ULTRA FAST PIX

SECOND FEATURE

Use this convenient *ampersand routine to save and retrieve Hi-Res screens at lightning speed.*

There are two ways to approach loading and saving Hi-Res screens on the Apple: the wait-and-see method and the let's see it now method. One way leaves you looking at a blank screen as your disk drive cranks around and finally loads the picture. The other way is the Ultra Fast Pix way. It loads Hi-Res pictures four times faster than ProDOS. In fact it can load a picture in about 560 milliseconds when reading multiple pictures. If you prefer to load data instead of pictures, Ultra Fast Pix loads 8K or more of data at high speed.

To display a Hi-Res slide show, 17 pictures stored on a disk can be reviewed in 9.5 seconds from a standing start (if you can see that fast!). A single picture is loaded in about 700 milliseconds, including drive startup. Compare that with these times: DOS 3.3 — 10.5 seconds; ProDOS — 3.0 seconds; and Fastpix — 2.0 seconds.

Ultra Fast Pix uses a novel technique introduced by Ken Manly's "Fastpix" in *Nibble* Vol. 3/No. 2. The Fastpix program loads Hi-Res pictures quickly, but not fast enough for me. In Ultra Fast Pix, I changed the disk format so it converts disk bytes on the fly. The load time has been reduced to about a third of what can be achieved with the original Fastpix program.

The demonstration program shown in **Listing 1** illustrates a typical use. Place an initialized disk in drive 2. Running the program will create 17 simple Hi-Res pictures and store them in Ultra Fast Pix format on the disk in drive 2. Note that the process of storing Ultra Fast Pix data on a disk destroys any previous contents. The program then initializes both Hi-Res pages. After you press Return, the speaker beeps and a Hi-Res picture is loaded first into page 1 and then into page 2. While the pictures are being loaded, the screen switches are used to display the other page. This gives the effect of snapping the pictures onto the screen. After 17 pictures are shown, the program beeps again and control returns to BASIC when you press Return.

ENTERING THE PROGRAMS

Key in the Applesoft program shown in **Listing 1** and save it with the command:

SAVE ULTRA.FAST.DEMO

If your system has only one drive, change "D = 2" in **line 250** to "D = 1" and change "DRIVE 2" in **line 280** to "DRIVE 1".

If you have an assembler, key in the source code from **Listing 2**, save it and assemble it using ULTRA.FAST as the object file name. If you are using Key Perfect and your assembler does not store zeros in the object file buffer for the .BS pseudo-opcode (or equivalent, such as DFS or .DS), enter the Monitor with CALL −151 and perform the following commands:

```
76E4:Ø
76E5<76E4.76FEM
77FF:Ø
79D4:Ø
79D5<79D4.79FEM
7A4Ø:Ø
7A41<7A4Ø.7A7EM
```

If you are using Key Perfect (regardless of assembler capabilities), BLOAD ULTRA.FAST and BSAVE it again using the command shown in the next paragraph.

If you don't have an assembler, key in the hex code from **Listing 2**. If you are using Key Perfect, zero the areas of memory reserved with .BS pseudo-ops with the Monitor commands shown above. Save the program with the command:

BSAVE ULTRA.FAST,A$7500,L$600

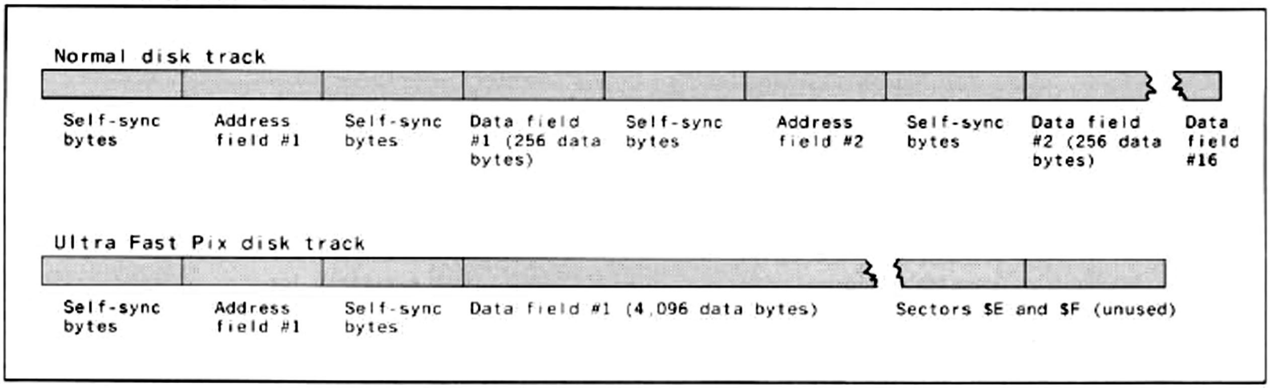
Because the program is self-modifying, it is important that you save the program before you run it. For help with entering *Nibble* programs, see the directions in the Program Listings section.

USING ULTRA FAST PIX IN YOUR PROGRAMS

When you BRUN ULTRA.FAST, an ampersand command is set up. The calling syntax is:

&C,N,P,D

FIGURE 1: Disk Track Formats



where *C* is the command to read or write (R,W); *N* is the picture number (with values of 0-16); *P* is the Hi-Res page number (with values of 1 and 2); and *D* is the disk drive (with values of 1 and 2).

To experiment with Ultra Fast Pix in your own programs, start by initializing a test diskette using either INIT with DOS 3.3 or the Filer under ProDOS. Now start the program by entering:

BRUN ULTRA.FAST

The program will load at $7500 and the ampersand vector will point to the beginning of the machine language program.

Type HGR to display Hi-Res page 1. Now load any Hi-Res picture from one of your own disks by typing:

BLOAD *picture*,A$2000

Insert the test diskette in drive 1 and try saving the picture by typing **&W,0,1,1**. Next, type **HGR** to clear the Hi-Res screen, then reload the picture by typing **&R,0,1,1**. You're now ready for more pictures.

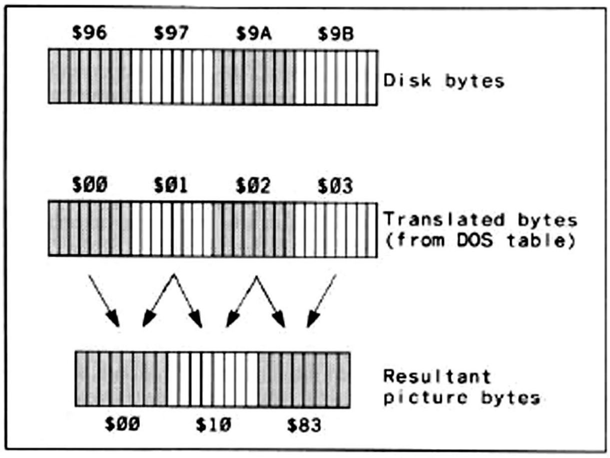
Creating a Picture Diskette

To create a picture diskette with up to 17 pictures, follow these steps:

1. Initialize a diskette using either DOS 3.3 or ProDOS.
2. **BRUN ULTRA.FAST** to load and initialize the program.
3. **BLOAD *picture*,A$2000** to load your Hi-Res picture.
4. Type **&W,*N*,1,*D*** to save a picture, where *N* is the picture number (0-16) and *D* is the drive number. For instance, **&W,0,1,1** will save the picture as picture 0 on drive 1.
5. Continue steps 3 and 4 until all of the desired pictures are stored.

For example, to read picture 0 onto Hi-Res page 1 from drive 1, you would type **&R,0,1,1**. To save picture 5 from Hi-Res page 2 to drive 2, you would type **&W,5,2,2**.

FIGURE 2: Byte Translation



PROGRAM OVERVIEW

Ultra Fast Pix gets its fast reading ability in two ways. First, it uses one sector per track, which contains 4,096 bytes ($1000) of data. (These 4,096 screen bytes are translated and written to disk as 5,462 bytes.) Second, it converts from disk bytes (which contain 6 bits of information) to data bytes (which contain 8 bits) as part of the disk reading process.

Disk Fundamentals

Before getting into details, let's review some disk fundamentals. The outermost track is track 0. From here stepping inward, the head can be moved to about 132 positions, only a quarter of which can be used as tracks. (This is because closer spacing would cause data to be mistakenly read from adjacent tracks.) DOS 3.3 and ProDOS both use the "even steps" as tracks. (When the head is moved outward as far as it will go, this position is defined as track 0.)

The disk speed is 300 rpm or 200 milliseconds per revolution. A disk byte can be written to disk every 32 microseconds, so the track capacity is theoretically 6,250 bytes. The real capacity is less than this because other bytes must be present to allow detection and synchronization of the data. For instance, both DOS 3.3 and ProDOS have 4,096 bytes of data per track.

DISK STRUCTURE

The structure of an ordinary disk track is shown in the upper diagram of **Figure 1**. It begins with a number of special "self-sync" bytes, which are used by the drive to align itself with an actual byte boundary. Since the data on a disk is nothing but a long stream of ones and zeroes, without the self-sync bytes there would be no way to tell where to begin reading (or writing).

