specification values are:

  'S' system (simulator) slot

  'I' invalid

'D' floppy disk drive

'A' RAM card of 16K or 32K.

'B' RAM card of 64K or more.

'F' Firmware card or ROM card.

'T' transparent.

If the specification for a slot is "transparent", any commands for the device in that slot will be given without any checking or conversion by the simulator. Transparent mode should not be used for:

Any devices such as RAM and ROM cards that bank select memory into the address range D000 to FFFF, which is used by the simulator.

Any devices such as disk drives which are timing dependent, as the simulator runs much slower than the 6502 in native mode.

Any devices which may use DMA (direct memory access) to modify memory from addresses $0000 to $07FF, as this memory is used by the simulator with a copy of the user's memory actually residing on the simulator's RAM board.

Address Compare Stop

The third line of the simulator control window starting with "PC" is the "PC compare stop" line. Up to four program counter values for "compare stop" can be specified. If the simulated program's PC equals one of these values, the simulator immediately enters idle mode. In addition, one "PC compare stop range" can be specified. To enter program counter stop values or a range, change the number (initially "0") to the number of stop addresses to be entered and then enter the addresses in the space provided. To disable PC compare stop, set the number back to "0".

The "MR" line of the simulator control window is the "memory read address compare stop" line. Like the "PC compare stop" line, up to four addresses and one range can be specified. Whenever the simulated program attempts to read one of these addresses, either by direct addressing, indirect addressing or stack fetch, the simulator enters idle mode.

The "MW" line of the simulator control window is the "memory write address compare stop" line. Idle mode is entered whenever the simulated program attempts to write to one of the addresses specified here.

Program Counter Swap

The "PCSW" area of the simulator control window is the "program counter swap" control area. Up to four address pairs can be specified here. If the simulated program's PC equals the first value of a pair, the PC is immediately set to the second value, and execution continues. This is very useful for eliminating slow timing loops, which are unnecessary in the simulator. Initially, 3 pairs of PCSW values are given. They are:

FCA8 FCB3 - This nullifies the monitor wait routine.

BA00 BA10 - This nullifies the DOS 3.3 seek delay routine.

BD9E BDAB - This nullifies the DOS 3.3 motor-on wait routine.

Program Counter Trace Table

The bottom eight lines of the simulator control window contain the PC trace table. The last 64 values of the program counter are kept here, so that when the simulated program is halted, a history of the last 64 instructions can be examined.

Program Halts

A program running under control of the simulator halts and the simulator enters idle mode whenever one of the following conditions is met:

The "stop" key is pressed by the user.

An invalid 6502 or 65C02 opcode is encountered. "???" is displayed in the information line where the opcode is normally displayed.

A JSR or RTS instruction is fetched while running with the "T" (trace) command.

A read or write to the device select addresses of a slot marked as "I" (invalid) in the slot table.

A compare stop occurs for PC, MR, or MW, while running.

An attempt is made to write to the floppy disk.

An attempt is made to reference certain I/O addresses. Among these are $C060 and $C068 for either read or write.

Note that in the case of a compare stop for MR or MW or an invalid device select reference, that idle mode is entered with the PC containing the address of the instruction <after> the one that caused the compare stop. Look at the last address in the trace table to find the correct address.

Internal Operational Notes

A few notes on the internal operation of the boot tracer / simulator / debugger:

Floppy disk reading is simulated by reading in an entire track of nibbles and passing them one at a time to the simulated program requesting them. Each time the simulated program requests a nibble, the next nibble in the buffer is returned. The simulated program never has to wait for a nibble by polling the high-order bit of the disk register. Because of this, framing bit timing is not preserved. In addition, the track is not synchronized to any other track upon reading. Floppy disk writing is not supported.

When reading a floppy disk, the simulator maintains the nibbles of the
most current track on the simulator's system RAM card. This track image
is valid until either the slot or drive number is changed or reselected,
or the read/write head is stepped to a different track. Only if the
current track image is invalid will the real floppy disk be read again.
Therefore, if the user performs a "CATALOG" operation while under control
of the simulator and then changes the diskette and performs another
"CATALOG" operation, the catalog information from the first disk will
still be displayed because the catalog (located entirely on track $11) did
not cause the head to change tracks and invalidate the track buffer. To
manually invalidate the track buffer, change the slot specification to 'I'
and back to 'D' while in the simulator control window.

