ProDOS POSITION COMMAND

TIPS 'N TECHNIQUES

Get around *the limits of the ProDOS POSITION command, with these practical machine language routines.*

The ProDOS POSITION command lets you easily move through a sequential file. Using the syntax POSITION,pn,F#, you can specify the number of carriage returns (F#) to pass by before stopping. A GET or INPUT positions the cursor at the next character. However, this powerful command has a bug in it. If you have a long field (over 239 characters between carriage returns), the cursor will stop at the 239th character. Then when you use GET or INPUT, it moves to the next character in the file.

It would not be so bad if an error message appeared when there were more than 239 characters to the next CR. But, instead, the command just executes and leaves you lost somewhere in the middle of a record. Or if F# is greater than one, it moves ahead the wrong number of records, and still leaves you lost. The Apple manual, *BASIC Programming with ProDOS* says nothing about this.

USING THE MOVE ROUTINES

The solution to the problem? Never use POSITION on files that might have more than 239 characters between CR's. If you must move ahead a certain number of records in such a file, make calls directly to the ProDOS machine language interface (MLI) and avoid using POSITION. Apple's *ProDOS Technical Reference Manual* explains how to do this.

Or you can simply type in the MOVE program in **Listing 1** and call it from BASIC.

It would not be so bad if an error message appeared . . .

Listing 2 is a demonstration program that shows how to use the MOVE routines. It creates a text file with 260 characters (10 of each letter in the alphabet), one carriage return, and the words "IT WORKS!" in it. Next, it attempts to use the POSITION command to read the file, and the result is "XYYYYYYYYYYZZZZZZZZZZ." It then uses the MOVE routines and the result is "IT WORKS!" Just type:

RUN MOVE.DEMO

to see the demonstration.

ENTERING THE PROGRAMS

To enter the MOVE routines, either type the assembly source code in **Listing 1** into your assembler and assemble the program, or enter the Monitor with CALL −151, type in the hexadecimal codes, and save it with:

BSAVE MOVE,A$300,L$85

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

SAVE MOVE.DEMO

For help with entering *Nibble* programs, see "A Welcome to New *Nibble* Readers" at the beginning of this issue.

HOW THE DEMONSTRATION WORKS

The MOVE demonstration program (**Listing 2**) begins by setting HIMEM to 37376 (**line 80**). This is necessary because the MOVE routines require a 400-byte data buffer at $9200. This address can be changed; see the section How MOVE Puts MLI Calls Together, below.

Next, the MOVE routines are loaded at $300, and the copyright notice is displayed (**lines 90-110**). **Lines 120-240** create a test file that contains 260 characters, followed by a carriage return, and the words "IT WORKS!"

TABLE 1: Parameter Lists for MLI Calls

Byte No.	Open	Read/Write	Newline	Close
1	$03	$04	$03	$01
2	Pathname pointer (low)	Reference number	Reference number	Reference number
3	Pathname pointer (high)	Data buffer pointer (low)	Enable flag	
4	I/O buffer pointer (low)	Data buffer pointer (high)	Newline character	
5	I/O buffer pointer (high)	Request count (low)		
6	Reference number	Request count (high)		
7		Transferred count (low)		
8		Transferred count (high)		

Lines 250-310 test the POSITION command. If it worked correctly, it would read up to the first carriage return, and then read the next field, which would be the words "IT WORKS!" Instead, it stops reading after 239 characters and returns the 21 characters between there and the first carriage return.

Lines 340-480 test the MOVE routines. Notice that **line 350** contains the full pathname for the file. If your disk is not named /NIBBLE, you must change this line to reflect the correct pathname. (The RENAME command can be used to change the name of a volume.) This time, the first carriage return is correctly located, and the words "IT WORKS!" are displayed on the screen.

Incidentally, the F# variable in the POSITION syntax is the same as the F# variable in the READ command. It doesn't work with READ either. You can test this by deleting **line 280** and changing **line 290** to:

```
PRINT CHR$(4);"READ TEST.FILE,F1"
```

You can also try it with the R# variable with similar results.

