SPECIAL PROGRAM

by Steve Ellis

PICTURE THIS

Enhance ProDOS 8 with a command for loading Super Hi-Res graphics

Until now, viewing Super Hi-Res pictures on the Apple IIGS has been rather cumbersome. Doing so used to require a ProDOS 16 paint program. But PLOAD (for Picture LOAD) adds a new command to ProDOS BASIC that allows you to load any Super Hi-Res picture in compressed or uncompressed format. The pictures are loaded as quickly as ProDOS allows — which from a RAM disk or 3.5-inch disk is pretty speedy. And you don't ever have to leave the familiar BASIC.SYSTEM environment.

USING THE PROGRAMS

Install PLOAD by BRUNning PLOAD.INSTALL from BASIC. It becomes a new command that is used with the following syntax:

```
PLOAD picname [,Sn][,Dn]
```

where *picname* is the pathname of the picture file, and the optional parameters S*n* and D*n* specify slot and drive.

PLOAD will automatically determine the filetype of the picture you choose (packed or unpacked) and load the picture onto the Super Hi-Res screen. All you have to do is set bit 7 of $C029 to turn on Super Hi-Res and display the picture. You can do this from BASIC with POKE 49193,193.To turn off Super Hi-Res, use the command POKE 49123,65. WARNING: Do not POKE any values at 49193 other than the two listed; doing so can have disastrous results.

Besides giving you the PLOAD command, installing PLOAD will affect the appearance of your disk directories when CATALOGed. PLOAD adds two new filename descriptors to BASIC.SYSTEM. PIC is the new filetype corresponding to unpacked, 32K picture files. PNT is the new filetype for packed picture files. Without PLOAD, these appear in your CATALOG as files of type $C1 and $C0, respectively.

I have included a slide show program that uses PLOAD to display all the Super Hi-Res pictures on a disk or in a directory. To use the slide show, simply RUN SLIDE.SHOW. Enter the prefix or slot and drive of the disk that contains the pictures you want to display. You can choose to have the pictures displayed with or without a title at the bottom of the screen. Be sure to end the program by choosing QUIT, or QuickDraw (the tools in the IIGS Toolbox that do all the Super Hi-Res graphics) will not shut down properly and may not be able to restart.

Steve Ellis, P.O. Box 237, New Ulm, TX 78950. The programs are compatible with ProDOS only.

ENTERING THE PROGRAM

If you have Merlin 16, just enter the source code in **Listing 1**, assemble it, and save the object code as PLOAD.INSTALL. Notice that in **line 183** the period performs an OR on the value COMMAND. If your assembler doesn't use a period to do an OR, change this line accordingly. If you don't have Merlin 16 or a comparable assembler that works with 24-bit addresses, get into the Monitor with CALL −151 and enter the hex dump in **Listing 2**.
Save the program with

```
BSAVE PLOAD.INSTALL,A$4000,L$3AD
```

To enter the slide show, type in the program in **Listing 3** and save it with the command

```
SAVE SLIDE.SHOW
```

The machine language driver for SLIDE.SHOW can be entered either with Merlin 16 (**Listing 4**) or directly from the hex dump (**Listing 5**). If entered from the hex dump, save the program with

```
BSAVE SLIDE.OBJ,A$300,L$63
```

For help on entering the programs, see the Typing Tips section.

HOW THE PROGRAM WORKS

PLOAD requests memory space from ProDOS via the GETBUFR routine. Since the address returned by ProDOS can vary depending on several factors, including the number of external commands and the number of open files, you can't know the final address of your code beforehand. Traditionally, external commands have used relocation routines to relocate the code. In contrast, PLOAD uses several of the machine instructions new to the Apple IIGS, which make writing position-independent code possible.

The major sections of the program are described below:

1. The first step is to install the command into ProDOS. The program checks to see if the command has already been installed; if so there is no need to continue and control returns immediately to BASIC. It then checks to see if BASIC version 1.1 is running. If so, two new filetypes, PIC and PNT, are added to the catalog descriptors. PIC is a 32K screen file and PNT is a compressed paint file. Next, the preinstallation top of free memory is saved and space for the program is requested from ProDOS. After changing a few values to relocate the program

to the new address returned by ProDOS, the program is protected in the system bitmap, a link is established to any previously installed commands, and PLOAD is moved to its final resting place in high memory.

2. Whenever ProDOS does not recognize a command, it passes the command through the external command vector at $BE06. Upon receiving control via that vector, ProDOS scans the input line to see if the command is PLOAD. Note the use of the PER instruction in **line 194.** PER is one of the new 65816 instructions that makes position-independent coding possible. PER calculates the offset from the program counter to the address given in the instruction and places the result on the stack. The address on the stack can then be accessed using the stack indirect addressing mode, as in **line 202**. In that line, each character from the input line (excluding spaces) is compared to the character at the address on the stack, indexed by Y (confusing, yes, but study it carefully, because it's a very powerful instruction). Once it's determined the command is PLOAD, the BI parameter list is set up to require a pathname and to allow optional slot and drive parameters, and the command is sent back to the BI to parse it. If the command is not PLOAD, a jump is made to any other external commands or back to the BI if there aren't any.

3. DO_CMD is where the actual picture loading begins. PLOAD requires several zero page locations, so the first thing it does is save their contents. Then several pointers are set up to point to the picture data buffer and the Super Hi-Res screen. Linear mapping of the Super Hi-Res screen is enabled in **lines 247-248**. The file is opened, and its reference number is copied. After the file's information is retrieved with a GET_FILE_INFO call, the main filetype is tested. A $C1 means the file is an unpacked, 32K image, and causes a branch to the appropriate loading routine. BIN files are also allowed to be loaded as if they were $C1 files, but if they are not precisely 32K long, an end of data error will occur. A filetype of $C0 causes a branch to the routine to handle packed picture files. Any other filetype results in a filetype mismatch error.

4. CLOSE is the PLOAD shutdown routine. The file is closed and the zero page contents it used are restored. Command returns via an RTS.

