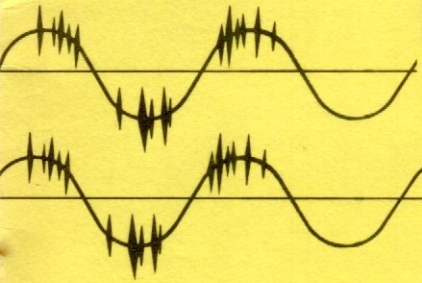


NIBBLE NEWS

VOLUME 3 / ISSUE 6 1984

IN THIS ISSUE:

STATE OF THE ART

NIBBLES AWAY was first introduced to the public in 1980. The ensuing popularity that followed was due in part to the comprehensive editing features and detailed documentation contained in the package. No other disk utility on the market came close to competing with all its features, making NIBBLES AWAY the #1 utility in the field.

NIBBLES AWAY II followed with new features, improved documentation, and a newsletter supporting its many users.

NIBBLES AWAY II remained 'State of the Art' for years, with only minor enhancements to its structure, while competitive products released many costly versions to their products trying to keep up with advancements being made in understanding the Apple II disk structure.

With the increased knowledge of the Apple's disk structure came protection techniques designed to foil the backup capability of NIBBLES AWAY II. Although the capability existed in NIBBLES AWAY II to edit and reproduce advanced techniques, many users lacked the knowledge required to successfully obtain a working backup.

NIBBLES AWAY II maintained its 'State of the Art' performance quite well without a major rewrite due to the planning and foresight that went into its development.

Now, after many requests from our customers, comes a NEW release of NIBBLES AWAY, with easy to use features that a novice can understand, as well as an advanced programmable language (that can actually be used without a degree) for the more adept user.

The article entitled 'PROTECTION UPDATES' will explain in brief some of the planned features in our new release.

Steve Pierce, President

PROTECTION UPDATES

In the past year or so there have been several new methods of protection which have come into use by various software manufacturers. Many of these are simply variations on old stand-by schemes, and others are completely new. More and more there a systems appearing which take advantage of features of the hardware which are undocumented and little known.

This makes it very difficult for a back-up program to determine the type of protection being used and the method required to decode it. NIBBLES AWAY II has a fairly structured approach to making back-ups of diskettes. This approach, while having a very wide range of effectiveness, can sometimes be to general for specialized cases. In the development of NAII, we chose the backup system which would require little or no user intervention for most back-ups, with changeable parameters so that more difficult disks could be backed-up.

With so many specialized protection systems on the market, it would be very difficult to add features to the current NAII to make it adapt to each of these. Since each one would have to be treated separately, the size of the program would very quickly get out of hand.

In our continuing development of the best all-around disk utility program, we have determined that the next generation of NIBBLES AWAY must incorporate some system whereby its internal approach to discovering a diskette's protection scheme can be altered easily and broadly enough so that many more types of protection can be handled.

This involves the creation of a type of language dedicated to backing up disks. In the same way that BASIC or PASCAL can read and write information from text or data files, this language will be able to read and write disk information, and easily determine the protection being used.

On the surface the language will appear very much like the current NAII; that is, the menus and functions will be very similar. Underneath, though, things will be quite different. The routines will be written in the NA language, and will be able to be changed either from a disk file (the new auto-load system), or by the user using the built in editor. This language will be very easy to understand and use, and will allow the user much more control over the backup processes.

For those people who do not wish to know how NA accomplishes its backups and only want to press a few keys and have them happen, the new language will provide this since the new auto-loads will be even easier to use than the old ones!

For the advanced user, the flexibility of a language whose built in commands include many low-level disk routines will make taking a disk 'apart' and finding out what makes it 'tick' that much easier and more enjoyable.

Another planned feature of the language is a direct link to Apple's new ProDOS language. This means that files will be stored on standard format disks which can easily be manipulated with other programs. (This feature will require a language card in your Apple, or a computer which has one already built in such as the Apple //e or Apple //c).

The language will be a structured one, with looping and conditionals as well as procedure and function calls. ProDOS will be fully supported with an enhanced Sector Editor (Sector/Block Editor) and many built in commands in the language to acces ProDOS disks. Eighty column screens will also be supported.

