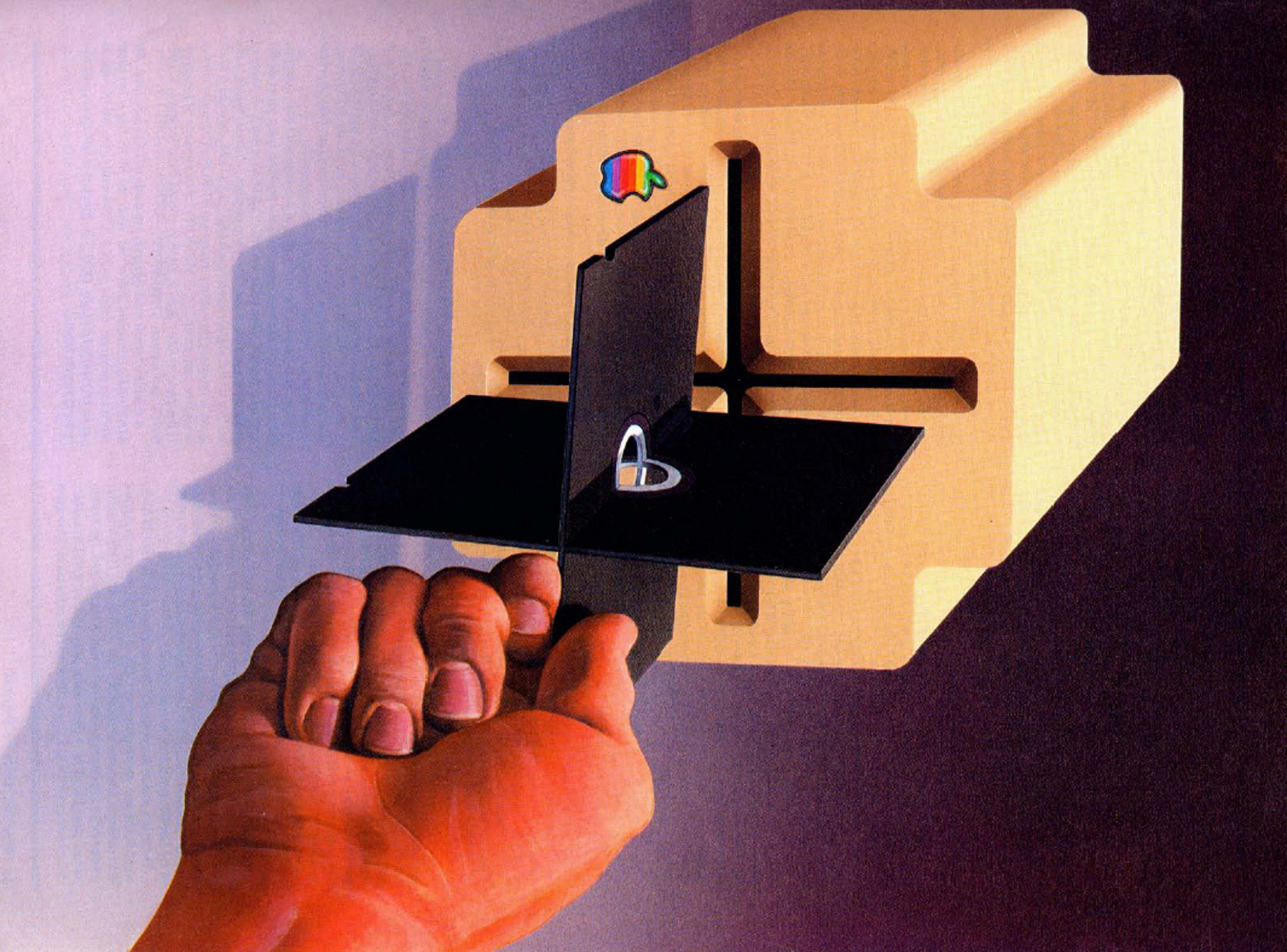

DOUBLEDOS

COVER FEATURE

Keep both DOS 3.3 and ProDOS in memory simultaneously, and switch back and forth between them with a single command. DoubleDOS lets you do it!

Since the release of ProDOS, Apple users have had the choice of two standard operating systems. I have had an Apple II for eight years, and have grown used to DOS 3.3. However, the power of ProDOS cannot be ignored, so I switch between the two. I used CONVERT to transfer the files that I use most often to ProDOS, but CONVERT has a major flaw: it cannot transfer random text files. It also takes a lot of time to set up and use, especially if you just want to transfer one file.

