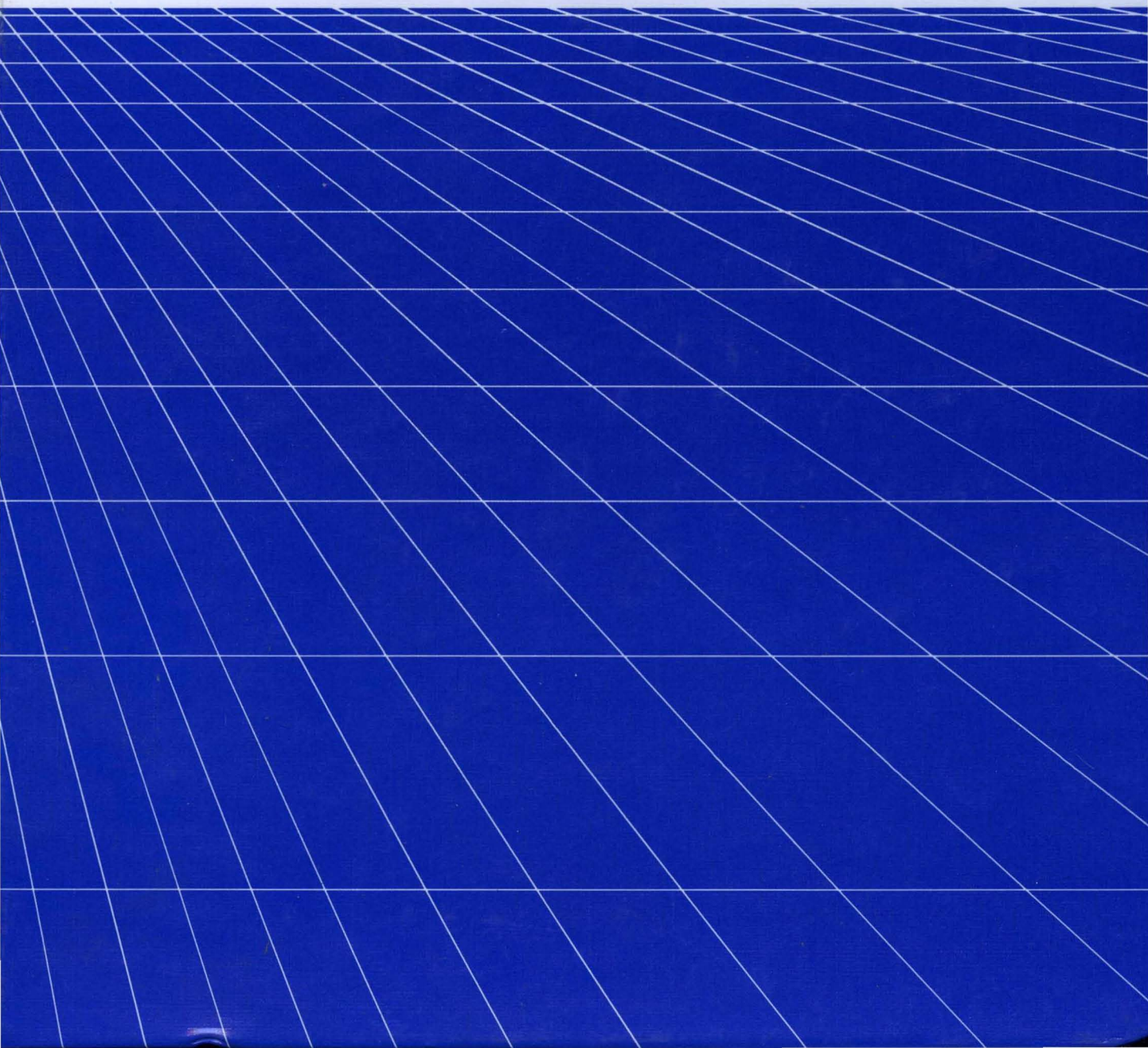
Digital Microsystems™

HINET PROGRAMMERS MANUAL



Digital Microsystems DMS™

PROGRAMMER'S MANUAL

Version 1.0

Digital Microsystems
1755 Embarcadero, Oakland, CA 94606
(415) 532-3686 TWX 910-366-7310

TABLE OF CONTENTS

1.0 INTRODUCTION

HiNet is a Local Area Network developed by Digital Microsystems. It links Z-80 and 8086/88 based microcomputers to centralized Hard Disk Storage and shared printers (via a high-speed print spooler). Local Hard Disk and Floppy Disk storage is available with two of the workstations. Several types of IBM PC compatible microcomputers can also be linked to the Network through a Z-80 based HiNet Adapter Card. MS-DOS, CP/M-86 and HIDOS (enhanced CP/M-80) operating systems can be run on the Network.

This Programmer's Manual is for anyone desiring to design or adapt programs and systems to work with HiNet. The basics of HiNet--its operation and utilities--are presented in DMS manuals for each workstation and the HiNet Master Manual. Some proprietary information that may be necessary to alter the BIOS is not included in this document. This information may be obtained from Digital Microsystems through a special licensing agreement.

2.0 NETWORK PROTOCOLS

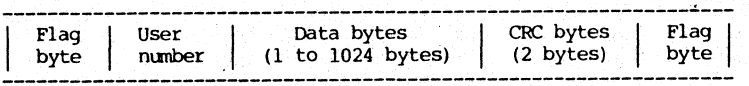
Data transmission and communication on the HiNet Network is governed by a set of protocols. These protocols are handled by the HiNet BIOS. They should not directly concern application programmers. Programs running under HiNet should work through the Operating System BDOS and the HiNet BIOS Interface Calls to manipulate data and communicate on the Network. However, to help

you better understand HiNet and its BIOS Interface, the basic Network protocols are briefly presented in this section.

2.1 NETWORK TRANSMISSION FORMAT

Each transmission on the Network is done in Synchronous Data Link Control (SDLC) format. SDLC was introduced by IBM for computer-to-computer communication. It was chosen for HiNet primarily because the widely available Zilog SIO chip implements most of the details of SDLC transmission and reception. The HiNet system automatically programs the SIO and DMA chips to send or receive blocks of data appropriately.

Each SDLC transmission has the following format:



Field	Description
Flag byte	A flag byte is the bit sequence '01111110'. At least two flag bytes surround each transmission. The SDLC standard requires a minimum of one flag byte before and after each transmission. However, HiNet forces several flag bytes at both ends because it is suspected that the SIO chip has a bug which causes it to miss a flag occasionally.

User number — Each station is assigned a unique identification number, a user number, when it logs in. Each and every transmission to a station must include its user number. The Master is always assigned user number 0, while all other stations are assigned numbers from 1 to 63. User numbers 251 thru 254 are reserved for special purposes, which are described in Section 2.

Data bytes — One or more bytes can be transmitted in the data portion of an SDLC transmission. In HiNet, the data bytes may specify a command, a response, or data to be read from or written to the Master disk.

CRC bytes — Each transmission is terminated by 16 bits of error-check information. These bits are computed when data is transmitted and are re-computed when data is received. If an error occurs in the middle of a transmission, the usual result is a detectable CRC error. HiNet will retry any Network transmission which has a CRC error.

Whenever it observes five consecutive ones in the data stream, the SIO chip inserts a zero bit automatically. These extraneous zero bits are removed by the receiving SIO chip. This zero-insertion method allows the chip to recognize flags, and thus to identify the beginning and the end of each data transmission.

2.2 HINET MASTER FUNCTIONS

On the Master station, the basic control loop is as follows:

1. The Master process is invoked each clock tick (usually 62 hertz). When the local user (commands from the Master's console) engages a private Floppy Disk operation, the Master process is delayed until the next clock tick. This is necessary because the DMA chip is shared by floppy and Network operations.

2. The Master polls each active user after invocation. Active users can respond with one of the commands listed below. All users that have acknowledged the Master's previous poll are active. A poll is a one byte command (50h). All Network transmissions are done in SDLC format, so the poll is actually preceded by the one-byte destination user number.

Description of HiNet Command	Command byte	Command length	Additional command parameters
Acknowledge	41h	1 byte	
Get who table	10h	1 byte	
Read 128 bytes	11h	8 bytes	dtn,src,dsk,trk,sec,vli
Read 1024 bytes	15h	8 bytes	dtn,src,dsk,trk,sec,vli
Write 128 bytes	12h	8 bytes	dtn,src,dsk,trk,sec,vli
Start spool file	14h	2 bytes	sid
Spool 128 bytes	1Ch	8 bytes	dtn,src,dsk,trk,sec,vli
End spool file	16h	2 bytes	sid
Assign partition	17h	15 bytes	nam,psw
Hog the Network	18h	1 byte	
Poll-Prime	55h	1 byte	
Lock record	19h	15 bytes	len,lck

Description of HiNet Command	Command byte	Command length	Additional command parameters
Unlock record	1Ah	15 bytes	len,lck
Clear all locks	1Bh	1 byte	
Get HD status	1Dh	1 byte	
Get date time	1Eh	1 byte	
Login	13h	20 bytes	usr,psw,ser#,prod
Instant logout	1Fh	2 bytes	src
Write Modes	20h	6 bytes	wmc,vli,dsk,val,usr
Network info	21h	1 byte	

Command Parameters	Parameter Length
dtn = destination station number (always 0)	1 byte
src = source station number (same as user number)	1 byte
dsk = partition number (0-63)	1 byte
trk = track number (0-511)	2 bytes
sec = sector number (1-128)	1 byte
vli = volume number (0-3)	1 byte
nam = partition name	8 bytes
usr = user name	8 bytes
psw = password	6 bytes
len = length of lock string (1-13)	1 byte
lck = lock string	13 bytes
wmc = write mode command (grt/rel/frc/qry)	1 byte
val = write mode force value or logical user number	1 byte
usr = write mode physical user number	1 byte
sid = spool job id number	1 byte

----------NOTE----------
Normally there is no need for an applications programmer to use these HiNet commands directly. The sections on the HiNet BIOS Calls and Interstation Communication explain the calls that will perform the above functions for the applications program.
---

3. If a user responds to a poll with an "acknowledge" command, then no further interaction with the user will be contemplated until the Master process re-awakes on the next clock tick. All other commands require an interchange of information between the Master and the User Station, as described in Section 3.0.

4. A user station is polled at the normal polling rate 160 times. The station then becomes a 'slow user' and is polled at 1/12 the normal polling rate for an additional 96 times. A user that responds to any of these polls regains normal status; otherwise, the user is logged out. When a user is logged out his or her user number and all locks are released. If a spool file is being created, it is erased. If the user owns any partitions, they are released.

5. After polling all active users, the Master checks the local user for pending requests. If a request is pending, the local user's Network command byte will be non-zero. Pending requests are processed, and their command bytes are set to zero, signaling command completion.

6. The Z-80 Network station bootstrap code is transmitted periodically (once per polling loop

and about once per second) to pseudo-user 254. The PROM at each Z-80 station has been programmed to receive 380h bytes addressed to 254's user number so that it can boot from the Network. The bootstrap code is loaded into memory at location 9000h, and executed. This boot code displays the "HiNet 2.2xx" message, waits for a poll of pseudo-user 253, and attempts to log in by PROM serial number. The 8086/8088 Network station PROM contains sufficient code to attempt the login by PROM serial number directly.

7. The Master polls pseudo-user 253 periodically (once per polling loop and about once per second). Any station that wants to connect to HiNet must respond to the user 253 poll with a series of interactions.

- The station sends the PROM serial number in ASCII, the serial number in binary, and product type to the Master.
- The Master accepts the login request unconditionally and responds with a unique user number, the login time, and the binary serial number.
- The Master consults the Machine table, Product Type table, and User table on the Hard Disk for a matching ASCII PROM serial number, and product type.
- If the PROM serial number is found in the USERS Table, the Master immediately sends Boot Phase 2, a loader for the BIOS. The station will then auto-boot with the appropriate BIOS and operating system.

- If the PROM Serial Number is not found in the USER Table, Boot Phase 2 loads the Login Please program into the station. This allows the user to enter a specific name and password that is stored in the USER Table. What is then sent to the station depends on whether the name/password was found in the User table and on the Network station's product type.

8. The Master polls pseudo-user 252 once per Master polling loop. A mimicking system (if any) must respond to these polls to come on-line and remain on-line.

9. The Master checks regularly (once every polling loop and also about once per second) whether the spool print buffer is empty and needs to be refilled. If so, the next sector of the printing spool file is read, and printing is restarted. If not, the spooler checks whether a new spool file should be opened and printed.

The following section details the various tables and buffers that are maintained in the Master.

2.3 TABLES AND BUFFERS

2.3.1 WHO TABLE

The who command can be used by any station to determine who is currently logged into HiNet, and who has active spool files.

The Who Table has either 32 or 64 entries, each 16 bytes long. The first byte of the return with a GETWHO indicates the maximum number of users. This value is also returned by NetInfo. Each entry corresponds to a single user. The first entry describes user 0 (the Master user); the remaining entries describe users 1 to 31 or 63. The NetInfo protocol outlined later on in this document may also be used to determine the maximum number of users. Each entry contains the following information:

FORMAT of WHO INFORMATION:

1 byte - max number of users on system (32 or 64)
16 byte entries * max users - USER ENTRIES
256 bytes (16 bytes * 16 entries) - SPOOL ENTRIES

FORMAT of User Entry:

1 byte - Activity count. 0FFh - active
0 - dead
other - logging out

8 bytes - User Name.
3 bytes - login time (secs,mins,hours)
3 bytes - last request time (secs,mins,hours)
1 byte - last request command byte.

2.3.2 SPOOL FILE TABLE

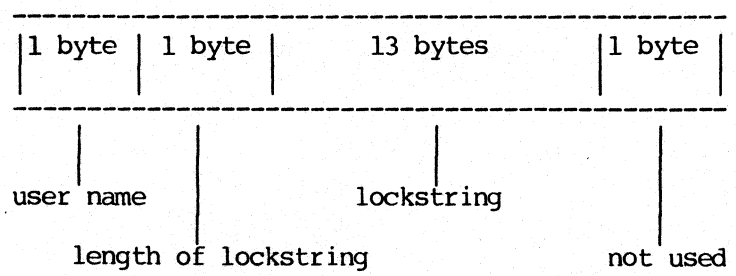
The spool table has 16 entries (the value currently returned by NetInfo), each 16 bytes long. Each entry corresponds to a single spool file. Each entry contains the following information:

1 byte - spool status
- 0 - starting spool
- 1 - spooling
- 2 - ready to print
- 3 - printing
- 4 - stop print
- 5 - waiting
- 0E5h - aborted or done

1 byte - user number of User spooling.
1 byte - spool ID (top nibble job #, low nibble block #).
2 bytes - time job started (min,hour).
2 bytes - track # of last sector of job.
1 byte - sector # of last sector of job.
8 bytes - spooler's name.

2.3.3 LOCK TABLE

Each lockstring is described by a 16-byte entry. The first byte is the number of the user who created the lock. If the entry is not in use, the first byte is 0FFh. The next byte is the lockstring length. The next 13 bytes are the lockstring. The last byte is not used.



See sections 3.0 and 5.0 for more information on lockstrings.

2.3.4 MASTER BUFFER

This is a general buffer for the Master's use. However, not all Hard Disk I/O operations use this buffer. It is located 400h bytes below the Who Table.

2.3.5 FLOPPY WRITE BUFFER

This optional buffer resides 100h bytes below Hard Disk buffer. This buffer is used for all double-density floppy write operations.

2.3.6 FLOPPY READ BUFFER

This optional buffer resides 100h bytes below write buffer. It is used for all double-density floppy read operations.