5. MLIERR is called whenever a ProDOS error occurs. The file is closed (via a call to CLOSE) and control jumps to the BASIC error handler in the BI global page.

6. If the filetype is $C1 or BIN, a 32K screen image is loaded. One data block is read in at a time (the size of the block depends on the setting of BLKSIZE). The MVN instruction is used to quickly and easily move the block into its destination in the Super Hi-Res page. Once $E1A000 is reached, the picture is completely loaded, the file is closed, and control returns to BASIC.

7. Packed paint files are further described by their auxiliary file types. A check is made to see what the file's aux type is, and the appropriate routine is called.

8. Aux type $00 files have the following format:

```
bytes $000-$01F: palette
bytes $020-$021: background color
bytes $022-$221: 16 patterns each with 32 bytes
bytes $222-end : packed picture data
```

First the palette is read. Then the SCB storage area is zeroed and the palette is moved to the palette area. The file mark is then set past the palette, background color, and patterns to byte $222 of the file, and a block of picture data is read. The data block is unpacked and if the picture is not finished, the program loops back to read another block.

9. Aux type $01 files are the easiest to unpack because they consist only of the screen image from $E12000-$E19FFF. The procedure to unpack them is similar to that for type $00, except no palette needs to be read in first. The file is simply read in and unpacked, block by block.

10. Aux type $02 files are the most complex of the three. Their format is shown in **Table 1.**

The first step in loading an aux type $02 file is determining its screen width. If the width is not 320 or 640, a range error occurs. The reason for not allowing other screen widths is this: In the file header, listed above, each screen line has an associated scan line entry. This is a 4-byte value that tells how many pixels are on the line. Unpacking nonstandard screen widths would require using this scan line entry. Instead, PLOAD ignores these entries and assumes the data is to be unpacked to a standard width screen. Doing otherwise would require reading the scan line entry, finding the pixel data corresponding to that line in the file, unpacking it, and repositioning to the next scan line entry. This would entail quite a lot of back and forth movement in the file, and would take up a large buffer space and more code for the PLOAD command.

Once PLOAD has determined you have a "legal" picture, it copies the master scan line control byte given in the file to the scan line control byte storage area from $E19D00-$E19DC7. The area from $E19DC8 through $E19DFF must contain 0's, so this is taken care of next. The program then counts the number of palettes in the file and moves each up to where it belongs. Of course, only one palette is displayed at a time on the screen, so only the first palette is of any consequence, but it doesn't hurt to move them all. At this point, you are positioned in the file just before the list of scan line directory entries (mentioned in the caveat above). Each entry is four bytes, so PLOAD simply skips four bytes for each vertical line in the picture, and you have (almost) the address of the start of the packed picture data. All that remains is to fudge the address to account for the 17 bytes that start the file. After all that the file position is correct, so the mark is set and a block is read. It is then unpacked, and a loop is executed just like in steps 8 and 9.

Table 1: AUX Type $02 File Formats

Byte	Function
$000-$003:	size of block
$004-$008:	STR 'MAIN'
$009-$00A:	SCB mode word, only low byte is significant
$00B-$00C:	number of pixels per scan line
$00D-$00E:	number of palettes
$00F-$XXX:	palette data, end address depends on number of palettes
$XXX+1-$XXX+2:	number of scan lines
$XXX+3-$XXX+*n*:	scan line directory entries, *n* depends on number of scan lines
$XXX+*n*+1-end:	packed picture data

11. The GS toolbox is called to unpack each block of a packed file. **Table 2** diagrams the stack usage of the call to UnPackBytes.

The tool number is $2703. The appropriate values are pushed onto the stack and the toolbox is called. The number of bytes actually unpacked is pulled from the stack and added to the file mark. Sometimes, the number of bytes actually unpacked is not the same as the number of bytes read in. This is because the unpacker cannot unpack bytes that haven't been read in yet, and sometimes a set of packed bytes crosses a data block boundary. So it unpacks as far as possible, and returns the number of bytes it was able to unpack. It then adds this number to the file mark and the subroutine returns.

Table 2: Stack Usage for UnPackByte Call

Stack before call:

2 bytes	:	room for result
4 bytes	:	pointer to buffer containing the packed data
2 bytes	:	size of the packed data buffer
4 bytes	:	pointer to pointer to area where data will be unpacked
4 bytes	:	pointer to word containing size of the area for unpacking

Stack after call:

2 bytes	:	number of bytes unpacked

CUSTOMIZING PLOAD

One feature of PLOAD you may wish to change is the size and location of the buffer used to load a picture. As listed, a $1000 byte block of data is read in at $5000. Changing either size or location is as easy as changing one or two constants in the assembly listing (making the change without an assembler is a little more complicated).

To change the buffer location, change SOURCE to the desired address. Make sure you keep it a 32-bit constant, with the high word 0. Change the buffer size by modifying BLKSIZE to the new size. BLKSIZE must be an even divisor of $8000 for PLOAD to work correctly. To check this, enter the Monitor and type

```
8000_nnnn
```

where *nnnn* is the BLKSIZE you want. If the display shows

```
R ->$00000000
```

the value of BLKSIZE is OK. Be sure that the combination of SOURCE and BLKSIZE does not cause conflicts with ProDOS or other programs.

If you don't have Merlin 16, use the information in **Table 3** to determine how to change the hex dump for the new BLKSIZE and/or SOURCE values.