DoubleDOS keeps both DOS 3.3 and ProDOS in memory at once, and lets you switch between them with a single command. You can load a file under DOS 3.3, switch to ProDOS, and save it on a ProDOS disk. You can load a ProDOS file, and save it on a DOS 3.3 disk. If you have two drives, you don't even have to switch disks. Even random access files can be moved easily from one operating system to the other. DoubleDOS will save you time and aggravation.

The only limitation is available memory. When you have both DOS 3.3 and ProDOS loaded, there is 25K of free memory left. This is enough for many programs, but you'll have to use CONVERT for programs larger than 100 sectors (50 blocks).

USING THE PROGRAMS

Before you can install DOS 3.3 and ProDOS together, you must create a new version of DOS 3.3 using the DOSLOAD program. To do this, type **RUN DOSLOAD**. You'll need a copy of your DOS 3.3 system master disk. A regular DOS 3.3 disk won't do; it must be the system master. When you run DOSLOAD, you are prompted to insert the system master. Remove the ProDOS disk from the drive and insert the system master in the same drive. Press Return. DOS 3.3 is loaded and relocated.

You are then prompted to insert the ProDOS disk. Remove the DOS 3.3 system master, and reinsert the ProDOS disk. DOS 3.3 is then saved on your disk under the name DOS38.5K.

Now, you can run the LOADER program (type **RUN LOADER**) and your system will be configured to contain both DOS 3.3 and ProDOS. You can switch to DOS 3.3 from ProDOS by typing **DOS**. You can switch from DOS 3.3 to ProDOS by typing **PRODOS**. It's as simple as that.

The only files you need to install DOS 3.3 are LOADER, PATCH.0, PATCH.1 and DOS38.5K. If you rename LOADER to STARTUP, you can boot the disk and have both ProDOS and DOS 3.3 installed automatically.

Two minor caveats:

1. If you do not have your disk drive in slot 6, DOS 3.3 will hang the first time you try to access a disk, unless you specify the slot number. For example, if your drive is in slot 5, to catalog the disk the first time, you would need to type **CATALOG,S5**. After that, you can omit the slot number, and it will be remembered.
2. If you run a program that tests for the presence of ProDOS by executing a PEEK (48896), the test will succeed even if DOS 3.3 is actually active. This is because DOS 3.3 has been relocated to run lower in memory. This may cause problems with some programs.

The only limitation is available memory.

Now, you can load BASIC programs under one system, enter the other system by typing either **DOS** or **ProDOS**, then save your program, or use any other of that system's commands.

ENTERING THE PROGRAMS

To enter the programs, first boot ProDOS and then type in the Applesoft program in **Listing 1**. Save it with:

SAVE DOSLOAD

Next, type in the Applesoft program in **Listing 2**, and save it with:

SAVE LOADER

Then, enter the Monitor with a CALL −151, and type in the hex code from **Listing 3**. Save it with:

BSAVE PATCH.0,A$7000,L$7C

Finally, enter the hex code from **Listing 4**, and save it with:

BSAVE PATCH.1,A$81D4,L$7B

Of course, if you have an assembler, you can enter the assembly language code from **Listings 3 and 4**, and assemble the programs.

For help with entering *Nibble* listings, see the beginning of the Program Listings Section.

HOW THE PROGRAMS WORK

Setting Up the System

When DOSLOAD is run under ProDOS, it POKEs a small machine language routine into page 3 of RAM. This routine uses the ProDOS machine language interface (MLI) to read blocks from a disk.

First, though, a call to FREBUF in the BASIC.SYSTEM global page at $BEF8 frees any reserved memory. FREBUF is one of two routines in BASIC.SYSTEM that handle reservation requests for RAM. FREBUF releases reserved RAM.

Another routine, called GETBUF, located at $BEF5, is used to reserve $2A00 bytes ($2A pages) of RAM. When it's called, GETBUF attempts to reserve the number of pages of RAM that is in the Accumulator. If it's successful, the Carry bit is cleared; if it's unsuccessful, the Carry bit is set. If GETBUF is successful, the Accumulator will contain the high byte of the starting address of the reserved block of RAM. This is where the DOS image will be stored. The address must be $7000, or the program will end with an error.