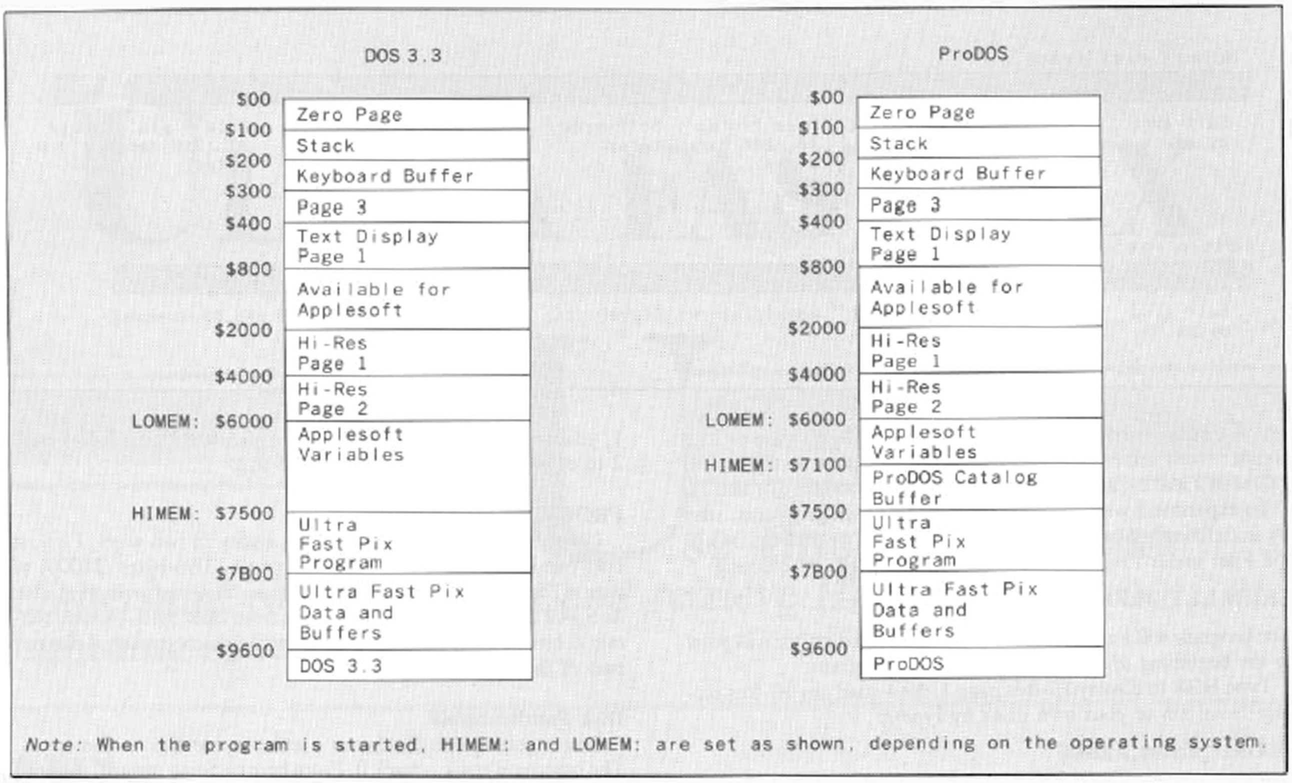
After the self-sync bytes is the first address field, which corresponds to sector $00 on the track. It contains some information that the operating system uses to determine which sector is which. Next, there are some more self-sync bytes, and then the first data field. The data field begins with a three-byte header, followed by the data itself, followed by a three-byte trailer. A normal data field contains 256 data bytes. There are actually more bytes than this on the disk, however, as I'll explain in a moment.

After the first data field are some more self-sync bytes, and then the second address field, and so on until the end of the track. There are 16 sectors and, therefore, 16 address and data fields on each track.

The structure of an Ultra Fast Pix disk track is slightly different, as shown in the bottom diagram of **Figure 1**. It starts with the same self-sync bytes and address field, but the data field is 4,096 data bytes long, and there is only one data field. This leaves a little extra space at the end of the track (where sectors $E and $F would normally be) but the object here is speed, not maximum storage space.

When Ultra Fast Pix stores a picture on the disk, it uses two adja-

FIGURE 3: Ultra Fast Pix Memory Maps



cent tracks to hold the 8,192 bytes of picture data. Each track holds half of the picture. The address field of sector zero is left on the disk, and the new, large data field is written right after it, overwriting the normal disk fields. This is why it is important to use a blank disk to store your pictures on; any other data on the disk is destroyed.

HOW DISK BYTES ARE WRITTEN

Interestingly, due to hardware limitations, an Apple floppy disk drive is not capable of reading and writing all of the 256 possible byte values, nor is any other floppy disk on the market. And yet, we do store data that contains all the possible values. This is accomplished by encoding the data into a form that requires the use of fewer byte values. Both DOS 3.3 and ProDOS use a form of encoding known as "6 and 2," which requires 342 disk bytes to represent 256 actual data bytes. Thus, each data field on a normal disk really contains 342 disk bytes, which are translated back into 256 data bytes when they are read.

Ultra Fast Pix uses its own encoding technique, which is slightly different from the one used by DOS 3.3 and ProDOS, though it uses the same translation table. It requires four disk bytes for every three data bytes, resulting in 5,462 disk bytes in the one large data field (the last byte contains only 2 bits of information). **Figure 2** shows an example of how the bytes are translated.

A full description of the various encoding techniques is beyond the scope of this article, but if you are interested there is a very good explanation in the books *Beneath Apple DOS* and *Beneath Apple ProDOS* by Don Worth and Pieter Lechner, available from Quality Software.

DETAILED PROGRAM WALKTHROUGH

SETUP

When **Listing 2** is BRUN, the SETUP routine (**lines 2040-2760**) sets the ampersand vector to the PARAM routine entry point. Next, the READ tables are set up using calculations on data from a single table instead of additional lengthy tables.

To set HIMEM, a check is made to identify the operating system. For DOS 3.3, HIMEM is simply set to the beginning of the program, and LOMEM is set just above Hi-Res page 2 at $6000.

For ProDOS, however, it's not as easy. ProDOS buffers may be active, so all files are closed. The ProDOS bit map is set up next. The bit map is a 24-byte field starting at address $BF58. A bit in the map is set for every page of memory in the lower 48K being used, starting with page zero. The pages in use are numbered sequentially starting with bit 0 in $BF58. So, for example, if $BF58 contains $CF, then locations $00-$7FF are in use, except for pages 2 and 3.

To make room for the catalog buffer in ProDOS, HIMEM is set an extra $400 bytes below the beginning of the program. During a CAT or CATALOG command, ProDOS uses memory locations $7100-$74FF, which are below the program. LOMEM is set to the same location as under DOS 3.3. The program memory map now looks like **Figure 3**.

PARAM

The beginning of the PARAM routine (**lines 2800-3410**) is the entry for the ampersand vector. When an ampersand is encountered in a program line, control passes through the ampersand vector to PARAM. The Accumulator contains the next character, and further characters are read by calling CHRGET at $B1.

After entry, the routine determines the command type (Read or Write) and saves this on the stack. Applesoft routines are used to get the rest of the parameters (Picture, Page and Drive) and check them for validity. The parameters can be numbers, variables or formulas.

SYNTAX ERRORs or RANGE ERRORs are dealt with using the appropriate Applesoft error handlers. This means that all errors can be handled from BASIC using ONERR GOTO.

All of the parameters are pushed onto the stack after checking. The stack is used here to keep the zero page untouched until it is swapped with the REGSAV area. During program development, allocating zero page variables for DOS 3.3 and ProDOS became such a headache that I knew there must be a better way. When

SWAP is called, the current zero page and the REGSAV area trade places. This technique positions zero page variables very neatly. ProDOS naturally saves the default slot in a different place than DOS 3.3. Under DOS 3.3, slot 6 is saved as $60 in location $5F8. This is convenient because it can be used as an index in the drive addresses. However, ProDOS simply saves slot 6 as $06, so it must be multiplied by 16 before it is saved in the register SLOT. The parameters are then pulled from the stack and saved in the new zero page area and program flow jumps to the appropriate READ or WRITE routine.

READ

The READ routine (**lines 3450-3810**) sets a counter called READCT to allow several reads with incorrect data trailers. The START and STOP addresses are set to the lower half of the appropriate Hi-Res page. When the correct data is read for the first 4,096 bytes, the track is advanced by one, and the second 4,096 bytes are loaded into the upper half of the Hi-Res page. Persistent disk errors are handled by the ERROR routine. If the second track read is okay, the drive is turned off, zero page is restored, and control returns to the Applesoft program.

WRITE

The WRITE routine (**lines 3850-4090**) is similar to the READ routine except that the data is first "pre-nibblized." The NIBBLE routine divides the 4,096 bytes of picture data into six-bit chunks and then translates them into disk bytes. There is no error checking to ensure that valid data was written to disk. Therefore, a disk that is write-protected will appear to save a picture.

SKTRK

SKTRK (**lines 4130-4610**) starts the drive and then waits for the drive to turn about one-half revolution so the speed is stabilized. If the drive has already been turning, the wait is skipped.

The SKABS routine is called to position the drive head at the correct track, and to read a disk address. If the track on the disk address matches the desired track, and the correct sector (always 0) is found, the routine exits. Sometimes, though, the head stepper misses steps. There is no reason to recalibrate if the routine knows where the head is positioned. If no disk address can be read, however, the head recalibrates by moving outward two tracks (unlike DOS 3.3). (Any unnecessary head movement wastes time; it is only necessary to move two tracks to get the stepper motor back in phase.)

At this point, RDADDR should be able to read a valid address. Two counters keep track of the maximum number of recalibrations (CALIB) and the number of times that a wrong track is read. When these counts are exceeded, the disk ERROR routine is called.

WR4096

WR4096 (**lines 4690-5160**) is a time-critical routine. If you make program modifications, make sure that no page crossing occurs. Also, if any of the zero-page variables are reallocated to non-zero page locations, this will also change the timing. The time for each instruction appears on the right-hand side of the comment field in the listing. This is convenient for counting cycles.

When WR4096 is called, the disk is moving and the correct disk address has been read. The drive is configured for writing and begins writing 40-cycle self sync bytes. The time between one LDA DRQ6L,X instruction and the next must be exactly 40 cycles. Five self sync bytes should be sufficient, but a few more won't hurt.

After the self syncs are written, the 40-cycle bytes must blend to 32-cycle bytes. To do this, the data in the buffer is written to disk in 32-cycle bytes. (The data was originally 4,096 bytes, but the conversion from eight-bit bytes to disk bytes containing six information bytes turns this into 5,462 disk bytes, plus the three header and three trailer bytes.)

The routine continues writing the buffer data in 32-cycle intervals until a zero is detected. Since the buffer contains the disk data, the only zero in it is the one at the end. If the drive is switched to read too quickly when the last byte is written, the last byte would be cut in half, making it unreadable. DOS 3.3 solves this problem by not checking trailers. Since checking trailers is the only way to detect a bad read, this last byte is important.

NIBBLE

The NIBBLE routine (**lines 5200-6100**) formats the data in preparation for the WR4096 routine. The pointers to the start of the "nibblizing" buffer are set, and the three data header bytes ($D5, $AA and $AD) are loaded in. The header bytes and trailer bytes in the buffer make the WR4096 timing simpler.

In **line 5340**, the conversion process starts. Sets of three picture bytes are converted to four disk bytes. In the first picture byte, the upper six bits are stripped and used to index into the disk byte table. The bottom two bits of the first picture byte and the upper four bits of the second picture byte are combined, and again used as indexes to the disk byte table. Similarly, the next two picture bytes are converted. After 4,096 bytes are converted in this manner, the last bytes placed in the buffer are the trailer bytes ($DE, $AA and $EB) and a zero to allow easy end-of-buffer checking by WR4096.

