

NORTH STAR ★ COMPUTERS, INC.

2547 Ninth Street • Berkeley, California 94710 • (415) 549-0858

North Star DOS

Version 2

Release 3

Copyright 1977, North Star Computers, Inc.

RELEASE 3 DOS

Release 3 of Version 2 DOS has several bugs fixed, and some changes also:

A) The GA command has been removed. Whenever the TY command specifies a type 1 file, then a third argument is expected which gives the go-address. E.G., TY BASIC 1 2A00.

B) The CF command now uses the 2.5K buffer immediately following the DOS.

C) Two editing characters are now accepted during the typing of DOS commands: at-sign (@) for deleting the entire current line, and underline (⎵) for deleting the most recently typed character. On some terminals a left-arrow character is equivalent to the underline.

D) The DLOOK subroutine may now be used to look up an "empty" entry in the directory, for the purpose of finding an entry which may be used for a new file. If the name pointed to by HL when DLOOK is called starts with a blank, then the first all-blank entry in the indicated directory will be found, if present. Thus, to create a new file using the DOS subroutines, first look up to desired new name to be sure it doesn't already exist. This lookup will return the first "free" disk address on the specified diskette. Next, do a lookup for a blank entry and copy in the name and disk address and length. Finally, do a DWRIT to write out the updated entry to diskette.

E) The DOS now puts 0 into the ACC for CIN and COUT calls.

F) The LI command will now list the go-address of type 1 files.

G) The personalizing instructions for BASIC on page 10 of the DOS manual should now indicate that Release 3 BASIC is to be created by doing:

*CR BASIC 45

*TY BASIC 1 2A00

INTRODUCTION

The North Star Disk Operating System (DOS) was designed and implemented by staff members of North Star Computers, Inc. for use in conjunction with the North Star MICRO DISK SYSTEM. The DOS permits a user to issue various "commands" from a terminal for maintaining and using files on the disk(s). The DOS also provides "library routines" which may be called from user software. These library routines will primarily be of interest to users who will be developing their own system software, as opposed to those users who will primarily use application programs such as BASIC.

The DOS occupies 2.5K (A00 hex) bytes of RAM, including 256 bytes of RAM for user I/O routines. The origin of the DOS is 2000 hex in the standard version.

The North Star DOS is intended for use only with the North Star MICRO DISK SYSTEM, and no license is granted for any other use. Improved copies of the Version 2 DOS, as they become available, may be obtained for a nominal copying charge.

Before the DOS can be used on a specific computer configuration, the instructions in the PERSONALIZING YOUR VERSION OF THE DOS section must be followed. It is suggested that you read the entire manual and then follow those instructions.

DISK ADDRESSES

As described in the hardware documentation, information is stored on the disk in 256-byte "blocks". Each diskette consists of 35 concentric "tracks" with 10 sector positions per track. A block exists at each sector position. Every block on the disk is identified by a unique "disk address" - an integer from 0 through 349. For example, the block at track 27, sector 3 has disk address 273. Track 0 is the "outermost" track, and track 34 is the "innermost" track.

FILES

The primary DOS function is to permit the creation, deletion and use of files on disk(s). A file is an integral number of blocks of data with sequential disk addresses. For example, a particular file might occupy disk addresses 17 through 95 on a diskette mounted on unit 2.

The first four blocks on each diskette contain a "file directory" which specifies a symbolic name, base address, length and type information for each file on that diskette. The symbolic name may be up to 8 characters long, and may include any characters except blank and comma. The length of a file may be up to 346 blocks. A directory may contain as many as 64 entries. No two files in a directory may have the same name, but it is possible for files of the same name to be in directories of diskettes mounted simultaneously on separate units in a multi-unit system.

space after a file is deleted or shortened. The CO command will not work properly if files on the disk have any overlap with each other. This command requires use of the 2.5K RAM area immediately following the DOS.

TY <file name> <file type> <optional go-address>

This command is used to change the type of the specified file on the indicated unit. If type 1 is specified, then the third argument must be supplied to specify the "go-address".

GO <file name>

This command is used to load the specified file into RAM from the indicated unit and begin execution. The GO command may be used only with type 1 files. The GO command will read the entire file into RAM beginning at the go-address, and then jump to the go-address. Obviously, the first byte of the file must be the entry point of the program.

JP <hex RAM address>

This command will cause the computer to jump to the specified RAM address. It provides a way of executing programs which exist in the address space of the computer. Do not confuse this command with the GO command.