The location of the default disk drive is retrieved from ProDOS and installed in the page 3 routine for doing the block reads from the disk. At this time, the last used drive must contain a DOS 3.3 system master disk. Since DOS 3.3 is stored on tracks 0, 1 and part of 2, the program reads blocks 0-23 (8 blocks per track) into RAM at locations $4000-$6FFF. The DOS 3.3 code is $2500 (9472) bytes long. This is 18.5 blocks long. More has been read in than we need, but we can just ignore the extra.

The calls to the MLI are at $300-$310. The entry point is $306. When the MLI is called, the following byte is the command number to MLI, which is $80 for BLOCKREAD. The next two bytes are the address of a parameter list ($300-$305) that the MLI uses.

The parameter list contains several items. First, it holds the number of items in the list — three in our case. The next byte is the address where the last unit number was POKEd by BASIC. The next pair of bytes, $00 and $40, refers to address $4000, where the block will be stored. BASIC increments the address by $200 before the next call. The last pair of bytes holds the block number. The second byte will always be zero in this program. The first byte is set by BASIC for each call to read blocks 0-$23.

Each block is $200 (512) bytes or two pages long. The pages in DOS 3.3 are not sequential, however, so the image must be shuffled around before it can be used.

The routine that does this shuffling is fairly simple. It uses a simple loop to move $100 bytes from the address that DOS was read into, to its correct location, as if it had been read in normally by a master boot. Even though the listing shows the LDA and STA referring to address $0000, the high bytes of these addresses will be POKEd in by BASIC for the calls from there.

There is a reason to use the DOS system master. In the old days, when RAM was very expensive, DOS 3.3 had to operate in Apples with different amounts of memory. To do that, the system master needed two extra pages (sectors) of code. When booted, the system master would load as if there were only 16K of RAM available, the minimum for DOS 3.3 use, then call the extra code. This extra code, called the relocator, would check each memory page starting at 48K size ($BF00), and work down in memory until it found usable space. It would then readdress all of DOS 3.3 to operate at the top of memory, and copy it into the available memory. Lastly, it would call the DOS, to RUN the Hello program.

We can use this relocator code ourselves to readdress the DOS to the memory level that we want. In this case, we tell it that the top of memory is page $99, which is just below BASIC.SYSTEM. Before we can call the relocator, we inhibit the jumping into DOS to continue the boot, then call the code. **Lines 270-310** set up and enter the DOS relocator code. A patch must be made in the DOS code that clears BASIC for the command FP. When the FP command is issued, normally the I/O buffers are rebuilt and HIMEM is reset. Since this would conflict with BASIC.SYSTEM's general purpose buffer, this cannot be allowed.

After the relocator code has finished executing, the I/O buffer building routine is called to set up the default number of buffers (three). This means that no more than three I/O operations may take place at once. For example, you may not have more than three files open at a time, or two open files and a CATALOG. The code that is executed by FP is modified to prevent setting the language card memory to an improper state. Now the BASIC program pointers must be cleared, and the DOS I/O hooks reconnected. Since DOS 3.3 supports more than just Applesoft BASIC, the program installs the I/O vectors for Applesoft in the DOS 3.3 addresses, which handle I/O for BASIC, to avoid any error. Finally, 64 ($40) is stored in the location used to tell DOS what language is in use. This forces DOS 3.3 to use Applesoft in ROM.

Now the changes are completed. DOS 3.3 is in memory at locations $7700-$99FF. The buffers for DOS 3.3 are built and in place at $7000-$76FF. Next, the program installs the code to set up the connection between DOS and ProDOS, and the code to do the switching. The file PATCH.0 is BLOADed at location $7000, the start of the lowest file buffer for DOS 3.3. This area will be initialized by DOS before its use, so it's safe to use. The file PATCH.1 is BLOADed at location $81D4, and that's where it operates. Last, the routine in BASIC.SYSTEM that frees all memory that was reserved earlier is called.

With both systems in place, Apple memory looks like this:

$9A00-$BFFF	BASIC.SYSTEM and both global pages
$7700-$99FF	Readdressed DOS 3.3
$7000-$76FF	Three DOS 3.3 buffers
$6C00-$6FFF	ProDOS general purpose buffer
$0800-$65FF	Free memory

How the Routines Work Together

There are two patches. PATCH.0 is the setup routine, which is not used after it has done its job. PATCH.1 is the code that actually connects the two operating systems together. It resides at $81D5-$8150, above the DOS 3.3 code that builds the I/O buffers. That is another reason for building them before the file DOS 38.5K is BSAVEd. If the buffer building routine is called, the RTS instruction at the beginning of PATCH.1 returns program flow to where it came from, leaving the routine undisturbed.