The real advantage to this new system is that as new protection systems appear, this language will be much more capable of adapting to those new systems.

This language will be accompanied by a full manual describing the language and how to use it along with a good deal of information about disk formats and protection schemes.

THE ANALYZER

During the process of backing-up a diskette, you will see four messages in the status line: READING, ANALYZING, WRITING and VERIFYING. It is during the analyzing process that most of the parameters in NIBBLES AWAY][are used.

In this article we will discuss the procedures which are performed to analyze the data from a disk, and where the various parameters come into play. Below is a basic outline of the procedure which NIBBLES AWAY][uses to analyze a track of data.

1) Read track into data buffer

2) Find first GAP (This is done in one of two ways:)
   a. If no address mark is specified, then NIBBLES AWAY][looks for the first section of data which has values between [GAPBYTE1] and [GAPBYTE2]-1. To be a GAP, there must be at least [GAP SIZE] bytes which fall in this range.
   The track buffer is scanned forward until the section of bytes which are valid for a GAP run out. At this point NIBBLES AWAY][tries to see if the GAP has really ended, or if there is just a glitch in the data. NA][looks forward [FALSE LO] bytes and scans up to [FALSE HI] bytes forward to see if there is more GAP ahead. If more GAP is found within this range, then the previously found GAP is considered invalid and is ignored, otherwise the end of the GAP is marked as the start of a trackfull of data.
   b. If an address mark has been specified, then this sequence of bytes is searched for, and the location where it is found is marked as the beginning of a track full of data.

3) Now NA][searches forward in the data to see if it can find a match for the data following the GAP. This is done to insure that the GAP which was found is consistent. The search for a match takes place starting [DATA MIN] pages (a page is $100 bytes) forward from the location of the initial GAP, and continues up to [DATA MAX] pages forward. When looking for a data match, [FIND MAX] bytes are required to be matched before the data is considered valid. If a match is found, then this is marked as the end of a track full of data, otherwise an error message is displayed.

4) Next the track full of data which has been marked is moved so that it ends at location $7FFF. While the data is being moved, all of those bytes which show as inverse in the [NIBBLE FILTER] are removed from the data. This gets rid of any garbage which may have shown up within the data in the read buffer.

5) If an insert mark has been selected, NA][now scans the data which has been moved to see if the desired insert mark exists. If so, the high bit of this byte is set to zero to tell the write routine to put it on the disk as a SYNC byte. If the parameters [OFFSET +] or [OFFSET -] have been set to a non-zero value, then the location which has its high bit changed will be shifted that number of bytes in the forward or backward direction respectively. This allows an insert mark to be added even if the actual byte which is SYNC is not constant, but the bytes previous to it or after it are.

6) If the [SYNC CONVERTER] is selected then one of two things can happen:
   a. If the [STANDARDIZER] is left on, then [FIX AMNT] bytes previous to every address mark will be set to the value of the parameter [FIX VALU]. This is normally $7F, which is a SYNC $FF. This adds a section of SYNC prior to every address mark for data reliability.
   b. If the [STANDARDIZER] is off, then [FIX AMNT] bytes previous to each address mark will be converted to SYNC. The values of these bytes will not change, but the high bits of each will be set to zero to make them into SYNC bytes.

7) The section of moved data is then written to the destination diskette, using either nine or ten bit sync, as specified by the value of the parameter [SYNC SIZ].

8) The data on the destination track is then read back in and matched against the data which was written out to verify the write operation.

'SNIGLET' TIDBITS

A free copy of the latest version of Nibbles Away II goes to whoever sends in the best SNIGLET. Although a computer related Sniglet would be nice, any sniglet will suffice for this contest. For those of you who don't know, a SNIGLET is any word that SHOULD be in the dictionary but ISN'T. c1984 HBO arr

EXAMPLE:

Postalports - Those annoying windows in an envelop which never seem to line up with the address.

Expandeloping - The act of blowing into the open end of an envelope.

TUTORIAL GUIDES

Many of our users have requested more in depth explanations of the various utility functions supported by NIBBLES AWAY II. Two booklets have been developed (entitled Nibble News Combined) containing articles compiled from past issues of NIBBLE NEWS clarifying the most asked about concepts of the Apple II disk architecture. These booklets should address the most asked about problem areas, and eliminate many of the telephone assistance requests from our clients.