----------NOTE----------
These tables can be manipulated while the Network is running by using DDT or ZDTI. However, extreme caution should be taken when altering these tables in memory. Any incorrect changes could bring down the Network or write data to the wrong area of the disk, thus destroying data.

2.3.7 HARD DISK TRACK AND SECTOR LAYOUT

The following tables list the track and sector layout for HIDOS (CP/M) and MS-DOS partitions. The contents of the disk parameter table for each possible partition size are shown below:

CP/M 2.2 Disk Parameters

	256K	512K	1MEG	2MEG
Sectors per track	128	128	128	128
Blockshift,mask	3,7,0	4,15,0	4,15,0	4,15,0
Block count - 1	255	255	511	1023
Directory count -	163	127	255	511
Directory blocks	0C000h	0C000h	0F000h	0FF00h
Check vector size**	16	32	64	128
Op sys tracks	0	0	0	0

	4MEG	8MEG	16MEG	32MEG
Sectors per track	128	128	128	128
Blockshift,mask	4,15,0	5,31,1	6,63,3	7,127,7
Block count - 1	2047	2047	2047	2047
Directory count -	1023	1023	1023	1023
Directory blocks	0FFFFh	0FF00h	0F000h	0C000h
Check vector size**	256	256	256	256
Op sys tracks	0	0	0	0

** HiDos does not allocate check vectors, size = 0.

CP/M FLOPPY DISK LAYOUT

	8" Floppy Single Density	8" Floppy Double Density	5.25" Double-Sided Floppy
Sectors per track	26	52	32
Blockshift,mask	3,7,0	4,15,0	5,31,0
Block count - 1	242	242	156
Directory count - 1	63	127	127
Directory blocks	0C000h	0C000h	08000h
Check vector size	16	32	32
Op sys tracks	2	2	3

MS-DOS DISK PARAMETERS

	256K	512K	1MEG	2MEG
Bytes per sector	128	128	128	128
Sectors per cluster	8	8	8	16
Reserved sectors	1	1	1	1
No. FATs	2	2	2	2
Root dir entries	256	256	256	512
No. sectors	2048	4096	8192	16384
Media byte	0	1	2	3
Sectors per FAT	3	6	12	12
Sectors per track	128	128	128	128

	4MEG	8MEG	16MEG	32MEG
Bytes per sector	128	128	256	512
Sectors per cluster	16	32	32	32
Reserved sectors	1	1	1	1
No. FATs	2	2	2	2
Root dir entries	1020	1020	1020	1020
No. sectors	32768	65535	65535	65535
Media byte	4	5	6	7
Sectors per FAT	24	24	12	6
Sectors per track	128	128	64	32

The first sector of each MS-DOS partition contains a copy of the above disk parameters. Microsoft refers to this data as the BPB and it is stored in the format described in the MS-DOS 2.0 Programmer's Reference Manual.

2.3.8 LAYOUT OF HARD DISK PARTITION ZERO

Tracks and Sectors are **CP/M Logical, not physical.**

track 0, sectors 01-1F
Controller Program

track 0, sectors 20-28
reserved for expansion of controller program

track 0, sectors 29-38
HiNet User Name Table
Up to 128 16-byte entries:

track 0, sectors 39-78
HiNet User Configuration Table

Up to 128 64-byte entries:
8 bytes: default A drive
8 bytes: default B drive
8 bytes: default C drive
8 bytes: default D drive
1 byte: length of typeahead
31 bytes: typeahead buffer

track 0, sectors 79-80
Disk Allocation Table

```
track 1, sectors 01-08
       Bad Sector Table

       Up to 64, 128, or 256 3-byte entries
       depending on drive type:
       1 byte:  track
       1 byte:  head
       1 byte:  sector

track 1, sectors 09-14
       Machine Table

       Up to 128 12-byte entries:
       4 bytes:  Serial Number
       1 byte:   Product Number
       6 bytes:  Option Map
       1 byte:   IOBYTE

track 1, sectors 15-16
       Write Mode Table

track 1, sector  17
       reserved

track 1, sector  18
       Reserved

track 1, sectors 19-20:
       Product Type Table

       Up to 40  25-byte entries:
       1 byte:   Product Type
       8 bytes:  Boot Phase 2 program name
       8 bytes:  Login Please program name
       8 bytes:  OS Menu program name
```

```
track 1, sectors 21-80:
       Operating System Table

       Up to 128 96-byte entries:
       1 byte:   OS number
      16 bytes:  Product Map
       6 bytes:  Option Map
      64 bytes:  Load List (8 names of 8 bytes)
       9 bytes:  --reserved--

track 2, sectors 01-02
       Cold Boot Loader

track 2, sectors 03-08
       reserved for use of Cold Boot Loader

track 2, sectors 09-20
       System Directory

       Up to 128 24-byte entries:
       8 bytes:  File Name
       5 bytes:  Disk Address
       2 bytes:  Length (128-byte records)
       4 bytes:  Load Address
       2 bytes:  Execution Address Offset
       1 byte:   Program/Data flag
       2 bytes:  --reserved--

track 2, sectors 21-80 -- Reserved
    Remainder of partition allocated according to
    contents of the System Directory.
```

3.0 THE HINET BIOS INTERFACE

The HiNet BIOS has several routines which can be called from a user program to send or receive data over the HiNet cable. Utility programs such as WHO, DIRNET, and ASSIGN use these routines to interface with HiNet. On Z-80s all four routines are accessible though jump vectors at fixed offsets from the base of the BIOS; on 8086/88s through BDOS Call 50 or Interrupt 78; under MS-DOS, the network calls must be accessed with INT 78 and the MACHINE BIOS Number.

Offset	Name	Description
6Fh	SENDNET	Transmit a block of data on the Network.

Input:
- HL = address of data to be transmitted
- BC = number of data bytes
- E = pre-transmission delay (Master only)
- A = user number of intended recipient

72h	RECNET	Receive a block of data from the Network.

Input:
- HL = address where data is to be stored
- BC = maximum number of data bytes
- DE = timeout delay (Master only)
- A = user number of recipient

Output:
- A = result status
  - bit7 =0 if timeout (block not received)
  - = 1 if block received
  - bit 6 = 0 if no CRC error

```
            = 1  if CRC error
      bit 5 = 0  if no receiver overrun
            = 1  if receiver overrun

Stations Only>
      bit 0 = 0  if a valid poll was not received
            = 1  if a valid poll was received
```

User station only

75h NACKPOLL Wait for next poll, then deactivate automatic poll acknowledgments.

Output: A = result status (same as RECNET)

Master station only

75h INTERCEPT Intercept and process a command from a user station.

Input: HL = address of HiNet command

Output: A = 0 if OK, non-zero if error

HINET BIOS JUMP VECTORS:

User station only:

78h ACKPOLL Reactivate automatic poll acknowledgments.

Master station only:

78h INTERRUPT Process a one-second interrupt.

84h HDstat Check status of local hard disk.

User station only:

87h HDstat Check status of Network volume(s).

Programs at any station other than the Master can communicate on the Network by calling the ACKPOLL routine first (to synchronize in case of a previous error), then the NACKPOLL routine. NACKPOLL either receives a poll or returns to the user after a four-second wait. When no poll is received NACKPOLL returns with an error status in A.

The proper way to test status is to test each bit individually, or AND the returned value with the significant status bits: 0E1h, and test for poll received with no error: 81h. When a poll is received, the program can then call SENDNET and RECNET to complete the desired transaction. When the transaction has been completed, the program should call ACKPOLL. ACKPOLL will force the BIOS to acknowledge polls automatically until NACKPOLL is called again.

When using SENDNET or RECNET at a user station, one does not need to specify a pre-transmission or timeout delay. The pretransmission delay will always be 500 microseconds while the timeout delay will always be about four seconds. The pretransmission delay should give the intended recipient of a message more than enough time to reprogram the DMA and SIO chips. For example, even if several interrupts (timer, spooler) occur while the chips are being reprogrammed, the recipient has 300 microseconds more than the usual 200 to prepare for reception of data from HiNet. The Master can vary the delays to minimize wasted time.

The INTERCEPT and INTERRUPT jump vectors are available only on the Master. The Master calls the INTERCEPT routine through the jump vector whenever it receives a command from a user station which has a command byte of 0. (The first byte of a command is called the "command byte".) Commands can be up to 15 characters long, including the command byte. By replacing the INTERCEPT jump vector, users can supply their own command processing routine to communicate directly with the Master station programs, other user stations, or anything they choose to do through their applications program.

The code to replace the jump vector might look something like this:

```
di              ; don't allow interrupts while changing
lhld  1         ; get address of base of BIOS + 3
lxi   D,73h     ; offset of INTERCEPT address
dad   D         ; compute address of INTERCEPT vector
lxi   D,INTERCEPT
mov   M,E       ; replace low byte of INTERCEPT vector
inx   H
mov   M,D       ; replace high byte of INTERCEPT vector
ei              ; interrupts OK now
; Here is the new INTERCEPT routine
INTERCEPT:
mov   a,m       ; ensure first command byte was zero
ora   a
mvi   a,0ffh    ; ret a = 0ffh if command wasn't zero
rnz
inx   H         ; skip first command byte
mov   A,M       ; look at second byte of command
sub   A         ; return status of OK (non-zero
ret             ; if error)
```

Note that the INTERCEPT routine is invoked also by the Master for illegal commands; so, the user should return a = ffh for any non-zero command.

The following example is an assembly language program which shows how a user station can access the WHO Table and the SPOOL Table. Both are maintained by the Master. These tables can be used to determine who is logged into the Network, and what each current user is doing. Note, the user number is stored in the same location at every Z-80 station: 47H. The User Number should be used to communicate with the Master; it should never be changed once the station has logged onto the Network.

```
;
; Determine size of who table
BeginWho:
          call WaitPoll ; wait for Network poll
          jrnz BeginWho ; start over again if error
;
          lxi  H,NetInfo ; point to command
          lxi  B,1       ; one byte command
          sub  A         ; master is user 0
          call SENDNET   ; send NetInfo to Master
;
          lxi  H,InfoTab ; place to store NetInfo table
          lxi  B,128     ; length of table
          call Receive   ; Receive from Network
          jrnz BeginWho  ; Try again if failed
;
; Now set up to get Who table
;
          call WaitPoll
          jrnz BeginWho ; start over again if error
```

```
;
; Send who command to Master
          lxi  H,WHOCMD  ; point to command
          lxi  B,1       ; this is a 1 byte command
          sub  A         ; the Master is always user 0
          call SENDNET   ; send "who" command to Master
;
; Get who table from Master
          lhld InfoTab+1 ; number of users x 16 + 1
          call Mul16
          inx  B
          lxi  H,WHOTAB  ; address of table
          call Receive   ; Receive data from Network
          jrnz GetWho    ; Try again if failed
;
; Get spool file table
          lhld InfoTab+5 ; number of spool entries x 16
          call Mul16
          lxi  H,SPLTAB ; address of table
          call Receive  ; Receive data from Network
          jrnz GetWho   ; Try again if failed
;
; Who table received successfully
          CALL ACKPOLL  ; re-activate automatic poll ACK
          .

NoTimeOut == 80h   ; set if message received
crc_ovr   == 60h   ; set if crc or overrun error
ValidPoll == 01h   ; set if valid poll received
NetUsr    == 47h   ; location in memory of user number
;
; General Wait to receive poll routine, returns TZ if poll
;
WaitPoll:
          call ACKPOLL   ; resume acking polls
          call NACKPOLL ; intercept the next one
          ani  NoTimeOut+ValidPoll+crc_ovr ; check status
```

```
        cpi  NoTimeOut+ValidPoll
        rz          ; return if succeeded
;
        push PSW    ; save error status for analysis
        call ACKPOLL ; resume automatic poll acknowlegement
        mvi  C,conouts ; poll not received, so ...
        lxi  D,NOPOLL ; print error message
        call BDOS   ; use the BDOS print function
        pop  PSW    ; return A reg and FZ
        ret
;
; General Receive Network routine, returns TZ if receive OK
;
Receive:
        lda  NetUsr    ; this station's user number at 47H
        call RECNET
        ani  NoTimeOut+ValidPoll+crc_ovr ; check status
        cpi  NoTimeOut ; should get non-poll msg rec'd
        rz             ; return if succeeded
;
        push PSW       ; save error status for analysis
        mvi  c,conouts
        lxi  d,xmitFailed
        call BDOS
        pop  PSW       ; return A reg and FZ
        ret
; Multiply L register by 16 and move result to BC
Mul16:
        mvi  H,0       ; zero upper byte
        dad  H         ; multiply by adding to self 4x
        dad  H
        dad  H
        dad  H
        push H        ; move result to BC
        pop  B
        ret
```

3.1 HINET BIOS CALLS

3.1.1 INTRODUCTION

Since the BIOS is specifically tailored to DMS hardware, programmer's should use the appropriate BIOS Calls whenever possible instead of writing to the hardware or using specific addresses in the BIOS. This section explains in detail the available calls to the BIOS. Section 4 details the BIOS Calls for the DMS-816's IBM ROM BIOS emulation.

3.1.2 Z-80 BIOS CALLS

To make a BIOS call on Z-80 stations, the BIOS function number is converted to an offset and the offset is added to the start of the BIOS Jump Vector. It is up to the application program to save any registers that may be destroyed by the BIOS. This method of calling the BIOS is demonstrated in the following example.

```
-------------------------------------------------
BOOT            equ     00000h
BIOS            equ     00001h

bioscall:   ;given  N = BIOS function number and
            ;all registers stuffed for the
            ;function being called, this
            ;routine will call the BIOS.
;ZILOG mnemonics                 TDL mnemonics
 push       IX                   push    X
 push       DE                   push    D
 ld         DE,(3 * N)           lxi     D,(3 * N)
```

```
ld        IX,(BIOS)              lixd      BIOS
add       IX,DE                  dadx      D
pop       DE                     pop       D
call      jmpbios                call      jmpbios
pop       IX                     pop       X
ret                              ret
jmpbios:   ;called to force a return address on
           ;the stack
jp        (IX)                   pcix
```

Function numbers 0 through 15 are standard to CP/M-80; they are handled exactly as documented in Digital Research's CP/M Operating System Manual. BIOS Function Numbers 30 through 49 are calls that are specific to Digital Microsystems' BIOS and should be used according to the above example. Each call is documented in the following sections for Z-80 HIDOS; 8086/88 CP/M-86 and MS-DOS. The function number table on the next three pages gives a complete list of the available function calls. A hyphen indicates that a call is not available for the particular operating system.