PATCH.0 handles the tricky part of working with both systems. It first frees all reserved memory that may have been set aside for special purposes, then calls BASIC.SYSTEM to get $2A pages of memory for program use. The page number returned, which is the bottom of the area assigned by BASIC.SYSTEM, *must* be $70. If it isn't, program flow drops through to ERROR. There all memory is set free, and an OUT OF MEMORY error message is displayed. If the page number is $70, DOS is copied from locations $2000-$49FF to locations $7000-$79FF.

A couple of small modifications are made to DOS 3.3. First, the MAXFILES command is disabled. Also, the MON flag is left alone when the program enters DOS 3.3. Normally, the MON state is set to NOMON I,O,C by DOS 3.3. Since it is left alone, you can reset it if you want.

After installing the command word PRODOS, the program sets up the page 3 vectors to access the DOS File Manager at the RWTS routines. The connection with BASIC.SYSTEM is set up to scan for DOS. The memory is marked in ProDOS's bit map as used so that it will not be overwritten. Last, the exit code is installed in page 3.

Changing From ProDOS to DOS 3.3

The programmers at Apple made it very easy to add a command to ProDOS BASIC.SYSTEM. Every time BASIC.SYSTEM gets a command that is not its own, it sends the command to BASIC via an address in its global page. That address is called XCOM, which stands for external command. Normally, the address just points to an RTS instruction.

Switching from ProDOS to DOS is achieved by the command DOS, issued from ProDOS in either mode. When ProDOS gets a command that is not one of its own, it exits through XCOM, which normally returns to BASIC. To add commands to the ProDOS environment, the vectors at XCOM ($BE07,$BE08) can be changed. When the new command is found by the added routine, flags for the external command are set, and the address of the routine to execute the command is set at $BE50,$BE51. To resolve memory conflicts, the command MAXFILES and the code that built the buffers are disabled.

When BASIC.SYSTEM gets a command it does not recognize, it exits via XCOM. That will take it to our program at location XCOM1. The program searches for the string "DOS". BASIC .SYSTEM has already done case conversion, so lower-case input works. Locations $BE6C and $BE6D contain the address of the string stored in BASIC.SYSTEM. The program checks four characters (including the ending RETURN) for a match with the new DOS command. If *all* four match, the command DOS has been found. If not, the Carry bit is set and the program exits with a JMP to the address that XCOM held in the first place. If a match is obtained, the real fun begins.

To continue to execute, BASIC.SYSTEM must be informed that an external command exists. BASIC.SYSTEM can also scan for additional parameters, if requested, but no more parameters are needed. The command number is stored by BASIC.SYSTEM at XCNUM (external command number) in location $BE53. To execute an external command, a zero is stored there. When control goes back to BASIC.SYSTEM, the zero tells it to send control to the address at XTRNADDR (external address) at $BE50,$BE51. That sends control to the code at the DOS command, which calls DOS 3.3.

Changing From DOS 3.3 to ProDOS

The simplest way to add a command to DOS 3.3 is to use an existing command for a different purpose. The command address can be changed by changing its vector in the address table. The command name characters can be changed, too, as long as the high bit for the last letter of the name is left off and the rest of the bits are left on.

I decided to use the DOS 3.3 INT command, since it is not used in ProDOS. Unfortunately, INT is only three letters long, and I needed six letters for the command PRODOS. I moved all of the commands between INIT and INT down by three letters, thereby giving me six letters at the INT spot, and changed INIT to $FC. This made it *almost* impossible to accidentally call INIT, but even if it were called, it would still need a file name.

To INIT a DOS 3.3 disk from within a program, use the command:

```
PRINT CHR$(4);CHR$(252);" HELLO"
```

From immediate mode, you can type:

```
| HELLO
```

Be careful not to accidentally initialize a disk with this command. Later, boot the disk under DOS 3.3, and run MASTER CREATE from your system master disk on it so that it will boot properly.

The PATCH.0 file frees all memory, then saves $2A00 bytes. It checks to see if free memory starts at $7000. If not, there's an error and the program halts. If it does, then an RTS instruction is stored in the MAXFILES address to disable the command. An address is changed so that the MON flag won't need to be reset to zero upon each reentry to DOS 3.3.