Table 3: BLKSIZE and SOURCE Values

Constant	Address
BLKSIZE	$415F
	$41ED
	$424F
BLKSIZE−1	$41DF
SOURCE	$4152
	$423B
SOURCE+9	$42DC
	$42E0
SOURCE+11	$42C8
SOURCE+13	$4302
SOURCE+15	$4311
	$431E

For example, if you wanted to make SOURCE $6000 and BLKSIZE $2000, you would enter (from the Monitor):

```
BLOAD PLOAD.INSTALL,A$4000
415A:00 20
415F:00 20
41ED:00 20
424F:00 20
41DF:FF 1F
4152:00 60
423B:00 60
42DC:09 60
42E0:09 60
42C8:0B 60
4302:0D 60
4311:0F 60
431E:0F 60
BSAVE PLOAD.INSTALL,A$4000,L$3AD
```

If you use the slide show, there are two things to watch out for. First, make sure SOURCE is not so low as to overwrite the BASIC program. Second, if you have changed SOURCE and/or BLKSIZE, make sure that the three pages from $4000-$4300 are not overwritten. If this happens, change the value $4000 in **line 34** of the slide show driver (**Listing 4**) to a new value.

REFERENCES

Apple IIGS Technical Note, no. 27: Graphics Image File Formats, Apple Computer, Inc., Cupertino, CA, October 1, 1987

Programming the 65816 by David Eyes & Ron Lichty, Brady/Prentice Hall Press, New York, 1986