BIOS Function Numbers

BIOS Function	CPM-80 function #	CPM-86 (BDOS 50) function #	COMMON (INT 78) function #
Cold Boot	-1	0	50
Warm Boot	0	1	-
Console Status	1	2	0
Console Flush	-	-	1
Console Input	2	3	2
Console Output	3	4	3
List Output	4	5	5
Punch Output	5	6	-
Reader Input	6	7	-
Aux Status	-	-	6
Aux Flush	-	-	7
Aux Input	-	-	8
Aux Output	-	-	9
HOME disk	7	8	-
Select disk	8	9	-
Set Track	9	10	-
Set Record	10	11	-
Set DMA offset	11	12	-
Read Record	12	13	10
Write Record	13	14	11
List Status	14	15	4
Get Memory size	-	-	12
Media Same	-	-	13
Spool Flush	-	-	14
Set I/O pointer	-	-	15

BIOS Function	function # CPM-80	function # CPM-86 (BDOS 50)	function # COMMON (INT 78)
Clear Locks	-	-	16
Aux Status 1	-	-	17
Record Translation	15	16	-
Set DMA segment	-	17	-
Get Segment Table	-	18	-
Get I/O byte	-	19	19
SET I/O byte	-	20	20
Make Assignment	-	21	21
Get Assignment	-	22	22
Set/reset visible error flag	-	25	24
Get Error Count	-	26	25
Set/reset Retry flag	-	27	26
WHO request	-	28	27
DIRNET request	-	29	28
Flush Flop Buff	30	-	-
Net Lock	31	30	29
CP/M MAP	32	-	-
Net Unlock	33	31	30
Coerce	-	32	31
Set I/O byte count	34	-	-
Version (location of)	35	-	-
SENDNET/sndnet	36	38	37
RECNET/rcvnet	37	39	38
NACKPOLL/clrpol	38	37	36
ACKPOLL/setpol	39	36	35
'USER' port output	40	-	-
Set Poll-Prime address	41	33	32
Set Receive timeout	42	-	-
Local Hard Disk Status	43	-	-
Net Hard Disk status	44	34	33
Set Network Mode	45	-	-

BIOS Function	function # CPM-80	function # CPM-86 (BDOS 50)	function # COMMON (INT 78)
Set List Type	46	-	-
Hard Disk Reset	47	-	-
Write Mode Request	48	-	42
BIOS information	49	-	41
Time and Date	-	35	34
Direct Gut bios call	-	40	-
816 Video Routine	-	-	39
Print Screen	-	-	40
Aux Init	-	-	43
Partition Table Address	-	-	44
Network Info Request	-	-	45
Set Print Escape	-	-	46
Spool Message flip/flop	-	-	47
Rectime	-	-	48
Ppsend	-	-	49

HIDOS BIOS FUNCTION NUMBERS ON THE Z-80

30. clearDDbuf (Flush Flop Buff) -- clears the double-density write buffer on DMS-3/Fs and DMS-3/501s.

    ON ENTRY -- Nothing

    ON EXIT -- Nothing

    REGISTERS DESTROYED-- All

31. NETLOCK - used for adding strings to the Master's lock table. See Section 8 for complete information on NETLOCK and NETUNLOK.

    ON ENTRY: Location 74 (4Ah) and 75 (4Bh) points to block where 14 byte lockstring is stored. The format of the lockstring is:

    1 byte - length of string (max 13)
    13 byte - string. String should be blank padded if it's less than 13 bytes.

    ON EXIT: In the accumulator (LOCKSTAT) and register L.

    LockStat = 00, L register = 00
    - accepted, lock successful

    LockStat = 01, L register = 01
    - denied, string locked by someone else

    LockStat = 02, L register = 02
    - denied, illegal string length

    LockStat = 01, L register = 81h
    - denied, string already locked by you

    LockStat = 02, L register = 82h
    - denied, lock table is full

    ----------NOTE----------
    The retry information is given to warn user that the lock may have been successful but the confirmation was missed because of Network noise. The lock is automatically tried again. However, since the first lock attempt may have successfully locked the string, the status says

it is locked. It is impossible to tell if it was locked on the first try or at some previous time.

---

32. cpmMap (CP/M MAP) -- get the current disk map. cpmMap returns in HL the address of the disk drive number or the partition unit number. This address is also one less than the address of the DPB for the drive or partition. If SELDSK has been called with a drive number of 0FFH, it returns in HL a pointer to the warm boot device drive/partition number.

    ON ENTRY -- none

    ON EXIT -- HL = pointer to unit number (DPBaddr-1)
    DE = pointer to DPH
    A = unit number

33. NETUNLOCK - release a locked string in the Master's Lockstring Table. See Section 5 for complete information.

    ON ENTRY: Location 74 (4Ah) and 75 (4Bh) points to block where 14 byte lockstring is stored. The format of the lockstring is the same as for NETLOCK.

    OM EXIT: NETUNLOK returns status in Accumulator (LockStat) and register L:

    LockStat = 00, L register = 00
    - accepted, unlock successful

    LockStat = 01, L register = 01
    - denied, string was locked by someone else

LockStat = 02, L register = 02
- denied, illegal string length

LockStat = 02, L register = 82h
- denied, lock string not in table

----------NOTE----------
Here, as in lock, a garbled response from the Master can lead to a retry, and then a spurious deny. This flag warns the user so he or she can use his or her best judgement as to how to interpret this.
---

34. SETBYT (SET I/O BYTE COUNT) -- sets the size of the DMA transfer.

    ON ENTRY -- BA has size of DMA.

    ON EXIT -- Nothing.

    REGISTERS DESTROYED -- None.

35. VERSION -- at this address is the BIOS revision number in binary format.

36. SENDNET - send data to the Master. Data can only be sent after a successful NAKPOLL so that the Master is waiting for data with the right User Number. Data must be sent immediately after the NAKPOLL or else the Master will time out waiting to poll the next station.

ON ENTRY: HL = block address
BC = byte count
E = delay time (master only)
A = user number
ON EXIT: None.

REGISTERS DESTROYED: A, BC, DE, HL.

NOTE: this routine assumes the send was successful since there is no way for a station to tell if the Master got the data.

37. RECNET - receive data from the Network. This is generally used after a NAKPOLL and a SENDNET have made a request from the Master.

ON ENTRY: HL = block address.
BC = maximum byte count.
DE = timeout count (Master only).
A = user number.

ON EXIT: A = error status:
bit 7 reset = timeout
bit 6 set = CRC error
bit 5 set = receiver overrun
bit 0 set = poll only received.

REGISTERS DESTROYED: A, BC, DE, HL.

38. NACKPOLL - wait for next poll from Master but don't acknowledge (ACK).

    ON ENTRY: nothing.

    ON EXIT: A = error status:
    bit 7 reset = timeout
    bit 6 set = CRC error
    bit 5 set = receiver overrun
    bit 0 set = poll only received.
    REGISTERS DESTROYED: BC, HL

39. ACKPOLL - this module programs the SIO to automatically ACK polls from the Master. This means that it will answer Network polls with an ACK and the user should be unaffected, except for time lost in interrupts.

    ON ENTRY: None.

    ON EXIT: None.

    REGISTERS DESTROYED: None.

40. PORTUout (USER PORT OUT) -- to output characters to CUSTOM Printer. Currently this is XON/XOFF on Port 2.

    ON ENTRY -- C has character to print.

    ON EXIT -- nothing.

    REGISTERS DESTROYED -- HL, A

41. SETppa (SET POLL PRIME ADDRESS) -- set the address of the poll-prime data structure. Return the last address in register HL.

    ON ENTRY -- BC has address of Poll-Prime block.

    ON EXIT -- HL has last address of data block from poll-prime.

    REGISTERS DESTROYED -- None.

42. RECTIME (RECEIVE TIMEOUT) -- sets the timeout interval for RECNET. (Assembled in HiNet station BIOS only.)

    ON ENTRY -- BC = timeout constant (in milliseconds).

    ON EXIT -- none.

    REGISTERS DESTROYED -- none.

43. HDSTAT (LOCAL HARD DISK STATUS) -- gets local Hard Disk status, (not Network Hard Disk status).

    ON ENTRY -- BC = block address
    On return, block has 8 byte Hard Disk command status followed by 128 byte volume information block.

    ON EXIT -- none.

    REGISTERS DESTROYED -- A, BC, DE, HL.

44. NEThdstat (NET HARD DISK STATUS) -- gets Network Hard Disk status.

ON ENTRY -- Register BC has address of block to load NET hdstat information.

HDSTAT info format:

```
1 byte  - 18h  (echoes command for info)
1 byte  -  ROM version.
1 byte  -  ROM revision.
1 byte  -  firmware version number.
1 byte  -  firmware revision number.
2 bytes - unused
1 byte  - status.  0 - disk present, info OK.
                  ~0 - no disk or other error,
                       rest of information invalid.
4  32 byte volume entries.

Format of Volume Entry -
1 byte   - status.       0FFh - volume present.
                            0 - volume not present.
1 byte   - current track for volume.
1 byte   - number of tracks on volume.
1 byte   - sectors per track.
1 byte   - head mask.
2 bytes  - location of partition offset table.
2 bytes  - location of bad sector table.
2 bytes  - location of bad sector dirty flag.
1 byte   - error in volume open during firmware
           initialization (0 = no error).
10 bytes - Volume Label.
10 bytes - unused
```

ON EXIT -- Block is loaded.
REGISTERS DESTROYED -- All.

45. SetNetMode (SET NETWORK MODE) -- this call allows multi-user application programs to set the local 1K Network Buffer mode to one of three states to ensure current data at all times. See Section 6 for more information. The three available states are:

    0) Always use the 1k Network Buffer. This is the default mode; it is automatically selected after a cold or warm boot.

    1) Do not use the buffer contents on the next NetRead request - force a network transmission to ensure current data. This will replace the 1k network buffer contents; all subsequent NetReads will use the buffer contents.

    2) Do not use the buffer contents until a cold or warm boot or until the program changes the network buffer usage mode.

    ON ENTRY -- C = New Mode (0, 1, 2)

    ON EXIT -- A = Old Mode

    REGISTERS DESTROYED -- HL

46. SetListType -- the sequence for accessing the printer attached to the console.

    ON ENTRY -- H = first character of enable sequence.
    L = second character of enable sequence.

D = first character of disable sequence.
E = second character of disable sequence.

ON EXIT -- None.

REGISTERS DESTROYED — None.

47. INITHARD (HARD DISK RESET) -- resets either the local or Network Hard Disk. Reset Network Hard Disk works only from the Master.

ON ENTRY -- C = 0, reset Network Hard Disk.
= 1, reset local Hard Disk.

ON EXIT -- Hard Disk reset.

REGISTERS DESTROYED — All

48. MakeModeRequest (WRITE MODE REQUEST) --

ON ENTRY -- HL - address of entry block.
DE - address of exit block.

ON EXIT -- data is in exit block.

Entry Block:	Exit Block:
Request Type	Reponse Status
Volume #	Unit Status
Unit #	8 byte Name
Value #	field
User #	
Drive # (logical)	

REGISTERS DESTROYED -- None.

49. DESCRIBE (BIOS INFORMATION)

ON ENTRY -- None.

ON EXIT -- HL has address of table.

REGISTERS DESTROYED -- None.

DESCRIBE Table:

Product type
Operating system type
Version #
Revision #
Patch
Mod
4 bytes, PROM Serial Number
6 bytes, option map
1 byte password
2 bytes, DMA Vector address
2 bytes, Lock byte
2 bytes, Lock DMA

3.1.3 BIOS CALLS THROUGH INTERRUPT 78

DMS has implemented Interrupt 78 as a COMMON BIOS entry point. The user loads the registers required by the specific BIOS call, moves the COMMON BIOS function number to the AH register and then issues an INT 78. This method may also be used by CP/M-86 programs; however, the "COMMON' Function Number--not the CP/M-86 Function Number--must be used.

The SI, BP and DS registers are saved. **All others (including the flags) may be modified by the BIOS.** The stack is manipulated to ensure the flags, as modified by the BIOS, are returned to the user.

3.1.4 CP/M-86 BDOS CALL 50 -- CALL THE BIOS

Digital Research supplies documentation on making BIOS calls via BDOS call 50 in the CP/M-

86 Operating System Manuals. When using BDOS Call 50, only the function number, CX and DX registers may be passed to the BIOS. See below for more information.

3.1.5 CP/M-86 and MS-DOS BIOS CALLS

For compatibility, the parameter block passed by the application is the same for CP/M-86 and MS-DOS. It is 5 bytes in length and contains:

BYTE	
0	BIOS function Number
1-2	value to be loaded into CX
3-4	value to be loaded into DX

If the function requires more data than can be passed in CX and DX, DX:CX will be the segment:offset of a parameter block tailored for the function being called. For example - the parameter block used by SNDNET and RCVNET is:

BYTE	
0	User number
1-2	offset of data to send/receive
3-4	segment of data to send/receive
5-6	length of data to send/receive

The following pages in this section show the currently implemented BIOS calls available in all three operating systems. The Z-80 and the Z816 share the same function number; however they may not always expect the same parameters or work exactly the same way. See the section on the particular function desired for specific

information. CP/M-86 uses the function numbers in the CPM-86 column when making BIOS calls via BDOS Call 50.

If using INT 78, the function numbers in the COMMON column of the BIOS Call Table should be used. MS-DOS uses the function numbers from the COMMON column for both the DMSBIOS and the INT 78 methods of accessing the BIOS. THE INT 78 METHOD IS PREFERRED.

3.2 MACHINE BIOS CALLS FOR 8086/88 DEVICES

This section describes the calls that can be made to the MACHINE BIOS (formerly called the GUT BIOS) on DMS 8086 and 8088 based machines. MACHINE BIOS calls are specific to the workstations (CPUs) while the SYSTEM BIOS calls are available through every type of station. The SYSTEM BIOS for CP/M-80 (on 816s), CP/M-86 and MS-DOS all use some of these calls. They can be called by a user through two methods:

A. USING INTERRUPTS TO CALL MACHINE BIOS FUNCTIONS:

For all functions under MS-DOS and CP/M-86 except video functions:

Put function number in AH,

load all registers as Function you are calling expects them,

call Interrupt 78.

For example, to call MACHINE BIOS function 33 HDSTAT:

```
mov ah,33
mov cx,offset HDBUFFER
mov dx,ds
int 78
```

B. DIRECT BIOS CALL through BDOS.

Under CP/M-86, BDOS call 50 calls the BIOS which can then call the MACHINE BIOS. To make the call you must set up two buffers: one for the BDOS to use to call the BIOS, and one for the BIOS to use to call the MACHINE BIOS.

Here is an example:

To call MACHINE BIOS function 33 HDSTAT.

HDSTAT expects DX and CX to hold the address of the buffer to read information into. Say the buffer is located at address DATASEG:HDBUFFER. Set up a buffer for call to HDSTAT from the BIOS called BIOScall (in DATASEG segment). This buffer contains DATASEQ equ 1000h (segment this program is running in).

1 byte - MACHINE BIOS FUNCTION NUMBER (in this case 33 (HDSTAT Call))

1 word - VALUE to load into CX for MACHINE BIOS call (in this case offset of HDBUFFER).

1 word - VALUE to load into DX for MACHINE BIOS call (in this case value of DATASEG).

In code:

```
BIOScall        db      33
                dw      HDBUFFER
                dw      DATASEG
```

Now set up a buffer (BDOScall) for BDOS to call BIOS with:

1 byte - BIOS FUNCTION NUMBER (always 40 (28h) - for Call to Machine).

1 word - VALUE to load into CX for BIOS call (in this case offset of BIOScall).

1 word - VALUE to load into DX for MACHINE BIOS call (in this case value of DATASEG).

```
In code:        db      28h
                dw      BIOScall
                dw      DATASEG
```

Now to call the Machine function we do this:
(assuming DS already has value DATASEG in it.)

```
mov  dx,offset BDOScall  ; buffer with call to BIOS
mov  cx,50               ; bios call function
int  0E0h                ; call bdos
```

INDEX OF 8086/88 MACHINE BIOS FUNCTION NUMBERS

3.2.1 8086/88 MACHINE BIOS FUNCTIONS

The 8086/88 MACHINE BIOS functions are listed below in numeric order. The function number must be placed in register AH before Interrupt 78 or BDOS call 50 are invoked.

The following numbers are for the INTERRUPT 78 method of calling the BIOS.

0. CONSTATUS - get status of console input.

   ON ENTRY: nothing.
   ON EXIT: If there is a char the zero flag is reset and char is in AX. Otherwise zero flag is set, 0 in AL.

1. CONFLUSH - empty the keyboard type-ahead buffer and input port. This routine keeps getting characters and throwing them away until there are no more.

2. CONINPUT - get character from console.

   ON EXIT: AX hold char. Routine does not return until it has a character.