The novice may use the tutorials as groundwork for a better understanding of the Apple II diskette structure.

The more advanced user may find referance material detailing the Apple II disk structure that has been previously unavailable.

Ordering information, along with content highlights of Nibble News Combined may be found on the back page of this newsletter.

NOTES FOR THE //c

Those people using NIBBLES AWAY on the Apple //c computer should be aware that NAII will only accept upper case characters for its commands. This means that the caps lock key should always be depressed when using NAII.

File names, however, can be entered in lower case, as long as you always refer to them in lower case. If you are creating auto-load files to be used on any other machine, you should stick to upper case only since some Apples cannot display lower case properly, or type in lower case to access the files.

Also, the up and down arrow keys will have no effect (or possibly undesired effects) and should therefore not be used.

RIDING OUT THE GLITCHES

If you've ever taken a look at the data which resides between sectors on a diskette, you may have been surprised to find that it was not always the same, and the same spot between sectors might read as different data on two different reads of that same spot. The culprit? Glitches.

Glitches occur because not all disk drives spin at the same rate, and disk drives do not spin at the same rate at any two times. The differences in speed are extremely small, but they are enough to cause glitches. What happens is that when a disk drive goes to write information over existing information, by the time it finishes writing out the data, it will not be in exactly the same place as it was the previous time that it wrote that data. Being off by just one bit (one eighth of a byte) is enough so that when the information is read back, the disk will get a piece of the information which was just written, and a piece of the data which was there from before.

When the disk drive reads this, it temporarily becomes confused and can return just about any data. After a few more bytes have gone by, things return to normal. How does this affect normal disk operation? It doesn't. The space between sectors is there for a very specific reason. This area gives the disk enough time to get itself straightened out from this temporary confusion, and it is alert again long before the next sector moves under the head.

However, for a program like NAII which does not use normal sector boundaries for its markers, this can cause eratic results. The problem usually arises when a write has taken place so that a section of all FF's ends up with a glitch in the middle because the first half was rewritten. This puts a glitch at the end of the rewritten section.

Since this is a common occurance, NAII has a built in system for avoiding any confusion. The 'FALSE LO' and 'FALSE HI' parameters control this feature. When NAII is scanning a section of memory and it finds something in the middle of a section which could be a glitch, it skips ahead by 'FALSE LO' bytes and looks some more. It looks up to 'FALSE HI' bytes forward to see if the section of data continues on the other side of the abnormality. If it does, then the offending data is considered a 'glitch' and is ignored.

This feature allows NAII to reliably disregard any sections of data which result from disk speed variations, allowing more reliable back-ups.

Y00

Have you ever sat down to back-up a diskette, received nothing but 'Y00's on the status display of NA]C, only to find that for some reason the disk does not work? That is the subject of this article.

First, the explanation of Y00 is in order. When you see Y00 on the screen, it means that NA]C has not detected an error in performing the back-up functions which you have selected. This means that the data read from the original disk was identified properly, that read verification was successful, and that the write operation produced a valid replica of the data which was read in on the destination diskette. It does not mean that the protection system on the particular disk must be satisfied. There are other things, like synchronization, insert marks and nibble counting which can be used on the disk.

In order to correct a disk which gets all Y00s but does not boot, it is necessary to find what type of additional protection is being used on the diskette.

Synchronization on the disk can easily be duplicated by selecting a 'synchronized copy' from the NA]C menu. This will take more time than a standard back-up, but it may turn up that hidden protection system.

Nibble counting can also be combated in the same manner, that is, by turning on nibble counting from the CONTROL parameter menu. This type of a back-up can take a LONG time due to the checking involved but be patient, it may correct your back-up problem.

Sync bytes are the most difficult to get around, since they must be detected manually, and no automatic system can be used for a hit and miss approach. The article 'In Search of Insert Marks' in the May 1983 Nibble News provides some insight into the detection of insert marks. (If you tried this method before with version A1 or B1, you may wish to try again with NA]C version C, since the SYNC read capabilities have been enhanced in the new version).

LIVING IN THE FAST LANE

Do you notice a large number of write errors when using NA]C? Do you find that the reliability of your disk drives is lower than you would care for? This may be a sign of disk drives which are adjusted too fast.