Switching

The routines that do the switching are basically similar. First, the current I/O vectors at $36-$39 are copied into the system that will be entered. This way, there won't be a problem if 80-column cards, printers, etc., are in use during the switch. Next, the I/O vectors for the new system are stored in temporary locations in the new system. In entering ProDOS, they must also be stored at $36-$39.

Last, the Trace flag is set for ProDOS, and cleared for DOS 3.3. The reason for changing the Trace flag status is that ProDOS uses the trace function to keep track of what is going on in BASIC. If a trace is in progress, ProDOS will take care of it. When switching to DOS 3.3, BASIC's Trace flag must be cleared because DOS 3.3 will not operate properly with a trace running. Then the old system is allowed to finish its function normally, so all pointers and flags are in their proper states. Here's how it's done: when the old system restores its I/O hooks before exiting to BASIC, it sets them to the system being entered!

Because interrupt handling is different in the two systems, the interrupts are disabled for a switch in either direction. Also, since RESET returns to the operating system through the I/O hooks, RESET will stop any operation and return to the system that was last in use.

SUGGESTIONS AND MODIFICATIONS

When you have finished using DOS 3.3, switch to ProDOS and CALL 768 in immediate or deferred mode. This will disconnect the XCOM link, reset HIMEM, and free all the memory that was used by DOS 3.3. If you disconnect DOS 3.3 while *in DOS*, you will remain in DOS until a PRODOS command is entered. However, DOS 3.3 will be vulnerable to damage from BASIC. For that reason, I suggest exiting only while in ProDOS. HIMEM will be reset to 38400 ($9600).

DoubleDOS transfers files more quickly than CONVERT and the CATALOG command is more convenient. If you have 128K RAM, you can use the /RAM disk for one environment and a floppy drive for DOS 3.3. With two drives, use one for each operating system. Whatever combination you set up, the operating system that you enter will default to the last device that *it* used.

If the exit code at the beginning of page 3 is in the way, it can be moved to another area that is called later. You may also BSAVE it to disk, then BLOAD it when you are ready to leave. Of course, you can always do a BYE to ProDOS, or reboot.

You will find that FILER and FID don't work due to the large amount of RAM they require. Do *not* attempt to execute any ProDOS SYS type routines because the SYS file will throw out any other system — in this case, both BASIC.SYSTEM and DOS 3.3.

Don't attempt to BLOAD a file above 28672 ($7000) under DOS 3.3 because DOS 3.3, ProDOS or both will be overwritten. The only exception might be a BLOAD of a custom patch to DOS 3.3. Such patches are common, but usually will be written for DOS 3.3 at 48K level, and would have to be modified for DOS 3.3 $2600 bytes lower. Be careful about patching DOS 3.3. The length and starting addresses for a binary type DOS 3.3 file will now be found at $8460, $8461 and $8472,$8473 (by PEEKing 33888, 33889, 33906 and 33907).

Listings for DoubleDOS begin on page 105

DoubleDOS

Article on page 24

THIS PROGRAM IS AVAILABLE ON DISK

If you'd rather not type in the listing for this program, you can buy it on disk, complete, free of typos and ready to run. DoubleDOS, Tape Library and Barricade are available on disk for an introductory price of $19.95 plus $1.50 shipping/handling ($2.50 outside the U.S.) from *Nibble*, 52 Domino Dr., Concord, MA 01742. Introductory price expires 6/30/87. See the coupon on the last page of the *Nibble* Software Catalog for ordering information.