LF <file name> <hex RAM address>

SF <file name> <hex RAM address>

These commands may be used to load or save a disk file to or from RAM. The entire contents of the file will be read or written to or from the specified RAM address.

CF <source file name> <destination file name>

This command may be used to copy one file to another. The two files may be on the same or separate units. The file copy is performed only if the destination file is at least as large as the source file. The file type and the type dependent information are also copied. This command requires use of the 2.5K RAM area immediately following the DOS.

CD <source unit #> <destination unit #>

This command will copy the entire contents of the diskette mounted on the specified source unit to the diskette mounted on the specified destination unit. The 2.5K of RAM area immediately following the DOS are required for this command.

RD <disk address> <hex RAM address> <# of blocks>

WR <disk address> <hex RAM address> <# of blocks>

These commands may be used to read or write a specified unit directly to or from RAM. The WR and RD commands should be used with great care, as typing errors can have catastrophic effects. The disk address may optionally be followed by ",1", ",2" or ",3" to indicate a particular unit. Otherwise, unit 1 is assumed. Note that a method of copying one diskette to another in a single drive system would involve repeated use of

COMMANDS

Instructions are issued to the DOS from the terminal by typing "commands". The command format is a 2-letter mnemonic followed by any required arguments. Arguments are separated from the command mnemonic and from each other by a single blank. A command must be terminated by a carriage return before the DOS takes any action. If a typing error occurs during typing of a command, an at-sign (@) may be typed to permit re-typing of the command. Also, an underline or left-arrow may be typed to erase the previously typed character.

When a file name is required as a command argument, the disk unit number (in a multi-unit system) may be specified by immediately following the file name with ",1", ",2" or ",3". Otherwise, unit 1 is assumed. Some sample file name formats are:

```
ABC TEST1234,3 BASIC,1
```

Commands may be typed whenever the prompt character (*) appears at the left margin of the terminal.

LI <optional unit #>

This command will list the entire contents of the directory on the diskette mounted on the specified unit. If no unit is specified, then unit 1 is assumed. For each file, its symbolic name, starting disk address, length and type will be printed. For type 1 files, the go-address will also be printed. To prematurely terminate a listing, a control-C may be typed.

CR <file name> <length> <optional start address>

This command will create a new file on the unit indicated by the file name. The length argument specifies the number of 256-byte blocks. If no starting address is given, then the file will start after the "last" (innermost) file currently allocated on the diskette. Otherwise, the supplied starting address will be used. This command will only create a file directory entry - no accessing of the file itself will be done.

DE <file name>

This command will delete an existing file directory entry on the indicated unit. No actual accessing of the file blocks will be done. The DE command, in conjunction with the CR command, may be used to change the length of a file on the disk.

CO <optional unit #>

This command may be used to "compact" the file space on the diskette mounted on the indicated unit. Any unused disk space between existing files will be eliminated by moving files toward track 0. The CO command may be used to reclaim file

DISK SYSTEM START-UP

After power-on, or when it is desired to re-start the disk system, the 8080 or Z80 computer must be forced to begin execution at the PROM bootstrap program starting address (E900 hex in the standard version). The PROM bootstrap program will read one 256-byte block from unit 1, disk address 4 into RAM at the DOS starting address (2000 hex in the standard version). After reading in the block, the bootstrap will branch to the DOS starting address. The program in the first block of the DOS will proceed to read in the nine blocks from disk addresses 5 through 13. Then the DOS will print the prompt character (*) and await a command from the terminal.

Once the DOS has been started, it is no longer necessary to leave the diskette in unit 1. The DOS is fully resident in RAM, and makes no disk accesses unless asked to do so. Furthermore, the DOS does not maintain any copies of the diskette file directory in RAM between commands. Thus it is possible, for example, to obtain listings of the file directories of several diskettes by inserting them one at a time and then issuing the LI command. Also, it is possible to copy one diskette to another in a single drive system by repeatedly exchanging diskettes and doing the appropriate sequence of RD and WR commands or LF and SF commands.

the RD and WR commands.

IN <optional unit #>

This command should be used to initialize each new diskette to be used in the system. The IN command writes each block on the specified drive with ASCII blank characters (20 hex). This initializes the directory and also guarantees that no "hard disk error" can result from access to an uninitialized file block. The IN command takes about 15 seconds. Needless to say, one should make sure that the proper diskette is mounted before issuing the IN command. Note that the IN command, in order to drive the disk at high speed, uses the 2.5K RAM area immediately following the DOS. Also note that an initialized diskette does not contain a copy of the DOS.