RD4096

The RD4096 routine (**lines 6180-7020**) must also be located so that no page crossing occurs within the routine. The routine is activated when the disk is spinning and the proper disk address has been read. When reading starts, self sync bytes are under the drive head. After the self sync bytes pass, the data header must be verified.

After verification of the data header ($D5, $AA and $AD), the 5,462 disk bytes are read. This part of the routine does the opposite of the WR4096 routine. Four disk bytes are quickly turned into three picture bytes; this is done on the fly. It's a little easier to write disk read routines because strict conformance with 32-cycle reads is not necessary. The only requirement is that the time between one LDX DRQ6L instruction and the next must be 32 cycles or less.

This routine, like the ProDOS routine, uses self-modifying code. It isn't faster, but it does save a register.

The normal method of reading from disk is LDA DRQ6L,X, where the X-Register contains the slot. This doesn't leave enough time for reloading registers. Instead, after the instruction for the slot is modified, an LDX DRQ6L is used. The X-Register then serves as an index to the picture byte in a lookup table. (Remember, since a disk byte contains only six bits of real information, a lookup table is used to convert it.)

After conversion and packing, the data is stored using the PAGE pointers. Since the routine reads 5,461 bytes in four-byte cycles, it can check for completion in just one place. The last byte contains just the last two bits. The data trailer ($DE, $AA and $EB) is checked now to ensure that "disk slip" didn't occur during this read. If the trailer is not correct, the Carry is used to flag an error.

RDADDR

The RDADDR routine (**lines 7080-7550**) reads a disk address and verifies it as valid. The address header sequence ($D5, $AA and $96) is looked for first. Once a valid address header is detected, the volume, track, sector and checksum are read. These are written in a less dense format than the disk data. Two disk bytes are read and merged together using the odd bits of the first disk byte ANDed with the even bits of the second disk byte, to form one data byte. These values are saved for later use and the checksum is verified.

Next, the address trailer bytes ($DE, $AA) are verified. Any errors are flagged with Carry set. Theoretically there should be three trailer bytes ($DE, $AA and $EB), but the routines that write addresses in DOS 3.3 and ProDOS both have a bug that chops off the $EB before it is completely written. The problem has been solved the easy way — forget the $EB.

Listing 1 for Ultra Fast Pix

ULTRA.FAST.DEMO

```
100  REM ***********************
110  REM *  ULTRA.FAST.DEMO    *
120  REM *                     *
130  REM * COPYRIGHT (C) 1987 *
140  REM * BY MICROSPARC, INC *
150  REM * CONCORD, MA  01742 *
160  REM ***********************
170  REM COMMAND STRUCTURE
180  REM &C,N,P,D
190  REM C=COMMAND ("R" OR "W")
200  REM N=PICTURE NUMBER (0 TO 16)
210  REM P=HIGH RES PAGE NUMBER (1 OR 2)
220  REM D=DISK DRIVE (1 OR 2)
230  ONERR  GOTO 620
240  PRINT  CHR$ (4);"BRUN ULTRA.FAST"
250 D$ =  CHR$ (4):XS = 140:YS = 96:P2 = 6.29
     :A = 90:D = 2:P = 2
260  TEXT : HOME : VTAB 12: HTAB 12
270  VTAB 12: HTAB 10: PRINT "INSERT INITIALI
     ZED DISK"
280  VTAB 14: HTAB 10: PRINT "   INTO DRIVE 2
      AND"
290  VTAB 16: HTAB 10: PRINT " PRESS RETURN T
     O START";: GET K$: PRINT
300  REM  *---- CREATE PICTURES
310  FOR N = 0 TO 16
320  HGR2 : HCOLOR= 3: HPLOT XS + A,YS
330  IF N < 10 THEN 400
340  FOR TH = 0 TO P2 STEP .03
350 R = A *  COS ((N - 8) * TH)
360 X = XS + R *  COS (TH):Y = YS - R *  SIN
     (TH)
370  HPLOT  TO X,Y
380  NEXT TH
390  GOTO 460
400  FOR S = 0 TO N + 3
410 TH = S * P2 / (N + 3)
420 X = XS + A *  COS (TH):Y = YS - A *  SIN
     (TH)
430  HPLOT  TO X,Y
440  NEXT S
450  REM  *---- SAVE PICTURES
460  & W,(16 - N),P,D
470  NEXT N
480  REM  *---- SHOW PICTURES
490  HGR2 : HGR : HOME
500  VTAB 22: HTAB 5: PRINT "PRESS RETURN TO
     VIEW PICTURES";: GET K$: PRINT
510  POKE  - 16302,0: POKE  - 16304,0: POKE  -
     16297,0
520  PRINT  CHR$ (7)
530  FOR N = 0 TO 16
540  POKE 49235 + (3 - P),0
550  & R,N,P,D
560 P = 3 - P
570  NEXT N
580  POKE 49235 + (3 - P),0
590  PRINT  CHR$ (7)
600  GET Z$: PRINT
610  HOME : TEXT : VTAB 12: HTAB 12: PRINT "T
     HAT'S ALL FOLKS!": END
620 ER =  PEEK (222): HOME : TEXT : VTAB 12: PRINT
     "AN ERROR HAS OCCURRED": PRINT : PRINT "
     RETURN TO TRY AGAIN, ESCAPE TO QUIT";: GET
     Z$: PRINT : ON Z$ =  CHR$ (27) GOTO 610:
      POKE  - 16302,0: POKE  - 16304,0: POKE
      - 16297,0: RESUME
```

END OF LISTING 1

```
              KEY PERFECT 5.0
                  RUN ON
                ULTRA.FAST
  ==========================================
     CODE-5.0    ADDR# - ADDR#     CODE-4.0
     --------    -------------     --------
     AD5E0895     7500 - 754F        267C
     44A280AD     7550 - 759F        25ED
     A0435C2A     75A0 - 75EF        247C
     B52FD77C     75F0 - 763F        25CF
     4381230E     7640 - 768F        2932
     03C5A15F     7690 - 76DF        27F6
     0B9135DF     76E0 - 772F        2457
     187E1660     7730 - 777F        28B8
     CEA0948F     7780 - 77CF        2700
     A3C254F1     77D0 - 781F        2CBA
     30FF59E7     7820 - 786F        2C82
     62C3183D     7870 - 78BF        2773
     946945E8     78C0 - 790F        2919
     3F56824C     7910 - 795F        2A17
     9F6867AC     7960 - 79AF        2AEF
     81A99BA4     79B0 - 79FF        279B
     D227ACCF     7A00 - 7A4F        2B50
     624EB0BB     7A50 - 7A9F        01D3
     C9B1F310     7AA0 - 7AEF        2B02
     FB23A67F     7AF0 - 7AFF        0653
     29F72855 = PROGRAM TOTAL =      0600
```