HOW THE MOVE ROUTINES WORK

Common MLI Calls

MOVE (**Listing 1**) makes calls to a part of ProDOS called the machine language interface (MLI). The call is always JSR $BF00, regardless of what you are trying to do. The first byte after the call tells ProDOS what to do. The following is a partial list of the codes you can use after JSR $BF00:

```
OPEN C8
READ CA
WRITE CB
NEWLINE C9
CLOSE CC
```

The second and third bytes after the JSR $BF00 tell ProDOS the starting address (low byte first) of a list of parameters for the call. The list is different for each call. The most common parameter lists are outlined in **Table 1**. ProDOS executes the command and returns to the fourth byte after the JSR $BF00 to continue with the program. If an error occurs during the call, the Accumulator contains the error code and the call is not executed. If no error occurs, the Accumulator is zero.

Each parameter list begins with a byte that indicates the number of parameters in the list. ProDOS uses this byte as a way to validate the parameter list. **Table 1** shows that the Open parameter list has three parameters: a pointer to the pathname (two bytes), a pointer to a buffer (two bytes), and a file reference number. The pathname pointer is simply the address in memory where you have stored the ASCII codes for a valid pathname (up to 64 characters long). The buffer pointer is the address of a 400-byte area in memory that ProDOS can use to store file information. The reference number is a number that is used to refer to the file as long as it is open. Many of the other file calls require this reference number as one of their parameters. To use the Open call, you supply the first two parameters, and Open then returns the file reference number.

The Read and Write calls use identical parameter lists, with four parameters: the file reference number (as returned by the Open call), a pointer to a data buffer (two bytes), a request count (two bytes), and a transferred count (two bytes). To use the Read or Write calls, you supply the file reference number you got from the Open call, a pointer to a data buffer (400 bytes), and the number of bytes you want to write to or read from the file (the request count). When the call returns, the actual number of bytes read or written will be returned in the transferred count parameter.

The Newline call has three parameters: the file reference number again, the Newline character (usually a $0D), and an enable flag. You supply all three parameters. The

character you supply as the Newline character determines what character is used as a field delimiter when the file is read. Set the enable flag to zero to disable the Newline mode, or to $7F to enable the Newline mode.

The Close call has one parameter: the file reference number. For more information on MLI calls, see the *ProDOS Technical Reference Manual*.

How MOVE Puts MLI Calls Together

Let's now examine how MOVE puts these MLI calls together. The first part of MOVE (**lines 1150-1180**) is the parameter list for the OPEN call. Notice the location of the I/O buffer above HIMEM and below BASIC .SYSTEM. This is the best place for it, unless you have other machine language code there. Below the I/O buffer is a four-page data buffer, starting at $9200, also above HIMEM. This buffer can be as big as you want — just make sure that it's bigger than the maximum number of bytes between any two CR's in your files. Notice that HIMEM was set at $9200 by **line 80** of the BASIC program (**Listing 2**).

Next is the pathname. There is space for the 64-character maximum.

Locations $346-$349 (**lines 1210-1240**) comprise the NEWLINE parameter list, with the enable mask set and an $0D (CR) as the NEWLINE character. Locations $34A-$351 (**lines 1260-1300**) comprise the Read parameter list, which requests the maximum number of bytes that the data buffer will hold. ProDOS returns the actual number that it read, up to and including the next CR, in locations $350-$351 (low byte first).

After the parameter lists are set up, the actual program is quite simple. The Open routine starting at **line 1330** opens the file, calls NEWLINE and returns to BASIC. The Read routine at **line 1460** loads the data buffer with all the characters up through the next CR. BASIC can then read them with PEEKs. If an error occurs, which will happen if the program is at the end of the file, then the error number is returned in the Accumulator. The error number is then stored in $352, so BASIC can tell if there is an error by PEEKing that location. It will contain a zero if there is no error.

The Close routine at $378 (**line 1530**) closes any file that was opened by these routines. It can be modified to close only specific files.

CONCLUSION

Making calls directly to the MLI avoids many of the difficulties of dealing with ProDOS from BASIC. And MLI calls are fast. If you have large files to sort through, MLI calls can cut your program execution times dramatically.