The simulator has code for "sector assist" built-in. This means that when
the simulated program requests a nibble followed immediately by testing
for disk register ready and a compare for $D5, the simulator immediately
finds the next $D5 in the track buffer and returns it to the simulated
program, instead of requiring the program to ignore each nibble until the
value $D5 is found.

The paddle I/O addresses ($C064-$C067 and $C06C-$C06F) are correctly
simulated if the code that accesses the I/O addresses is similar to the
monitor routine at $FB1E (PREAD). If the reference is not similar to the
monitor routine, idle mode will be entered.

CERTIFY   DISKETTE

[C] CERTIFY DSK

The disk certify function of Locksmith is invoked by pressing 'C' from the
main menu. It can be used to check a diskette for physical damage prior
to use.

THIS UTILITY DESTROYS DATA ON THE TARGET DISK.

Enter the output drive number, the starting track, ending track, and track
increment. Insert the diskette you wish to test, and press the space bar
in response to the prompt to insert the diskette.

This function works by writing a specific pattern of nibbles 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 is questionable, especially if several adjacent tracks display
errors.

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 (*).

16 SECTOR UTILITIES

This function gives you access to several utilities designed to work with normal 16 sector (generally unprotected) disks. These can be DOS 3.3, PRODOS, CP/M, or any standard 16-sector disk.

To invoke any of the 16-sector utilities, first press 'U' from the main menu.

16 SECTOR DISK VERIFY

[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.

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.

A number indicates that the sector was read correctly, but that it took several rereads to read it correctly.

16 SECTOR FORMAT

[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.

THIS FUNCTION 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 protected disks currently on the market use tracks $06.5 through $22.5. These tracks are formatted with track numbers $06 through $22.

16 SECTOR DISK COMPARE

[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.

16 SECTOR SYNC SIGNATURE

[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 varaiations 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
DOS INIT	ODA741EB85...	(-3)
Locksmith Format	048C048C...	(+4)
Locksmith Fast Backup	OFEDCBA987...	(-1)
Penulta Copy	OECA8...	(-2)
CopyWriter (no verify)	0000000....	(+0)
CopyWriter (verify)	OFEDCBA987...	(-1)
Disk Muncher	OD852FC964...	(**)
Pack Rat	OD85630DA5...	(**)

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):

    OFEDCBA9876543210FEDCBA9876543210FE

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:

    OFEDCBA9873210FEDCBA6543210FED98765
              ^   ^   ^

Note that the progression is the same (-1), except that at 3 places
(identified by arrows below the signature), where the progression is (-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 BACKUP 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.

DOS 3.3 UTILITIES

[D] DOS3.3 UTIL

Pressing the 'D' key enters the DOS 3.3 utilities menu. Select any option
from this menu and you will be prompted for the input drive to use. The
input drive will also be used to rewrite data, if the function requires.

CATALOG DISK

Pressing 'C' from the DOS 3.3 utilities menu will display the catalog of
the disk in the current drive. Deleted files are marked with a "D" in the
front of the file name.

LOAD DOS FILE INTO MEMORY

Pressing the 'L' key will allow you to load a DOS file into memory for
editing with the disk editor. After answering the prompt for the
filename, the file is read into memory from $2000 to $7EFF. The TSL
(track / sector list) is placed in memory at $7F00, so that the file can
be rewritten to disk from the disk editor.

SHOW DISK SPACE MAP

Pressing the 'M' key will display a disk space map on the screen. The
sectors are displayed down the left and right sides of the screen. Free
sectors are indicated by "-" and used sectors are indicated by "#". Press
the space bar to enter the utilities menu.

FIX SECTOR COUNTS

Due to errors within DOS 3.3, the sector count of a file in the catalog
sometimes gets set incorrectly, usually to zero. Pressing 'K' from the
DOS 3.3 utilities menu will allow you to automatically correct the sector
counts on the disk. The sectors for each file are counted and compared to
the information in the catalog. If the counts are not equal, the filename
and the counts are displayed and you are prompted to press the space bar
to correct the sector count for the file. Press the space bar to correct
the sector count for the file or press the ESC key to abort the function.

VERIFY VTOC INTEGRITY

The VTOC (volume table of contents) is a bit-map of the free sectors of
the disk volume. The catalog is a list of names, each with a chain of one
or more TSLs (track/sector lists) which define the rest of the sectors in
the file. If a user's program fails while files are open for output and
they are not properly closed, the VTOC and catalog can get out of step.