LISTING 1: PLOAD.INSTALL Source Code

```
*
* PLOAD.INSTALL Source Code
* BY STEVE ELLIS
* COPYRIGHT (C) 1989
* MICROSPARC, INC.
* CONCORD, MA 01742
*
* MERLIN 16 ASSEMBLER
         XC                  ;turn on 65816 opcodes
         XC
         ORG    $4000        ;run at $4000

* ProDOS equates
HIMEM    EQU    $73          ;himem pointer
EXTCMD   EQU    $BE06        ;vector to external commands
ERROUT   EQU    $BE09        ;ProDOS error handler
XTRNADR  EQU    $BE50        ;external command address for BI
XLEN     EQU    $BE52        ;length of command string minus 1
XCNUM    EQU    $BE53        ;BASIC command number (0 if external)
PBITS    EQU    $BE54        ;BI parms to be parsed
MLI      EQU    $BE70        ;MLI interface
FIFILID  EQU    $BEB8        ;file ID type
FIAUXID  EQU    $BEB9        ;auxiliary file type
SREFNUM  EQU    $BEC7        ;GET_FILE_INFO reference number
MARK     EQU    $BEC8        ;in-file position mark
OSYSBUF  EQU    $BECE        ;buffer for OPEN
OREFNUM  EQU    $BED0        ;OPEN file reference number
RWRFNUM  EQU    $BED6        ;READ/WRITE file reference number
RWDATA   EQU    $BED7        ;pointer to data to be used
RWCOUNT  EQU    $BED9        ;number of bytes to read/write
RWTRANS  EQU    $BEDB        ;returned # of bytes read
CREFNUM  EQU    $BEDE        ;CLOSE file reference number
GETBUFR  EQU    $BEF5        ;ProDOS buffer allocation routine
BITMAP   EQU    $BF58        ;ProDOS system bit map
IVERSION EQU    $BFFD        ;BI version number
COUT     EQU    $FDED        ;character out routine

*  Storage for program variables
PTR      EQU    $00
UPKAR    EQU    $02
UPKSZ    EQU    $06

*  Constants
BLKSIZE  EQU    $1000        ;size of each data block
                             ;BLKSIZE MUST be an even divisor of $8000!
SOURCE   EQU    $00005000    ;source address of data block
DEST     EQU    $00E12000    ;super hi-res page

* General purpose macros:                                  *
*  Put the 65816 in emulation mode, 8 bit acc. and registers
EMULATE  MAC
         SEC
         XCE
         <<<
*  Switch to native mode, 16 bit acc. and registers
NATIVE   MAC
         CLC
         XCE
         REP    #$30
         <<<
*  Macro to simulate a branch to subroutine instruction
BSR      MAC
         PER    *+5
         BRL    ]1
         <<<
*  Following are macros to perform MLI calls and           *
*  file error handling:                                    *
FILERR   MAC
         BCC    *+5
         BRL    MLIERR
         <<<
GET_FILE_INFO MAC
```

```
          LDA   #$C4
          JSR   MLI
          FILERR
          <<<
OPEN      MAC
          LDA   #$C8
          JSR   MLI
          FILERR
          <<<
READ      MAC
          LDA   #$CA
          JSR   MLI
          FILERR
          <<<
SET_MARK  MAC
          LDA   #$CE
          JSR   MLI
          FILERR
          <<<

          EMULATE
          LDA   EXTCMD+2   ;get page of other commands
          CMP   #$BE       ;there are none
          BEQ   GETROOM    ; so don't bother looking
          STA   SRCHNG+2   ;save the address in zero page
SEARCHØ   LDY   #Ø         ;start at byte Ø
SEARCH    INY              ;bump that to byte 1
          BEQ   ALREADY    ; means we have a match
SRCHNG    LDA   $ØØØØ,Y    ;get a byte
          CMP   COMMAND+256,Y ;compare to our code
          BEQ   SEARCH     ;if equal, look some more
          INC   SRCHNG+2   ;otherwise look at next higher page
          LDA   SRCHNG+2
          CMP   #$9A       ;up to start of DOS yet?
          BCC   SEARCHØ    ; no, search some more
          BRA   GETROOM    ; yes, skip installed message

*  Print an error message stating that PLOAD has already
*  been installed and return to BASIC.
ALREADY   LDY   #Ø
:1        LDA   AINSTL,Y   ;get a character
          BEQ   :2         ;stop on Ø
          JSR   COUT       ;print the char.
          INY              ;finish message
          BNE   :1         ;always
:2        RTS              ;back to BASIC

*  Ask ProDOS for room for our command.          *
GETROOM   LDA   HIMEM+1    ;get top of free memory
          CLC
          ADC   #4         ; add to that the ProDOS general buffer
          STA   OHIMEM     ; save the result
          LDA   #>CMDEND-COMMAND ;get number of pages for our command
          INC              ;add one for total pages needed
          JSR   GETBUFR
          BCC   GOTBUF     ;got them
          JMP   ERROUT     ;otherwise exit with an error

*  Now that we've got the space, we relocate a few   *
*  addresses, and move our code up to its new home.  *
GOTBUF    STA   REL1+2
          STA   REL2+2
*  Update the system bitmap
MRKPAGE   TAX              ;get page number into acc.
          PHA              ;save it
          LSR              ;shift it right a few times
          LSR
          LSR
          TAY              ; to address byte in bitmap
          TXA
          AND   #7         ;isolate bit position
          TAX
          LDA   #Ø
          SEC              ;mark the page with a 1 bit
:1        ROR
          DEX
          BPL   :1
          ORA   BITMAP,Y   ;mask with previous value
          STA   BITMAP,Y   ; and store it
          PLA              ;get page number
          INC              ;bump it
          CMP   OHIMEM     ;done all the pages?
          BCC   MRKPAGE    ; no, finish it up

*  Check for BASIC version 1.1
CATMOD    LDA   IVERSION   ;get BI version number
          CMP   #1         ;must be version 1.1 for catalog mods
          BNE   DCHAIN     ;don't change anything
          LDA   #$CØ       ;replace IVR and INT file desciptors
          STA   $B98E      ; with PNT and PIC file descriptors
          INC
          STA   $B98D
          LDA   #"P
          STA   $B9AF
          STA   $B9B2
          LDA   #"I
          STA   $B9B3
          LDA   #"C
          STA   $B9B4
          LDA   #"N
          STA   $B9BØ
          LDA   #"T
          STA   $B9B1

*  Daisy-chain our command
DCHAIN    NATIVE
          LDA   EXTCMD+1   ;get previous address of ext. commands
          STA   CMDLINK+4  ; and save so we can jump to it
REL1      LDA   #COMMAND   ;put address of our command
          STA   EXTCMD+1   ; into external jump
          LDA   #CMDEND-COMMAND.$ØØFF ;last byte of program
          LDX   #COMMAND   ;get source address
REL2      LDY   #$ØØØØ     ; and destination address
          MVN   $ØØ,$ØØ    ; and move the program up
          EMULATE
          RTS
OHIMEM    DS    1          ;room for old himem value
          DS    \          ;skip to next page

*  Scan the input line for our command
COMMAND   CLD              ;valid command handler identifier
          PER   PLOAD      ;push run-time address of string PLOAD
          LDY   #Ø         ;scan for command
          TYX              ; on input line
:1        LDA   $2ØØ,X     ;get a char
          INX
          CMP   #" "       ;skip blanks
          BEQ   :1
          AND   #$DF       ;convert lower case to upper
          CMP   (Ø1,S),Y   ;compare char. to command string
          BNE   CMDLINK
          INY
          CPY   #5         ;got the whole word?
          BCC   :1         ; no, keep looking
          DEY
          STY   XLEN       ;put the len-1 in BI global page
          STZ   XCNUM      ;command code = Ø means external handler
          NATIVE
          PER   DO_CMD     ;push address of command handler
          PLA              ;find out what it is
          STA   XTRNADR    ; and let the BI know where it is
          LDA   #$Ø4Ø1     ;require pathname 1, allow slot & drive
          STA   PBITS      ; for BI parser
          PLA              ;pull address of 'PLOAD' off the stack
          EMULATE
          CLC              ;let BI parse it
          RTS
CMDLINK   PLA              ;clean up the stack
          PLA
          SEC              ;not our command
          JMP   $ØØØØ      ; so jump to any other handlers

DO_CMD    NATIVE
          PER   SAVBUF     ;push run-time address of save area
          LDY   #6
:1        LDA   PTR,Y      ;get a zero page byte
          STA   (Ø1,S),Y   ;save it
          DEY
          DEY
          BPL   :1         ;finish all the 8 bytes
          PLA
          LDA   #SOURCE
          STA   PTR        ;point to source data area
          STA   RWDATA     ;tell MLI where to load data
          LDA   #DEST
          STA   UPKAR      ;pointer to super hi-res screen
          LDA   #BLKSIZE
          STA   RWCOUNT    ;read one data block at a time
          STZ   MARK       ;zero file mark (start at byte Ø)
          LDA   #$ØØE1     ;hi word of super hi-res screen location
          STA   UPKAR+2
          LDA   HIMEM      ;set HIMEM address
          STA   OSYSBUF    ; as buffer for OPEN
          EMULATE
          LDA   #$4Ø       ;initialize super hi-res
          TSB   $CØ29      ; without changing its current status
          OPEN
          LDA   OREFNUM    ;copy our reference number
          STA   RWRFNUM    ; to read/write,
          STA   CREFNUM    ; close, and
          STA   SREFNUM    ; get_info refnums
          GET_FILE_INFO
          LDA   FIFILID    ;check file ID type
          CMP   #$C1       ;full 32K image, no need to unpack
          BEQ   BIGPIC
          CMP   #$Ø6       ;assume BIN files are 32K images
          BEQ   BIGPIC
          CMP   #$CØ       ;packed picture image
          BEQ   PACPIC
BADTYPE   LDA   #$ØD       ;FILE TYPE MISMATCH
          BRA   MLIERR     ;exit with error back to BASIC

CLOSE     EMULATE
          LDA   #$CC       ;CLOSE the file
          JSR   MLI
          PER   SAVBUF
          LDY   #7
:1        LDA   (Ø1,S),Y   ;restore the ZP we trampled
          STA   PTR,Y
          DEY
          BPL   :1
          PLA
          PLA
          RTS              ;return to BASIC

MLIERR    PHA              ;save acc.
          BSR   CLOSE      ;close the file
          PLA              ;get acc.
          JMP   ERROUT     ;abort

*  Load 32K images.
BIGPIC    EMULATE
          READ
          NATIVE
          PHB              ;save data bank
          LDA   #BLKSIZE-1 ;move one data block
          LDX   PTR        ; from source address
          LDY   UPKAR      ; to super hi-res page
          MVN   SOURCE,DEST
          PLB              ;restore data bank
          LDA   UPKAR      ;find location on super hi-res page
          CLC
          ADC   #BLKSIZE   ;increment screen pointer by size of block
          STA   UPKAR
          CMP   #$AØØØ     ;done with the picture (up to $AØØØ)?
          BNE   BIGPIC     ; no, do some more
          BRA   CLOSE      ;close the file and exit

PACPIC    NATIVE
          LDA   FIAUXID    ;get file AUX type
          BEQ   TYPEØØ
          CMP   #ØØØ1
```