Listing 2 for Ultra Fast Pix

ULTRA.FAST

```
                1000 *       ULTRA.FAST
                1010 *
                1020 *       COPYRIGHT (C) 1987
                1030 *       BY MICROSPARC, INC.
                1040 *       CONCORD, MA 01742
                1050 *       S-C MACRO ASSEMBLER 1.1
                1060 *
                1070 *-------------------------------
                1080 *
                1090 *       ADDRESSES
                1100 *
B1-             1110 CHRGET .EQ $B1       GET A CHARACTER
03D0-           1120 WARM33 .EQ $3D0      DOS  3.3 BASIC WARMSTART
03F5-           1130 AMPVEC .EQ $3F5      AMPERSAND JUMP VECTOR
05F8-           1140 SLOT33 .EQ $5F8      DOS 3.3 SLOT * 16
6000-           1150 P2END  .EQ $6000     LOMEM SETTING ABOVE PAGE 2
BE00-           1160 WARMPR .EQ $BE00     PRODOS BASIC WARMSTART
BE3C-           1170 SLOTPR .EQ $BE3C     PRODOS DEFAULT SLOT
BE70-           1180 GOSYST .EQ $BE70     PRODOS COMMAND ENTRY
BEDO-           1190 SCLOSE .EQ $BEDO     PRODOS BASIC SYSTEM CLOSE
BF00-           1200 GLOBAL .EQ $BF00     PRODOS GLOBAL PAGE
BF58-           1210 BITMAP .EQ $BF58     PRODOS MEMORY USAGE BITMAP
BF94-           1220 LEVEL  .EQ $BF94     PRODOS SYSTEM LEVEL
DEC9-           1230 SYNTAX .EQ $DEC9     SYNTAX ERROR ROUTINE
DEBE-           1240 CHKCOM .EQ $DEBE     CHECK FOR COMMA - SYNTAX ERROR IF NOT
E6F8-           1250 GETBYT .EQ $E6F8     EVAL A FORMULA TO AN INTEGER
F2E9-           1260 ERRHND .EQ $F2E9     APPLESOFT ON ERROR GOTO HANDLER
F6E6-           1270 ILLQTY .EQ $F6E6     ILLEGAL QUANTITY ROUTINE
FBDD-           1280 BEEP   .EQ $FBDD     BEEP THE SPEAKER
FCA8-           1290 WAIT   .EQ $FCA8     MONITOR DELAY ROUTINE
FDED-           1300 COUT   .EQ $FDED     OUTPUT A CHARACTER
                1310 *
                1320 *-------------------------------
                1330 *
                1340 *       PAGE ZERO
                1350 *
                1360 *
69-             1370 LOMEM  .EQ $69       BASIC VARIABLE START (2 BYTES)
6B-             1380 STARY  .EQ $6B       BASIC START OF ARRAYS (2 BYTES)
6D-             1390 ENDARY .EQ $6D       BASIC END OF ARRAYS (2 BYTES)
6F-             1400 STSTR  .EQ $6F       BASIC START OF STRINGS (2 BYTES)
73-             1410 HIMEM  .EQ $73       BASIC HIGHEST MEMORY (2 BYTES)
D8-             1420 ONERR  .EQ $D8       ON ERROR GOTO ACTIVE WHEN BIT 7 SET
                1430 *
00-             1440 REG    .EQ $00       DEFINE REG AREA START
                1450 *
00-             1460 ADDTRY .EQ REG       READ ADDRESS RETRY COUNTER
01-             1470 BUFF   .EQ REG+1     NIBBLIZED DATA BUFFER POINTER (2 BYTES
03-             1480 CALIB  .EQ REG+3     DISK RECALIBRATION COUNTER
04-             1490 CHECK  .EQ REG+4     DISK CHECKSUM TEMPORARY
05-             1500 CURTRK .EQ REG+5
06-             1510 DISKCK .EQ REG+6     DISK CHECK,SECTOR,TRACK,VOLUME (4 BYTES)
0A-             1520 DRIVE  .EQ REG+10    DRIVE PARAMETER
0B-             1530 HDDIR  .EQ REG+11    DISK HEAD DIRECTION
0C-             1540 HDDLY  .EQ REG+12    DISK PHASE ON DELAY BEFORE NEXT STEP
0D-             1550 HDMOVE .EQ REG+13    DISK HEAD MOVEMENT REQUIRED
0E-             1560 MERGE  .EQ REG+14    MERGE BYTE FOR RDADDR
0F-             1570 PAGE   .EQ REG+15    HI-RES PAGE TO READ/WRITE (2 BYTES)
11-             1580 READCT .EQ REG+17    TRACK READ RETRY COUNTER
12-             1590 RETRY  .EQ REG+18    READ RETRY COUNTER
13-             1600 SLOT   .EQ REG+19    CURRENT DISK SLOT * 16
14-             1610 SHDATA .EQ REG+20    TEMPORARY SHIFT DATA
15-             1620 START  .EQ REG+21    HIRES PAGE HI-BYTE FROM PARAM
16-             1630 STOP   .EQ REG+22    HIRES STOP COUNTER (2 BYTES)
18-             1640 STEPS  .EQ REG+24    DISK HEAD MOVEMENT SO FAR
19-             1650 TRACK  .EQ REG+25    TRACK TO READ/WRITE
                1660 *
19-             1670 REGNUM .EQ TRACK-REG DEFINE HOW MANY REGISTERS WE NEED
                1680 *
                1690 *-------------------------------
                1700 *
07-             1710 ATSECT .EQ DISKCK+1 CURRENT DISK SECTOR LOCATED
08-             1720 ATTRK  .EQ DISKCK+2 CURRENT DISK TRACK LOCATED
                1730 *
                1740 *-------------------------------
                1750 *
                1760 *       DISK DRIVE ADDRESSES
                1770 *
C080-           1780 PHASE  .EQ $C080     BASE OF HEAD STEPPER MOTOR PHASES
C080-           1790 PH0OFF .EQ PHASE     PHASE 0 OFF
C081-           1800 PH0ON  .EQ PHASE+1   PHASE 0 ON
C082-           1810 PH1OFF .EQ PHASE+2   PHASE 1 OFF
C083-           1820 PH1ON  .EQ PHASE+3   PHASE 1 ON
```

```
C084-          PH2OFF .EQ PHASE+4   PHASE 2 OFF
C085-          PH2ON  .EQ PHASE+5   PHASE 2 ON
C086-          PH3OFF .EQ PHASE+6   PHASE 3 OFF
C087-          PH3ON  .EQ PHASE+7   PHASE 3 ON
C088-          DRMOFF .EQ $C088     DRIVE MOTOR OFF
C089-          DRMON  .EQ $C089     DRIVE MOTOR ON
C08A-          DRSEL1 .EQ $C08A     SELECT DRIVE 1
C08B-          DRSEL2 .EQ $C08B     SELECT DRIVE 2
C08C-          DRQ6L  .EQ $C08C     SHIFT WHILE WRITING/READ DATA
C08D-          DRQ6H  .EQ $C08D     LOAD WHILE WRITING/READ WRITE PROTECT
C08E-          DRQ7L  .EQ $C08E     READ
C08F-          DRQ7H  .EQ $C08F     WRITE
               *
               *-------------------------------
               *
7500-          BEGIN  .EQ $7500     BEGINNING OF PROGRAM
               *  NOTE: BEGIN MUST BE ON A PAGE BOUNDARY
               *
                      .OR BEGIN     PACK IT AT THE TOP
               *
7500- A9 85    SETUP  LDA #PARAM    SET UP AMPERSAND VECTOR
7502- 8D F6 03        STA AMPVEC+1
7505- A9 75           LDA /PARAM
7507- 8D F7 03        STA AMPVEC+2
750A- A2 80           LDX #$80      SET UP READ TABLES
750C- BD 00 7A .1     LDA READ6R,X  GET STANDARD READ TABLE
750F- 48              PHA           SAVE FOR LATER
7510- 4A              LSR           SHIFT RIGHT 2 PLACES
7511- 4A              LSR
7512- 9D 00 7B        STA READ4R,X  FORM 4R TABLE
7515- 4A              LSR           SHIFT RIGHT 2 PLACES
7516- 4A              LSR
7517- 9D 00 7C        STA READ2R,X  FORM 2R TABLE
751A- 68              PLA           GET ORIGINAL AGAIN
751B- 0A              ASL           SHIFT LEFT 2 BITS
751C- 0A              ASL
751D- 9D 00 7D        STA READ6L,X  FORM 6L TABLE
7520- 0A              ASL           SHIFT LEFT 2 BITS
7521- 0A              ASL
7522- 9D 00 7E        STA READ4L,X  FORM 4L TABLE
7525- 0A              ASL           SHIFT LEFT 2 BITS
7526- 0A              ASL
7527- 9D 00 7F        STA READ2L,X  FORM 2L TABLE
752A- E8              INX           NEXT
752B- D0 DF           BNE .1        DONE ?
               *
752D- AD 00 BF        LDA GLOBAL    CHECK IF PRODOS
7530- C9 4C           CMP #$4C      JMP OPCODE IF PRODOS
7532- D0 34           BNE .3        MUST BE DOS 3.3
7534- A9 00           LDA #$00
7536- 8D 94 BF        STA LEVEL     SET SYSTEM LEVEL TO ZERO
7539- 8D DE BE        STA SCLOSE+1  MARK "ALL FILES"
753C- A9 CC           LDA #$CC      "CLOSE" COMMAND
753E- D8              CLD           LEGAL CALL
753F- 20 70 BE        JSR GOSYST    CLOSE 'EM
7542- A0 75           LDY /BEGIN    GET START OF PAGES USED
7544- 98       .2     TYA
7545- 29 07           AND #$7       GET PAGE MOD 8
7547- AA              TAX           INDEX INTO BIT MASK
7548- BD CC 79        LDA MASK,X    GET BIT MASK
754B- 48              PHA           HOLD IT
754C- 98              TYA           GET PAGE AGAIN
754D- 4A              LSR           DIVIDE BY 8
754E- 4A              LSR           AND FIND BYTE TO SET
754F- 4A              LSR
7550- AA              TAX           BYTE IS INDEX
7551- 68              PLA           GET BIT MASK BACK
7552- 1D 58 BF        ORA BITMAP,X  MARK PAGE AS USED
7555- 9D 58 BF        STA BITMAP,X
7558- C8              INY           NEXT PAGE
7559- C0 95           CPY /BUFEND+1 ALL PAGES USED MARKED ?
755B- 90 E7           BCC .2        NOT YET
755D- 38              SEC
755E- A9 75           LDA /BEGIN    MOVE ANOTHER $400 FOR CAT BUFFERS
7560- E9 04           SBC #4
7562- 85 74           STA HIMEM+1   SET HIMEM
7564- 85 70           STA STSTR+1   SET START OF STRINGS
7566- D0 06           BNE .4        BRANCH ALWAYS
7568- A9 75    .3     LDA /BEGIN
756A- 85 74           STA HIMEM+1   SET HIMEM
756C- 85 70           STA STSTR+1   SET START OF STRINGS
756E- A9 00    .4     LDA #BEGIN    SET HIMEM TO BEGIN
7570- 85 73           STA HIMEM
7572- 85 6F           STA STSTR
7574- A9 00           LDA #P2END    SET LOMEM ABOVE PAGE 2
7576- 85 69           STA LOMEM     SET LOMEM
7578- 85 6B           STA STARY     SET START OF ARRAYS
757A- 85 6D           STA ENDARY    SET END OF ARRAYS
757C- A9 60           LDA /P2END    HIGH BYTE
757E- 85 6A           STA LOMEM+1   SET LOMEM
7580- 85 6C           STA STARY+1   SET START OF ARRAYS
7582- 85 6E           STA ENDARY+1  SET END OF ARRAYS
7584- 60              RTS
               *
               *-------------------------------
               *
7585- C9 52    PARAM  CMP #'R'      READ TWO TRACKS ?
7587- D0 04           BNE .1        CHECK WRITE
7589- A9 00           LDA #$00      READ
758B- F0 06           BEQ .2        SKIP AROUND
758D- C9 57    .1     CMP #'W'      WRITE TWO TRACKS ?
758F- D0 60           BNE .6        GIVE SYNTAX ERROR
7591- A9 FF           LDA #$FF      WRITE
7593- 48       .2     PHA           SAVE COMMAND ON STACK
7594- 20 B1 00        JSR CHRGET    GET NEXT CHAR
7597- 20 BE DE        JSR CHKCOM    SHOULD BE A COMMA
759A- 20 F8 E6        JSR GETBYT    GET DESIRED PICTURE NUMBER
759D- E0 11           CPX #17       TOO HIGH ?
759F- B0 53           BCS .7        GIVE QUANTITY ERROR
75A1- 8A              TXA           GET IN X
75A2- 0A              ASL           DOUBLE IT
75A3- 48              PHA           SAVE DESIRED TRACK ON STACK
75A4- 20 BE DE        JSR CHKCOM    NEED A COMMA HERE
75A7- 20 F8 E6        JSR GETBYT    EVAL PAGE
75AA- E0 01           CPX #1        PAGE 1 ?
75AC- D0 04           BNE .3        NO
75AE- A9 20           LDA #$20      BASE OF PAGE ONE
75B0- D0 06           BNE .4        BRANCH ALWAYS
75B2- E0 02    .3     CPX #2        PAGE 2 ?
75B4- D0 3E           BNE .7        QUANTITY ERROR
75B6- A9 40           LDA #$40      BASE OF PAGE TWO
75B8- 48       .4     PHA           SAVE PAGE ON STACK
75B9- 20 BE DE        JSR CHKCOM    GET ANOTHER COMMA
75BC- 20 F8 E6        JSR GETBYT    WHICH DRIVE ?
75BF- E0 01           CPX #1        DRIVE 1 ?
75C1- F0 06           BEQ .5        YES
75C3- E0 02           CPX #2        DRIVE 2 ?
75C5- F0 02           BEQ .5        YES
75C7- D0 2B           BNE .7        GIVE A RANGE ERROR
75C9- 8A       .5     TXA           GET DRIVE NUMBER
75CA- 48              PHA           SAVE IT
75CB- 20 75 79        JSR SWAP      BRING IN OUR ZERO PAGE
75CE- 68              PLA           GET DRIVE
75CF- 85 0A           STA DRIVE     SAVE FOR LATER
               *
               * CHECK FOR PRODOS / DOS 3.3
               *
75D1- AD 00 BF        LDA GLOBAL    CHECK PRODOS GLOBAL PAGE
75D4- C9 4C           CMP #$4C      IS IT A JMP ?
75D6- F0 05           BEQ .51       YES - IT'S PRODOS
75D8- AD F8 05        LDA SLOT33    GET DOS 3.3 SLOT
75DB- D0 07           BNE .52       BRANCH ALWAYS - NEVER ZERO
75DD- AD 3C BE .51    LDA SLOTPR    GET PRODOS DEFAULT SLOT
75E0- 0A              ASL           MULTIPLY BY 16
75E1- 0A              ASL
75E2- 0A              ASL
75E3- 0A              ASL
75E4- 85 13    .52    STA SLOT      THIS IS OUR SLOT NOW

75E6- 68              PLA           GET PAGE
75E7- 85 15           STA START     SAVE AS START
75E9- 68              PLA           GET TRACK NUMBER
75EA- 85 19           STA TRACK     SAVE AS TRACK
75EC- 68              PLA           GET COMMAND
75ED- F0 08           BEQ READ      READ 2 TRACKS
75EF- D0 54           BNE WRITE     WRITE 2 TRACKS
75F1- 4C C9 DE .6     JMP SYNTAX    GIVE SYNTAX ERROR
75F4- 4C E6 F6 .7     JMP ILLQTY    GIVE ILLEGAL QUANTITY ERROR
               *
               *-------------------------------
               *
75F7- A9 05    READ   LDA #5        SET NUMBER OF RETRIES
75F9- 85 11           STA READCT
75FB- A9 FF    .1     LDA #$FF      SET STOP ADDRESS
75FD- 85 16           STA STOP
75FF- A9 00           LDA #$00      SET UP PAGE
7601- 85 0F           STA PAGE
7603- A5 15           LDA START     READ FROM THIS PAGE
7605- 85 10           STA PAGE+1
7607- 09 0F           ORA #$0F      SHOULD NOW BE $2F OR $4F
7609- 85 17           STA STOP+1    THIS IS HALF OF PAGE
760B- 20 7D 76        JSR SKTRAK    GET TO THE TRACK
760E- 20 00 78        JSR RD4096    READ 4096 BYTES
7611- 90 06           BCC .2        NO SLIPPED DISKS HERE
7613- C6 11           DEC READCT    TRY AGAIN ?
7615- D0 E4           BNE .1        YES
7617- F0 20           BEQ .4        GIVE 'EM THE ERROR EXIT
7619- A9 05    .2     LDA #5        SET RETRY COUNTER
761B- 85 11           STA READCT
761D- A9 00    .3     LDA #$00      RESET PAGE LOW BYTE
761F- 85 0F           STA PAGE
7621- A5 15           LDA START     DO SECOND HALF
7623- 09 10           ORA #$10      SHOULD NOW BE $30 OR $50
7625- 85 10           STA PAGE+1
7627- 09 0F           ORA #$0F      SHOULD NOW BE $3F OR $5F
7629- 85 17           STA STOP+1
762B- E6 19           INC TRACK     NEXT TRACK
762D- 20 7D 76        JSR SKTRAK    MOVE THE ARM
7630- 20 00 78        JSR RD4096    READ 4096 BYTES
7633- 90 07           BCC .5        THIS TRACK READ OKAY ?
7635- C6 11           DEC READCT    TRY AGAIN ?
7637- D0 E4           BNE .3        YES
7639- 4C 85 79 .4     JMP ERROR     GIVE DISK ERROR EXIT
               *
763C- A6 13    .5     LDX SLOT      GET SLOT
763E- BD 88 C0        LDA DRMOFF,X  TURN OFF THE DRIVE
7641- 20 75 79        JSR SWAP      RESTORE ZERO PAGE
7644- 60              RTS
               *
               *-------------------------------
               *
7645- A9 00    WRITE  LDA #$00      SET PAGE LOW BYTE TO ZERO
7647- 85 0F           STA PAGE
7649- A5 15           LDA START     GET PICTURE START
764B- 85 10           STA PAGE+1    SET PAGE HIGH BYTE
764D- 09 10           ORA #$10      DO HALF OF IT
764F- 85 17           STA STOP+1
7651- 20 54 77        JSR NIBBLE    PRENIBBLE HALF OF PICTURE
7654- 20 7D 76        JSR SKTRAK    MOVE TO THE TRACK
7657- 20 00 77        JSR WR4096    WRITE HALF A PAGE
765A- A9 00           LDA #$00      RESET PAGE LOW BYTE
765C- 85 0F           STA PAGE
765E- A5 15           LDA START     GET START OF PICTURE
7660- 09 10           ORA #$10      NOW IT'S THE MIDDLE
7662- 85 10           STA PAGE+1
7664- 18              CLC           NO CARRY
7665- 69 10           ADC #$10      ADD HALF A PICTURE
7667- 85 17           STA STOP+1    STOP HERE
7669- 20 54 77        JSR NIBBLE    PRENIBBLE OTHER HALF
766C- E6 19           INC TRACK     NEXT TRACK
766E- 20 7D 76        JSR SKTRAK    MOVE TO IT
7671- 20 00 77        JSR WR4096    WRITE OTHER HALF
7674- A6 13           LDX SLOT      GET SLOT
7676- BD 88 C0        LDA DRMOFF,X  TURN OFF DRIVE
7679- 20 75 79        JSR SWAP      RESTORE ZERO PAGE
767C  60              RTS
               *
               *-------------------------------
               *
767D- A5 13    SKTRAK LDA SLOT      GET SLOT NUMBER
```