Pressing 'V' from the DOS 3.3 utilities menu will perform a VTOC/catalog
integrity verification. The display will appear much like the disk space
map display (function 'M') and contain both "-" (free sectors) and "#"
(used sectors). In addition, 3 types of errors are detected and

displayed:

Sectors which are allocated in the bit map but not used in any files are
displayed with an inverse "A" (allocated, not used).

Sectors which are used in a file but not allocated are displayed with an
inverse "U" (used, not allocated). This is a critical situation because
the file data of the affected file can be overwritten if another file is
allocated over it. If this situation occurs, do not use this disk.
Immediately copy the files to a blank disk using the FID utility found on
the system master diskette.

Sectors which are allocated to more than one file at a time are displayed
with an inverse "2" (2 files allocated) or an inverse "3", etc. This
situation indicates that one of the files overwrote the other one. You
should copy the files to a blank disk using the FID utility found on the
system master diskette.

REMOVE DOS FROM DISK

Pressing the 'R' key will allow you to remove the DOS boot code from the
disk, freeing up 32 more sectors. Note that only tracks 1 and 2 (16
sectors each) are freed. Track 0 could be freed but DOS is unable to
allocate files on track zero even if it is marked free.

A disk space map is displayed. After pressing the space bar, DOS is
removed from the diskette, the VTOC is rewritten, and a disk space map is
again displayed. You will notice that tracks 1 and 2 now contain free
sectors.

UN-DELETE A FILE

Pressing the 'U' key from the DOS 3.3 utilities menu will allow you to
un-delete a file which was previously deleted. Enter the filename when
prompted. If the filename cannot be found, an error message will be
displayed. If the file has not been overwritten by another file, it will
be un-deleted and returned to the catalog. If the file has already been
overwritten by another file, a message will be displayed and the undelete
will be aborted. Press the space bar to return to the utilities menu.

ALPHABETIZE CATALOG

Pressing the 'A' key will allow you to alphabetize the filenames in the
catalog. After the names have been read into memory and sorted, you will
be prompted to press the space bar to allow the alphabetized catalog to be
written back to the diskette.

ENCRYPT A FILE

After pressing the 'E' key (encrypt file), you will be asked to supply the
name of a file to encrypt. The catalog is checked to verify that the file
exists. You are then asked to supply a password for the encryption. The
password you enter will not be displayed on the screen, but you will need
to enter the password again to verify that you entered it correctly.

If both password entries are identical, the file is encrypted using the
password you supplied. Do not forget the password – it is the only way to
decrypt this file. If the encrypted file is displayed, it will appear as
meaningless garbage. If the password is forgotten, the file will probably
remain that way forever.

DECRYPT FILE

Pressing 'D' from the DOS 3.3 utilities menu will perform a file
decryption function. This is the opposite of the encryption function
described above. The procedures are the same as the ones for encryption
of a file. If you decrypt using the wrong password, the file will remain
as meaningless garbage.

Note that if a file is encrypted twice (perhaps by two individuals, each
entering their own password), that the decryption should be done in the
reverse order. For example, if a file is encrypted first with password
"LOCK" and then with password "SMITH", the order for proper decryption is
first with the password "SMITH" and then with "LOCK".

After data is encrypted, it should be backed up. This is because if an
attempt is made to decrypt the file with the wrong password, the file is
scrambled more. To reverse a decryption in error, perform the encryption
using the same password, and vice versa.

ADVANCED DISK RECOVERY

[X] DSK RECOVER

When you notice a problem with your disk drive which causes disk I/O errors, either on a permanent or intermittent basis, it's best to have your drive checked.

What if you have important data which you've backed up using the mis-aligned or faulty drive?

Locksmith Advanced Disk Recovery (ADR) will recover data which cannot be read from a diskette due to one of three problems: track mis-alignment, off-center hub clamping, and improper disk speed.

Proper track alignment is more critical than disk speed when reading and writing data to a diskette. While disk speed, once set, usually remains constant; track alignment, because it is affected by normal mechanical wear of the drive mechanism, can change with time.

While disk drive track alignment should only be attempted by a qualified technician, there are some warning signs that indicate improper track alignment of your disk drive. Track mis-alignment should be suspected if data which was written on a specific disk drive is difficult or impossible to read using a different disk drive. This symptom does not indicate which drive is mis-aligned, but merely that an incompatibity exists between the two drives. Sometimes the mis-alignment problem is compounded because the writing and reading disk drives are mis-aligned in the opposite direction. For example the writing drive may record a track 1/4 track too low and the reading drive may attempt to read the track 1/4 track too high. The resulting difference of 1/2 of a track is enough to make the data impossible to read. Even if the reading drive alignment is correct, the resulting difference of 1/4 track is enough to make the data read unreliably, especially on the inner tracks of the diskette (higher track numbers) where the data density is greater.