Listing 1 for DoubleDOS

DOSLOAD

```
10  REM  ************************
20  REM  * DOSLOAD               *
30  REM  * BY WILLIAM REYNOLDS   *
40  REM  * COPYRIGHT (C) 1987    *
50  REM  * BY MICROSPARC, INC.   *
60  REM  * CONCORD, MA  01742    *
70  REM  ************************
80  TEXT : HOME
90  PRINT "DOSLOAD": PRINT "BY WILLIAM REYNOL
    DS III"
100  PRINT "COPYRIGHT 1987 BY MICROSPARC, INC
    ."
110  VTAB 10: PRINT  CHR$ (7);"INSERT DOS 3.3
     SYSTEM MASTER DISKETTE   INTO THIS DRIV
    E NOW, PRESS <RETURN>";: GET A$: PRINT
120  FOR X = 768 TO 809: READ Y: POKE X,Y: NEXT

130  DATA 3,0,0,64,0,0,32,0,191,128,0,3,141,1
    6,3,96,0,162,0,138,189,0,0,157,0,0,232,2
    08,247,96,32,248,190,169,42,32,245,190,1
    41,42,3,96
140  DATA 32,75,214,32,81,130,96
150  CALL 798: IF  PEEK (810) <  > 112 THEN  PRINT
    "MEMORY CONFIGURATION ERROR !!!": END
160  POKE 769,( PEEK (48701) - 1) * 128 + ( PEEK
    (48700) * 16)
170  FOR X = 0 TO 23: POKE 771,X * 2 + 64: POKE
    772,X
180  CALL 774: IF  PEEK (784) <  > 0 THEN  PRINT
    : PRINT  CHR$ (7)"READING ERROR": END
190  NEXT : GOTO 210
200  POKE 790,A1: POKE 793,A2: CALL 785: RETURN

210 :A1 = 64:A2 = 54: GOSUB 200
220  FOR B = 0 TO 4:A1 = A1 + 1:A2 = 31 - B: GOSUB
    200: NEXT
230  FOR B = 0 TO 8:A1 = A1 + 1:A2 = 63 - B: GOSUB
    200: NEXT
240 :A1 = A1 + 1:A2 = 32: GOSUB 200:A1 = A1 +
    1:A2 = A2 + 1: GOSUB 200
250  FOR B = 0 TO 13:A1 = A1 + 1:A2 = 47 - B:
     GOSUB 200: NEXT
260 :A1 = A1 + 1:A2 = 48: GOSUB 200:A1 = A1 +
    1:A2 = A2 + 1: GOSUB 200
270 :A1 = 106: FOR B = 0 TO 3:A1 = A1 + 1:A2 =
    53 - B: GOSUB 200: NEXT
280  POKE 6916,153: POKE 7646,234: POKE 7205,
    96
290  CALL 6915
300  POKE 33247,169: POKE 33248,3: POKE 33249
    ,234
310  CALL 33236
320  FOR X = 32634 TO 32640: READ Y: POKE X,Y
    : NEXT : POKE 30696,94
330  FOR X = 1 TO 12: POKE 30549 + X, PEEK (3
    0571 + X): NEXT
340  POKE 33974,64
350  HOME : PRINT  CHR$ (7);"INSERT THE PRODO
    S DISKETTE, PRESS <RET>";: GET A$: PRINT

360  PRINT  CHR$ (4)"BSAVE DOS38.5K,A$7000,L$
    2A00"
370  END
```

END OF LISTING 1

Listing 2 for DoubleDOS

LOADER

```
10  REM  ************************
20  REM  * LOADER               *
30  REM  * BY WILLIAM REYNOLDS   *
40  REM  * COPYRIGHT (C) 1987    *
50  REM  * BY MICROSPARC, INC.   *
60  REM  * CONCORD, MA  01742    *
70  REM  ************************
80  TEXT : HOME
90  PRINT "LOADER": PRINT "BY WILLIAM REYNOLD
    S III"
100  PRINT "COPYRIGHT 1987 BY MICROSPARC, INC
    ."
110  FOR X = 1 TO 2000: NEXT : VTAB 12: PRINT
    "LOADING AND CONFIGURING DOS..."
120  DATA 169,158,141,7,190,169,190,141,8,190
    ,32,248,190,169,42,32,245,190,141,34,3,1
    44,5,169,0,24,144,246,169,76,141,10,3,96
130  FOR X = 768 TO 801: READ Y: POKE X,Y: NEXT

140  CALL 768
150  IF  PEEK (802) = 112 THEN 170
160  HOME : PRINT  CHR$ (7);"MEMORY CONFIGURA
    TION ERROR.": END
170  ONERR  GOTO 250
180  PRINT  CHR$ (4);"BLOAD DOS38.5K": REM  A
    $7000
190  PRINT  CHR$ (4);"BLOAD PATCH.0": PRINT  CHR$
    (4);"BLOAD PATCH.1,A$7080": REM  $7000,A
    ND $7080
200  POKE 216,0
210  POKE 33896,1: POKE 33898,6
220  CALL 28672
230  IF  PEEK (0) = 0 THEN  HOME : PRINT : VTAB
    10: PRINT "DOS 3.3 INSTALLED, 25K BYTES
    FREE.": PRINT : PRINT "TYPE 'DOS' TO MOV
    E TO DOS 3.3,": PRINT "'PRODOS' TO RETUR
    N TO PRODOS'": END
240  GOTO 160
250  HOME : VTAB 8: PRINT "PATCH.0, DOS 38.K,
     OR PATCH.1": PRINT "FILE NOT FOUND": END
```