DT <optional unit #>

The DT command may be used to test the unit or to verify the usability of a diskette. This command will continuously write a changing pattern and then read the diskette on the specified unit. Note that all information previously stored on the diskette will be overwritten, and that a tested diskette must be initialized before use. If a hard disk error occurs, then the test will stop and print out the hard disk error message. The command may be stopped by typing a control-C. Note that the 2.5K block of memory immediately following the DOS will be used for this command.

PERSONALIZING YOUR VERSION OF BASIC

When you have successfully created your personal version of the DOS on the second diskette, you may proceed to creating your personal version of BASIC on the second diskette. First, insert the pre-recorded diskette in unit 1, and read BASIC into RAM at the location where it is intended to be run ('2A00 hex in the standard version). BASIC requires at least 10K of RAM.

```
*LF BASIC 2A00
```

Now remove the pre-recorded diskette and insert the second diskette. Create an entry in the file directory for BASIC, set the type and set the go-address:

```
*CR BASIC 45  
*TY BASIC 1 2A00
```

The region in RAM where BASIC allocates user BASIC programs and data is set up in the BASIC initialization sequence (see Appendix 2). No modification is necessary if you use a standard version and are using 16K of memory beginning at 2000 hex. If you have a non-standard version of BASIC or you wish to change the region where BASIC allocates programs and data, then you must modify the appropriate LBI instructions in the BASIC software. If you decide to make such modifications, stop your computer at this point and make the appropriate modifications to the copy of BASIC now in RAM. Then re-start the DOS by branching to the bootstrap address. Whether or not you made the above modification, now write BASIC out onto the second diskette:

```
*SF BASIC 2A00
```

It should now be possible to start BASIC by typing

```
*GO BASIC
```

The I/O requirements of BASIC are handled by calling the DOS terminal I/O routines.

PERSONALIZING YOUR VERSION OF THE DOS

The following procedure must be followed the first time you operate the DOS after installing it in your computer system. Read this entire section before starting the procedure. After you have followed this procedure, the DOS will communicate directly with your terminal immediately after disk system start-up.

The DOS is designed to be able to interface to any conceivable terminal I/O conventions. There are four routines used by the DOS: character input (CIN), character output (COUT), control-C detect (CONTC), and terminal initialization (TINIT). In the DOS which you receive with your MICRO DISK SYSTEM, each of these routines is merely a jump to self loop. The location of these routines is shown in Appendix 1. Thus, when you first perform a disk system start-up sequence, the DOS will be stuck in a branch to self loop at TINIT.

At this point, remove the pre-recorded diskette and insert the second diskette supplied. Now stop the computer and enter your own terminal I/O subroutines in the last 256 bytes of the DOS (from 2900 hex through 29FF hex in the standard version), carefully following the interfacing rules described in Appendix 1. (There is a sample set of I/O routines in Appendix 3.) Next, patch the four JMP instructions to contain the addresses of your routines.

Now, force your computer to branch to TINIT. (It is important that during the entering of your I/O routines, you do not change the computer stack pointer.) The terminal should print out an asterisk (*) and the DOS should be awaiting a command.

Be sure the second diskette, and not the pre-recorded diskette is properly inserted in unit 1. Now, initialize the second diskette with the IN command.

```
*IN 1
```

Next create a file with the name DOS. This will discourage your later allocating a file on top of the disk space that will hold the DOS.

```
*CR DOS,1 10
```

Now write out the DOS from RAM (2000 hex in the standard version) to disk unit 1.

```
*SF DOS 2000
```

You should now be able to start your personalized version of the DOS by branching to the PROM bootstrap start address (E900 hex in the standard version).

FILE DIRECTORY STRUCTURE

This section gives a detailed description of the format of entries in the file directory on a diskette. The file directory occupies disk addresses 0 through 3, with each of these four blocks holding sixteen 16-byte entries. The symbolic name of the entry is the first 8 bytes of an entry. An empty entry is an entry with 8 blanks (20 hex). Following the symbolic name in an entry, the disk address (2 bytes), the file size (two bytes) and the type (1 byte) follow. The last three bytes of an entry are type dependent. In particular, for a type 1 file (GO file), the two bytes following the type byte contain the go-address, and for a type 2 file (BASIC program) the byte following the type byte specifies how many blocks of the file actually contain valid data.