While track alignment problems usually manifest themselves gradually over time, a problem due to off-center hub clamping (more common on older drives or diskettes without hub rings) is usually quite sudden and severe. When the disk drive door is closed, the hub clamps down on the center hole of the diskette. If the media is slightly off-center when the door is closed, it is usually centered as the hub clamps it into position. Sometimes the diskette is clamped off-center. If this is done before reading a diskette, the resulting abnormal head-seek noise is usually noticed and simply opening and closing the drive door will re-center the diskette properly. If, however, the diskette is clamped off-center prior to formatting and writing, the diskette will be formatted and written with no adverse symptoms. The data on the diskette can even be read and verified, however, once the diskette is removed, the data will be unreadable because it was written with the diskette off-center. Any attempt to read this off-center diskette will fail.

Writing sectors to a diskette on a drive which is running too fast can cause the following sector's address field to be over-written. Although the data field of the sector is intact, the missing address field will cause an I/O error when attempting to read the sector.

Locksmith Advanced Disk Recovery will recover the data from a diskette

recorded either off-center, with bad track alignment, or with missing sector address fields, and write the data to a blank formatted disk. You must previously format the disk upon which you wish to place the recovered data. (You may use the Locksmith 16-sector disk format utility to do this.)

Specify the source and target drive numbers and the starting and ending track numbers (usually 00 to 22) and Locksmith Advanced Disk Recovery will read the problem disk on the integral as well as half-tracks and quarter-tracks as needed to recover the data sector by sector. If you wish to perform a test run to determine if Locksmith ADR can recover the data without actually writing the recovered data, specify a target drive number of zero. Six tracks are processed at a time. The recovered data is written to the target disk, unless zero was specified for the target disk drive, in which case no writing of recovered data is performed.

The track status display area at the top of the screen shows the status of writing the data to the target disk. The bottom part of the screen is used as the sector status display. The numbers appearing in inverse text down the left and right sides of the sector status area are the sector numbers for each track. The order of the sectors in the sector status area is the same as the sector order on the track (0,7,E,6,D,....). A single status character is placed in the sector status area for each sector to indicate the status of the sector read:

A period (.) indicates that the sector was read from the correct track with no errors.

An (inverse A) indicates that the address field for the sector was never found.

An (inverse D) indicates that the address field was found but the data field was not read correctly.

A plus sign (+) indicates that the sector was correctly read from a higher track, half-track, or quarter track.

A minus sign (-) indicates that the sector was correctly read from a lower track, half-track, or quarter track.

A semicolon (;) indicates that the sector was missing an address field, but that this problem was corrected.

Note that only the (inverse-A) or (inverse-D) indicate an unrecoverable error.

The following figure shows a sample screen after Advanced Disk Recovery has recovered data from an off-center disk:

```
                DSK RECOVER
    .00
    .25
    .50
    .75
    HEX  0000000000000000011111111111111110000
    TRK  0123456789ABCDEF0123456789ABCDEF0123
      0++--.+.-.+.-+.-.+.-.+.--+.--+.--+.-0
      7+.--++--.+-.+.-++--.+.-.+.--+.-.+.-7
      E+.-.+.-.++-.+-.+.--.+--.+.-.+--.+--E
      6+.-.+.-.+.-.+.-.+.-.+.--.+--..--.+--6
      D+--.+.-.+.-+.-.+.-.+.-.+.--..--+.--D
      5.--++--++.-+.-++--.+.-.+.-.+.-.+.--5
      C.-.+.-.++--+.-+.--.+--.+.-.+--.+.-.C
      4.-.+.-.+.-.+-.+.-.+.--.+--.+--.+--.4
      BD-++.-.+.-.+-.+.-.+.-.+.--.+--+.--+B
      3A.++--++.-...-.+--.+.-.+.-.+.-.+.--+3
      AA.+.-.++--+.-+.--+.--.+.-.+.-.+--.+A
      2A.+.-.+.-.+.-+.-.+.--+.--+.-.+.-.--+2
      9.++-.+.-.+-.+.-.+.-.+.--+.-+.--.+.-9
      1.++--++.-.+-.+--.+.-.+.-.+.-.+.-.+.1
      8.+.-.++--++.-+.-+.--.+.-.+.-.+--.+.8
      F.+.-.+.-.+.-+.-.+.--+.--.+--+.--.+-F
```

Note that when the sectors are arranged in the same order in which they
appear on a track, the plus signs and minus signs on a track appear in
clusters.   This indicates which sectors were found on higher track numbers
(inward toward the center of the disk) and which were found on lower track
numbers (outward toward the edge of the disk).