Listing 2 for for Ultra Fast Pix

ULTRA.FAST *(continued)*

```
767F- AA                  TAX            KEEP IT IN X
7680- 05 0A               ORA DRIVE      ADD IN DRIVE SELECT
7682- A8                  TAY            USE AS INDEX
7683- B9 89 C0            LDA DRSEL1-1,Y  SELECT ENABLE 1 OR 2
7686- BD 8E C0            LDA DRQ7L,X    SET DRIVE FOR READING
7689- BD 8C C0            LDA DRQ6L,X
768C- A9 02               LDA #2         ALLOW TWO RECALIBRATIONS
768E- 85 03               STA CALIB      SET RECALIBRATION COUNTER
7690- A0 08               LDY #8         WAIT 90 USEC FOR DATA CHANGE
7692- BD 8C C0            LDA DRQ6L,X    GET SOME DATA
7695- DD 8C C0     .1     CMP DRQ6L,X    SAME ?
7698- D0 0B               BNE .2         NO - IT CHANGED
769A- 88                  DEY            DONE WAITING ?
769B- D0 F8               BNE .1         NOT YET
769D- BD 89 C0            LDA DRMON,X    TURN ON THE DRIVE
76A0- A9 F3               LDA #243       WAIT FOR 150 MSEC
76A2- 20 A8 FC            JSR WAIT       USE MONITOR DELAY ROUTINE
76A5- A9 0A        .2     LDA #10        TRY TEN TIMES
76A7- 85 12               STA RETRY      SET RETRY COUNTER
76A9- BD 89 C0            LDA DRMON,X    TURN IT ON
76AC- 20 A3 78     .3     JSR RDADDR     READ AN ADDRESS
76AF- 90 17               BCC .4         OKAY - WE HAVE ONE
76B1- C6 12               DEC RETRY      TRY READING SOME MORE
76B3- D0 F7               BNE .3         NOTHING HERE ?
76B5- C6 03               DEC CALIB      TRY RECALIBRATING
76B7- 30 28               BMI .6         GIVE AN I/O ERROR
76B9- A9 02               LDA #2         PRETEND WE'RE A LITTLE BIT OFF
76BB- 85 05               STA CURTRK
76BD- A9 00               LDA #0         MOVE HERE
76BF- 20 01 79            JSR SKABS      MOVE THE HEAD TO TRACK ZERO
76C2- A9 00               LDA #0         CURRENT TRACK IS NOW ZERO
76C4- 85 05               STA CURTRK
76C6- F0 DD               BEQ .2         ALWAYS
76C8- A5 08        .4     LDA ATTRK      WHERE ARE WE ?
76CA- C5 19               CMP TRACK      SAME ?
76CC- F0 0E               BEQ .5         WE'RE HERE
76CE- 85 05               STA CURTRK     WE'RE REALLY AT THIS TRACK
76D0- A5 19               LDA TRACK      GET DESIRED TRACK
76D2- 0A                  ASL            MULTIPLY BY TWO
76D3- 20 01 79            JSR SKABS      GET THE TRACK
76D6- C6 12               DEC RETRY
76D8- D0 D2               BNE .3         MOVE THE HEAD AND TRY AGAIN
76DA- F0 05               BEQ .6         GIVE I/O ERROR
76DC- A5 07        .5     LDA ATSECT     CHECK SECTOR
76DE- D0 CC               BNE .3         LOOK FOR SECTOR ZERO
76E0- 60                  RTS
                   *
76E1- 4C 85 79     .6     JMP ERROR      USE THE DISK ERROR EXIT
                   *
                   *--------------------------------
                   *
                   * MAKE SURE THAT ROUTINE DOESN'T OVERLAP ON TWO PAGES
                   *
76E4-                     .BS $7700-*
                   *
7700- A9 00        WR4096 LDA #BUFMEM    SET UP BUFFER POINTER
7702- 85 01               STA BUFF
7704- A9 80               LDA /BUFMEM
7706- 85 02               STA BUFF+1
7708- BD 8D C0            LDA DRQ6H,X    IGNORE WRITE PROTECT
770B- BD 8E C0            LDA DRQ7L,X    CONFIRM READ
770E- A9 FF               LDA #$FF       WRITE A SELF SYNC BYTE
7710- 9D 8F C0            STA DRQ7H,X    LOAD DATA                      5
7713- DD 8C C0            CMP DRQ6L,X    WRITE IT                       4
7716- EA                  NOP            WASTE 9 CYCLES                 2
7717- 48                  PHA                                           3
7718- 68                  PLA                                           4
7719- A0 10               LDY #$10       WRITE 16 MORE SELF SYNCS       2
771B- 48           .1     PHA            WASTE 7 CYCLES                 3
771C- 68                  PLA                                           4
771D- 20 4B 77            JSR .6         WRITE 40 CYCLE DATA            6
7720- 88                  DEY            ANOTHER ?                      2
7721- D0 F8               BNE .1         YES                            2/3
7723- EA                  NOP            6 CYCLES TO BLEND TO 32        2
7724- EA                  NOP                                           2
7725- EA                  NOP                                           2
7726- B1 01        .2     LDA (BUFF),Y   GET DISK DATA                  5
7728- F0 15               BEQ .4         AT THE END ?                   2/3
772A- 9D 8D C0            STA DRQ6H,X    WRITE DATA                     5
772D- DD 8C C0            CMP DRQ6L,X    SHIFT DATA                     4
7730- C8                  INY            NEXT DISK DATA BYTE            2
7731- D0 06               BNE .3         FINISHED THIS PAGE ?           2/3
7733- E6 02               INC BUFF+1     BUMP POINTER HIGH BYTE         5
7735- EA                  NOP            WASTE 4 CYCLES                 2
7736- EA                  NOP                                           2
7737- D0 ED               BNE .2         BRANCH ALWAYS                  3
7739- EA           .3     NOP            WASTE 8 CYCLES                 2
773A- EA                  NOP                                           2
773B- EA                  NOP                                           2
773C- EA                  NOP                                           2
773D- D0 E7               BNE .2         BRANCH ALWAYS                  3
773F- A0 03        .4     LDY #3         MAKE SURE YOU WRITE $EB
7741- 88           .5     DEY
7742- D0 FD               BNE .5
7744- BD 8E C0            LDA DRQ7L,X    SET BACK TO READ
7747- BD 8C C0            LDA DRQ6L,X
774A- 60                  RTS
                   *
774B- 48           .6     PHA                                           3
774C- 68                  PLA                                           4
774D- 9D 8D C0            STA DRQ6H,X    WRITE DATA                     5
7750- DD 8C C0            CMP DRQ6L,X    SHIFT DATA                     4
7753- 60                  RTS                                           6
                   *
                   *--------------------------------
                   *
7754- A9 00        NIBBLE LDA #BUFMEM    SET UP BUFFER POINTER
7756- 85 01               STA BUFF
7758- A9 80               LDA /BUFMEM
775A- 85 02               STA BUFF+1
775C- A0 00               LDY #$00       NO INDEXING
775E- A9 D5               LDA #$D5       WRITE DISK HEADER BYTES
7760- 91 01               STA (BUFF),Y
7762- E6 01               INC BUFF       NO CARRY TO WORRY ABOUT YET
7764- A9 AA               LDA #$AA
7766- 91 01               STA (BUFF),Y
7768- E6 01               INC BUFF
776A- A9 AD               LDA #$AD
776C- 91 01               STA (BUFF),Y
776E- E6 01               INC BUFF       NO CARRY
7770- B1 0F        .1     LDA (PAGE),Y   GET PICTURE BYTE
7772- 85 14               STA SHDATA     SAVE FOR LATER USE
7774- 4A                  LSR            CONVERT TOP 6 BITS TO LOWER 6 BITS
7775- 4A                  LSR
7776- AA                  TAX            INDEX INTO WRITE TABLE
7777- BD 00 7A            LDA WRTABL,X   GET DISK BYTE
777A- 91 01               STA (BUFF),Y   SAVE IT
777C- E6 01               INC BUFF       NEXT BUFFER LOCATION
777E- D0 02               BNE .2         BUMP HIGH BYTE ?
7780- E6 02               INC BUFF+1
7782- E6 0F        .2     INC PAGE       NEXT PICTURE BYTE
7784- D0 08               BNE .3         BUMP HIGH BYTE ?
7786- E6 10               INC PAGE+1
7788- A5 10               LDA PAGE+1
778A- C5 17               CMP STOP+1     DONE HALF OF PICTURE YET ?
778C- F0 4E               BEQ .8         YES
778E- B1 0F        .3     LDA (PAGE),Y   GET PICTURE BYTE
7790- 48                  PHA            SAVE FOR LATER USE
7791- 66 14               ROR SHDATA     GET BOTTOM TWO BITS OUT
7793- 6A                  ROR            MERGE INTO FOUR BITS FROM THIS BYTE
7794- 66 14               ROR SHDATA
7796- 6A                  ROR
7797- 4A                  LSR            SHIFT TO BOTTOM 6 BITS
7798- 4A                  LSR
7799- AA                  TAX            INDEX INTO WRITE TABLE
779A- BD 00 7A            LDA WRTABL,X
779D- 91 01               STA (BUFF),Y   SAVE IN BUFFER
779F- E6 01               INC BUFF       NEXT BUFFER LOCATION
77A1- D0 02               BNE .4
77A3- E6 02               INC BUFF+1     BUMP HIGH BYTE ?
77A5- E6 0F        .4     INC PAGE       NEXT PICTURE BYTE
77A7- D0 02               BNE .5