LISTING 1: PLOAD.INSTALL Source Code *continued*

```
         BEQ   TYPE01
         CMP   #0002
         BEQ   T02JMP
         EMULATE
         BRL   BADTYPE     ;not a recognized packed file type
T02JMP   BRL   TYPE02      ;can't reach it with a normal branch

* Load and unpack aux type $00 files.
TYPE00   NATIVE
         LDA   #$7D00      ;only interested in data for unpacking
         STA   UPKSZ
         LDA   #$0020      ;read the palette
         STA   RWCOUNT
         EMULATE
         READ
         LDX   #0
         TXA
SCBLP0   STAL  $E19D00,X   ;zero out the scan line area, since all
         INX               ;Paintworks pictures are 320 mode, palette 0
         BNE   SCBLP0
         LDX   #$1F
PALTLP0  LDA   SOURCE,X
         STAL  $E19E00,X   ;move palette to palette area
         DEX
         BPL   PALTLP0
         NATIVE
         LDA   #$222       ;position past palette in file
         STA   MARK
         LDA   #BLKSIZE
         STA   RWCOUNT
T00LOOP  EMULATE
         SET_MARK
         READ
         NATIVE
         BSR   UNPACK
         LDA   UPKAR
         CMP   #$9D00
         BLT   T00LOOP
         BRL   CLOSE

* Load and unpack aux type $01 files.
TYPE01   NATIVE
         LDA   #$8000      ;SHR pic is $8000 bytes long
         STA   UPKSZ       ;tell toolbox
T01LOOP  EMULATE
         SET_MARK
         READ
         NATIVE
         BSR   UNPACK
         LDA   UPKSZ       ;unpacked the entire picture?
         BNE   T01LOOP     ; no, do some more
         BRL   CLOSE       ; else exit through CLOSE

* Load and unpack aux type $02 files.
TYPE02   NATIVE
         LDA   #$7D00      ;unpack only screen data (not SCB's, etc.)
         STA   UPKSZ
         EMULATE
         READ
         NATIVE
         LDA   SOURCE+11   ;get number of horizontal pixels
         CMP   #320
         BEQ   PIXOK
         CMP   #640
         BEQ   PIXOK

* If the picture doesn't have either 320 or 640 pixels, exit
* to BASIC with a RANGE ERROR.
         EMULATE
         LDA   #2
         BRL   MLIERR

* Continue unpacking after determining a standard screen width.
* First, copy the screen control byte for each scan line.
PIXOK
         MX    00
         LDA   SOURCE+9    ;get hi-byte of SCB byte
         XBA               ;move it to high-byte of acc
         ORA   SOURCE+9    ; and get it in low-byte of acc
         AND   #$F0F0      ;only interested in high nibbles
         LDX   #0
SCBLP2   STAL  $E19D00,X   ;put it in SCB storage area
         INX
         INX
         CPX   #$C8        ;only up to $E19DC7
         BNE   SCBLP2
         LDA   #0          ;zero out from $E19DC8 -> $E19DFF
:1       STAL  $E19D00,X
         INX
         INX
         CPX   #$100       ;done the whole page?
         BNE   :1          ; no, finish it up

* Count the number of palettes and move them to where
* they belong (from $E19E00 up).
         LDX   SOURCE+13   ;index with number of palettes
         LDA   #$00        ;use acc. to hold address
PALTLP2  CLC
         ADC   #$20        ;point to next palette
         DEX
         BNE   PALTLP2     ;more palettes
         TAX               ;copy address of the end of the palettes
         PHA               ; and save it
:2       LDA   SOURCE+15,X
         STAL  $E19E00,X   ;move the data into palette area
         DEX
         DEX
         BPL   :2          ;more palette data

* We have the start of the ScanLineDirectory now.  Skip
* over each entry (4 bytes) to find the beginning of the
* packed picture data.
         PLY
         TYA               ;get start addr. of entries in A
         LDX   SOURCE+15,Y ;get number of scan lines as index
PICLP2   CLC
         ADC   #4          ;skip an entry
         DEX
         BNE   PICLP2      ;more to do
         ADC   #17         ;adjust pointer to correct address
         STA   MARK        ;load from that point in file
         LDA   #BLKSIZE
         STA   RWCOUNT
T02LOOP  EMULATE
         SET_MARK
         READ
         NATIVE
         BSR   UNPACK
         LDA   UPKAR
         CMP   #$9D00
         BLT   T02LOOP
         BRL   CLOSE

* Call the toolbox to unpack the picture.
UNPACK   NATIVE
         LDX   RWTRANS     ;number of bytes actually read
         LDA   #0
         PHA               ;space for result
         PHA               ;pointer to buffer holding packed data
         LDY   PTR         ;low word of buffer
         PHY
         PHX               ;number of bytes read
         PHA               ;pointer to pointer to
         PEA   #UPKAR      ; area to unpack into
         PHA               ;pointer to word holding length
         PEA   #UPKSZ      ; of size of area to unpack into
         LDX   #$2703      ;tool number for UnPackBytes
         JSL   $E10000     ;call the toolbox
         PLA               ;get number of bytes unpacked
         CLC
         ADC   MARK        ;update the file mark by adding the number
         STA   MARK        ; of bytes unpacked to previous mark
         RTS

PLOAD    ASC   "PLOAD"
AINSTL   HEX   8D
         ASC   "PLOAD ALREADY INSTALLED"
         HEX   8D8D00
SAVBUF   DS    8
CMDEND   EQU   *
```