It is common, when recovering data from an off-center disk, to have some
sectors on track 00 which could not be recovered, because to recover them
would require reading below track zero, which is physically impossible.
This is usually not a problem, since DOS does not normally use track 00
for storing user data.

DISK SPEED TEST

[S] DISK SPEED

Pressing 'S' from Locksmith's main menu will put you into the 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,
irregardless 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.

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 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.

ERASE DISK

[E] ERASE DISK

This function is used to entirely or partially erase a diskette.

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 disk which has been previously formatted.

INSPECTOR / WATSON

[I] INSPECTOR

If you have previuosly loaded Inspector/Watson onto your RAM card with the main menu 'L' (load RAM card) function, 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, with the exception of the ESC function (see next paragraph). 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 information regarding the use of Inspector/Watson, please refer to their respective user's manuals.

LOCKSMITH PROGRAMMING LANGUAGE

Locksmith Programming Language (LPL) is a tool which the Locksmith user can use to specify how Locksmith is to perform certain functions. With it, the user can set up a procedure to backup a difficult-to-copy disk, automatically search for and change data on a disk, repair damaged data, etc. Almost anything that can be done with Locksmith can be done automatically by specifying the proper LPL commands.

LPL commands, or statements, are collected into a file. This file can be entered from the keyboard by the user or may be loaded from a special-format parameter disk (which is included with your Locksmith 6.0).

THE INCLUDE (.I) COMMAND

LPL files can be saved with their own name on the parameter disk. These files can refer to other named LPL files on the parameter disk and can include parts of these files or entire named files within them. In this way you can use a technique used in another file without repeating the LPL statements, simply by referring to the name of the other file. For example, if a file called "BRODBUND" exists on the parameter disk, and it contains LPL statements which you wish to use in the current file, use the following statement in your LPL file:

    .I BRODBUND

The ".I" must appear as the first characters on the line, and be followed by a space. All of the statements in the named file "BRODBUND" would be included when the file you are entering was expanded by the text editor during either a syntax check or backup operation. You can save your file to the parameter disk either before expansion (to save disk space) or after expansion. Included files can, in turn, refer to additional included files. If you wish to include part of a file, you can specify the starting and ending line numbers after the file name. For example:

    .I TESTFILE,4-1F

The above example would include only line numbers 4 through 1F from the file named "TESTFILE".

There is no limit to the number of .I include statements which you can use, except that the total number of lines in the file cannot exceed FF (decimal 255).

Each line entered into the LPL file is given a line number (the line number appears to the left of the line in "inverse" characters). The line number is printed in hexadecimal to save space on the display, but the line numbers are not important, except when including text from other LPL files with the .I directive. An LPL line can be up to 38 characters long. The following is an example of an LPL line:

FIND D5 AA 96

The above example is an LPL line which consists of a single LPL statement. You can enter several statements on a single line by separating them by colons (:). For example:

SLOT 6 : IN.DRIVE 1 : OUT.DRIVE 2

The spaces before and after each colon are not necessary, but are included
for easier readability. If a statement is too long for the current line,
it can be continued onto the next line by coding a "-" as the last
character on the line. For example:

DEPOSIT (DF) (F4) (DF) (D4) (FF) -
D5 AA 96 AA AA

You may insert remarks or comments into your file at any time to help
document what you are doing. An asterisk (*) causes the rest of the
statement to be ignored. Note that the comment is a statement, and that
any statements following it on the line are evaluated. For example:

* SET SLOT NUMBER : SLOT 6
SLOT 6 : * SET SLOT NUMBER

Both the above lines are equivalent. Blank lines can be included anywhere
for ease of readability and are ignored.

LPL STATEMENTS

LPL statements consist of "tokens" which can be thought of as words in a
sentence. Tokens are separated by blanks. The first token used in a
statement determines the type of the statement. A token can represent a
variable name or parameter name, a constant, or a processing routine name.
Statements can be grouped into the following categories:

        comment or blank line
        statement label
        assignment statement
        processing

Any statement may have a statement label preceding it. Statement labels
are optional and are used for branching within the LPL file with the GOTO
statement. A statement label begins with the keyword "LABEL", and is
followed by a name. Label names can be of any length, and are any
sequence of alphabetic or numeric characters or the period, with the first
character being alphabetic. The following are examples using valid
statement labels:

LABEL READ.TRACK.AGAIN :

GOTO READ.TRACK.AGAIN

LABEL A2.ERROR : PRT "LENGTH ERROR"

GOTO A2.ERROR

TYPES OF CONSTANTS

Several types of constants are used within LPL. They are described here,
with examples of each.