77A9- E6 10               INC PAGE+1
77AB- B1 0F        .5     LDA (PAGE),Y   GET NEXT PICTURE BYTE
77AD- 85 14               STA SHDATA     SAVE FOR SHIFTING
77AF- 68                  PLA            GET BACK LAST PICTURE BYTE
77B0- 29 0F               AND #$0F       GET LOWER 4 BITS
77B2- 26 14               ROL SHDATA     PLUS UPPER TWO OF ADJACENT BYTE
77B4- 2A                  ROL
77B5- 26 14               ROL SHDATA
77B7- 2A                  ROL
77B8- AA                  TAX            INDEX INTO WRITE DATA TABLE
77B9- BD 00 7A            LDA WRTABL,X   GET DISK DATA BYTE
77BC- 91 01               STA (BUFF),Y   SAVE IN BUFFER
77BE- E6 01               INC BUFF       NEXT BUFFER LOCATION
77C0- D0 02               BNE .6         BUMP HIGH BYTE ?
77C2- E6 02               INC BUFF+1
77C4- B1 0F        .6     LDA (PAGE),Y   GET PICTURE BYTE AGAIN
77C6- 29 3F               AND #$3F       LOWER 6 BITS ONLY
77C8- AA                  TAX            INDEX INTO WRITE DATA TABLE
77C9- BD 00 7A            LDA WRTABL,X   GET DISK DATA BYTE
77CC- 91 01               STA (BUFF),Y   PUT IN BUFFER
77CE- E6 01               INC BUFF       NEXT BUFFER LOCATION
77D0- D0 02               BNE .7         BUMP HIGH BYTE ?
77D2- E6 02               INC BUFF+1
77D4- E6 0F        .7     INC PAGE       NEXT PICTURE BYTE
77D6- D0 98               BNE .1         BUMP HIGH BYTE ?
77D8- E6 10               INC PAGE+1
77DA- D0 94               BNE .1         BRANCH ALWAYS
77DC- A5 14        .8     LDA SHDATA     WRITE LAST TWO BITS
77DE- 29 03               AND #$03       JUST BOTTOM TWO
77E0- AA                  TAX            USE AS INDEX
77E1- BD 00 7A            LDA WRTABL,X   GET DISK BYTE
77E4- 91 01               STA (BUFF),Y   SAVE IN BUFFER
77E6- E6 01               INC BUFF       NEXT BUFFER LOCATION
77E8- A9 DE               LDA #$DE       LAST BYTES ARE DISK TRAILER BYTES
77EA- 91 01               STA (BUFF),Y   SAVE IT
77EC- E6 01               INC BUFF       NO CARRY TO WORRY ABOUT
77EE- A9 AA               LDA #$AA
77F0- 91 01               STA (BUFF),Y
77F2- E6 01               INC BUFF
77F4- A9 EB               LDA #$EB       LAST TRAILER BYTE
77F6- 91 01               STA (BUFF),Y
77F8- E6 01               INC BUFF
77FA- A9 00               LDA #$00       LAST BYTE IS ZERO
77FC- 91 01               STA (BUFF),Y
77FE- 60                  RTS
                   *
                   *--------------------------------
                   *
                   *      MAKE SURE THAT READ DOESN'T OVERLAP ON TWO PAGES
                   *
77FF-                     .BS $7800-*
                   *
7800- 8A           RD4096 TXA            GET SLOT IN ACC
7801- 09 8C               ORA #$8C       MAKE Q6L ADDRESSES CORRECT
7803- 8D 32 78            STA .05+1
7806- 8D 44 78            STA .06+1
7809- 8D 56 78            STA .07+1
780C- 8D 68 78            STA .08+1
780F- 8D 79 78            STA .09+1
7812- BD 8C C0     .01    LDA DRQ6L,X    READ SHIFT REGISTER
7815- 10 FB               BPL .01        WAIT FOR FULL BYTE
7817- C9 D5        .02    CMP #$D5       FIND $D5 ?
7819- D0 F7               BNE .01        NO
781B- EA                  NOP            DELAY
781C- BD 8C C0     .03    LDA DRQ6L,X    READ SHIFT REGISTER
781F- 10 FB               BPL .03        WAIT FOR FULL BYTE
7821- C9 AA               CMP #$AA       FIND $AA ?
7823- D0 F2               BNE .02        NO
7825  EA                  NOP            DELAY
7826- BD 8C C0     .04    LDA DRQ6L,X    READ SHIFT REGISTER
7829- 10 FB               BPL .04        WAIT FOR FULL BYTE
782B- C9 AD               CMP #$AD       FIND $AD ?
782D- D0 E8               BNE .02        NO
```

```
                  6390 *--------------------------------
                  6400 * START READING 4096 BYTES
782F- A0 00       6410        LDY #$00      INITIALIZE Y
                  6420 *--------------------------------
7831- AE 8C C0    6430 .05    LDX DRQ6L     READ DISK BYTE 1            4
7834- 10 FB       6440        BPL .05       WAIT FOR 8 BITS             2/3
7836- BD 00 7D    6450        LDA READ6L,X  CONVERT IT                  4
7839- C4 16       6460        CPY STOP      DONE ?                      3
783B- D0 06       6470        BNE .06       NO                          2/3
783D- A6 10       6480        LDX PAGE+1    CHECK HIGH BYTE             3
783F- E4 17       6490        CPX STOP+1    DONE ?                      3
7841- F0 35       6500        BEQ .09       YES - ONE MORE BYTE         2/3
                  6510 *--------------------------------
7843- AE 8C C0    6520 .06    LDX DRQ6L     READ DISK BYTE 2            4
7846- 10 FB       6530        BPL .06       WAIT FOR 8 BITS             2/3
7848- 1D 00 7C    6540        ORA READ2R,X  ADD INTO FIRST DISK BYTE    4
784B- 91 0F       6550        STA (PAGE),Y  SAVE IN HIRES AREA          6
784D- BD 00 7E    6560        LDA READ4L,X  GET NEXT PART               4
7850- C8          6570        INY           NEXT HIRES BYTE             2
7851- D0 02       6580        BNE .07       NEXT 256 BYTES ?            2/3
7853- E6 10       6590        INC PAGE+1    BUMP HIGH BYTE OF POINTER   5
                  6600 *--------------------------------
7855- AE 8C C0    6610 .07    LDX DRQ6L     READ DISK BYTE              4
7858- 10 FB       6620        BPL .07       WAIT FOR 8 BITS             2/3
785A- 1D 00 7B    6630        ORA READ4R,X  ADD INTO SECOND DISK BYTE   4
785D- 91 0F       6640        STA (PAGE),Y  SAVE IN HIRES AREA          6
785F- BD 00 7F    6650        LDA READ2L,X  GET NEXT PART               4
7862- C8          6660        INY           NEXT HIRES BYTE             2
7863- D0 02       6670        BNE .08       WRAPPED 256 BYTES ?         2/3
7865- E6 10       6680        INC PAGE+1    BUMP HIGH BYTE OF POINTER   5
                  6690 *--------------------------------
7867- AE 8C C0    6700 .08    LDX DRQ6L     READ DISK BYTE              4
786A- 10 FB       6710        BPL .08       WAIT FOR 8 BITS             2/3
786C- 1D 00 7A    6720        ORA READ6R,X  ADD INTO THIRD BYTE         4
786F- 91 0F       6730        STA (PAGE),Y  SAVE IN HIRES AREA          6
7871- C8          6740        INY           256 BYTES DONE ?            2
7872- D0 BD       6750        BNE .05       KEEP GOING                  2/3
7874- E6 10       6760        INC PAGE+1    BUMP POINTER HIGH BYTE      5
7876- D0 B9       6770        BNE .05       BRANCH ALWAYS               3
                  6780 *--------------------------------
7878- AE 8C C0    6790 .09    LDX DRQ6L     READ LAST DISK BYTE         4
787B- 10 FB       6800        BPL .09       WAIT FOR 8 BITS             2/3
787D- 1D 00 7A    6810        ORA READ6R,X  DON'T SHIFT LAST 2 BITS     4
7880- 91 0F       6820        STA (PAGE),Y  SAVE IT                     6
                  6830 *--------------------------------
7882- A6 13       6840        LDX SLOT      GET SLOT
7884- BD 8C C0    6850 .10    LDA DRQ6L,X   READ CHECK BYTE
7887- 10 FB       6860        BPL .10       GOT EIGHT BITS ?
7889- C9 DE       6870        CMP #$DE      IS IT $DE ?
788B- D0 14       6880        BNE .13       NO
788D- BD 8C C0    6890 .11    LDA DRQ6L,X   READ NEXT CHECK BYTE
7890- 10 FB       6900        BPL .11       EIGHT BITS ?
7892- C9 AA       6910        CMP #$AA      IS IT $AA ?
7894- D0 08       6920        BNE .13       NO
7896- BD 8C C0    6930 .12    LDA DRQ6L,X   READ LAST CHECK BYTE
7899- 10 FB       6940        BPL .12       EIGHT BITS ?
789B- C9 EB       6950        CMP #$EB      IS IT $EB ?
789D- D0 02       6960        BNE .13       NO
789F- 18          6970        CLC           SHOW GOOD READ
78A0- 60          6980        RTS
                  6990 *
                  7000 *
78A1- 38          7010 .13    SEC           SHOW SLIPPED DISK
78A2- 60          7020        RTS
                  7030 *
                  7040 *--------------------------------
                  7050 *      MAKE SURE THAT RDADDR
                  7060 *       DOESN'T OVERLAP PAGE BOUNDARY
                  7070 *
78A3- A0 24       7080 RDADDR LDY #$24      TRY A FEW TIMES
78A5- 84 00       7090        STY ADDTRY    ABOUT 6274 TIMES (FULL REVOLUTION)
78A7- 88          7100 .1     DEY
78A8- D0 04       7110        BNE .2        BORROW ?
78AA- C6 00       7120        DEC ADDTRY    DEC HIGH BYTE
78AC- F0 51       7130        BEQ .11       GIVE AN ERROR
78AE- BD 8C C0    7140 .2     LDA DRQ6L,X   READ DISK BYTE
78B1- 10 FB       7150        BPL .2        WAIT FOR 8 BITS