3. CONOUTPUT - Outputs a character to the console.

   ON ENTRY: Char is in AL.
   ON EXIT: nothing.

4. PRNSTATUS - check printer status.

   ON EXIT:
   For all devices except spooler:
   Zero flag reset if printer is ready.

   For Spooler: If a failed attempt was made to print a character since the last job end, the status is bad, and every time status is called another attempt is made to open

a spool block. Status is based on the results of this attempt.

Under MS-DOS a timing loop prevents the operating system from prematurely aborting the print routine if the printer signals it is not ready (e.g., its buffer is full).

If ready: zero flag reset, AL has FF
If not ready: zero flag set, AL has 0

5. PRNOUTPUT - output a char to the list device.

ON ENTRY: char is in AL.
ON EXIT:
IF SPOOLER
char is in AL
success - zero flag reset.
failure - zero flag set.

6. AUXSTATUS - get status from the auxiliary output device.

ON EXIT: AL contains status from Read Reg 0 of SIO Zero flag is reset if device is ready to transmit.

7. AUXFLUSH - empty the auxiliary buffer, if any. **(NOT IMPLEMENTED as of 9/1)**.

8. AUXINPUT - get character from auxiliary device.

ON EXIT: char is in AL.

9. AUXOUTPUT - write character to AUX device.

   ON ENTRY: char is in AL.
   ON EXIT: nothing

10. DISKREAD - read from disk.

    ON ENTRY: AL - logical drive number
              ES - buffer segment
              DI - buffer offset
              DX - sector
              BX - track
              CX - number of 128b sectors to read

    ON EXIT: Info is read into the buffer at ES:DI. If success zero flag is reset else zero is set and CL has number of sectors successfully read.

11. DISKWRITE - parameters identical to disk read. (see above)

12. GETMEMSIZE - returns the physical size of memory.

    ON EXIT: CX has memory size in (16 byte) paragraphs. (E.g., 1000H = 64K memory.)

13. MEDIASAME - returns the current value of the media change flag, and resets the media change flag to true. Used by MS-DOS only, to see if it needs to re-read a directory before doing some file I/O. Media change flag is set true whenever

there is an assign to make MS-DOS keep up with the latest.

14. SPOOLFLUSH - Ends a spool job if one is active.

15. SETIOPTR - relocates the iobyte to the address asked for.

    ON ENTRY: CX has offset of address.
    DX has segment of address.
    If addr is 0:0 it means reset to default IOBYTE address.

    ON EXIT: moves the current IOBYTE to this address, and thereafter accesses that address when it needs IOBYTE.

16. CLRLOCK - sends byte to Master that says clear all locks for our user number.

    ON ENTRY: nothing
    ON EXIT: AX is 0, zero flag set.

17. AUXSTATUS1 - Get status of aux device from SIO read reg 1. This call is needed only by the ROM BIOS Emulation.

19. GETIOBF - Get the IOBYTE.

    ON EXIT: AL has the IOBYTE.

20. SETIOBF - Set the IOBYTE.

    ON ENTRY: value to set IOBYTE is in CL.

21. MAKASS - This call attempts to assign a partition to a logical drive number. The only valid drive types are HiNet and Memory disk.

    (IMPORTANT: BIOS Memory disk assignments are only valid on CP/M-86. MS-DOS does not allow BIOS manipulation of the Memory Allocation Table. An attempt to assign a Memory Disk while running under MS-DOS will result in a crash. However MS-DOS Memory Disks are available through the MS-DOS CONFIG.SYS device drivers.)

    The format of the assign request buffer is:

    byte:

0	-	logical drive number	(set by caller)
1-8	-	partition name	(set by caller)
9	-	partition size	
10	-	partition number	
11	-	control byte	(caller sets OS type)
12	-	volume number	
13	-	drive type	(set by caller)
14	-	write status	
15-20	-	password	(set by caller)

    OS type set by caller in control byte: Low bit determines OS. See DIRNET for size and layout of the Control byte.

        CP/M-80, HIDOS, & CP/M-86 - 0
        MS-DOS - 1

The possible drive type values are listed here. Only the first two are currently valid for the 86 stations. The rest are found in use with Z-80 stations:

Network Partition	-	60h
Memory Disk	-	E0h
8 inch Hard Disk	-	40h
5 inch Hard Disk	-	C0h
mini-floppy(one side)	-	80h
mini-floppy(double side)		A0h
8" floppy (single dn)	-	20h
8" floppy (double dn)	-	00h

ON ENTRY: CX has offset of request buffer
DX has segment of request buffer

ON EXIT:

IF SUCCESS
AL = 0 and all bytes in buffer not set by caller are filled in by bios. Note that the write status field is set on the basis of the control byte, so if its an ownable partition it is marked as unowned. No query about its present status is made by this module.

IF FAIL
AL = 1 assignment invalid.
AL = 2 invalid partition name or password
AL = 3 insufficient memory for memdisk.
AL = 4 mem disk already in use.
AL = 6 partition is not of requested OS type.

NOTE: even if an assign fails, the ownership of the previously assigned partition on that logical drive is released.

22. GETASS: Get information about the partition currently assigned to a logical drive.

Format of buffer:

BYTE		
0	-	logical drive number (set by caller)
1-8	-	partition name
9	-	partition size
10	-	partition number
11	-	control byte
12	-	volume number
13	-	drive type
14	-	write status
15	-	unused
16	-	unused

ON ENTRY: DX:CX is address of buffer.

ON EXIT:

IF SUCCESS, Buffer is filled out with all current info. AL = 0

IF FAILURE, AL = 5 meaning drive number in request was illegal.

NOTE: A partition number of 0FFh means drive is unassigned.

ON PARTITION WRITE OWNERSHIP:
Whenever a Network partition is assigned, the control byte is checked. If the control byte indicates a write status of R/W, R/O or HIDOS, the write status is set to that value. Otherwise the write status is set to unknown. Whenever a memory disk is assigned, its write status byte is set to read/write.

On OS initialization, if any of user's default partition names have the high bit set on the first character, the OS attempts to gain ownership status for that drive through a Network ownership request. If ownership is denied, a message is given to user warning them that they cannot write to that drive.

Whenever a write is attempted, the write status of the drive is checked. If it is not equal to the user number or read/write status or shared (when running HIDOS), the user is informed that he or she can't write to that partition, and a failure is returned. CP/M then warm boots, while MS-DOS will query what the user wants. No matter what the user says, he or she cannot write to the partition without an abort and an explicit write ownership request through ASSIGN.

Whenever an owned drive is re-assigned, ownership is automatically released by MAKASS. If the assign fails, the status of the drive remains unowned unless the assigning application makes a new explicit ownership request.

24. SETERR - turns on and off the display of error messages when there is a net error. Sets variable ERRFLG. Default setting is 0.

    ON ENRTY: CL has value to load ERRFLG with.
     0 - error messages off
    ~0 - error messages on
    ON EXIT: ERRFLG is set.

25. GETERR - get the cumulative error count. The error count is incremented whenever there is some kind of net error. Variable is called ERRNUM.

    ON ENTRY: nothing
    ON EXIT: AX contains number of errors.

26. SETTRY - sets variable TRYFLG, which says whether to automatically retry when there is a net error. Default setting is 0FFh.

    ON ENTRY: CL has value to set
    0 - don't retry, ~0 - do retry.

    ON EXIT: TRYFLG is set.

27. WHO - gets the who info from the Master.

    ON ENTRY: DX:CX is address of table to load.
    ON EXIT: AL has 0 for success, 1 for failure.

    IF success the table is filled with WHO info.

    FORMAT of WHO INFORMATION:

    1 byte - max number of users on system (32 or 64)
    16 byte entries * max users - USER ENTRIES
    256 bytes (16 bytes * 16 entries) - SPOOL ENTRIES

    FORMAT of User Entry:

    1 byte - Activity count. 0FFh - active
    0 - dead
    other - logging out

8 bytes - User Name.
3 bytes - login time (secs,mins,hours)
3 bytes - last request time (secs,mins,hours)
1 byte - last request command byte.

FORMAT of Spool Entry

1 byte - spool status
- 0 - starting spool
- 1 - spooling
- 2 - ready to print
- 3 - printing
- 4 - stop print
- 5 - waiting
- 0E5h - aborted or done

1 byte - user number of User spooling.
1 byte - spool ID (top nibble job #, low nibble block #).
2 bytes - time job started (min,hour).
2 bytes - track # of last sector of job.
1 byte - sector # of last sector of job.
8 bytes - spooler's name.

28. DIRNET - gets DIRNET info from Master (reads ALLOC Table);

ON ENTRY: DX:CX has address of table to fill out. First byte is the volume number (0,1,2,3) of the Hard Disk to read.

ON EXIT: AL has 1 for failure, 0 for success.

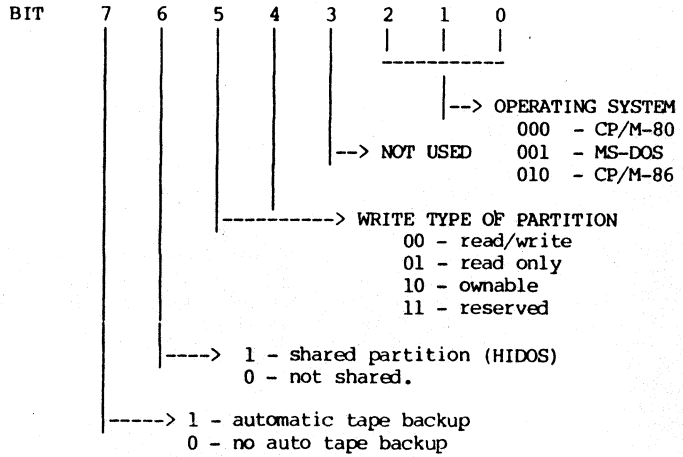
On success table is filled with a table with 64 entries. Total size = 1K.

FORMAT of DIRNET INFORMATION: (same as ALLOC Table)

1 byte - size (0 - end of table,1 - 256k,
2 - 512K, 3 - 1MEG, 4 - 2MEG,
5 - 4MEG etc. up to 32MEG).

8 bytes - Partition name.
6 bytes - Reserved
1 bytes - Control byte.

FORMAT OF CONTROL BYTE -



29. LOCK - used for adding strings to the Masters lock table.

ON ENTRY: DX:CX has addr of lock string.
ON EXIT: AH and AL tell status of request.
Format of lock string:

1 byte - length of string (max 13)
13 byte - string. String should be blank padded if it's less than 13 bytes.

Return status:

reg AL	reg AH	
00	00	- success
01	01h	- denied, owned by someone else
01	81h	- denied, already owned by you
01	83h	- denied, already owned by you, after a retry.
02	02h	- denied, illegal string length
02	82h	- denied, table full
02	03h	- denied, bad transmission (only appears when retry flag is not set).

----------NOTE----------
The retry information is given to warn user that the lock may have been successful but the confirmation was missed because of Network noise. The lock is automatically tried again. However, since the first lock attempt may have successfully locked the string, the status says it's locked. It is impossible to tell if it was locked on the first try or at some previous time.
---

30. UNLOCK - release a locked string in Masters lock table.

ON ENTRY: DX:CX has addr of string to release. Format is the same as in LOCK (above).

ON EXIT: AX has status of request, as follows:

reg AL	reg AH	
00	00	- success
01	01h	- denied, locked by someone else
02	02h	- denied, illegal string length
02	82h	- denied, string not found in table
02	83h	- denied, string not found in table after a retry.
02	03h	- denied, bad transmission (only appears when retry flag is unset).

----------NOTE----------
Here, as in lock, a garbled response from the Master can lead to a retry, and then a spurious deny. This flag warns the user so he or she can use his or her best judgement as to how to interpret this.
---

31. COERCE - changes NETPAR to 0FFh, causing a flush of disk buffers on the next read or write.

    ON ENTRY: nothing
    ON EXIT: NETPAR is 0FFh.

32. SETPPA - Sets pointers in BIOS to users poll prime command.

    ON ENTRY: DX:CX address of users command block. If DX = 0 pointers are reset to warm boot values.

    ON EXIT: EX:BX has address of old command block.

    Format of Poll-Prime Command Block is:

1 byte - Status byte:
00h - station naks Poll-Primes.
01h - station will ACK Poll-Primes.
11h - station will ACK Poll-Primes and then pass control to user routine addressed in this block. User routine returns in AL this routines response to the poller.
02h (internal) -

2 bytes - reserved
129 bytes - Poll-Prime Data
2 bytes - offset of user routine
2 bytes - segment of user routine

33. HDSTAT - gets the hdstat info from the Master.

ON ENTRY: DX:CX is address of table to load.
ON EXIT: AL has 0 for success, 1 for failure.

IF success the table is filled with HDSTAT information as follows.

HDSTAT info format:

1 byte - 18h (echoes command for info.)
1 byte - ROM version.
1 byte - ROM revision.
1 byte - firmware version number.
1 byte - firmware revision number.
2 bytes - unsed
1 byte - status. 0 - disk present, info OK
~0 - no disk or other error, rest of information invalid.
4 32 byte volume entries.

Format of Volume Entry -

1 byte	- status.	0FFh - volume present. 0 - volume not present.
1 byte	- current track for volume.	
1 byte	- number of tracks on volume.	
1 byte	- sectors per track.	
1 byte	- head mask.	
2 bytes	- location of partition offset table.	
2 bytes	- location of bad sector table.	
2 bytes	- location of bad sector dirty flag.	
1 byte	- error in volume open during firmware init (0 = no error).	
10 bytes	- Volume Label.	
10 bytes	- unused.	

34. TIMDAT - gets the time and date info from the Master.

ON ENTRY: DX:CX is address of buffer to load.
ON EXIT: AL has 0 for success, 1 for failure.

IF success the buffer is filled with time and date info. Format of time date info:
One byte each for:
ticks, secs, mins, hours, month, day ,year

62 ticks per second.

35. SETPOL - this module puts the BIOS in automatic poll ACKing mode. This means that it will answer Network polls with an ACK and the user should be unaffected, except for time lost in interrupts. SETPOL also enables NMI, in case it was disabled by CLRPOL. This routine also polls the keyboard to make sure that too long a time has not elapsed while NMI was disabled.

ON ENTRY: nothing
ON EXIT: SIO is reprogrammed, NMI is enabled on DMS-816 and polls are ACKed.

36. CLRPOL - disables NMI on the DMS-816 and waits for a Network poll. If a poll is received, a success is returned. If no poll comes for 10 milliseconds it calls SETPOL which enables NMI and polls the keyboard. The routine starts again, looping till approximately 5 second have passed. The routine then displays a waiting message and returns a failure to user. The routine will have performed a SETPOL in this case. If an invalid interrupt is received, SETPOL is not called.

ON ENTRY: nothing.

ON EXIT: IF success - carry flag is reset if we have just recieved a poll and not acked. NMI is disabled. Automatic acking is disabled. IF failure - carry flag is set, no poll was received. AL contains RECSTAT. NMI may or may not be enabled. Auto ACK mode may or may not be on.

Bit Format of RECSTAT: (7 is high bit)
7 - reset if timeout.
6 - set on CRC error.
5 - set if overrun.
4 - undefined.
3 - set if underrun.
2 - set on error.
1 - undefined.
0 - reset on error.

37. SNDNET - send data to the Master. Data can only be sent after a successful CLRPOL so that the Master is waiting for data with the right User Number. Data must be sent immediately after the CLRPOL or else the Master will time out waiting to poll the next station.

    ON ENTRY: DX:CX points to send command buffer.
    ON EXIT: AX = 0, carry flag reset.

    NOTE: this routine assumes the send was successful since there is no way for station to tell if Master got the data.