END OF LISTING 1

LISTING 2: PLOAD.INSTALL

```
Start: 4000             Length: 3AD

4A|4000:38 FB AD 08 BE C9 BE F0
6F|4008:2A 8D 13 40 A0 00 C8 F0
3D|4010:14 B9 00 00 D9 00 42 F0
14|4018:F5 EE 13 40 AD 13 40 C9
62|4020:9A 90 E9 80 0E A0 00 B9
34|4028:8A 43 F0 06 20 ED FD C8
18|4030:D0 F5 60 A5 74 18 69 04
6A|4038:8D B5 40 A9 02 1A 20 F5
D2|4040:BE 90 03 4C 09 BE 8D A2
CD|4048:40 8D AE 40 AA 48 4A 4A
F7|4050:4A A8 8A 29 07 AA A9 00
21|4058:38 6A CA 10 FC 19 58 BF
D6|4060:99 58 BF 68 1A CD B5 40
7B|4068:90 E2 AD FD BF C9 01 D0
89|4070:25 A9 C0 8D 8E B9 1A 8D

CD|4078:8D B9 A9 D0 8D AF B9 8D
90|4080:B2 B9 A9 C9 8D B3 B9 A9
88|4088:C3 8D B4 B9 A9 CE 8D B0
75|4090:B9 A9 D4 8D B1 B9 18 FB
C6|4098:C2 30 AD 07 BE 8D 3B 41
C4|40A0:A9 00 41 8D 07 BE A9 FF
63|40A8:02 A2 00 41 A0 00 00 54
9E|40B0:00 00 38 FB 60 00 00 00
9E|40B8:00 00 00 00 00 00 00 00
F3|40C0:00 00 00 00 00 00 00 00
B6|40C8:00 00 00 00 00 00 00 00
69|40D0:00 00 00 00 00 00 00 00
3A|40D8:00 00 00 00 00 00 00 00
3B|40E0:00 00 00 00 00 00 00 00
2A|40E8:00 00 00 00 00 00 00 00
91|40F0:00 00 00 00 00 00 00 00
4E|40F8:00 00 00 00 00 00 00 00
8E|4100:D8 62 81 02 A0 00 BB BD

35|4108:00 02 E8 C9 A0 F0 F8 29
EA|4110:DF D3 01 D0 22 C8 C0 05
8C|4118:90 ED 88 8C 52 BE 9C 53
E5|4120:BE 18 FB C2 30 62 15 00
16|4128:68 8D 50 BE A9 01 04 8D
B2|4130:54 BE 68 38 FB 18 60 68
97|4138:68 38 4C 00 00 18 FB C2
EA|4140:30 62 61 02 A0 06 00 B9
27|4148:00 00 93 01 88 88 10 F7
8F|4150:68 A9 00 50 85 00 8D D7
A5|4158:BE A9 00 20 85 02 A9 00
8F|4160:10 8D D9 BE 9C C8 BE A9
8F|4168:E1 00 85 04 A5 73 8D CE
D2|4170:BE 38 FB A9 40 0C 29 C0
C7|4178:A9 C8 20 70 BE 90 03 82
A5|4180:40 00 AD D0 BE 8D D6 BE
12|4188:8D DE BE 8D C7 BE A9 C4
7D|4190:20 70 BE 90 03 82 2A 00
```

```
02 |4198:AD B8 BE C9 C1 F0 2E C9
09 |41A0:06 F0 2A C9 C0 F0 51 A9
F2 |41A8:0D 80 17 38 FB A9 CC 20
92 |41B0:70 BE 62 F0 01 A0 07 B3
0C |41B8:01 99 00 00 88 10 F8 68
0D |41C0:68 60 48 62 02 00 82 E2
B1 |41C8:FF 68 4C 09 BE 38 FB A9
63 |41D0:CA 20 70 BE 90 03 82 E9
62 |41D8:FF 18 FB C2 30 8B A9 FF
9A |41E0:0F A6 00 A4 02 54 E1 00
24 |41E8:AB A5 02 18 69 00 10 85
BC |41F0:02 C9 00 A0 D0 D7 80 B3
22 |41F8:18 FB C2 30 AD B9 BE F0
16 |4200:12 C9 01 00 F0 78 C9 02
BB |4208:00 F0 05 38 FB 82 97 FF
B8 |4210:82 9B 00 18 FB C2 30 A9
D5 |4218:00 7D 85 06 A9 20 00 8D
AE |4220:D9 BE 38 FB A9 CA 20 70
16 |4228:BE 90 03 82 94 FF A2 00
6A |4230:8A 9F 00 9D E1 E8 D0 F9
EF |4238:A2 1F BD 00 50 9F 00 9E
BF |4240:E1 CA 10 F6 18 FB C2 30
32 |4248:A9 22 02 8D C8 BE A9 00
B0 |4250:10 8D D9 BE 38 FB A9 CE
AA |4258:20 70 BE 90 03 82 62 FF
EA |4260:A9 CA 20 70 BE 90 03 82
54 |4268:58 FF 18 FB C2 30 62 02
89 |4270:00 82 E9 00 A5 02 C9 00
DC |4278:9D 90 D9 82 2D FF 18 FB
E4 |4280:C2 30 A9 00 80 85 06 38
95 |4288:FB A9 CE 20 70 BE 90 03
F3 |4290:82 2F FF A9 CA 20 70 BE
3C |4298:90 03 82 25 FF 18 FB C2
CD |42A0:30 62 02 00 82 B6 00 A5
63 |42A8:06 D0 DC 82 FD FE 18 FB
42 |42B0:C2 30 A9 00 7D 85 06 38
81 |42B8:FB A9 CA 20 70 BE 90 03
3F |42C0:82 FF FE 18 FB C2 30 AD
2F |42C8:0B 50 C9 40 01 F0 0C C9
92 |42D0:80 02 F0 07 38 FB A9 02
5B |42D8:82 E7 FE AD 09 50 EB 0D
54 |42E0:09 50 29 F0 F0 A2 00 00
7A |42E8:9F 00 9D E1 E8 E8 E0 C8
CA |42F0:00 D0 F5 A9 00 00 9F 00
EF |42F8:9D E1 E8 E8 E0 00 01 D0
B7 |4300:F5 AE 0D 50 A9 00 00 18
0A |4308:69 20 00 CA D0 F9 AA 48
3B |4310:BD 0F 50 9F 00 9E E1 CA
DE |4318:CA 10 F5 7A 98 BE 0F 50
3C |4320:18 69 04 00 CA D0 F9 69
61 |4328:11 00 8D C8 BE A9 00 10
BE |4330:8D D9 BE 38 FB A9 CE 20
BB |4338:70 BE 90 03 82 83 FE A9
EB |4340:CA 20 70 BE 90 03 82 79
8A |4348:FE 18 FB C2 30 62 02 00
FD |4350:82 0A 00 A5 02 C9 00 9D
67 |4358:90 D9 82 4E FE 18 FB C2
51 |4360:30 AE DB BE A9 00 00 48
FC |4368:48 A4 00 5A DA 48 F4 02
B8 |4370:00 48 F4 06 00 A2 03 27
40 |4378:22 00 00 E1 68 18 6D C8
88 |4380:BE 8D C8 BE 60 D0 CC CF
C1 |4388:C1 C4 8D D0 CC CF C1 C4
1D |4390:A0 C1 CC D2 C5 C1 C4 D9
8A |4398:A0 C9 CE D3 D4 C1 CC CC
E5 |43A0:C5 C4 8D 8D 00 00 00 00
05 |43A8:00 00 00 00 00

TOTAL: 89D2
```