LISTING 1: MOVE

```
                1000 *--------------------------------
                1010 * MOVE
                1020 * BY STEVEN BIRGE
                1030 * COPYRIGHT (C) 1987
                1040 * BY MICROSPARC, INC.
                1050 * CONCORD, MA  01742
                1060 *--------------------------------
                1070 * S-C MACRO ASSEMBLER 2.0
                1080 *--------------------------------
BF00-           1090 MLI     .EQ $BF00
9600-           1100 BUFF    .EQ $9600     ;I/O BUFFER
9200-           1110 DATA    .EQ $9200     ;DATA BUFFER
                1120 *
                1130         .OR $0300
                1140 *
0300- 03        1150 OPARMS  .HS 03        ;OPEN PARAMETER LIST
0301- 06 03     1160         .DA PATH      ;POINTER TO PATHNAME BUFFER
0303- 00 96     1170         .DA BUFF      ;POINTER TO I/O BUFFER
0305- 00        1180         .HS 00        ;REF. NO. RETURNED HERE
0306- 00 00 00
0309- 00 00 00
030C- 00 00     1190 PATH    .HS 0000000000000000 ;64 BYTE BUFFER FOR PATHNAME
030E- 00 00 00
0311- 00 00 00
0314- 00 00     1200         .HS 0000000000000000
0316- 00 00 00
0319- 00 00 00
031C- 00 00     1210         .HS 0000000000000000
031E- 00 00 00
0321- 00 00 00
0324- 00 00     1220         .HS 0000000000000000
0326- 00 00 00
0329- 00 00 00
032C- 00 00     1230         .HS 0000000000000000
032E- 00 00 00
0331- 00 00 00
0334- 00 00     1240         .HS 0000000000000000
0336- 00 00 00
0339- 00 00 00
033C- 00 00     1250         .HS 0000000000000000
033E- 00 00 00
0341- 00 00 00
0344- 00 00     1260         .HS 0000000000000000
                1270 *
0346- 03        1280 NPARMS  .HS 03        ;NEWLINE PARAMETER LIST
0347- 00        1290         .HS 00        ;REF. NO.
0348- 7F        1300         .HS 7F        ;ENABLE NEWLINE MODE
0349- 0D        1310         .HS 0D        ;$0D IS NEWLINE CHARACTER
                1320 *
034A- 04        1330 RPARMS  .HS 04        ;;READ PARAMETER LIST
034B- 00        1340         .HS 00        ;REF. NO.
034C- 00 92     1350         .DA DATA      ;POINTER TO DATA BUFFER
034E- 00 04     1360         .DA $400      ;REQUEST $400 BYTES FROM DISK
0350- 00 00     1370         .DA $0000     ;NO. OF BYTES READ
0352- 00        1380         .HS 00        ;ERROR CODE
                1390 *
0353- D8        1400 OPEN    CLD
0354- 20 00 BF  1410         JSR MLI
0357- C8        1420         .HS C8        ;OPEN CODE
0358- 00 03     1430         .DA $300      ;POINTER TO PARM LIST
035A- 8D 52 03  1440         STA RPARMS+8  ;STORE ERROR CODE
035D- AD 05 03  1450         LDA OPARMS+5  ;GET REF. NO. FROM OPEN LIST
0360- 8D 47 03  1460         STA NPARMS+1  ;PUT IN NEWLINE LIST
0363- 8D 4B 03  1470         STA RPARMS+1  ;PUT IN READ LIST
0366- 20 00 BF  1480         JSR MLI
0369- C9        1490         .HS C9        ;NEWLINE CODE
036A- 46 03     1500         .DA NPARMS    ;POINTER TO PARM LIST
036C- 60        1510         RTS
                1520 *
036D- D8        1530 READ    CLD
036E- 20 00 BF  1540         JSR MLI
0371- CA        1550         .HS CA       ;READ CODE
0372- 4A 03     1560         .DA RPARMS    ;POINTER TO READ LIST
0374- 8D 52 03  1570         STA RPARMS+8  ;PUT ERROR CODE IN READ LIST
0377- 60        1580         RTS
                1590 *
0378- D8        1600 CLOSE   CLD
0379- 20 00 BF  1610         JSR MLI
037C- CC        1620         .HS CC       ;CLOSE CODE
037D- 83 03     1630         .DA CPARMS    ;POINTER TO CLOSE LIST
037F- 8D 52 03  1640         STA RPARMS+8  ;PUT ERROR CODE IN READ LIST
0382- 60        1650         RTS
                1660 *
0383- 01 00     1670 CPARMS  .HS 01.00     ;CLOSE PARM LIST
END OF LISTING 1
```