Single byte constant (hex):

D5
6
00
(FF)
Single byte constants, when enclosed in parenthesis, indicate self-sync.

Multiple byte constant (hex):
(FF) D5 AA 96
00 10 20 30 40 50 60 70 80

Double byte constant (hex):
1A70
Double byte constants usually represent addresses or lengths.

Track value constant (hex):
12.5
1A.75
11.0
0.
Track value constants contain a decimal point.

Single byte character constant:
"X"
'X'
Note that either single or double quotes can be used.

Multiple byte character constant:
"DON'T FORGET TO COVER THE NOTCH."
If a quote is to appear in the constant, use the other quote to delimit
the string.

Multiple byte constants can consist of a mix of hex and ASCII data:
89 'GRD' 8D

Special constant:
?
The "?" represents a "don't care" value, when used in search patterns.

Flag constants:
YES
NO
ON
OFF
These constants are equivalent to 0 and FF, but are preferred because of
their readability.

TYPES OF VARIABLES

Variable names used within LPL usually have a specific use for each name,

although some general-purpose names have been provided for the user. Several types of variable names are provided within LPL. They are described here, with examples for each.

Single byte variable:
SLOT
IN.DRIVE
OUT.DRIVE

Multiple byte pattern/string variable:
GR.CHARS
PAT1
SYNC.PAT
Pattern/string variables are variable length, and have a one byte length as the first byte of the variable. Although pattern/string variables have a variable length, each is allowed a maximum length of 15 (decimal).

Multiple byte variable:
NIB.TRANS
SEC.TRANS
INV.TAB
These variables refer to multiple-byte areas of memory of fixed length, such as lookup tables. Because they are fixed length, there is no length byte prefix associated with the variable.

If you have a need to refer to a byte within a multiple byte variable, you can code a plus sign followed by the displacement into the table. For example:
NIB.TRANS +2F

Double byte pointer variable:
START
END
TRK.LEN
CURSOR
PTR.W
PTR.X
PTR.Y
PTR.Z
Double byte pointer variables can also represent lengths.

Track variable:
BEGIN.TRK
END.TRK
INCR.TRK
SYNC.TRK
Track variables can represent integral track values, or 1/4, 1/2, or 3/4 tracks.

Flag variable:
SYNC
COUNT

SHOW.ADDR
Flag variables are single byte variables that contain a flag of YES/NO, ON/OFF, or FF/0 and act as switches which can be set and tested. Values other than these may have unpredictable results.

THE ASSIGNMENT STATEMENT

The assignment statement is used to assign a value to a variable. The variable can be assigned the value of a constant or another variable.

The variable to be assigned and the variable or constant to which it is assigned must be of the same type. For example, a pointer variable can only be assigned to another pointer variable or a two-byte value (3 or 4 hex digits).

Some examples of valid assignment statements are:

START CURSOR
Set the variable START (start of track data) to CURSOR (current cursor location).

CURSOR 2040
Sets the value of CURSOR to point to 2040.

SLOT 6
Sets the one-byte variable SLOT to the value 6.

BEGIN.TRK 12.5
Sets the track variable BEGIN.TRK to the track value 12.5.

CUR.TRK SYNC.TRK
Sets the track variable CUR.TRK to the track variable SYNC.TRK.

COUNT YES
Sets the flag variable to the flag value YES.

PAT4 D5 AA 96
PAT4 PAT7
Both of the above string variable assignments are valid.

GR.CHARS 89 'GRD'
Sets the string variable GR.CHARS to the 4-byte string consisting of hex 89 (control-I character) followed by the ASCII data 'GRD'.