END OF LISTING 2

Listing 3 for DoubleDOS

PATCH.0

```
                    1   ******************************
                    2   * PATCH 0                    *
                    3   * BY WILLIAM REYNOLDS III    *
                    4   * COPYRIGHT (C) 1987         *
                    5   * BY MICROSPARC, INC.        *
                    6   * CONCORD, MA  01742         *
                    7   ******************************
                    8   * MERLIN PRO ASSEMBLER       *
                    9   ******************************
                   10   *
                   11   FREBUF   EQU   $BEF8
                   12   GETBUF   EQU   $BEF5
                   13   PVEC     EQU   $7C52
                   14   PRODOS   EQU   $81D5
                   15   XCOM     EQU   $8220
                   16   *
                   17            ORG   $7000
                   18   *
7000: 20 F8 BE     19   START    JSR   FREBUF     Free all buffers
7003: A9 2A        20            LDA   #$2A       We want $2A pages
7005: 20 F5 BE     21            JSR   GETBUF     Get the buffer
7008: 90 03        22            BCC   A1         If carry set,
700A: 4C 72 70     23   A2       JMP   ERROR      then go Error
700D: C9 70        24   A1       CMP   #$70       High byte must be $70
700F: D0 F9        25            BNE   A2
7011: A9 60        26            LDA   #$60       RTS instruction
7013: 8D 51 7C     27            STA   $7C51      Inhibit Maxfiles command
7016: A9 52        28            LDA   #$52       Don't reset MON flag when
7018: 8D D8 77     29            STA   $77D8      re-entering DOS
701B: A2 00        30            LDX   #$00
701D: BD 88 82     31   A3       LDA   $8284+4,X  Shift most of the command
7020: 9D 85 82     32            STA   $8284+1,X  letters by 3 to make room
7023: E8           33            INX   for        'PRODOS' command in
7024: E0 6A        34            CPX   #$6A       spot that 'INT' was
7026: D0 F5        35            BNE   A3
7028: A9 FC        36            LDA   #252       'INIT' is now 'I' instead
702A: 8D 84 82     37            STA   $8284
702D: A2 05        38            LDX   #$05       install 'PRODOS' command
702F: BD 76 70     39   A4       LDA   NAME,X
7032: 9D EE 82     40            STA   $82EE,X
7035: CA           41            DEX
7036: 10 F7        42            BPL   A4
7038: A2 7A        43            LDX   #$7A       install code to detect
703A: BD 80 70     44   A5       LDA   $7080,X    'DOS' and 'PRODOS'
703D: 9D D4 81     45            STA   $81D4,X
7040: CA           46            DEX
7041: 10 F7        47            BPL   A5
7043: A2 18        48            LDX   #$18       set page 3 vectors into
7045: BD 57 78     49   A6       LDA   $7857,X    machine language entry
7048: 9D D6 03     50            STA   $03D6,X    points into DOS
```

Listing 3 for DoubleDOS

PATCH.0 *(continued)*

```
704B: CA          51            DEX
704C: 10 F7       52            BPL   A6
704E: A9 D4       53            LDA   #PRODOS-$01 Entry point to DOS to
7050: 8D 4C 77    54            STA   $774C       goto ProDOS
7053: A9 81       55            LDA   #>PRODOS
7055: 8D 4D 77    56            STA   $774D
7058: A9 20       57            LDA   #XCOM       ProDOS external command
705A: 8D 07 BE    58            STA   $BE07       vector goes to code at XCOM
705D: A9 82       59            LDA   #>XCOM
705F: 8D 08 BE    60            STA   $BE08
7062: A2 03       61            LDX   #$03        save current ProDOS vectors
7064: BD 34 BE    62   A7       LDA   $BE34,X     for later re-use
7067: 9D 52 7C    63            STA   PVEC,X
706A: CA          64            DEX
706B: 10 F7       65            BPL   A7
706D: A9 00       66            LDA   #$00        if no error, put $00 at $0000
706F: 85 00       67            STA   $00         BASIC program will test
7071: 60          68            RTS   for         error condition
7072: A9 FF       69   ERROR    LDA   #$FF        if error, put $FF AT $0000.
7074: D0 F9       70            BNE   ERROR-$03
                  71   *
7076: 50 52 4F    72   NAME     ASC   'PRODO'
7079: 44 4F
707B: D3          73            ASC   "S"
```