    Format of send buffer:
    1 byte - Network user number.
    2 bytes - offset of data buffer to send.
    2 bytes - segment of data buffer to send.
    2 bytes - length of data buffer to send.

38. RCVNET - receive data from the net. This is generally used after a CLRPOL and a SNDNET have made a request from the Master.

    ON ENTRY: DX:CX points to RECEIVE request buffer.
    ON EXIT: IF success - carry flag is reset.
    IF failure - carry flag set.
    AL has RECSTAT.

    For format of RECSTAT see CLRPOL.

    Format of Receive command buffer:

1 byte - Network user number.
2 bytes - offset of data buffer to receive.
2 bytes - segment of data buffer to receive.
2 bytes - length of data buffer to receive.

39. VIDEO (816 ONLY) - this call services all the DMS-816 video output needs. Entries and exits are determined by the function called.

FUNCTIONS are based on value of AH:

0 - Set mode - only valid mode B&W 80 x 25
1 - Set cursor type.
2 - Set cursor pos
3 - Read cursor position
4 - Read Light Pen Position (not implemented)
5 - Select Active Page (not implemented)
6 - Scroll active page up
7 - Scroll active page down
8 - Read char/attribute
9 - Write char/attribute
10 - Write char only
11 - Set color palette (not implemented)
12 - write dot (not implemented)
13 - read dot (not implemented)
14 - write char in tty mode
15 - return video status

VIDEO - SET MODE initialzes video controller.
ON ENTRY: AH - 0
ON EXIT: screen is blank. Cursor is a 0,0.

VIDEO - SET CURSOR DISPLAY
Sets cursor display type, top and bottom cell line to display or cursor.

On Entry: AH - 1
CH - start line
CL - end line
ON EXIT: nothing

VIDEO - SET CURSOR POSITION
On entry: AH - 2
DX - row,col of cursor
ON EXIT: cursor is moved to new position.

VIDEO - GET CURSOR POSITION to cursor postion
On entry: AH - 3
On exit : DX - row and col of cursor.
CX - cursor mode.

VIDEO - SCROLL UP SCREEN
Scrolls the defined window the defined number of lines.

On entry: AH - 6
AL - numbers of rows to scroll (0 for blank the area)
CX - row/col of upper left corner
DX - row col of lower right corner
BH - attribute of blanked lines

VIDEO - SCROLL UP DOWN

Scrolls the defined window the defined number of lines.

On entry: AH - 7
AL - numbers of rows to scroll (0 for blank the area)
CX - row/col of upper left corner
DX - row col of lower right corner
BH - attribute of blanked lines

VIDEO - READ CHAR and ATTRIBUTE

Reads the character at the current cursor position of screen.

On Entry: AH - 8
On Exit: AL - char
AH - attr

VIDEO - WRITE CHAR and ATTRIBUTE

Writes chars and attributes to screen at current cursor position.

On Entry: AH - 9
CX - count of chars to write
AL - char
BL - attribute
On Exit: Cursor position is not changed.

VIDEO - WRITE CHAR ONLY (no attribute)

Writes characters to screen at current cursor position, not changing the attributes.

On Entry: AH - 10
CX - count of chars to write
AL - char
On Exit: Cursor position is not changed.

VIDEO - WRITE CHAR IN TTY MODE - outputs a single character with whatever is the current attribute (CURATTR), and updates the cursor. It interprets these characters - CR, BS, LF and BELL. If cursor gets to end of screen the screen

automatically scrolls. Default screen length is 25 lines, but this can be changed to 24 using SPECIAL CALLS.

On Entry: AH - 11
AL - char to output
On Exit: nothing

VIDEO - RETURN VIDEO STATUS
ON ENTRY: AH - 12
On Exit: AH - cols on screen (always 80)
AL - current video mode (always 7)
BH - current active page (always 0)

40. PRINTSCREEN (816 ONLY) - sends screen image to printer. Checks flag at 50:0 to see if a screen print is already in progress (1 = in progress). If so it does not initiate another screen print.

ON ENTRY: nothing.
ON EXIT: If success 50:0 hold 0, else 50:0 holds 0FFh.

41. DMSINFO - gets the DMS INFO table into user buffer.

ON ENTRY: DX:CX is address of buffer to load.
ON EXIT: The buffer is filled with information as follows:

FORMAT of DMS INFO table:
1 byte - product type:
1 - ZSBC3 machines. (5080,3/B,3/F,501,3/10X)
2 - DMS-4
3 - DMS-1280
4 - DMS-3C
5 - DMS-86,5086
6 - 5016
7 - 816
8 - PC card

1 byte - OS type 11h - CP/M-80 (2.x)
12h - CP/M-86
13h - HIDOS
21h - MS-DOS (2.x)

4 bytes - Gutbios Version number
(vers,rev,assem,pass)
4 bytes - Prom serial number

6 bytes - Option Map:

BIT	DEVICE DRIVER
0	8-inch Floppy Disks (SD and DMS DD)
1	5-inch DS, DD Floppy Disks
2	5-inch IBM format Floppy Disks
3	8-inch Hard Disk with DMS controller
4	5-inch Hard Disk with Xebec controller
5	5-inch Hard Disk with Adaptec controller
6	Port 0 type-ahead (console)
7	Port 0 polled (console)
8	Port 2 polled (printer)
9	Port 3 type-ahead (aux. comm.)
10	Port 3 polled (aux. comm.)
11	Parallel Port 1 (console)
12	Parallel Port 1 (printer)

13 Parallel Port 2 (DMS-3/F printer)
14 Console/Printer Mux (ADDS, 1280, 5000)
15 Spooler
16 Net Buffer (1K)
17 Real-Time Clock
18 Front-panel Interrupt
19 Number of logical drives (bit 0)
20 Number of logical drives (bit 1)
21 Number of logical drives (bit 2)
22 Number of logical drives (bit 3)
23 Memory Mapped console (DMS-816)
remaining 25 bits unused.

1 byte - Password mode
0FFh - not auto mode.
0 thru 9 - auto mode
with this PW #.

2 bytes - Segment address of this table.

2 bytes - Offset addres of this table

1 byte - Local Network user number.

4 bytes - Mother BIOS version number
vers,rev,mod,change

42. WRSCOMMAND - this is used to make write status requests from the BIOS and the Master.

ON ENTRY: DX:CX points to WRITE STATUS COMMAND BUFFER.

ON EXIT: AL and response status field of COMMAND BUFFER are loaded with Network response.

Success - AL has 04Dh (mesask)
Command buffer is filled out
Failure - other value (currently 04Fh - cmddeny)

----------NOTE----------
Success indicates a successful request dialogue, not that the request was granted. You must check Info in the command buffer for that. Failure is generallly due to invalid parameters in the request (e.g., the partition number doesn't match the partition number in the BIOS partition table for the logical drive number given.)

Format of command buffer:

1 byte - Request type 0 - grant ownership
1 - release onwership
2 - force partition to a state
3 - query currrent status
4 - release all this users partitions
1 byte - volume number of partition.
1 byte - unit number of partition.
1 byte - value.
1 byte - physical user
1 byte - logical drive (not sent to Master)
1 byte - response status.
1 byte - Unit status.
8 bytes - User name.

WRScommand - entry and exit parameters for each type of request.

GRANT and QUERY request -

ON ENTRY: User must fill out:
Request type - 0 for grant,3 for query.
Volume number of partition.
Unit number of partition.
Logical drive of partition --
(A-0,B-1 etc.)

ON EXIT: If request dialogue was successful
(response status = 4D):
Unit status has status of drive.
User name has ASCII name of drive
owner, if there is one.

Unit status values:
0FFh - ownable partiton but not owned.
0FEh - read/write partition.
0FDh - read only partition.
0FCh - HIDOS partition. (Can be written to only by HIDOS but can be read by CP/M-86).
0FBh - multi. This status is returned on a grant request for a drive that is already owned by your user number. It may indicate that you own it on another drive.
0FAh - illegal request.
other - Network user number of current owner of patition.

RELEASE request -
ON ENTRY: User must have filled out:
Request type - 2.
Volume number of partition.
Unit number of partition.
Logical drive of partition
(A-0,B-1 etc.)

ON EXIT: If request dialogue was successful

(response status = 4D):
Unit status has status of drive.
User name has ascii name of drive owner, if there is one.

----------NOTE----------
An attempt to release a partition you don't own returns a failure in response status.

RELEASE ALL request -

ON ENTRY: User must have filled out:
Request type - 4.

ON EXIT: If request dialogue was successful (response status = 4D):
Write ownership is released an all your partitions.

FORCE request -
This forces the status of the partition named. You can force the write status on any drive, whether you are assigned to it or not. If you are not assigned to it, set logical drive to 0FFh.

ON ENTRY: User must have filled out:
Request type - 1.
Volume number of partition.
Unit number of partition.
Logical drive of partition A-0,B-1 etc. if you are assigned to it. 0FFh if not assigned.

ON EXIT: If request dialogue was successful (response status = 4D):
Unit status has status of drive.
User name has ASCII name of drive owner, if there is one.

43. AUXINIT - FOR 816 ONLY. A service routine for the virtual IBM ROM BIOS. It sets parameters for SIO channel A.

ON ENTRY: AL parameter byte.
ON EXIT : paramters are set.

FORMAT of parameter byte:

```
bit  7  6  5  4  3  2  1  0
     x  x  x  p  p  s  l  l
```

x bits - don't care (IN IBM used for baud rate which can only be set on 816 with dip switches).
p bits - parity setting
- x0 - no parity
- 01 - odd parity
- 11 - even parity

s bit - stop bits
- 0 - 1 stop bit
- 1 - 2 stop bits

l bits - character length
- 10 - 7 bits
- 11 - 8 bits

44. PARTADDR - return address of the 86 BIOS Partition Table.

ON ENTRY: DX:CX points to buffer to load with address of Partition Table.

ON EXIT : Buffer has offset and segment of partition table loaded into it. Format of address is: low byte offset, high offset, low segment, high segment

Partition Table has 8 entries corresponding to Drive A-H. The format is as follows:

```
8 bytes - partition name
1 byte  - partition size
1 byte  - unit number
1 byte  - volume number
1 byte  - drive type
1 byte  - write status of drive
1 byte  - logical drive number (A-0,B-1, etc.)
1 byte  - media same flag (for MSDOS)
```

For values of parition size see DIRNET.
For values of drive type see MAKASS.
For values of Write Status see WRScommand.

45. NETINFOREQ - Gets Network info into a user buffer.

```
ON ENTRY: DX:CX points to buffer to load.
ON EXIT : IF failure - AL = 1
          IF success - AL = 0
                     -  buffer is loaded with
                        net info.
```

FORMAT of NETINFO:

```
    1 byte  - reserved
    1 byte  - max number of users on system
    3 bytes - reserved
    1 byte  - numer of entries in spool table
    2       119 bytes - maximum number of lockstrings
  119 bytes - unused
```

46. SETPRESC - This sets the escape codes used for output to a printer attached to console. When outputing to such a device up to a 2 byte escape code is sent before and another after each char is output. This tells the console to pass the character to the printer. If a byte in an escape sequence is 0 it marks the end of that sequence.

    ON ENTRY: DX:CX points to a buffer with the two sequences in it.

    ON EXIT: The sequences are loaded into the BIOS sequence variables.

    FORMAT of sequence buffer:

```
    byte one of first sequence
    byte two of first sequence
    byte one of second sequence
    byte two of second sequence
```

47. SPOOLMESS -- this call allows turning on and off console messages from the spool device. The default setting is messages on. Messages should be turned off, for example, when you are running a background spooler and you do not want messages displayed in the middle of another application.

ON ENTRY: CL has setting
0 - turn messages off.
~0 - turn message on.
ON EXIT: message display flag (spMessOn) is set.

48. PPSEND - (HiNet Release 6.2) this call sends a Poll-Prime to a specified station. It incorporates many of the individual commands that were previously required to send a Poll-Prime.

ON ENTRY: DX:CX - buffer
1 byte - target User Number
129 bytes - Poll-Prime data

ON EXIT: AL = 0 data sent
AL = 1 no poll from master
AL = 2 could not Hog Network
AL = 3 target station not accepting message
AL = 4 transmission error

50. COLD BOOT -- releases all owned partitions and jumps to PROM cold boot routine to rejoin the Network.

4.0 DMS-816 IBM ROM BIOS EMULATION

4.1 INTRODUCTION

Part of the Hardware specific BIOS for IBM PCs is programmed into a ROM chip. DMS's emulation of the IBM ROM BIOS for the DMS-816 is not in hardware but in the MACHINE BIOS. This section details the various interrupt calls that are available on the DMS-816. Programmers should always use these Interrupts instead of specific addresses in the Hardware or BIOS. DMS cannot guarantee that Hardware or BIOS addresses will remain constant with future enhancements.

4.2 PRINT SCREEN

INTERRUPT = 5 HEX

This interrupt prints the screen display through the assigned port (parallel, serial, Network spooler). The cursor's position is saved and restored upon completion. Interrupts can be left enabled when this routine is called. Results of the Print Screen routine are contained at address **50:0.**

If 50:0 = 0 it indicates that either Print Screen has not been called or the routine has been successfully completed.

If 50:0 = 1, Print screen is in progress.

If 50:0 = 255 (0FFH), an error was encountered during the routine.

4.3 TIMER INTERRUPT

INTERRUPT = 8H
ADDRESS = 20-23 Hex

INT 8 can be used by a subroutine by placing the address of the subroutine at the address 20-23 Hex. Whenever a clock tick occurs and INT 8 is called, the BIOS will jump to the subroutine. Be sure to save the current vector address before initiating the interrupt and restore the vector address upon return. Use IRET at the end of the subroutine to return to the jump point.

----------NOTE----------
On the DMS-816, clock ticks are not reported uniformly. Some ticks may be skipped while other functions are being executed. (Ticks should come about 18.2 times a second.)

4.4 TIME OF DAY

INTERRUPT = 1AH

Set register AH = 0 to read the current clock setting. Returns:

HIGH PORTION OF COUNT = CX
LOW PORTION OF COUNT = DX

AL = 0 if timer has not passed 24 hours since last count.

AL = ~0 if 24 hours have passed since last request.

Set register AH = 1 to set the current clock.

CX = HIGH PORTION OF COUNT
DX = LOW PORTION OF COUNT

----------NOTE----------
When working on the Z-80 side of the DMS-816, time of day is stored at address 40H. Every two minutes the BIOS gets the time from the Master. Therefore, if SETTIME is used to set a different time than what is on the Master, the local time will be reset to Network time every two minutes.

4.5 DMS-816 VIDEO ROUTINES

INTERRUPT = 10 Hex
ADDRESS = 40-43 Hex

SET CURSOR TYPE **AH = 1**
CH bits 0-4 = start line for cursor.
CH bits 5-7 = 0
CL bits 0-4 = end line for cursor.
CL bits 5-7 = 0

SET CURSOR POSITION **AH = 2**
DH,DL = Row,column (0,0) is upper left.

READ CUSOR POSITION **AH = 3**
ON EXIT: DH,DL = Row, Column of current setting.
CH,CL = cursor mode currently set.

SCROLL SCREEN UP **AH = 6**
AL = number of lines, input lines blanked at bottom of screen.

AL = 0 blank entire screen.

CH,CL = Row,Column of upper left corner of scroll.

DH,DL = Row,Column of lower right corner of screen.

BH = attribute to be used on blank line.

SCROLL SCREEN DOWN **AH = 7**

AL = number of lines, input lines blanked at top of screen.

AL = 0 blank entire screen.

CH,CL = Row,Column of upper left corner of scroll.

DH,DL = Row,Column of lower right corner of screen.

BH = attribute to be used on blank line.

READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION **AH = 8**

ON EXIT: AL = character read.
AH = character attribute.

WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION **AH = 9**

CX = count of characters to write.
AL = character to write
BL = attribute of character to write.

WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION AH = 10

CX = count of characters to write.
AL = character to write.

VIDEO ATTRIBUTE BYTE:

```
                          BITS
                    7  6  5  4  3  2  1  0

NORMAL              B  0  0  0  I  0  0  0
REVERSE             B  1  1  1  I  0  0  0
NON-DISPLAY BLACK   B  0  0  0  I  0  0  0
NON-DISPLAY WHITE   B  1  1  1  I  1  1  1
                    |           |
                    |           |
                    |        I = 0 NORMAL INTENSITY
                    |        I = 1 HIGH INTENSITY
                    |
                 B = 0 NON-BLINKING
                 B = 1 BLINKING FOREGROUND
```

4.6 EQUIPMENT CHECK

INTERRUPT = 11 Hex
ADDRESS = 44-47 Hex

This interrupt lets you check to see what equipment is attached to the DMS-816. Register AX is set to indicate bit map of equipment:

```
AX = 0100 0010 0011 1100 for DMS-816.
     |                   |
     |                   |
  bit 15              bit 0
```

The bit map of AX is as follows:

BIT	EQUIPMENT CHECK (2 BYTES IN AX)
15, 14	Number of printers attached, always 1 for DMS-816.
13	Not used
12	Game I/O attached, none for DMS-816.
11,10,9	Number RS232 Cards attached, always 1 for DMS-816.
8	Not used.
7,6	Number of diskette drives, none for DMS-816.
5,4	Video mode, always 11 to indicate B&W 80 X 25 display for DMS-816.
3,2	PLANAR RAM size = 64K for DMS-816.
1	Not used.
0	Not used for DMS-816.

4.7 MEMORY SIZE DETERMINATION

INTERRUPT = 12 Hex

This routine determines how much RAM is installed on the DMS-816's Mother board. Memory is reported in register **AX** in 1K blocks.

4.8 DISKETTE I/O

INTERRUPT = 13 Hex

At this time there are no 48 TPI PC-compatible Floppy Diskette Drives for the HiNet Network. However, Interrupt 13 can still be used to read and write files but the data is directed to the Network's Hard Disk partitions. DMS recommends that only BDOS calls be used to read and write files and to avoid the use of Interrupt 13 if possible.

INPUT:

AH = 0, Reset diskette system; Hard reset to NEC controller chip, recal required on all drives.

AH = 1 Read the status of the system into register AL. Diskette status from last operation is used.

Registers for Read/Write:

DL -- drive number (0-3 allowed value checked).

DH -- head number -- not supported by DMS.

CH -- Track number (0-39 HEX, not value checked).

CL -- Sector number (1-8, not value checked).

AL -- number of sectors (max = 8, not value checked).

ES:BX -- address of buffer.

AH = 2 -- Read the desired seectors into memory.

AH = 3 -- Write the desired sectors from memory.

Note--AH = 4, verify desired sectors and AH = 5, from desired track, are not supported at this time by DMS.

4.9 COMMUNICATIONS (RS-232)

INTERRUPT = 14 Hex

This routine provides I/O to the RS-232 Port (DMS-816 PORT2).

AH = 0 initialize port.
AL has initialization parameters:

7 6 5	4 3	2	1 0
BAUD RATE	PARITY	STOP BIT	WORD LENGTH
*NA	x0 = NONE	0 = 1	10 = 7 BITS
	01 = ODD	1 = 2	11 = 8 BITS
	11 = EVEN		

*NOTE---BAUD is set with DIP switches on DMS-816.

AH = 1 - send character in AL through RS-232
ON EXIT: return status of port (see below)

AH = 2 -- receive character through RS-232
ON EXIT: AH - line status, non-zero
only on error.
AL - has character received.

AH = 3 -- return complete status (both AH and AL)

STATUS BYTE IN AL:

BIT	
4	CTS
5	DSR
6	RING INDICATE (SYNC)
7	RECEIVE LINE DETECT (DCD)

STATUS BYTE IN AH:

BIT	
0	RECEIVE DATA READY
1	OVERRUN
2	PARITY ERROR
3	FRAMING ERROR
4	BREAK
5	READY TO TRANSMIT
6	READY TO TRANSMIT
7	TIMEOUT

AH = 80 -- turn off auto enable, use DCD for receive.

AL = 0 -- auto enable off
~0 -- auto enable on

4.10 KEYBOARD

INTERRUPT = 16 Hex

Routines for reading input from the keyboard.

INPUT:

AH = 0 Read the next character from the keyboard. Return the result in register AL and the scan code in register AH.

AH = 1 Set the Z flag to indicate that another character is available to be read from the keyboard.

If ZF = 1 there is no character available.

If ZF = 0 the next character in the buffer to be read is in AX; the character remains in the buffer.

AH = 2 Return the current shift status in register AL. (See note on KB_FLAG.)

OUTPUT:
Same as Input except AX and Flags change. All registers are preserved.

KB_FLAG status byte is set if AH = 2. The upper four bits of the staus byte tell which keyboard modes are on (1) and which are off (0). The lower four bits tell if the ALT, CTRL or SHIFT keys are pressed. An additional byte (KB_FLAG_1) gives more keyboard information. However,

KB_FLAG_1 is not placed in a register; it is used internally by the KEYBOARD_IO routine. To find the contents of KB_FLAG1 look at the contents of address 418H. (KB_FLAG is at 417H.)

Bit = 1	KB_FLAG	KB_FLAG1
0	RIGHT SHIFT KEY	
1	LEFT SHIFT KEY	
2	CTRL	
3	ALT	CTRL NUM LOCK IS ON
4	SCROLL LOCK IS ON	SCROLL LOCK KEY PRESSED
5	NUM LOCK IS ON	NUM LOCK KEY PRESSED
6	CAPS LOCK IS ON	CAPS LOCK KEY PRESSED
7	INSERT MODE ON	INSERT KEY PRESSED

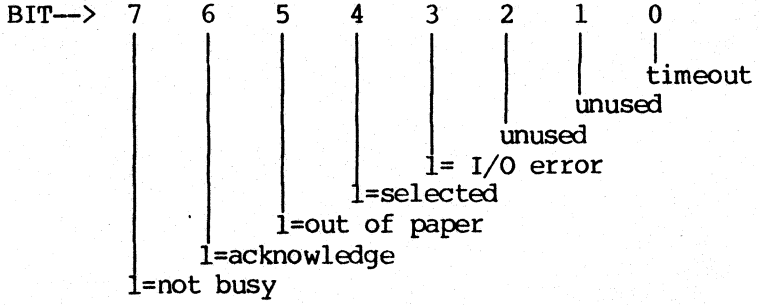
4.11 PRINTER

INTERRUPT = 17 Hex
ADDRESS = 5C-5F Hex

To communicate with the printer use INT 17.

INPUT:
AH = 0 print the character in register AL. On return AH = 1 if character could not be printed (timed out). The other bits are set as on any normal status call.
AH = 1 initialize the printer port.
AH = 2 read the printer status into AH.

STATUS BYTE FOR PRINTER



DX = printer to be used--0,1,2 corresponds to values in printer base area.

4.12 **BOOTSTRAP** (COLD BOOT)

INTERRUPT = 19 Hex
ADDRESS = 64-67 Hex

Initiates a cold boot, logs user off Network, workstation attempts to rejoin Network.

4.13 DMS 816 I/O ADDRESSING

PORT	TYPE	ADDR RANGE	ADDR BITS 76543210	IBM
CRT CARD:				
1) BW6845CS/		0B0-0B7	10110XXX	6845CS/ (Mono)
a) Index reg	O	0B4	10110100	Index reg
b) Data reg	I/O	0B5	10110101	Data reg
COMMUNICATION PORTS:				
2) SIOCS/		098-09F	10011XXX	Not used
a) RS232 control (A)	O	09A		
b) RS232 data (A)	I/O	098		
c) HINET control (B)	O	09B		
d) HINET data (B)	I/O	099		
KEYBOARD PORTS:				
3) KI/STAT	I (bit 0)	0B8-0BF	10111XXX	CRT Control &
a) PPSIN (parallel port status in)	I	0B8		Status Port
b) PPSOUT (parallel port status out)	I	0B9		
c) HS (horizontal sync)	I	0BA		
d) KBDIN (keyboard input)	I	0BB		

PORT	TYPE	ADDR RANGE	ADDR BITS 76543210	IBM
e) TEST (formerly TOUT)	I	0BC		
f) PERR (parity error)	I	0BE		
g) CPBUSY (Centronics port busy)	I	0BF		
h) spare (jumpered to 1 or 0)		0BD		
---				
4) NMIRST/ (NMI reset)	STROBE	090-097	10010XXX	Not used
5) PPORTI/ (parallel port input)	I	0F0-0F7	11110XXX	Diskette
6) Z-80SWO/ (Z-80 switchover)	STROBE	088-08F	10001XXX	SDLC Comm
7) PPORTO/ (parallel port output)	O	0A8-0AF	10101XXX	Reserved
8) CNTRL/ (All control bits are set to zero on reset)	O (bit 0)	0E8-0EF	11101XXX	Not used
a) FRVV (full reverse video)	O	0E8		
b) KBDOUT (keyboard output)	O	0E9		

	PORT	TYPE	ADDR RANGE	ADDR BITS 76543210	IBM
c)	DISVID*	disable=0 enable =1 (disable video)	0EA		
d)	BELL	1 then 0	0EB		
e)	BLINK	alternate (video blink)	0EC		
f)	KBDINT/	0 then 1 (keyboard interrupt)	0ED		
g)	Z-80INT/	0 (Z-80 interrupt)	0EE		
h)	ENNMI	disable=0 enable =1 (enable NMI)	0EF		
9)	SIORETI/	STROBE (SIO return from interrupt)	0F8-0FF	11111XXX	Asynch Comm (Primary)
10)	PARRST/	STROBE (parity reset)	0C0-0C7	11000XXX	
11)	C6845CS/		0D0-0D7	11010XXX	6845CS (Color)
12)	CPORTO/	0 (Centronics port output)	080-087	10000XXX	
13)	CPORTS/	STROBE (Centronics port strobe)	0D8-0DF	11011XXX	

*FRVV must be set to a 0 when DISVID = 0

4.14 DMS-816 Z-80 I/O OPERATIONS

Turning control over to the 8088 is done by executing an OUT to any port. An IN instruction should never be executed by the Z-80 since it could cause a false parity error.

4.15 DMS-816 CENTRONICS INTERFACE

Data is sent to the Centronics port by issuing an OUT to port 80h (CPORTO/) with the data in the AL register and then issuing an OUT to port D8h (CPORTS/) which sends a strobe pulse to the port. The strobe pulse must be on for at least 1 microsecond. Before another byte of data can be sent, a "0" must be read from port BFh (CPBUSY).

4.16 PROGRAMMING FUNCTION KEYS

The 93 available levels of function keys on DMS-816 and DMS-5000 can be loaded from both the keyboard (by the operator) and from the Host via a program. This feature can be invaluable for customizing the terminal to specialized application programs.

Across the top of the keyboard are sixteen function keys. Each are programmable with up to three separate strings of variable length. In addition, the ten numeric keys, the decimal point, +, - and * keys on the numeric/cursor-control keypad and the ALT keys in the main key group, are all programmable. The four arrow keys next to the Space Bar always have the same

values that the arrow keys in the keypad (shifted 2,4,6 and 8). Each key may hold three separate values, one for the key alone, one for the key with the SHIFT key held down, and one for the key with the CTRL key held down. This gives you 93 programmable keys in all.

To reprogram the function keys the following code is used:

ESC 1
function key number
length of string
string

FUNCTION KEY NUMBER -- "Function key number" is the one-byte binary identifier of the function key you want to program. To calculate the function key number of a function key, add the hex value of the key (e.g., F1=1H, F16=10H) to 80H. For example, the function key number for F17 is:

17=11H 11H + 80H = 91H F17=91H.

To program the SHIFTED value of a function key, set bit 5 in its function-key number byte to one. To program the CONTROL value of a key, set bit 6 in the byte to one. See Diagram 4-1 for the byte structure of the function key numbers.

To find the function key number for the shifted value of a function key, add A0H to the key number. For example:

SHIFT/F6 = 6 + A0H = A6H

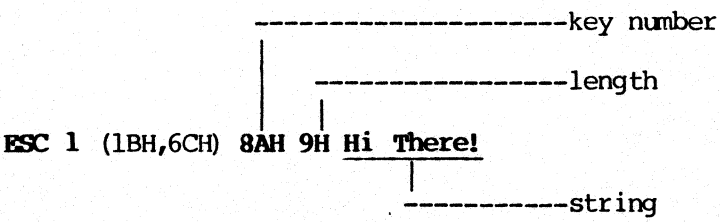
To find the function key number for a CTRL/Function key, add C0H to the key number. For example:

CTRL/F6 = 6 + C0H = C6H

As another example, if you wanted to program key F10 to output the string:

Hi There!

the programming command would look like this:



In BASIC the program code would be:

```
10 PRINT CHR$(1BH);CHR$(6CH);CHR$(8AH);CHR$(9H);
20 PRINT Hi There!;
```

Strings can be between 1 and 125 bytes. The maximum amount of RAM storage available for programming the function keys is 1K bytes. Any entries over the 1K limit will not be accepted.

BYTE STRUCTURE FOR FUNCTION KEYS

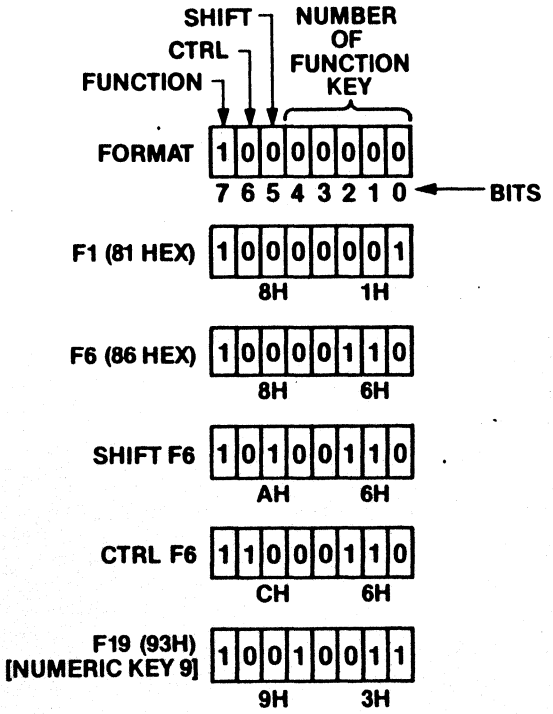


Diagram 4-1. Byte structure of the Function Keys and examples for calculating the Function Key Number:

- For all function keys, bit 7 is set to 1.
- Bit 6 is set to 1 only for the CTRL value of a key.
- Bit 5 is set to 1 only for the SHIFT value of a key.
- Bits 4 through 0 are for the hex value of the function key label.

FUNCTION KEY NUMBER REPORT -- Normally, when a function key is pressed, the string programmed into it is sent to the host CPU. A mode can be entered in which only the function key number (in hex) will be reported to the host CPU when a function key is pressed. For example, F1 will send 81H and F6 will send 86H.

ESC p (1BH,70H) - Report function key number only.

ESC P (1BH,50H) - Report function key's programmed string.

FUNCTION KEY REPORT -- When the host CPU sends the sequence:

ESC g (1BH,67H) **keynum**

to the CRT controller, the controller sends back the length of the string that is programmed into the function key and then the string.