File directory entry:

bytes 0-7	symbolic name of entry
bytes 8-9	disk address
bytes 10-11	number of blocks in file
byte 12	file type
bytes 13-15	type-dependent information

DISK ERRORS

Every disk operation is tried 10 times by the DOS before reporting failure. After the 10 tries, the disk address is printed followed by the message "HD?", and the DOS will await further commands. For example,

```
1 234HD?  
*
```

informs of a disk error on unit 1, at track 23, sector 4.

A disk error can result from attempting to use an improperly inserted diskette (e.g., forgetting to close the drive latch) or from attempting to read uninitialized or improperly written data.

It is possible to specify to the DOS that after every write operation performed, an attempt be made to verify the written data against the data in RAM. This modification will result in slower operation, and most users should find that it is not needed. To make the modification, load a copy of the DOS into high RAM, change the RWCHK byte from 0 to 1 (see Appendix 1), then write the updated copy of the DOS to diskette.

depth at the time of a branch to HDERR from within the DOS is indeterminate. [Note: Software for dealing with hard disk errors is notoriously difficult. It is suggested that due to the expected low frequency of hard disk errors, for most applications the existing HDERR action will be sufficient. Hard disk errors will result primarily from careless use (e.g. forgetting to initialize a diskette, or from removing a diskette while writing is in progress). Hard disk errors can also result from power failure during writing, or from a hardware system failure.]

LIST

This routine will list the file directory of the specified drive. The listing format will be exactly the same as the listing format obtained with the DOS LI command.

Note: The procedure for creating a new file using the above routines would be the following: First use DLOOK to search for the desired new name - if DLOOK succeeds then a file of that name already exists and should not be created. On failure, HL will have the disk address which should be used as the starting address of the new file. Next, use DLOOK to find an empty directory entry by locking up a blank name. If this call to DLOOK fails, then the directory fails. On success, use the pointer in HL to copy the new file name into the directory entry, and copy in the disk address and length and type information. Finally, call DWRIT to copy the new directory entry back to the disk.

DOS LIBRARY ROUTINES

This section describes how user machine language software may interface to the DOS for the accessing of disk files.

Appendix 1 shows the entry points for each of the routines to be described here. The exact interfacing requirements are described in the appendix. The DOS uses the stack pointer existent at call time, and some of the DOS library routines may require as much as 30 bytes of stack storage. Note that the DOS may be re-entered without using the bootstrap PROM. Now follows a discussion of each library routine.

DLOOK

This routine searches for a specified file name in the directory of the indicated disk unit. If the specified name begins with a blank, then an "empty" file directory entry is looked up. On failure, HL is set to the value of the first free disk address on the indicated unit following the last file on the diskette.

On success, HL contains a pointer into a buffer in DOS RAM that has a copy of the sought entry. The pointer addresses the first byte following the symbolic name i.e., byte 8). Also, on return, the ACC specifies the disk unit which was determined from the name passed as argument.

DWRIT

This routine is used to write back to diskette an updated file directory entry which was previously found using DLOOK. No disk activity may occur between the DLOOK and the DWRIT call.

DCOM

This routine may be used to issue an arbitrary disk read or write command. On a read request, DCOM will try 10 times for a successful read before giving up and branching to HDERR. DCOM will fail return if the supplied arguments are out of bounds. However, great care should be used before calling DCOM with incorrect arguments.

DOS This is an entry point to the DOS command processor. It can be used to return control to a loaded DOS without requiring a PROM bootstrap load.