Disk speed can be a critical factor in the disk back-up process. If a disk is too fast, it can mean that when NA]C tries to write a track of data to it, the tail end of the track will wrap around and erase the beginning of the track. This causes the first section of data to be lost, and in most cases a write error will occur.

This problem becomes especially critical in the case of nibble counted disks, where NA]C must try to make the back-up disk look as much like the original as possible. If the destination drive is too fast, there may not be enough room for the data to fit, and errors will occur. NA]C can expand the data if the destination drive is a bit too slow, but it is not possible to compress data.

To compensate for the problem, it is a good idea to slow your drives down to about -5 on the NA]C speed test (this is the drive speed that Apple recomends.) Having both drives very close in speed is also a critical factor.

Drive speed variations can be a problem if they are too extreme. Your drives should not fluctuate by more than about eight in either direction from their center value. A variation higher than that can upset a back-up process very badly.

Once your drives are adjusted to the new speed, you should find that all of your software has no problem with the drives and the disks which used to give you trouble during back-ups now have no problem.

We have found that careful drive speed adjustment can make a huge difference in the success rate for making back-ups. Sometimes during a nibble counted back-up it can even help to vary the drive speed slightly while NA]C is running to receive a better back-up.

Overall, drive speed can be the most important factor in reliable back-ups.

USING CONTROL-R

The control-R function of the Track/Bit Editor (TBE) in NIBBLES AWAY II is a very useful tool for figuring out the protection systems on certain disks. It's function is to detect and display the SYNC bytes which exist on a particular diskette.

To see this, try the following example. Boot NIBBLES AWAY II, enter the TBE, then place a DOS 3.3 master diskette in drive one. Press control-R and watch the screen. If you use the forward and backward arrows to move through the buffer, you will see that there are sections of bytes which are shown in inverse video. These bytes represent SYNC bytes on the disk. In the case of the 3.3 master disk, these bytes are FF's. The bytes which follow the FF's make up the address mark for that particular track. This information is very useful if you have a disk which has a non-standard SYNC byte. Using the control-R option you can locate the areas of SYNC, and use the bytes that follow in the address mark parameter in NIBBLES AWAY II.

When looking through the memory dump, you may see one or two bytes off by themselves which are shown in inverse. Most of the time these are just glitches and not really SYNC bytes, so you can ignore them. The two places where SYNC bytes are normally used as part of a protection system are after the address and data fields.

The address field normally consists of 13 bytes. The first three are the address mark (D5 AA 96 for DOS 3.3), then there are two bytes for the volume number, two for the track number, two for the sector number, two for the checksum, and finally two for the closing mark. Sometimes either the second byte of the closing mark, or the byte after the closing mark will always be sync. This is what is known as an 'INSERT MARK'. To duplicate this on a backup disk, you should key in the sequence of bytes up to and including the byte which is SYNC (2 or 3 bytes is enough) into the insert mark parameter in NIBBLES AWAY II.

There is also a closing mark for the data field. The data field begins a few bytes after the end of the address field (It begins with D5 AA AD for DOS 3.3). At the end of the data field is a section of SYNC leading up to the next address mark. In between the data field and the next section of SYNC, is the data field closing mark (DE AA for 3.3). As with the closing mark for the address field, sometimes the second byte and sometimes the one after that are SYNC, and are handled in the same way as are those for the address field.

The control-R function is set up to display in inverse for 10 bit SYNC, which is what DOS 3.3 and all insert marks use. If a disk has 9 bit SYNC (used by DOS 3.2) then the areas where this type of SYNC exists will appear as alternating inverse and non-inverse. This is easy to see if you have a DOS 3.2 diskette and read it with control-R. The sections of FF's between the data will show alternating between inverse and normal video.

In this article we discussed some basic uses for control-R. In future issues we will delve more deeply into its possible uses for decoding disk protection schemes.

THE BEGINNERS

This month we will discuss how the Apple communicates with the disk drive and how the Apple reads or writes information to and from the disk. The information presented may be a little technical at some points but there is no need to worry if everything is not crystal clear... All that we need at this stage of the game is a basic idea of what is happening. Besides, all will become clear later. Now to begin...