END OF LISTING 2

LISTING 3: SLIDE.SHOW

```
37  | 10  REM *
C0  | 20  REM * SLIDE.SHOW
B9  | 30  REM * BY STEVE ELLIS
AE  | 40  REM * COPYRIGHT (C) 1989
CB  | 50  REM * MICROSPARC, INC.
24  | 60  REM * CONCORD, MA 01742
45  | 70  REM *
3F  | 80 D$ =  CHR$ (4): DIM F$(45):F = 0:P1$ = ""
49  | 90  ONERR  GOTO 600
E6  | 100  PRINT D$"-PLOAD.INSTALL"
CD  | 110  ONERR  GOTO 610
69  | 120  PRINT D$"BLOAD SLIDE.OBJ,A$300"
0D  | 130  PRINT D$"PR#3"
73  | 140  ONERR  GOTO 390
C2  | 150  HOME : PRINT : GOSUB 620: VTAB 1: PRINT :
    |     PRINT "Super Hi-Res Slide Show": PRINT "By
    |      Steve Ellis": PRINT "Copyright (C) 1989":
    |     PRINT "MicroSPARC, Inc.": PRINT : PRINT
7A  | 160  PRINT "1) Enter Slot/Drive"
64  | 170  PRINT : PRINT "2) Enter Prefix": PRINT :
    |     PRINT "3) Turn picture names off": PRINT :
    |     PRINT "4) Turn picture names on": PRINT :
    |     PRINT "5) See slide show": PRINT : PRINT "
    |     6) Quit": PRINT : PRINT "Your Choice: ";
63  | 180  POKE  - 16368,0: GET AN$: IF AN$ < "1" OR
    |     AN$ > "6" THEN 180
09  | 190  PRINT AN$;:
57  | 200  IF AN$ = "1" THEN  GOSUB 510: GOTO 140
B6  | 210  IF AN$ < > "2" THEN 270
72  | 220 PP$ = P1$: HOME : INPUT "Enter Prefix: /";P
    |     1$: IF P1$ < > "" THEN P1$ = "/" + P1$: IF
    |       LEFT$ (P1$,2) = "//" THEN P1$ =  RIGHT$ (
    |     P1$, LEN (P1$) - 1): GOTO 240
7F  | 240 ERR = 0:FL = 1: PRINT D$"prefix ";P1$: PRIN
    |     T D$"catalog": PRINT : PRINT "Press Return
    |     to continue ";: GET AN$: PRINT AN$;:
4A  | 250 FL = 0: IF ERR THEN P1$ = PP$
42  | 260  GOTO 150
1D  | 270  IF AN$ = "3" THEN F = 1: GOTO 150
EA  | 280  IF AN$ = "4" THEN F = 0: GOTO 150
46  | 290  IF AN$ = "6" THEN  CALL 852: HOME : VTAB 2
    |     3: END : REM shut down QuickDraw and end
F6  | 300  IF P1$ = "" THEN  PRINT D$;"PREFIX": INPUT
    |      P1$: GOTO 320
1F  | 310  IF  RIGHT$ (P1$,1) < > "/" THEN P1$ = P1$
    |     + "/"
48  | 320 X = 1
AA  | 330  PRINT D$"OPEN"P1$",TDIR": PRINT D$"READ"P1
    |     $
E0  | 340  INPUT A$: INPUT A$: INPUT A$
4E  | 350  INPUT A$:T$ =  MID$ (A$,18,3): IF T$ < >
    |     "PIC" AND T$ < > "PNT" AND T$ < > "$C1"
    |     AND T$ < > "$C0" THEN 350
16  | 360  FOR Y = 15 TO 2 STEP  - 1: IF  MID$ (A$,Y,
    |     1) = " " THEN  NEXT
20  | 370 F$(X) =  MID$ (A$,2,Y)
45  | 380 X = X + 1:Y = 2: NEXT Y: GOTO 350
3A  | 390  PRINT D$"CLOSE": IF FL THEN ERR = 1: VTAB
    |     16: PRINT "Error using prefix ";P1$: PRINT
    |     "Press Return to continue ";: POKE  - 16368
    |     ,0: GET AN$: VTAB 16: PRINT  SPC( 79): PRIN
    |     T  SPC( 79);: GOTO 250
7B  | 400  IF X = 1 THEN  HOME : PRINT "Disk error or
    |      no pictures on that disk...press Return to
    |      continue ";: POKE  - 16368,0: GET AN$:
    |     PRINT AN$;: GOTO 140
39  | 410  ONERR  GOTO 490
56  | 420  CALL 768: REM Startup Quickdraw
02  | 430 Y = 1
0E  | 440  PRINT D$"PLOAD"P1$ + F$(Y)
A8  | 450  FOR Z = 1 TO  LEN (F$(Y)): POKE 735 + Z,
    |     ASC ( MID$ (F$(Y),Z,1)): NEXT : POKE 735 +
    |      Z,0: CALL 794: REM poke title into memory
    |     and calc. its pixel width
E3  | 460 H = (320 -  PEEK (866)) / 2: POKE 824,H: RE
    |     M center the title
E8  | 470  IF F = 0 THEN  CALL 819: REM draw the titl
    |     e
62  | 480  GET A$: IF  ASC (A$) = 27 THEN 500
9B  | 490 Y = Y + 1: IF Y < X THEN 440
51  | 500  CALL 852:F = 0: GOTO 140
23  | 510  ONERR  GOTO 580
90  | 520  HOME : VTAB 1: PRINT "Slot:";: POKE  - 163
    |     68,0: GET S: IF S < 1 OR S > 7 THEN 520
58  | 530  PRINT S
71  | 540  HTAB 1: VTAB 3: PRINT "Drive:";: POKE  - 1
    |     6368,0: GET D: IF D < 1 OR D > 2 THEN 540
7E  | 550  PRINT D: PRINT D$;"PREFIX,S";S;",D";D:
    |     PRINT D$"catalog": PRINT : PRINT "Press Re
    |     turn to continue ";: GET AN$: PRINT AN$;:
33  | 560  PRINT D$"PREFIX": INPUT P1$
18  | 570  POKE 216,0: HOME : RETURN
8D  | 580  CALL  - 3288: IF  PEEK (222) = 16 THEN 570
25  | 590  PRINT : PRINT "Invalid Slot and Drive. Pre
    |     ss RETURN to continue.";: POKE  - 16368,0:
    |     GET AN$: GOTO 520
8E  | 600  HOME : PRINT "CANNOT FIND PLOAD.INSTALL":
    |     END
5B  | 610  HOME : PRINT "CANNOT FIND SLIDE.OBJ": END
FE  | 620  VTAB 22: HTAB 1: PRINT "While slide show i
    |     s running, press Return for next picture":
    |     PRINT "and Escape for this menu";: RETURN

TOTAL: 101D
```