HDERR

HDERR branches to DOS code that prints an error message and then enters the DOS command processor. HDERR is branched to within the DOS whenever a read attempt is impossible to successfully complete after 10 retries. If your software wishes to retain control in the event of a hard disk error, your software should modify the address of the HDERR JMP instruction (e.g., LXI H,ADDR; SHLD HDERR+1). The stack

```

2019          *DOS LIBRARY ROUTINE ENTRY POINTS
2019          *
2019          *THIS ADDRESS IS BRANCHED TO ON HARD DISK ERRORS
2019 C30000    HDERR JMP 0          0 IS NOT THE REAL ADDRESS
201C          *
201C          *THIS IS THE FILE DIRECTORY LOOKUP ROUTINE
201C          *ACC MUST CONTAIN THE DEFAULT UNIT NUMBER (NORMALLY 1)
201C          *HL=POINTER TO FILE NAME IN RAM,
201C          *FOLLOWED BY EITHER A BLANK OR CARRIAGE RETURN.
201C          *FAILURE IF CARRY SET. ON FAILURE, HL=FIRST FREE DISK ADDRESS
201C          *ON SUCCESS, ACC=THE DISK UNIT INDICATED, AND HL HAS A POINTER
201C          *TO THE EIGHTH BYTE OF A COPY OF THE ENTRY IN DOS RAM
201C C30000    DLOOK JMP 0          0 IS NOT THE REAL ADDRESS
201F          *
201F          *THIS ROUTINE WILL WRITE A DIRECTORY ENTRY BACK TO DISK
201F          *NO ARGS ARE NEEDED. MUST FOLLOW DLOOK.
201F C30000    DWRIT JMP 0          0 IS NOT THE REAL ADDRESS
2022          *
2022          *THIS ROUTINE MAY BE USED TO ISSUE A DISK COMMAND
2022          *ACC=NUMBER OF BLOCKS
2022          *B=COMMAND (0=WRITE, 1=READ, 2=VERIFY), C=UNIT NUMBER
2022          *DE=STARTING RAM ADDRESS, HL=STARTING DISK ADDRESS
2022          *RETURN WITH CARRY SET MEANS ARGUMENTS WERE ILLEGAL
2022 C30000    DCOM  JMP 0          0 IS NOT THE REAL ADDRESS
2025          *
2025          *THIS ROUTINE MAY BE USED TO LIST A FILE DIRECTORY
2025          *ACC=DISK UNIT
2025 C30000    LIST  JMP 0          0 IS NOT THE REAL ADDRESS
2028          *
2028          *THIS ADDRESS IS AN ENTRY POINT TO THE LOADED DOS
2028 C30000    DOS   JMP 0          0 IS NOT THE REAL ADDRESS
202B          *
202B          *THIS NEXT BYTE IS A FLAG USED BY DOS.
202B          *IF 0, THEN READ-AFTER-WRITE CHECK IS NOT DONE,
202B          *IF 1, THEN READ-AFTER-WRITE CHECK IS DONE.
202B 00        RWCHK DB 0
202C          *
202C          *

```

Appendix 1: DOS I/O Routines and Entry Points

```

0000      *
0000      *NORTH STAR DISK OPERATING SYSTEM
0000      *
0000      ORG 2000H          STANDARD VERSION ORIGIN VALUE
2000      DS 13            THESE CELLS ARE RESERVED
200D      *
200D      *
200D      *THIS IS THE CHARACTER OUTPUT ROUTINE
200D      *THE CHARACTER TO BE OUTPUT MUST BE IN THE B REGISTER.
200D      *DEVICE NUMBER MAY BE SUPPLIED IN ACC, IF DESIRED.
200D      *ON RETURN THE CHARACTER MUST ALSO BE IN THE ACC.
200D      *NO OTHER REGISTERS MAY BE MODIFIED.
200D C30D20  COUT JMP COUT          YOUR ROUTINE MUST DO A RET
2010      *
2010      *THIS IS THE CHARACTER INPUT ROUTINE.
2010      *DEVICE NUMBER MAY BE SUPPLIED IN ACC, IF DESIRED.
2010      *THE 7-BIT ASCII CODE MUST BE RETURNED IN THE ACC.
2010      *NO OTHER REGISTERS MAY BE MODIFIED
2010 C31020  CIN  JMP CIN          YOUR ROUTINE MUST DO A RET
2013      *
2013      *THIS IS THE TERMINAL INITIALIZATION ROUTINE
2013      *ALL REGISTERS MAY BE USED.
2013      *IF NOT NEEDED, MERELY PATCH IN A RET.
2013 C31320  TINIT JMP TINIT
2016      *
2016      *THIS ROUTINE DETECTS A CONTROL-C
2016      *IF Z IS SET ON RETURN, THAT MEANS A CONTROL-C WAS TYPED.
2016      *OTHERWISE, IF NO CHARACTER WAS TYPED OR A CHARACTER OTHER
2016      * THAN CONTROL-C WAS TYPED, Z MUST NOT BE SET.
2016      *CONTC SHOULD RETURN IMMEDIATELY IF NO CHAR WAS TYPED,
2016      * NOT WAIT FOR A CHARACTER AND THEN RETURN.
2016      *ALL REGISTERS MAY BE USED.
2016 C31620  CONTC JMP CONTC
2019      *