NOTE: Due to speed considerations, most accesses to the I/O locations assigned to a disk drive must be done through a machine language program. It is not absolutely necessary, however, that everyone understand the machine language portions of this discussion. What is important is that we grasp the basic concepts involved. Example 1 is a listing of a program that performs an operation similiar to the R)ead function of the Track/Bit editor built into Nibbles Away II. Although ours is not nearly as elegant, they both read a single track into the computers memory.

A diskette is constructed of a circular piece of plastic that has been covered with a coating of a metallic oxide (also known as rust) very similiar to a regular cassette tape. All information, be it your tax records, your general ledger, a letter to mom, or your 15th level Ninga in Wizardry, is stored as a series of magnetic impulses in this coating. Through the miracals of modern technology, these impulses are translated into binary 1s ands 0s denoting the presence or absence of a pulse. These 'bits' are put together into groups of eight to become disk 'bytes'. These bytes can then be logically separated into different groupings on a disk. Each disk is divided into (usually) 35 tracks. These tracks are cocentric circles evenly spaced out across the disk a bit like ripples in a pond after a pebble has been tossed in except that they do not move. Each of these tracks can be further subdivided into as many as 16 sectors. (16 is not the only number of sectors possible. Some disks only have 1 sector per track.) The track and sector divisions are determined by the software. For example DOS 3.3, which we have all used, divides the disk into 35 tracks of 16 sectors. DOS 3.2 on the other hand, used 35 track of 13 sectors. The number of different combinations is almost limitless, which makes the job of backing up programs all the more interesting. Luckly most of the messy details required to read or write a byte

are taken care of for us by the disk interface card when we need to read or write a byte to the disk. The disk interface card, also known as the disk controller card, is the card that connects the cable from your disk drive to your Apple and allows the two of them to communicate. It plugs into the Apple in one of the 8 Input/Output (I/O) slots located in the back.

The Apple disk controller card is a truly amazing piece of electronics. Like most Apple I/O devices it can communicate with the Apple via a series of 16 memory locations assigned to each slot. These locations are tucked away somewhere between $C080 and $C0FF. We say somewhere because their exact location is slot dependent, meaning that it varies depending on which slot the disk controller is currently plugged into. (Usually slot six). These locations can be used for a variety of purposes depending on the requirements of a particular interface card. The disk controller uses its share of these locations to move the read/write head (called 'seeking the head'), turn the drive motor on or off, select drive one or two, to check for a write protect tab, and to transfer data between the Apple and the disk drive through what are known as the data latches (the term 'data latch' in this context merely refers to specific locations that are used to pass data from the Apple to the disk drive and vice versa.) In some cases all that is necessary is to access the location to achieve the desired result. In other cases it is possible to transfer data through the location, as when reading or writing. Even though these locations move around, it is not as hard as it might sound to figure out exactly which set of 16 bytes a disk controller might be assigned. If an interface card is in slot 1, then it is given the space starting at $C090 and extending to $C09F. If a card is plugged into slot 2 then it has locations $C0A0 thru $C0AF and so on down the line. By adding hex $x0 to $C080, where x is the slot number, we get the I/O locations for a particular slot. Slot six then has the space between $C0E0 and $C0E9. ($60 + $80 = $E0) For this and later discussions we will assume that the disk controller card is plugged into slot 6.

As the diskette spins past the Read/Write head of the disk drive, the data bits are read and transfered to the data latch. As subsequent bits are read, they are added to the data already in the latch. The newcomers are added by shifting the existing bits to the left (see figure 1) and placing the new bit into the Least Significant Bit position

of the latch. ('Least Significant Bit' is a fancy term for the bit representing the 1's place in a byte. In this case the rightmost bit.) This process continues until the Most Significant Bit (another fancy term; this one meaning the bit representing the 128s) place in the latch becomes a 1. (See NIBBLE NEWS, September 1982, 'Fun with the Sector Editor' for a more detailed description of the hex and binary numbering systems.) This is how the controller knows that it has read an entire byte. Iet is while this bit is a one that the data is considered 'valid'. This makes it easy to read information from the disk because, in machine language, a byte may be considered 'plus' if the Most Significant Bit (MSB) is a zero. Therefore it is possible to write a loop that checks the controller data latch and Branch Plus (BPL) back the check. In this way it will wait until the MSB is set, meaning that the byte is no longer 'plus' and that the data is valid. See Program Listing 1 for more details.