78B3- C9 D5       7160 .3     CMP #$D5      IS IT $D5 ?
78B5- D0 F0       7170        BNE .1        NO - KEEP LOOKING
78B7- EA          7180        NOP           WAIT
78B8- BD 8C C0    7190 .4     LDA DRQ6L,X   READ NEXT DISK BYTE
78BB- 10 FB       7200        BPL .4        WAIT FOR EIGHT BITS
78BD- C9 AA       7210        CMP #$AA      IS IT $AA ?
78BF- D0 F2       7220        BNE .3        NO - MIGHT BE $D5
78C1- A0 03       7230        LDY #$03      GET READY FOR INDEXING
78C3- BD 8C C0    7240 .5     LDA DRQ6L,X   READ LAST ADDRESS HEADER BYTE
78C6- 10 FB       7250        BPL .5        WAIT FOR EIGHT BITS
78C8- C9 96       7260        CMP #$96      IS IT $96 ?
78CA- D0 E7       7270        BNE .3        NO - MIGHT BE $D5
78CC- A9 00       7280        LDA #$00      RESET CHECKSUM
78CE- 85 04       7290 .6     STA CHECK
78D0- BD 8C C0    7300 .7     LDA DRQ6L,X   READ DISK BYTE
78D3- 10 FB       7310        BPL .7        WAIT FOR EIGHT BITS
78D5- 2A          7320        ROL           SHIFT IT
78D6- 85 0E       7330        STA MERGE     SAVE TEMPORARILY
78D8- BD 8C C0    7340 .8     LDA DRQ6L,X   READ DISK BYTE
78DB- 10 FB       7350        BPL .8        WAIT FOR EIGHT BITS
78DD- 25 0E       7360        AND MERGE     MERGE IT
78DF- 99 06 00    7370        STA DISKCK,Y  DISKCK   = CHECKSUM
78E2- 45 04       7380        EOR CHECK     DISKCK+1 = SECTOR
78E4- 88          7390        DEY           DISKCK+2 = TRACK
78E5- 10 E7       7400        BPL .6        DISKCK+3 = VOLUME
78E7- A8          7410        TAY           CHECK CHECKSUM
78E8- D0 15       7420        BNE .11       SET CARRY FOR ERROR
78EA- BD 8C C0    7430 .9     LDA DRQ6L,X   READ DISK BYTE
78ED- 10 FB       7440        BPL .9        WAIT FOR 8 BITS
78EF- C9 DE       7450        CMP #$DE      VALID TRAILER ?
78F1- D0 0C       7460        BNE .11       GIVE AN ERROR
78F3- EA          7470        NOP           WAIT A LITTLE
78F4- BD 8C C0    7480 .10    LDA DRQ6L,X   READ DISK BYTE
78F7- 10 FB       7490        BPL .10       WAIT FOR EIGHT BITS
78F9- C9 AA       7500        CMP #$AA      VALID TRAILER ?
78FB- D0 02       7510        BNE .11       NO - GIVE ERROR
78FD- 18          7520        CLC           SHOW NO ERRORS
78FE- 60          7530        RTS
78FF- 38          7540 .11    SEC           SHOW ERROR
7900- 60          7550        RTS
                  7560 *
                  7570 *--------------------------------
                  7580 *
7901- 48          7590 SKABS  PHA           SAVE DESIRED TRACK
7902- 06 05       7600        ASL CURTRK    TWO PHASES PER TRACK
7904- A0 00       7610        LDY #$00      SET INDEX TO ZERO
7906- 84 18       7620        STY STEPS     SET STEPS SO FAR TO ZERO
7908- 38          7630        SEC           ENTER WITH DESIRED TRACK.2 IN ACC
7909- E5 05       7640        SBC CURTRK    HOW FAR DO WE MOVE ?
790B- 85 0B       7650        STA HDDIR     IN = + , OUT = -
790D- 10 07       7660        BPL .1        KEEP IT POSITIVE
790F- 85 0B       7670        STA HDDIR     SET IT OUTWARDS
7911- 18          7680        CLC
7912- 49 FF       7690        EOR #$FF      INVERT IT
7914- 69 01       7700        ADC #01       AND ADD ONE
7916- 85 0D       7710 .1     STA HDMOVE    SAVE IT
7918- 66 05       7720        ROR CURTRK    DIVIDE BY TWO
791A- 66 05       7730        ROR CURTRK    CHECK IF TRACK ODD ?
791C- A5 0B       7740        LDA HDDIR     CHECK HEAD DIRECTION TOO
791E- 90 04       7750        BCC .2        IT'S EVEN
7920- 10 04       7760        BPL .3        ODD - AND INWARD (Y=2)
7922- 30 04       7770        BMI .4        ODD - AND OUTWARD (Y=0)
7924- 10 02       7780 .2     BPL .4        EVEN - AND INWARD (Y=0)
7926- A0 02       7790 .3     LDY #$02      (EVEN AND OUT) OR (ODD AND IN)
7928- A5 0D       7800 .4     LDA HDMOVE    HOW FAR NOW ?
792A- F0 42       7810        BEQ .11       WE'RE DONE
792C- AA          7820        TAX           NOW X IS CLEAR
792D- E4 18       7830        CPX STEPS     HOW FAR HAVE WE GONE
792F- 90 02       7840        BCC .5        HDMOVE < STEPS
7931- A6 18       7850        LDX STEPS     GET THE LOWEST
7933- E0 08       7860 .5     CPX #8        KEEP IT UNDER 7
7935- 90 02       7870        BCC .6        ALREADY LESS THAN 7
7937- A2 07       7880        LDX #7        MAKE IT 7
7939- BD C4 79    7890 .6     LDA DLYTBL,X  GET THE DELAY
793C- 85 0C       7900        STA HDDLY     SAVE THE DELAY
793E- B9 C0 79    7910        LDA PHSTBL,Y  FIND PHASE TO TURN ON
7941- 05 13       7920        ORA SLOT      OR IN THE SLOT
7943- AA          7930        TAX           USE AS INDEX
7944- BD 80 C0    7940        LDA PHASE,X   TURN IT ON
7947- A9 13       7950 .7     LDA #19       DELAY FOR HDDLY * 100 MICROSECONDS
7949- E9 01       7960 .8     SBC #1        DECREMENT
794B- D0 FC       7970        BNE .8
794D- C6 0C       7980        DEC HDDLY     DECREMENT DELAY
794F- D0 F6       7990        BNE .7        KEEP GOING
7951- CA          8000        DEX           TURN IT BACK OFF
7952- BD 80 C0    8010        LDA PHASE,X   OFF THIS TIME
7955- A5 0B       8020        LDA HDDIR     WHAT DIRECTION ?
7957- 30 09       8030        BMI .9        OUTWARDS IF NEGATIVE
7959- C8          8040        INY           NEXT PHASE
795A- C0 04       8050        CPY #4        OUT OF TABLE ?
795C- D0 09       8060        BNE .10       NOT YET
795E- A0 00       8070        LDY #0        RESTART AT ZERO
7960- F0 05       8080        BEQ .10       SKIP OVER
7962- 88          8090 .9     DEY           NEXT PHASE
7963- 10 02       8100        BPL .10       STILL IN TABLE
7965- A0 03       8110        LDY #3        START AT END
7967- C6 0D       8120 .10    DEC HDMOVE    NEXT MOVEMENT
7969- E6 18       8130        INC STEPS     BUMP THE STEP COUNT
796B- 4C 28 79    8140        JMP .4        KEEP GOING
796E- 68          8150 .11    PLA           GET DESIRED TRACK
796F- 4A          8160        LSR           DIVIDE BY TWO FOR ACTUAL TRACK
7970- 85 05       8170        STA CURTRK    PASS IT BACK
7972- A6 13       8180        LDX SLOT      GET SLOT BACK
7974- 60          8190        RTS
                  8200 *
                  8210 *--------------------------------
                  8220 *
7975- A2 19       8230 SWAP   LDX #REGNUM   GET NUMBER TO SWAP
7977- B5 00       8240 .1     LDA REG,X     GET ZERO PAGE REG
7979- BC D4 79    8250        LDY REGSAV,X  GET SAVE AREA
797C- 9D D4 79    8260        STA REGSAV,X  SAVE ZERO PAGE REG
797F- 94 00       8270        STY REG,X     MOVE REG SAVE TO ZERO PAGE
7981- CA          8280        DEX           NEXT REG
7982- 10 F3       8290        BPL .1        MORE ?
7984- 60          8300        RTS
                  8310 *
                  8320 *--------------------------------
                  8330 *
7985- A6 13       8340 ERROR  LDX SLOT      GET SLOT
7987- BD 88 C0    8350        LDA DRWOFF,X  TURN OFF THE DRIVE
798A- 24 D8       8360        BIT ONERR     IS ON ERROR GOTO ACTIVE ?
798C- 10 05       8370        BPL .1        NO - PRINT MESSAGE
798E- A2 08       8380        LDX #$08      SHOW AN I/O ERROR
7990- 4C E9 F2    8390        JMP ERRHND    USE APPLESOFT ERROR HANDLER
7993- 20 DD FB    8400 .1     JSR BEEP      BEEP THE SPEAKER
7996- A2 00       8410        LDX #$00      PRINT ENTIRE MESSAGE
7998- BD B3 79    8420 .2     LDA MESS,X    GET CHARACTER
799B- F0 06       8430        BEQ .3        DONE
799D- 20 ED FD    8440        JSR COUT      SEND IT
79A0- E8          8450        INX           NEXT CHARACTER
79A1- D0 F5       8460        BNE .2        BRANCH ALWAYS
79A3- 20 75 79    8470 .3     JSR SWAP      RESTORE THE ORIGINAL ZERO PAGE
79A6- AD 00 BF    8480        LDA GLOBAL    GET PRODOS GLOBAL START
79A9- C9 4C       8490        CMP #$4C      IT'S A JMP IF PRODOS
79AB- F0 03       8500        BEQ .4        YES - IT'S PRODOS
79AD- 4C D0 03    8510        JMP WARM33    GO DO DOS 3.3 BASIC WARMSTART
79B0- 4C 00 BE    8520 .4     JMP WARMPR    GO DO PRODOS BASIC WARMSTART
                  8530 *
79B3- 8D          8540 MESS   .DA #$8D      RETURN
79B4- C4 C9 D3
79B7- CB A0 C5
79BA- D2 D2 CF
79BD- D2          8550        .AS -"DISK ERROR"
79BE- 8D 00       8560        .DA #$8D,#$00  RETURN , END
                  8570 *
79C0- 03 05 07
79C3- 01          8580 PHSTBL .DA #3,#5,#7,#1   PHASE-ON ADDRESSES
79C4- 70 6C 68
79C7- 64 60 5C
79CA- 5A 58       8590 DLYTBL .DA #$70,#$6C,#$68,#$64,#$60,#$5C,#$5A,#$58
                  8600 *
79CC- 80 40 20
79CF- 10 08 04
79D2- 02 01       8610 MASK   .HS 8040201008040201  BIT MASK FOR PRODOS
                  8620 *
```