5.0 SHARING PARTITIONS UNDER HIDOS

To enable more than one person to use a CP/M disk at the same time, a method must be devised so that when one person makes a change to the directory or the Allocation Vector (AV), everyone will know.

The method DMS chose puts the AV on the disk instead of in the BIOS for shared partitions. Each user's OS knows (by virtue of a flag in the ALLOC table) that the drive is shared. Whenever the directory or the AV is to be updated, the user's OS locks the partition (via a HiNet BIOS lock command) so that it has sole write access to the drive. The operating system updates the AV and when finished, unlocks the partition so that another user can make changes. In this way blocks are not allocated to more than one file and the directory is always kept up to date. Any user can read when the drive is locked, but only the person who has locked the drive can write to it.

HIDOS allows more than one person to work in the same partition at the same time but NOT on the same file. When working on a file, CP/M keeps in local memory a copy of the directory entry (in the form of a File Control Block), and modifies this copy as changes are made to the file (changes meaning adding or erasing blocks). The changes are not reflected in the disk's directory until the file is closed, or a new extent is needed.

Since a local copy is kept by CP/M, the locking mechanism used above will not work. In

fact, it is extremely impractical for any distributed-processor CP/M network to take care of this situation on the OS level. It would, of course, be desirable for the OS to take care of everything so that existing software could run with no modifications.

5.1 USING HIDOS--LIMITS AND RESTRICTIONS

Important! Do not use these BIOS calls on a shared partition:

Home	SetDma
SelDisk	Read
SetTrk	Write
SetSec	SecTran

Only use BDOS calls for Reads and Writes!

It is permissible to use DMS extended BIOS calls with the exception of SendNet and RecNet. If you must use HiNet commands with SendNet or RecNet, be sure you understand how HIDOS and HiNet work.

Do **NOT** change DPB's since HIDOS has special entries in the DPB that are non-standard.

Do **NOT** allow more than one person to use or modify a file at the same time. The application program that manipulates a file can allow this if it uses HiNet record locking functions on records or files during read/modify/write routines. If the application program does not explicitly do locking then do not share files. Also, take care that no one is using a file in a

shared partition when that file is erased by another user.

It is a good idea to establish a method for identifying files so that those people working on the same shared partition will not confuse their files with someone else's. If it is desired that more than one person have access to the same files, use the NetLock/NetUnLock utilities or some other method to avoid more than one person working on the same file at the same time. Always use the filename for the lockstring, rather than a shared partition name.

In a shared partition, do not use any program that creates a temporary file of a fixed name. If two people are using such a program, the temporary files will get confused with disastrous results. For example, many compilers and word processors create temporary files of a fixed name. (WORDSTAR creates a temporary file called EDBACKUP.$$$ for every large file that is opened for editing or reading.) You will probably need to experiment or talk to the program manufacturer to be sure of this.

Directory information is volatile for shared disks. When a 'DIR' is done, remember that the information is instantly 'old' and may be incorrect. Someone else may have modified the directory information immediately after you asked to get it. Thus, files may disappear even though they were there for a 'DIR'. Therefore, you must make sure that your files are only used by you (you could use the NetLock/NetUnlock utilities).

Information on disk space usage may not be correct. To get the most up-to-the-minute information on how much space is left on the disk, do a warm boot first. Remember, between the time you warm boot and a program such as STAT checks the disk space usage, someone else working in that partition could have changed the value without your local computer knowing about it.

5.2 FILE AND RECORD LOCKING

The idea behind file and record locking is to allow more than one person the ability to access and modify the same data at the same time, with each person getting the most recent data. In order to assure that you always have the most recent data, you need to do a "read" knowing that no one else has accessed the data file or record with the intention of modifying it. The procedure is to get ownership of the right to update the data (LOCK the data in question), read it, modify it, write it back, and then release ownership so another person can gain ownership. Read access without locking could always be granted with the understanding that someone else may be currently modifying what you have read.

As an example, consider an airline reservation system. The operator does an unlocked read to check seating availability. When the customer agrees to an available seat, the operator does a lock, then a read and checks to make sure the seat is still available--since someone else may have taken it between the time he or she did the

unlocked read and the locked read. If the seat is still available, the operator reserves the seat by updating the record with that data, writing it back and unlocking the record.

If everyone only did locked reads, system performance would suffer greatly with people waiting for access to be granted for their locks before they could read or examine data. Such waiting is not necessary since most reads don't need to be locked.

5.3 RECORD LOCKING PROCEDURES

It is important to realize that you must lock before rereading any data that is to be modified before modification/writing since what you have read without locking may not be current. Someone else may be changing the data while you are examining it in the unlocked state.

You should develop a method for naming what needs to be locked. The file name is fine for file locking; for records, the filename and record number combined could be a good name. The HiNet locking mechanism locks a string of, at most, 13 bytes.

To update a record, follow these procedures: lock, or wait for the lock to be granted, read the record, update it, write it back, and unlock it. If the record does not exist (i.e., it is not yet there to read) skip the read step. Probably some initialization should be done to the record. The method of determining if the record is there or not is

application-dependent. For some applications all records can be allocated initially. For others, only file extension may be allowed so that all allocated records are contiguous.

To extend a file you need to know which record is the current end. A specific record (say the first) can hold a pointer to the end. In this case, lock the record with the pointer. Using a random write (or sequential, if appropriate) write the record after the last record. This becomes the new last record; update the pointer accordingly. Write the pointer record back to the disk. Close the file to ensure that the directory is updated. Unlock the record with the pointer.

----------NOTE----------
If you are using BDOS calls you do not have to reopen a file after closing it in order to reuse the file. From a high level language the file will have to be reopened.

5.4 DATA RECORD SIZE VS. CP/M RECORD SIZE

The logical record size equals the data record size and the application program record size. Complications can arise if the logical record size to be locked is not the same size as, or is not a multiple of, the CP/M record size (128 decimal, 80 hex bytes). It is highly recommended that the data record size be 128 bytes or an integer multiple of 128. The problem is that a CP/M record can contain parts of more than one logical record. Thus the logical record can be locked, but not the CP/M record. There-

fore, more than one person can have the CP/M record in CP/M memory, each thinking he or she has sole ownership to modify that record. When they write back the logical record, that part of the CP/M record corresponding to some other logical record will be set to what it was when the read was done, overwriting any changes someone else may have made.

If you decide that you want a logical record size which is not equal to an integral number of CP/M records, you must lock the CP/M records, i.e., use the CP/M record name(s) that are being used by more than one logical record. There will be one or two records to lock.

Let us consider these four aspects of the problem:

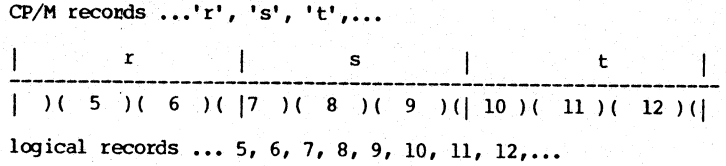
1. The data record is much smaller than the CP/M record.

2. The data record is slightly smaller than the CP/M record.

3. The data record is much larger than the CP/M record.

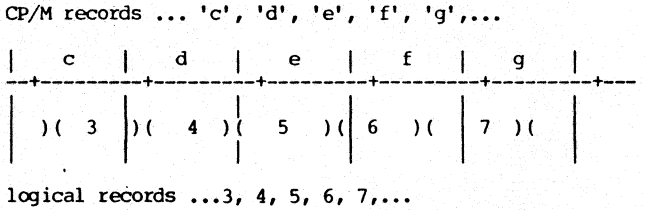
4. The data record is slightly larger than the CP/M record.

Example 1. The data record is much smaller than the CP/M record.



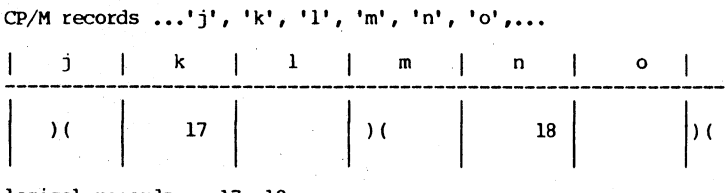
If logical record 6 is locked, read, changed, written back, and unlocked by user A, and at the same time User B locks, reads, changes and writes back logical record 5, the last one to write will overwrite the previous user's change. This occurs because the same CP/M record "r" is read and written each time.

Example 2. The data record is slightly smaller than the CP/M record.



The same problem exists as in #1. Notice this time that the logical record generally crosses a physical record boundary.

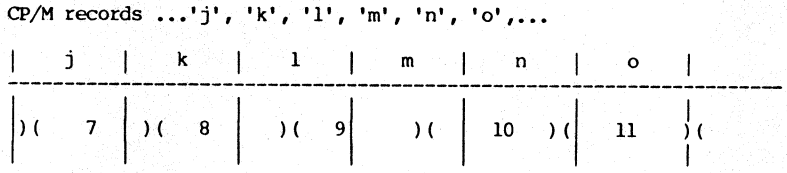
Example 3. The data record is much larger than the CP/M record.



logical records ...17, 18,...

In this case if user A works on logical record 17 and user B on logical record 18 the conflict arises in CP/M record 'm'.

Example 4. The data record is slightly larger than the CP/M record.



The same situation as #3 occurs here, only now almost all the CP/M records are shared by two logical records (except CP/M record 'o' which is totally contained in logical record 11, so no problem there).

5.5 CALCULATION OF CP/M RECORDS

Given a logical record we need to find the CP/M records that must be locked to avoid logical record conflict. There are one or two CP/M records in each of the four cases. The procedure is to find the CP/M records used by the first and last bytes of the logical record. We assume that the logical records are logically continuous and linearly numbered (i.e., records are numbered 2,3,4,5...).

To find the CP/M record used by the last byte of the logical record, first get the logical record number. If the first logical record is record "0" then add one to the logical record number. Now multiply this number by the logical record size and then divide by the CP/M record size (128 decimal). If there is a remainder, round up. The result is the last CP/M record used by the logical record.

Now, to find the CP/M record used by the beginning of the logical record, repeat the above procedure for the logical record just before the CP/M record. In this case, before dividing by the CP/M record length, add one so that the first byte of the logical record in question will be included.

These two records are the ones to lock. If they are the same record then only one record needs to be locked. If locking two CP/M records, watch out for lock-out. If you lock one record and the other is locked, unlock the first, wait a random amount of time and retry, since you may

be competing with someone else for the same records.

It is assumed that all CP/M records between the first and last CP/M records of the logical record do not need to be locked since anyone wanting to read them must also lock the ends. This assumes no overlap of logical records.

If the logical data file has something other than logical records (such as a file header or record headers) then the size of this must be taken into account.

EXAMPLES

1: Logical file name = DBASE1
logical record size = 136 bytes
logical records = 1,2,3,4,5,.....
(Note: first record=1)

no headers or inter record info.

Want to lock logical record 23.

(23 * 136) / 128 = 24.44 ---> 25
((22 * 136) + 1) / 128 = 23.38 ---> 24

So lock 24 and 25. Lockstrings could be DBASE24 and DBASE25.

2: Logical file name = DBASE1
logical record size = 136 bytes
logical records = 0,1,2,3,4,5,.....
(Note: first record=0)
No headers or inter record info.

Want to lock logical record 23.

```
    ( (23+1) * 136 ) / 128 = 25.5   ---> 26
( [(22+1) * 136] + 1 ) / 128 = 24.44  ---> 25
```

So lock 25 and 26. Lockstrings could be DBASE25 and DBASE26.

3: Logical file name = DBASE1
logical record size = 136 bytes
logical records = 1,2,3,4,5,....
(Note: first record=1)

Assume there is a 32-byte file header before logical record 1.

Want to lock logical record 75.

```
    [(75 * 136) + 32] / 128 = 79.9  --> 80
{[(74 * 136) + 32] + 1} / 128 = 78.88 --> 79
```

So lock 80 and 79. Lockstrings could be DBASE79 and DBASE80.

4: Logical file name = SMALLDATA
logical record size = 18 bytes
logical records = 1,2,3,4...
(Note: first record=1)

Assume no headers or inter-record data.

Want to lock logical record 345.

```
      (345 * 18) / 128 =   48.5 --> 49
{(344 * 18) + 1 } / 128 =   48.3 --> 49
```

So lock 49. Lockstring could be SMALLDATA49.

5.6 HINET BIOS LOCK AND UNLOCK

Record locking and unlocking are invoked by first constructing a "lockstring" and then calling the BIOS Function Number for Lock or Unlock. See Section 3.1.2 for information on using BIOS Function Calls.

The lockstring should indicate the file and record to be locked. Note that the lockstring can, in fact, contain any sequence of bytes. However, to allow different applications to utilize record locking on the same HiNet system requires that a convention be established. The recommended convention is to use the file name as the first 8 characters and the record number as the last 5 characters of the lock string.

----------NOTE----------
HiNet Release 6 has implemented additional return values for BIOS Lockstrings. These additions will not affect programs that use the older Lockstring routines. The new Lockstring returns documented here give additional information that help programs prevent "deadly embrace".

Before calling the BIOS Lock or Unlock Functions, locations 74 (4A hex) and 75 (4B hex) should point to the lockstring, i.e., contain the address of the string to be locked. The first byte of the string is an integer from 1 to 13, indicating the length of the string.

The BIOS routines return immediately and put the outcome of the request in location 73 (49 hex). This is the status of the request.

Z-80 RETURN STATUS:

Lock returns status in LockStat and register L>

LockStat = 00, L register = 00
- accepted, lock successful

LockStat = 01, L register = 01
- denied, string locked by someone else

LockStat = 02, L register = 02
- denied, illegal string length

LockStat = 01, L register = 81h
- denied, string already locked by you

LockStat = 02, L register = 82h
- denied, lock table is full

Unlock returns status in LockStat and register L>

LockStat = 00, L register = 00
- accepted, unlock successful

LockStat = 01, L register = 01
- denied, string was locked by someone else

LockStat = 02, L register = 02
- denied, illegal string length

LockStat = 02, L register = 82h
- denied, lock string not in table

ClearLock returns status in LockStat and register L>

LockStat = 00, L register = 00
- accepted, ClearLock successful (always)

The CBASIC functions "fn.lock" and "fn.unlock" can be used to interface with the lock and unlock routines in the BIOS. Similar interface functions can easily be written for other compilers.

```
-----------------------------------------------
DEF FN.LOCKWORK%(STRING$,FUNC%)
    ADDR% = SADD(STRING$)
    HIGH% = (ADDR%/100h) AND 0FFh
    IF ADDR% < 0 THEN HIGH% = HIGH% - 1
    POKE 4AH,ADDR% AND 0FFH
    POKE 4BH,HIGH%
    CALL ((PEEK(2)*100h) OR PEEK(1)) + FUNC%
    FN.LOCKWORK% = PEEK(49H)
RETURN
FEND
DEF FN.LOCK%(STRING$)
    FN.LOCK% = FN.LOCKWORK%(STRING$,5DH)
    RETURN
FEND
DEF FN.UNLOCK%(STRING$)
    FN.UNLOCK% = FN.LOCKWORK%(STRING$,63H)
    RETURN
FEND
-----------------------------------------------
```

The following program demonstrates how to use the record locking functions. First, a file containing 128 records is created. Several users can then simultaneously run this program, and update different records in the file at will. The program will allow only one user at a time to update any particular record; however, several users are allowed to update DIFFERENT records in the file simultaneously. The lock functions are on the "LOCKFNS.BAS" file.