END OF LISTING 3

Listing 4 for DoubleDOS

PATCH.1

```
1    ********************************
2    * PATCH.1                      *
3    * BY WILLIAM REYNOLDS III      *
4    * COPYRIGHT (C) 1987           *
5    * BY MICROSPARC, INC.          *
6    * CONCORD, MA  01742           *
7    ********************************
8    * MERLIN PRO ASSEMBLER         *
9    ********************************
10   *
11   FREBUF   EQU   $BEF8
12   GETBUF   EQU   $BEF5
13   PVEC     EQU   $7C52
14   *
15            ORG   $81D4
16   *
```

```
81D4: 60          17   CMPTCH   RTS
81D5: 78          18   PRODOS   SEI              ;Disable interrupts
81D6: A9 EA       19            LDA   #$EA       let DOS finish a lot
81D8: 8D B9 79    20            STA   $79B9      of things that it
81DB: 20 83 79    21            JSR   $7983      needs to, but force
81DE: A9 9A       22            LDA   #$9A       return to here
81E0: 8D B9 79    23            STA   $79B9
81E3: A2 03       24            LDX   #$03       save current I/O vectors
81E5: BD 53 84    25   A11      LDA   $8453,X    and set ProDOS intercepts
81E8: 9D 30 BE    26            STA   $BE30,X
81EB: BD 52 7C    27            LDA   PVEC,X
81EE: 9D 34 BE    28            STA   $BE34,X
81F1: 95 36       29            STA   $36,X
81F3: CA          30            DEX
81F4: 10 EF       31            BPL   A11
81F6: A9 A5       32            LDA   #$A5       set TRACE flag
81F8: 85 F2       33            STA   $F2        let DOS exit, but it will jump
81FA: 4C B6 79    34            JMP   $79B6      into ProDOS now
                  35   *
81FD: 78          36   DOS      SEI
81FE: A2 03       37            LDX   #$03       save current I/O vectors
8200: BD 30 BE    38   A12      LDA   $BE30,X
8203: 9D 53 84    39            STA   $8453,X
8206: CA          40            DEX
8207: 10 F7       41            BPL   A12
8209: A9 BD       42            LDA   #$BD       set DOS intercepts
820B: A0 81       43            LDY   #$81
820D: A2 78       44            LDX   #$78
820F: 8D 34 BE    45            STA   $BE34
8212: 8C 36 BE    46            STY   $BE36
8215: 8E 35 BE    47            STX   $BE35
8218: 8E 37 BE    48            STX   $BE37
821B: A9 00       49            LDA   #$00
821D: 85 F2       50            STA   $F2        disable TRACE flag
821F: 60          51            RTS
                  52   *
8220: AD 6C BE    53   XCOM     LDA   $BE6C      check for 'DOS'
8223: 85 00       54            STA   $00
8225: AD 6D BE    55            LDA   $BE6D
8228: 85 01       56            STA   $01
822A: A0 04       57            LDY   #$04
822C: B1 00       58   A13      LDA   ($00),Y
822E: D9 49 82    59            CMP   NAMDOS-$01,Y
8231: D0 15       60            BNE   NOTDOS
8233: 88          61            DEY
8234: D0 F6       62            BNE   A13
8236: 8C 54 BE    63            STY   $BE54      set external command flags
8239: 8C 53 BE    64            STY   $BE53
823C: A9 FD       65            LDA   #DOS       set external routine address
823E: 8D 50 BE    66            STA   $BE50
8241: A9 81       67            LDA   #>DOS
8243: 8D 51 BE    68            STA   $BE51
8246: 18          69            CLC               ;carry = 0 because we had
8247: 60          70            RTS               ;command
8248: 38          71   NOTDOS   SEC               ;carry = 1 because we did
8249: 60          72            RTS               ;not find command
                  73   *
824A: 44 4F 53    74   NAMDOS   ASC   'DOS'
824D: 0D          75            DFB   $0D
824E: 00          76   END      DFB   $00
```

END OF LISTING 4