DF.HDR3 B4 : AF.HDR3 DD
These assignment statements set the third data field header byte to B4 (normally AD) and the third address field header byte to DD (normally 96).

```
AF.TRL1.TEST IGNORE.TRL
AF.TRL2.TEST IGNORE.TRL
DF.TRL1.TEST IGNORE.TRL
DF.TRL2.TEST IGNORE.TRL
```
These assignment statements patch RWTS to ignore the trailer nibbles of
the address field and data field.

```
AF.TRL1.TEST NORM.TRL1
AF.TRL2.TEST NORM.TRL2
DF.TRL1.TEST NORM.TRL1
DF.TRL2.TEST NORM.TRL2
```
These statements patch RWTS back to normal trailer nibble checking.

```
AF.CSUM.TEST IGNORE.CSUM
DF.CSUM.TEST IGNORE.CSUM
AF.CSUM.TEST NORM.AF.CSUM
DF.CSUM.TEST NORM.DF.CSUM
```
These are used to patch RWTS to ignore or to use the address field and
data field checksums.

```
AF.CSUM.SEED 00
DF.CSUM.SEED 00
```
These statements set the checksum seed values for RWTS read address field
and read data field.

```
AF.HDR1 D5
AF.HDR2 AA
AF.HDR3 96
AF.TRL1 DE
AF.TRL2 AA
DF.HDR1 D5
DF.HDR2 AA
DF.HDR3 AD
DF.TRL1 DE
DF.TRL2 AA
```
These statements set the default header and trailer values for address
fields and data fields which are used for reading.

```
DF.HDR1.WRT D5
DF.HDR2.WRT AA
DF.HDR3.WRT AD
DF.TRL1.WRT DE
DF.TRL2.WRT AA
DF.CSUM.SEED.WRT 00
```
These statements set the header and trailer nibbles to be used when
writing a data field. The last statement sets the seed to be used for the
calculation of the checksum for writing the data field.

PROCESSING ROUTINES

LPL processing routines (referred to as "algorithms" in earlier versions
of Locksmith) are routines that the user can invoke on demand using LPL.
Some of the routines are complex and are driven by many variables, while
others are simple and operate using only one or two variables. Some
routines use parameters consisting of constants or pointers passed to them
and some accept no parameters at all. The valid syntax of each processing
routine is described in an addendum to this manual, but we will describe
some of the processing routines here and give some examples of their use.
The Parameter Diskette included with your Locksmith is an excellent source
of examples for the use of LPL processing routines and variable names.
You may also wish to refer to a "Quick Reference of LPL Names" in the
addendum to this manual for a complete list of all variable names and
processing routine names.

```
ABORT "DATA NOT FOUND"
```
Aborts the current operation and displays the message on the screen, then
returns to the main Locksmith menu.

```
PAUSE "ENTER 1,2, OR 3:"
```
After a key is pressed, the value of the key is placed in the single-byte
variable KEY.IN

```
PRT "DISK MUST BE WRITE/ENABLED"
```
Prints the message on the screen. If the last character of the message is
a blank, the cursor remains on the same line for additional output.

```
SHOW SLOT
```
Prints the value of the variable name SLOT. If the flag variable
SHOW.ADDR is YES or ON, the address of the variable SLOT is also printed.

```
GOTO READ.AGAIN
```
Branches to the label READ.AGAIN elsewhere in the LPL program.

```
FIND D5 AA 96
```
Causes a search from the current CURSOR pointer until the END pointer,
searching for the string D5 AA 96.

```
FIND PAT4
```
This is the same as the previous example, except that the search is for
the contents of the string variable PAT4. If a processing routine accepts
a string variable, it also accepts an explicitly coded string.

```
ERROR '5'
```
This causes the character '5' to be placed in the current track of the
status display area.

```
PAT4 D4 DD FF
PAT5 D5 AA 96
CHANGE PAT4 PAT5
```
This processing routine changes all occurrences of the first string to the
second for the entire range of START to END. Note that the CHANGE
statement and other statements that accept two strings as parameters do
not accept multiple byte constants. For example, the following use of the
CHANGE statement is invalid:

```
CHANGE  D5  AA  96   PAT4
CHANGE  PAT4   D5  AA  96
CHANGE  D5  AA  B4   TO   D5  AA  96


COPY  0  22  1     (invokes  track  procedure)
FCOPY  0  22     (Fast  Disk  Backup  copy)
```

ERASE
This causes the track to be erased on the output drive.

FORWARD TRK.LEN
BACK TRK.LEN
These statements cause the CURSOR to be moved forward or backward by the
length determined by the variable TRK.LEN. These routines can also be
coded with a 2-byte constant. For example:
FORWARD 024E
BACK 0042

CODE AD CUR.TRK 0A 0A 0A 0A 60
The CODE processing routine is provided for the 6502 assembly language
programmer. The parameters supplied are decoded, placed in a contiguous
area of memory, and given control with a JSR instruction.