```


Appendix 3: Sample I/O Routines

```

0000      *
0000      *
0000      *SAMPLE I/O ROUTINES FOR PERSONALIZING DOS
0000      *
0000          ORG 2900H
2900
2900      STAT EQU 0          FOR THIS EXAMPLE, ASSUME I/O STATUS PORT IS 0
2900      IBIT EQU 1          ASSUME BIT 0 IS KEYBOARD STATUS BIT
2900      OBIT EQU 2          ASSUME BIT 1 IS OUTPUT STATUS BIT
2900      DATA EQU 1         ASSUME DATA IN AND OUT PORT IS 1
2900
2900      CHIN IN STAT        GET KEYBOARD INPUT STATUS
2902      ANI IBIT           TEST FOR INPUT STATUS READY
2904      CA0029             JZ CHIN             LOOP IF NOT READY
2907      DB01               IN DATA          READ THE CHARACTER
2909      E67F               ANI 7FH          MASK DOWN TO 7-BIT ASCII
290B      C9                 RET
290C
290C      CHOUT IN STAT      GET OUTPUT STATUS
290E      E602               ANI OBIT         MASK DOWN TO OUTPUT STATUS BIT
2910      CA0C29             JZ CHOUT        LOOP IF NOT READY FOR OUTPUT
2913      78                 MOV A,B          COPY THE CHARACTER TO ACC
2914      D301               OUT DATA
2916      C9                 RET             NOTE THAT CHAR IS NOW IN ACC TOO
2917
2917      C9                 INIT RET         TINIT NOT NEEDED IN MANY SYSTEMS
2918
2918      DB00               CCONT IN STAT    GET STATUS BYTE
291A      E601               ANI IBIT         SELECT KEYBOARD STATUS BIT
291C      EE01               XRI IBIT        SET Z FLAG FALSE IF NO INPUT
291E      C0                 RNZ            RETURN IMMEDIATELY IF NO CHAR TYPED
291F      DB01               IN DATA          GET THE TYPED CHAR
2921      E67F               ANI 7FH          MASK DOWN TO 7-BIT ASCII
2923      FE03               CPI 3           SET Z IF CONTROL-C, ELSE CLEAR Z
2925      C9                 RET
2926      *
2926      *AFTER ENTERING THESE ROUTINES, PATCH THE CORRECT ADDRESSES
2926      * IN THE DOS JMP INSTRUCTIONS TO THESE ROUTINES.
2926      *

```

Appendix 2: BASIC Entry Points

```
0000      *
0000      *NORTH STAR BASIC, VERSION 6
0000      *
0000      *          ORG 2A00H          STANDARD VERSION ORIGIN
2A00      *
2A00 AF    EP0    XRA A              INITIALIZATION ENTRY POINT
2A01 C3052A    JMP EP11
2A04 37     EP1    STC              CONTINUE ENTRY POINT
2A05 210000    EP11   LXI H,ENDBAS  FIRST CELL OF PROGRAM REGION
2A08 11FF5F    LXI D,5FFFH        LAST CELL OF CONTIGUOUS MEMORY
2A0B C30000    JMP START          ENDBAS AND START ARE NOT REALLY 0
2A0E      *
```

RELEASE 3
NORTH STAR VERSION 6 BASIC

Release 3 BASIC fixes all reported bugs in Release 2 BASIC, and also makes several additions. Release 3 is a superset of Release 2 - all programs which run under Release 2 should run under Release 3. Note that the size of BASIC has increased - now a 45 block file is required.

A) Multiple input/output devices. Now PRINT, INPUT, LIST, and LINE may specify a "device expression" as an optional first argument preceded by a number sign, e.g.

```
PRINT #3,A,B,C
INPUT #X,"TYPE A NUMBER: ",N
LIST #2,100,500
LINE #2,64
```

Device 0 is assumed if the device expression is missing. Also, device 0 is always used for command communication in direct mode and in the DOS. LINE becomes an executable statement instead of a command, so that device line lengths can also be set up within programs as well as in direct mode.

The device value, which must be between 0 and 7 inclusive, is communicated in the ACC to the DOS CIN and COUT routines. The DOS routines can then decide which device to communicate with based on the contents of the ACC. Note that no changes are needed for the DOS I/O routines if this feature will not be used. Also note that this feature may be used to communicate string values to machine language subroutines.

B) Delete command. The DEL command has been added to allow the deletion of a block of program statements. For example,

```
DEL 100,200
```

will delete all program statements with line numbers in the range of 100 through 200 inclusive.

C) Byte access to data files. It is now possible to access a data file in BASIC at the byte level. A numeric value in the range of 0 to 255 may be written into or read from a specified byte address in a data file. These byte accesses may be performed in either sequential or random access mode. To indicate byte mode, an ampersand (&) is placed before the variable to be read into or the expression to be written out. For example,

```
10 READ #0,A,&B
```