Listing 2 for for Ultra Fast Pix

ULTRA.FAST *(continued)*

```
                *       ZERO PAGE SWAP AREA
                *
79D4-           REGSAV .BS REGNUM   RESERVE JUST ENOUGH ROOM
                *
79ED-                  .BS $7A00-*  MOVE TO NEAREST PAGE BEGINNING
                *
7A00- 96 97 9A
7A03- 9B 9D 9E
7A06- 9F A6     WRTABL .HS 96979A9B9D9E9FA6
7A08- A7 AB AC
7A0B- AD AE AF
7A0E- B2 B3            .HS A7ABACADAEAFB2B3
7A10- B4 B5 B6
7A13- B7 B9 BA
7A16- BB BC            .HS B4B5B6B7B9BABBBC
7A18- BD BE BF
7A1B- CB CD CE
7A1E- CF D3            .HS BDBEBFCBCDCECFD3
7A20- D6 D7 D9
7A23- DA DB DC
7A26- DD DE            .HS D6D7D9DADBDCDDDE
7A28- DF E5 E6
7A2B- E7 E9 EA
7A2E- EB EC            .HS DFE5E6E7E9EAEBEC
7A30- ED EE EF
7A33- F2 F3 F4
7A36- F5 F6            .HS EDEEEFF2F3F4F5F6
7A38- F7 F9 FA
7A3B- FB FC FD
7A3E- FE FF            .HS F7F9FAFBFCFDFEFF
                *
                *
                *       READ TABLE DEFINITION
                *
                *       SIX DATA BITS ARE 1-6
                *
                *       READ6L = 65432100
                *       READ2R = 00000065
                *       READ4L = 43210000
                *       READ4R = 00006543
                *       READ2L = 21000000
                *       READ6R = 00654321
                *
                *       SO FOUR BYTES READ ARE
                *       SPLIT INTO THREE BYTES AS:
                *
                * BYTE 1 = D1(READ6L)+D2(READ2R)
                * BYTE 2 = D2(READ4L)+D3(READ4R)
                * BYTE 3 = D3(READ2L)+D4(READ6R)
                *
7A00-           READ6R .EQ WRTABL   PLACE WRTABL IN SPARSE READ6R
7A40-                  .BS $7A80-* MOVE UP TO LAST 80 BYTES IN PAGE
                *
7A80- 00 00 00
7A83- 00 00 00
7A86- 00 00            .HS 0000000000000000 80-87
7A88- 00 00 00
7A8B- 00 00 00
7A8E- 00 00            .HS 0000000000000000 88-8F
7A90- 00 00 00
7A93- 00 00 00
7A96- 00 01            .HS 0000000000000001 90-97
7A98- 00 00 02
7A9B- 03 00 04
7A9E- 05 06            .HS 0000020300040506 98-9F
7AA0- 00 00 00
7AA3- 00 00 00
7AA6- 07 08            .HS 0000000000000708 A0-A7
7AA8- 00 00 00
7AAB- 09 0A 0B
7AAE- 0C 0D            .HS 000000090A0B0C0D A8-AF
7AB0- 00 00 0E
7AB3- 0F 10 11
7AB6- 12 13            .HS 00000E0F10111213 B0-B7
7AB8- 00 14 15
7ABB- 16 17 18
7ABE- 19 1A            .HS 001415161718191A B8-BF
7AC0- 00 00 00
7AC3- 00 00 00
7AC6- 00 00            .HS 0000000000000000 C0-C7
7AC8- 00 00 00
7ACB- 1B 00 1C
7ACE- 1D 1E            .HS 0000001B001C1D1E C8-CF
7AD0- 00 00 00
7AD3- 1F 00 00
7AD6- 20 21            .HS 0000001F00002021 D0-D7
7AD8- 00 22 23
7ADB- 24 25 26
7ADE- 27 28            .HS 0022232425262728 D8-DF
7AE0- 00 00 00
7AE3- 00 00 29
7AE6- 2A 2B            .HS 0000000000292A2B E0-E7
7AE8- 00 2C 2D
7AEB- 2E 2F 30
7AEE- 31 32            .HS 002C2D2E2F303132 E8-EF
7AF0- 00 00 33
7AF3- 34 35 36
7AF6- 37 38            .HS 0000333435363738 F0-F7
7AF8- 00 39 3A
7AFB- 3B 3C 3D
7AFE- 3E 3F            .HS 00393A3B3C3D3E3F F8-FF
                *
7B00-           READ4R .BS $100
7C00-           READ2R .BS $100
7D00-           READ6L .BS $100
7E00-           READ4L .BS $100
7F00-           READ2L .BS $100
8000-           BUFMEM .BS 5469     WRITE PRENIBBLE BUFFER
955D-           BUFEND .EQ *
                *---------------------------------
                *
205D-           ZZSIZE .EQ *-SETUP  PROGRAM SIZE
```

END OF LISTING 2