PRT "ASCII TEXT TO PRINT"
The "PRT" routine prints data on the display screen. If the last byte of
the string is a blank, no new line is started. In this way, you can
display text with the value of a variable name. For example:
PRT "TRACK INCREMENT IS " : SHOW INCR.TRK

Some processing routines function slightly differently depending on
whether in "nibble-mode" or in "byte-mode". If the last read or write
command was NREAD or NWRITE, then nibble-mode is in effect. If the last
read or write command was SREAD, SWRITE, TREAD or TWRITE, then byte-mode
is in effect.

The routines which function differently depending on whether in nibble
mode or byte mode are FIND, VER, and CHANGE.

In nibble mode, a zero value in the pattern specified acts as a "don't
care" value. It can be coded as either "?" or "0".

In byte mode (used when reading or writing sectors), the zero value in a
pattern is significant.

For example:

NREAD : FIND D5 00 96

The above statements perform a nibble-read of the current track, and place
the CURSOR on the first occurrence of a D5 value, followed by any value,
followed by a 96 value.

The statements:

TREAD : FIND D5 00 96

The above example performs a track-read of the current track, and places
the CURSOR on the first occurrence of a D5, followed by a 00, followed by
a 96.

In byte-mode, "don't care" values in pattern search strings are not
allowed.

The following processing routines accept either string variables or
multiple byte constants:
FIND
VER
REP
DEPOSIT

The nibble or byte mode can be changed manually by setting the flag
variable BYTE.MODE to YES if byte mode is desired, or to NO if nibble mode
is desired.

TRACK-PROCEDURES

A track procedure is a sequence of LPL statements that are to be executed
for each track that is being copied. (One may consider the entire LPL
file as a disk procedure, because it is involved with processing the
entire disk to be copied.) The track procedure is defined with the
BEGIN.PROC and END.PROC statements. The track procedure is then later
invoked for each track to be copied, by a processing routine which invokes
the track procedure. There is currently only one routine which invokes
the current track procedure. The COPY processing routine (either
nibble-copy or sector-copy) invokes the current track procedure. The
"current" track procedure is the last one which was defined. If no track
procedure was defined in the LPL file, then the default Locksmith track
procedure is used.

For example:

BEGIN.PROC

  ...
  ... track procedure A

  ...
END.PROC

COPY 0 8 2
COPY 12 18 2

BEGIN.PROC

  ...
  ... track procedure B

  ...
END.PROC

COPY 1 9 2
COPY 13 19 2

In the above example, the first two COPY processing routines will use
track procedure A, and the second two COPY routines will use track
procedure B.

The LPL statements within the track procedure can be used to process a
track in one of two modes. Either in nibble-mode or sector-mode.
Nibble-mode processing routines are NREAD, NWRITE, NVERIFY, which process
a track in the form of nibbles. Sector-mode processing routines are
TREAD, TWRITE, TVERIFY, which process a track in the form of 16 separate
sectors.

The format of the COPY command is:

    COPY <begin.trk> <end.trk> <incr.trk>

where <begin.trk> is the starting track, <end.trk> is the ending track,
and <incr.trk> is the track increment. These values are stored in the
variable names BEGIN.TRK, END.TRK, and INCR.TRK, respectively.

If tracks are to be synchronized or nibble-count preservation is to be
performed, set the variable names SYNC or COUNT to the value YES. The
SYNC and COUNT keywords are not specified on the COPY command, as in
version 5.0 of Locksmith.

LPL ERROR CODES

During execution of the LPL program, some errors may be encountered which
cannot be checked for during syntax check of the LPL statements.

If an error is encountered during execution of LPL code, processing
immediately stops and the following message is displayed on the screen:

    LPL CODE ABORTED

     ERROR CODE xx

where "xx" is one of the following values:

01 GOTO statement encountered without finding the matching LABEL
statement.

02 PROC.BEGIN statement encountered while already within a track
procedure.

03 PROC.END statement encountered while already outside of a track
procedure.

04 COPY statement encountered within a track procedure. The COPY
statement invokes a track procedure and cannot occur within one.

05 USE.DEFAULT.PROC encountered while within a track procedure.

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 AE  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:BA 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 FF  7E:BF FF  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

DATA FIELD NIBBLE ENCODING TABLE

The following translate table is used for calculating data field checksums. It is described in the chapter on the disk 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
```

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 appendix 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 were changed to some other value. (see "History of Locksmith and Copy Protection" chapter of this manual)