Writing to the disk is much more complicated than reading from it. When in the 'write mode' the disk controller sends out bits from the latch so that every 32 microseconds it has written all 8 bits and it needs a new data byte. If the controller is not fed at exactly 32 micro second intervals, then it begins to write 0 bits to the disk. (This feature is actually used to create 'sync bytes' which are used to control the disk controller.) This is because, in the same way that reading shifts data into the latch, writing shifts data out of the latch and on to the disk. The bits that are left open by this shifting now contain zero. See Figure 2 to see how a byte is written to disk.

Now that we have a basic idea of how the hardware works, we are able to discuss how the software works.

Figure 1

Read bits from the disk

Data latch	Bit from disk
_ _ _ _ _ _ _ 1	<---- 1
_ _ _ _ _ _ 1 0	<---- 0
_ _ _ _ _ 1 0 1	<---- 1
_ _ _ _ 1 0 1 1	<---- 1
_ _ _ 1 0 1 1 0	<---- 0
_ _ 1 0 1 1 0 1	<---- 1
_ 1 0 1 1 0 1 1	<---- 1
1 0 1 1 0 1 1 1	<---- 1

Figure 2

Write bits to the disk

Bit to disk	Data latch
1 <----	1 0 1 1 0 1 1 1
0 <----	0 1 1 0 1 1 1 0
1 <----	1 1 0 1 1 1 0 0
1 <----	1 0 1 1 1 0 0 0
0 <----	0 1 1 1 0 0 0 0
1 <----	1 1 1 0 0 0 0 0
1 <----	1 1 0 0 0 0 0 0
1 <----	1 0 0 0 0 0 0 0

EXAMPLE 1

```
                1000 ************************
                1005 *                      *
                1010 *   READ A TRACK INTO  *
                1020 *    MEMORY DIRECTLY   *
                1030 *    FROM THE DISK.    *
                1040 *                      *
                1050 * $1000 TO $8000 IS    *
                1060 *   WHERE THE DATA IS  *
                1070 *   STORED.            *
                1080 *                      *
                1090 *                      *
                1100 ************************
                1200
0000-           1210 POINT    .EQ $00
                1220
FCA8-           1221 WAIT     .EQ $FCA8
                1222
C089-           1230 DRIVEON  .EQ $C089
C088-           1240 DRIVEOFF .EQ $C088
C08A-           1250 DRIVE1   .EQ $C08A
C08B-           1260 DRIVE2   .EQ $C08B
C08E-           1270 READMODE .EQ $C08E
```

```
C08C-                 1280 READ     .EQ $C08C
                      1290
                      2000
                      2005 * SET UP OUR POINTER INTO THE
                      2006 *     DATA BUFFER
                      2007
0800- A9 00           2010 START    LDA #$00
0802- 85 00           2020          STA POINT
0804- A9 10           2030          LDA #$10
0806- 85 01           2040          STA POINT+1
                      2050
                      2060 * SET UP X REGISTER TO USE SLOT
                      2070 * SIX
                      2080
0808- A9 06           2090          LDA #$06
080A- 0A              2100          ASL
080B- 0A              2110          ASL
080C- 0A              2120          ASL
080D- 0A              2130          ASL
080E- AA              2140          TAX
                      2150
                      2160 * TURN DRIVE ON AND SELECT DRIVE
                      2170 * ONE AND SELECT READ MODE
                      2180
080F- BD 89 C0        2190          LDA DRIVEON,X
0812- BD 8A C0        2200          LDA DRIVE1,X
0815- BD 8E C0        2205          LDA READMODE,X
                      2210
                      2220 * WAIT A BIT FOR THE DRIVE TO
                      2230 * TO COME UP TO SPEED
                      2240
0818- A9 FF           2250          LDA #$FF
081A- 20 A8 FC        2260          JSR WAIT
                      2270
                      2280 * HERE IS THE READ LOOP.  THIS
                      2290 * READS IN A TRACK ABOUT 3 TIMES
                      2300 *
                      2310
081D- A0 00           2311          LDY #$00
                      2312
081F- BD 8C C0        2320 LOOP1    LDA READ,X
0822- 10 FB           2330          BPL LOOP1
0824- 91 00           2340          STA (POINT),Y
0826- C8              2350          INY
0827- D0 F6           2360          BNE LOOP1
0829- E6 01           2370          INC POINT+1
082B- A5 01           2380          LDA POINT+1
```



```
082D- C9 80           2390          CMP #$80
082F- D0 EE           2400          BNE LOOP1
                      2410
                      2420 * TURN DRIVE OFF AND EXIT
                      2430
0831- BD 88 C0        2440 EXIT     LDA DRIVEOFF,X
0834- 60              2450          RTS

SYMBOL TABLE

C08A- DRIVE1
C08B- DRIVE2
C088- DRIVEOFF
C089- DRIVEON
0831- EXIT
081F- LOOP1
0000- POINT
C08C- READ
C08E- READMODE
0800- START
FCA8- WAIT
```