LISTING 2: MOVE.DEMO

```
10  REM  ************************
20  REM  * MOVE.DEMO             *
30  REM  * BY STEVEN BIRGE       *
40  REM  * COPYRIGHT (C) 1987    *
50  REM  * BY MICROSPARC, INC.   *
60  REM  * CONCORD, MA  01742    *
70  REM  ************************
80  HIMEM: 37376: REM   $9200 BELOW DATA BUFF
    ER
90  ONERR  GOTO 490
100  PRINT  CHR$ (4);"BLOAD MOVE": REM   PLAC
    E MACHINE LANGUAGE ROUTINE AT $300
110  POKE 216,0: TEXT : HOME : NORMAL : PRINT
    "MOVE DEMO": PRINT "BY STEVEN BIRGE": PRINT
    "COPYRIGHT 1987 BY MICROSPARC, INC.": POKE
    34,5
120  VTAB 10: PRINT "CREATING DEMO FILE, PLEA
    SE WAIT..."
130  REM  CREATE DEMO FILE
140  PRINT  CHR$ (4);"OPEN TEST.FILE"
150  PRINT  CHR$ (4);"CLOSE TEST.FILE"
160  PRINT  CHR$ (4);"DELETE TEST.FILE"
170  PRINT  CHR$ (4);"OPEN TEST.FILE"
180  PRINT  CHR$ (4);"WRITE TEST.FILE"
190  FOR J = 0 TO 25
200 E$ =  CHR$ (J + 65): REM  E$=EACH LETTER
    OF ALPHABET
210  FOR K = 1 TO 10: PRINT E$;
220  NEXT K,J
230  PRINT : PRINT "IT WORKS!"
240  PRINT  CHR$ (4);"CLOSE"
250  REM  TEST POSITION COMMAND
260  HOME : PRINT "NOW TESTING POSITION COMMA
    ND"
270  PRINT  CHR$ (4);"OPEN TEST.FILE"
280  PRINT  CHR$ (4);"POSITION TEST.FILE,F1"
290  PRINT  CHR$ (4);"READ TEST.FILE"
300  INPUT A$
310  PRINT  CHR$ (4);"CLOSE"
320  PRINT : PRINT A$: PRINT : FOR X = 1 TO 2
    000: NEXT
330  REM  TEST MOVE ROUTINES
340  PRINT : PRINT "NOW TESTING MOVE ROUTINES
    "; PRINT
350 N$ = "/NIBBLE/TEST.FILE": REM  N$ MUST CONT
    AIN THE FULL PATHNAME FOR THE FILE
360  POKE 774, LEN (N$): REM  PUT LENGTH OF P
    ATHNAME AT $306
370  FOR N = 1 TO  LEN (N$)
380  POKE 774 + N, ASC ( RIGHT$ (N$, LEN (N$)
     - N + 1)): REM   PUT N$ IN $307 ON
390  NEXT
400  CALL 851: REM    $353 OPEN FILE
410  CALL 877: CALL 877: REM  READ AFTER FIRS
    T CR
420 E = 0:A$ = ""
430 E$ =  CHR$ ( PEEK (37376 + E)): REM  READ
     CHARACTER
440  IF E$ =  CHR$ (13) THEN 460: REM   CR

450 A$ = A$ + E$:E = E + 1: GOTO 430
460  CALL 888: REM  CLOSE FILE
470  PRINT A$
480  TEXT : VTAB 20: END
490  HOME : VTAB 7: PRINT "THIS PROGRAM REQUI
    RES THE BINARY FILE": PRINT "'MOVE' TO B
    E ON THE SAME DISK": END
```

END OF LISTING 2