END OF LISTING 3

LISTING 4: SLIDE.SHOW.OBJ Source Code

```
*
* SLIDE.OBJ Source Code
* BY STEVE ELLIS
* COPYRIGHT (C) 1989
* MICROSPARC, INC.
* CONCORD, MA 01742
*

         XC                    ;turn on 65816 opcodes
         XC
         ORG   $0300
TOOL     EQU   $00E10000

*
* Macro to put the 65816 in native mode, 16 bit acc. & regs.
*
NATIVE   MAC
         CLC
         XCE
         REP   #$30
         <<<
*
* Macro to return to emulation mode
*
EMULATE  MAC
         SEC
         XCE
         <<<
*
* Start up Quickdraw II.  Use $4000-$4300 as the 3 pages
* of direct space it needs.
*
STARTUP  NATIVE
         PEA   $4000           ;change this if $4000-$4300 is needed
                               ; for something else
         PEA   $0000           ;master SCB of 0 (320 mode, palette 0)
         PEA   $0000           ;size of largest pixel map, 0 = screen width
         PEA   $1000           ;arbitrary ID number, since we should be
                               ; the only application running that requires
                               ; an ID
         LDX   #$0204          ;QDStartUp
         JSL   TOOL
         EMULATE
         RTS
*
* Calculate the width of the string ending with 0 that has been
* put in memory at $2E0.
*
CALCWID  NATIVE
         PHA                   ;space for result
         PEA   $0000
         PEA   $02E0           ;pointer to C string
         LDX   #$AA04          ;CStringWidth
         JSL   TOOL
         PLA
         STA   WIDTH           ;put the width where Applesoft can get it
         EMULATE
         RTS
*
* Position the pen at the bottom of the screen and draw the
* picture title.
*
DRAWSTR  NATIVE
         PEA   $0000           ;horizontal pos., changed by calling program
         PEA   $00C7           ;vertical pos., bottom of screen
         LDX   #$3A04          ;MoveTo
         JSL   TOOL
         PEA   $0000
         PEA   $02E0           ;pointer to C string
         LDX   #$A604          ;DrawCString
         JSL   TOOL
         EMULATE
         RTS
*
* Shut down Quickdraw and release the direct page space used.
* Be SURE to CALL this routine if you accidentally press RESET,
* or the program won't be able to restart Quickdraw correctly.
*
SHUTDOWN NATIVE
         LDX   #$0304          ;QDShutDown
         JSL   TOOL
         EMULATE
         RTS
WIDTH    DS    1               ;width of picture title
```

END OF LISTING 4

LISTING 5: SLIDE.OBJ

```
Start: 300             Length: 63

AD|0300:18 FB C2 30 F4 00 40 F4
8E|0308:00 00 F4 00 00 F4 00 10
5B|0310:A2 04 02 22 00 00 E1 38
B8|0318:FB 60 18 FB C2 30 48 F4
7B|0320:00 00 F4 E0 02 A2 04 AA
C1|0328:22 00 00 E1 68 8D 62 03
9C|0330:38 FB 60 18 FB C2 30 F4
16|0338:00 00 F4 C7 00 A2 04 3A
36|0340:22 00 00 E1 F4 00 00 F4
6D|0348:E0 02 A2 04 A6 22 00 00
45|0350:E1 38 FB 60 18 FB C2 30
8C|0358:A2 04 03 22 00 00 E1 38
6A|0360:FB 60 00

TOTAL: 7AAD
```

END OF LISTING 5