Request-Line

Below is a listing of programs most frequently requested by NIBBLES AWAY II customers seeking back-up parameters. If you have been successfull in developing parameters for these programs, we would appreciate your submitting them for publication in future issues of "Nibble News" so that we may share them fellow NIBBLES AWAY owners. We will give you credit toward an extension on your subscription.

Thank you for your support.

BARON'S SAT PREPARER
BPI INVENTORY
CHOPLIFTER
DB MASTER
DROL
FINANCIAL COOKBOOK
FLIGHT SIMULATOR II
GENERAL MANAGER
JUGGLER'S RAINBOW
KEYSTROKE
KRELL LOGO
LIST HANDLER
LODERUNNER
MASTER TYPE
PFS:WRITE
PFS:FILE IIE
PINBALL CONSTRUCTOR
ROCKY'S BOOTS
SENSIBLE SPELLER IV
SPIDER EATER
STICKYBEAR SERIES
SAT (HARCOURT, BRACE, JOVANOVITCH)
TEMPLE OF APSHAI (NEW VERSION)
THE VAULT
ULTIMA III
WORD HANDLER

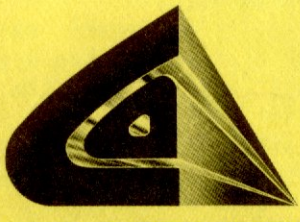
NIBBLE NEWS COMBINED #1
(8/82 - 1/83)

BEGINNERS GUIDE.....Step by step guide to backing-up disks.
BOOTSTRAP.....A discussion of the boot process for Apple DOS 3.3 with a look at different boot systems and the methods used to trace them.
EASY AUTO-LOADS.....Creating auto loads using Videx keyboard enhancer.
FUN WITH SECTOR EDITOR.....Tutorial on Apple DOS disk layout, format of disk files, basic disk repair and exploring disks from other operating systems.
MAKING AUTO LOADS.....Make your own personalized Auto-Load disk.
PARAMETER HUNT.....Tutorial on finding address marks.
PATCH-WORK.....Global mods to NAII (V. B1) to provide extra features: greater printer capability, higher speed auto load execution.

NIBBLE NEWS COMBINED #2
(2/83 - 10/83)

BOOTSTRAP.....Discussion on Apple DOS 3.3.
DISK SCANNER.....Utility program to scan a disk in search of sequence of bytes.
FUN WITH SECTOR EDITOR.....Helpful utilities and disk data encoding program.
IN SEARCH OF INSERT MARKS...How to identify insert marks.
MEMORY/DISK CONV. TABLE.....Shows correlation between a given DOS memory address and the disk track/sector location.
QUESTIONS AND ANSWERS.....Commonly asked questions answered.
SPIRALING AWAY.....Discussion of track arching as used in disk protection.
SYNC BYTES.....Methods used to create sync bytes.

Combined #1 .... $7.50
Combined #2 .... $7.50



COMPUTER: applications, Inc.
SOFTWARE BY DESIGN

13300 S.W. 108 STREET CIRCLE • MIAMI • FLORIDA 33186
(305) 385-4277

SOURCE: TCD 328


Scanned by cvxmelody
http://www.cvxmelody.net/AppleUsersGroupSydneyAppleIIDiskCollection.htm