The statement "READ #1,R;" is needed after a write to force CBASIC to flush its I/O buffer for file number 1. Without this statement, the record will not be updated on the disk until the next random read or write to that file. This is due to a peculiarity in the I/O algorithms used by CBASIC. Similar problems may be encountered with other compilers.

```
%INCLUDE LOCKFNS
    FILENAME$ = "DEMO.DAT"
    INPUT "ENTER 0 TO CREATE, 1 TO
                          UPDATE DEMO FILE";I
    IF I = 0 THEN \
       CREATE FILENAME$ RECL 128 AS 1 :\
       FOR I = 1 TO 128 :\
       PRINT #1;I :\
       NEXT I :\
       CLOSE 1
    OPEN FILENAME$ RECL 128 AS 1

100 INPUT "RECORD NUMBER";R
    LOCKSTRING$ = "DEMO     "+STR$(R)
    WHILE FN.LOCK%(LOCKSTRING$) <> 0
       WEND
    READ #1,R;I
    PRINT "OLD VALUE";I
    INPUT "NEW VALUE";I
    PRINT #1,R;I
    READ #1,R;      REM  flush the record
    I% = FN.UNLOCK%(LOCKSTRING$)
    GO TO 100
    END
```

5.7 8088/8086 RECORD LOCKING

This section describes the calls the can be made to the MACHINE BIOS on DMS 8086 and 8088 based machines. The SYSTEM BIOS for CP/M-80 (on DMS-816s), CP/M-86 and MS-DOS all use some of these calls. They can be called by a user through the use of INTERRUPT 78.

5.7.1 USING INTERRUPT 78 TO CALL MACHINE BIOS

For all functions except video functions under HIDOS, CP/M-86 and MS-DOS:

Put function number in AH.

Load all registers as Function you are calling expects them.

Call Interrupt 78 (4E Hex).

To hold the 13 byte lockstring plus 1 byte length, you must set up a buffer for the lockstring with the format:

1 byte - length of string,
13 bytes - lockstring; string should be blank padded, especially if it is less than 13 bytes.

In code this would be written as:

```
mov ah,29
mov cx,offset STRINGBUFFER
mov dx,ds
int 78
```

where the offset address of the STRINGBUFFER is loaded into register CX, and the data segment register is loaded into DX. When INT 78 is called it knows to look at AH, CX and DX for the appropriate information.

5.7.2 BIOS FUNCTION CALLS FOR RECORD LOCKING

MACHINE BIOS NUMBER 29 --LOCK

LOCK adds lockstrings to the Master's Lock Table.

ON ENTRY: DX:CX has address of lock string.
ON EXIT: AH and AL tell status of request.

Format of lock string:

1 byte - length of string (max 13)
13 byte - string. String should be blank padded if its less than 13 bytes.

Return status:

reg AL	reg AH	
00	00	- success
01	01h	- denied, owned by someone else
01	81h	- denied, already owned by you
01	83h	- denied, already owned by you, after a retry.
02	02h	- denied, illegal string length
02	82h	- denied, table full
02	03h	- denied, bad transmission (only occurs when retry flag is in reset state).

---NOTE---

The retry information is given to warn the user that the lock may have been successful but the confirmation was missed because of Network noise. The lock is tried again. However, since the first lock attempt may have successfully locked the string the status says it is locked. It is impossible to tell if it was locked on the first attempt or on a previous try.

UNLOCKING

Before unlocking a record under MS-DOS, the program must use MS-DOS Function 0DH. (Load AH = 0DH; see MS-DOS Programmer's Manual for details.) This function flushes the 'dirty' local buffers to ensure that the directory will be up to date when the file is closed. (MS-DOS Function 10H closes a file and updates the FAT.) This is very important when extending files. Ideally, in multi-user environments, file space should be pre-allocated before any extensions are attempted.

DMS MACHINE BIOS FUNCTION 30 -- UNLOCK

UNLOCK releases a locked string in Master's Lock Table.

ON ENTRY: DX:CX has addr of string to release.
Format is the same as in LOCK (above).

ON EXIT: AX has status of request, as follows:

reg AL reg AH

00	00	- accepted, unlock successful
01	01h	- denied, locked by someone else
02	02h	- denied, illegal string length
02	82h	- denied, string not found in table
02	83h	- denied, string not found in table after a retry
02	03h	- denied, bad transmission (only occurs when retry flag is in a reset state)

----------NOTE----------
Here, as in LOCK, a garbled response from the Master can lead to a retry, and then a spurious deny. This flag warns the user so they can used their best judgement as to how to interpret this.

MACHINE BIOS FUNCTION 16 -- CLRLOCK

CLRLOCK sends byte to Master to clear all locks under the sending station's User Number.

ON ENTRY: nothing

ON EXIT: AX is 0, zero flag is set.

6.0 THE NETWORK BUFFER

The HiNet BIOS normally provides a 1k network buffer to enhance system performance. However, for some programs such as multi-user data bases, data must not be buffered or obsolete data may mistakenly be taken as current.

In the past, programs that had to ensure that all data was current would first read (unwanted) data into the 1k buffer so that the read of desired data would come across the network and not from the 1k buffer. This is neither elegant or efficient. Starting with the HiNet BIOS version 247 there is a DMS-specific BIOS jump vector (SetNetMode) that allows a transient (i.e., user) program to select the Network Buffer usage mode. The three buffer modes are:

0) Always use the 1k Network Buffer. This is the default mode; it is automatically selected after a cold or warm boot.

1) Do not use the buffer contents on the next NetRead request - force a network transmission to ensure current data. This will replace the 1k network buffer contents; all subsequent NetReads will use the buffer contents.

2) Do not use the buffer contents until a cold or warm boot or until the program changes the network buffer usage mode.

The SetNetMode jump vector is available in both the network Master and the network Stations but will result in a Call Error on a stand-alone system. Since the network Master never has the 1k network buffer, the SetNetMode jump vector will do nothing - it is there simply so that networking programs do not have to check to see if they are running on a Master or Station.

To call the SetNetMode vector perform the following steps:

1) Load locations 0001 and 0002. This is the address of the warm boot vector.

2) Add 93 (5d) to the warm boot address. This is the offset to the lock function.

3) Add the value of register DE to the contents of register HL.

4) Load register C with the desired mode:

   0 => always use the network buffer

   1 => don't use the network buffer the next time only

   2 => never use the network buffer

5) Execute the code at the address obtained in step 2.

The previous NetMode value is returned in register A in case you wish to restore the NetMode to its previous state.

Reproduced below is a tested assembly program fragment that sets the NetBufMode to Buffer Mode 1, "Do not use the 1k buffer for the next NetRead only".

```
        .ident netjmp
        .pabs
        .phex
        .loc    100h
;
BiosVector  ==  01h     ;CP/M W B jump address
DMSoffset   ==  (5Dh-3) ;First jump in DMS table
Netmodedisp ==  (15*3)  ;# of jumps to SetNetMode
NotNextTime ==  01      ;Direct read next time
;
        lhld    BiosVector      ;CPM warm boot
        lxi     D,DMSoffset+Netmodedisp
                        ;# of bytes to SetNetMode
        dad     D       ;HL = addr of code in bios
        mvi     C,NotNextTime ;direct read next time
        pchl            ;execute it, return to
                        ;calling routine
;
        .END
```

At system assembly time, the choice may be made to not include the 1k Network Buffer in the system at all; this will automatically ensure that all NetRead requests get current data from the network. This generally provides poorer performance than when using the Network Buffer in conjunction with the SetNetMode vector. The distribution versions of the HiNet BIOS all use the Network Buffer for the stations. If the HiNet BIOS is assembled without the Network Buffer then the SetNetMode vector is still present but does nothing.

7.0 INTERSTATION COMMUNICATIONS

INTRODUCTION

Normally the Master can transmit polls and data on the Network to only one station at a time followed by a transmission from a station to the Master. Poll-Prime and Hog protocols enable a station to transmit data directly to another station without going through the Master. In overview, the procedure for this station to station communication is as follows:

- A station sends a Hog command to the Master requesting permission to transmit over the Network to another station.
- The Master grants the Hog to the station for a specified period of time.
- Using the WHO Table to determine the User Number of the station that is to be the target of the communication, the sending station sends a Poll-Prime addressed to the appropriate User Number.
- The receiving station ACKs the Poll-Prime indicating it is ready to receive data.
- The sending station sends the data to the receiving station's User Number.
- The receiving station ACKS (or NAKS) the reception of data.

- The sending station transmits a Hog Release to the Master, relinquishing control of the Network.

----------IMPORTANT----------
When using Poll-Primes under the 8086/88 BIOS, BIOS Call 49 (via INT 78) PPSEND greatly simplifies the process by performing all of the above steps for you. See Section 3.0 for more information on PPSEND.

The exact protocols for interstation communication on stations are described in this section. To avoid semantic difficulties the following 'glossary' is provided.

Poll-Prime: A one-byte transmission from one workstation to another requesting permission to send Poll-Prime data.

Poll-Prime Data: The actual data transferred from one station to another.

Poll-Prime Data Block: An area of memory reserved for receiving Poll-Prime data.

Poll-Prime Ack: A one-byte transmission from one station to another signifying one of two things:

1. A Poll-Prime has been received and it is alright to send the Poll-Prime data.

2. Poll-Prime data has been received without error.

Poll-Prime Nak: A one byte transmission from one station to another signifying one of two things:

1. A Poll-Prime has been received but the station will not accept the Poll-Prime data at this time.

2. Poll-Prime data has been received incorrectly.

Poll-Prime User: A pseudo user number (251).

Hog: A one-byte transmission from a station to the Master requesting permission to control the Network for a limited amount of time.

Hog Ack: A one-byte transmission from the Master to a station allowing the station to control the Network.

Hog Nak: A one-byte transmission from the Master to a station refusing permission to control the Network.

Hog Release: A one-byte transmission from station to Masterr terminating the Hog sequence.

Z816: The 816 running in Z-80 mode (as opposed to 8088 mode).

7.1 THE POLL-PRIME DATA BLOCK

To receive data, the user must first create a Poll-Prime data block. The format differs between Z-80 and 8086/88.

FIELD	Z-80	8086/88	DESCRIPTION
PPstat	01 bytes	01 bytes	Status byte
reserved	02 bytes	02 bytes	(for future enhancements)
PPdata	129 bytes	129 bytes	Received data
PPuoff		02 bytes	offset to user routine
PPuseg		02 bytes	segment of user routine
Size	132 bytes	136 bytes	

PPstat may contain one of the following values:

00h Station will NAK Poll-Primes. (Set at warm boot.)

01h Station will ACK Poll-Primes.

11h **8086/88 ONLY:** Station will ACK Poll-Primes. When the Poll-Prime data is received the receive status will be passed to the user routine addressed by PPuoff and PPuseg. The user routine must determine if the transmission is good or bad. The user routine will return a Poll-Prime Ack or a Poll-Prime Nak in AL. The BIOS will send the users response to the Poll-Prime user. 02h (internal only) Station has acked a poll-prime and is waiting for the Poll-Prime data.

12h **8086/88 ONLY:** (internal only) Station has acked a Poll-Prime and is waiting for the poll-prime data. When the Poll-Prime data is received a user routine will be called. (See status 11 above.)

83h Station has received Poll-Prime data without error. Station will not ACK Poll-primes until the status has been changed to 01 (or 11h for the 8086/88).

PPuoff and PPuseg are defined for the 8086/88 BIOS only. They point to a user routine accessed by the BIOS if the user sets the status byte to 11h. The user routine is called with a CALLF instruction and must terminate with a RETF instruction.

PPdata is the user data area where the poll-prime data is stored. The first byte is, by convention only, the user number of the sending station.

7.2 RECEIVING POLL-PRIMES

The BIOS must be told the address of the Poll-Prime data block before Poll-Primes can be received. Until this happens, the BIOS uses an internal 1 byte status buffer set to 0 - not accepting Poll-Primes. A BIOS function (SETPPA) has been implemented to perform this function.

To use the function, the user must allocate memory for his Poll-Prime data block, set the PPstat byte to the appropriate value, and then call SETPPA to tell the BIOS where the poll-

prime data block is. Use BIOS Function Call 41 on Z-80 based stations and Function number 32 with Interrupt 78 on 8086/88 stations.

Once the SETPPA has been called, the user may turn on and off the acking of Poll-Primes by setting the PPstat byte.

When Poll-Prime data has been received, the BIOS changes the PPstat byte to 83h. The BIOS will not accept Poll-Primes until the user resets the PPstat byte to 01h (or optionally 11h for the 8086/88). When a warm boot is performed in CP/M-80 or CP/M-86, the BIOS will reset an internal pointer to point to a dummy PPstat byte set to 0 (not accepting Poll-Primes).

7.3 MS-DOS CONSIDERATIONS

MS-DOS does not have a warm boot process. For this reason the user MUST call SETPPA with the DX register set to 0 before the program using Poll-Primes terminates. This will clear the MS-DOS BIOS internal pointer. If this is not done, the BIOS pointer will be pointing to an area in memory that is no longer a Poll-Prime data block. The next Poll-Prime received by the station will interpret the byte with unpredictable results.

If the program using Poll-Primes aborts, the RELEASE utility may have to be run to reset the BIOS Poll-Prime pointer. If RELEASE fails due to a reception of a Poll-Prime before resetting the pointer, the station must be reset.

The Z-80 BIOS will only set the status byte to 83h if the Poll-Prime data is received without error. If received incorrectly, the BIOS will reset the status byte to its previous value, i.e., receiving Poll-Primes. The 8086/88 will act the same as the Z-80 if the PPstat byte is set to 01h. If set to 11h the user program will be called at address PPuseg:PPuoff with the received status byte in register AL. The user routine may decide to NAK an otherwise good transmission (or ACK a bad one). The BIOS checks the AL register returned by the user interface. If it is **not** an ACK the BIOS will NAK the Poll-Prime data.

The application program need only test the PPstat byte to determine if Poll-Prime data has been received. The BIOS will refuse Poll-Primes after reception until the PPstat byte is changed to 01h (or 11h for the 8086/88).

7.4 SENDING POLL-PRIMES FROM A Z-80 STATION

The following is an overview of the steps required to send data from one Z-80 station to another user station on the Network. The user Number of the User currently logged in at the receiving station must be known. If the receiving station has a unique user name a WHO request may be used to translate the user name into a user number. Stations may also be logged in under their PROM Serial Number (auto-boot stations).

On 8086/88 workstations, the BIOS Call **PPSEND** should be used instead of the following Z-80 procedure. PPSEND greatly simplifies the Poll-Prime process; see Section 3.0, BIOS Call 48.

1. Wait for a poll from the Master but do not ACK it.

2. Send a HOG request to the Master requesting control of the Network.

3. Receive from the Master. Set up a data area to receive acknowledgement bytes and save the byte received from the Master. If the Master has sent a HOG ACK, continue with step 4; if not, go to step 10.

4. Set the station's timeout value to **4 msec** with the BIOS RECTIME call.

5. Send a Poll-Prime to the workstation you wish to communicate with.

6. Using the Poll-Prime user number, receive from the station sent to in step 5. Save the byte received in the same data area as in step 3. If a Poll-Prime ACK is received, continue with step 7; otherwise skip to step 9.

7. Send Poll-Prime data to the station.

8. Using the Poll-Prime user number, receive from the station sent to in step 7. Save the byte received (again, in the same data area as before).

9. Send a Hog Release to the Master. This is required to tell the Master that it has control of the Network again.

10. Tell the station BIOS to start acking polls again. If this is not done the station will soon log out.

11. Test the byte received and stored in steps 3, 6 or 8. If this byte was not a Poll-Prime ACK the transmission was unsuccessful and must be repeated. The program may retry this sequence a set number of times (5 works well) before deciding the transmission cannot be made.

The BIOS contains entry points for five of the functions described above, they are:

1. Wait for a poll from the Master but do not ACK. (NACKPOLL on the Z-80, CLRPOL on the 8086/88). This turns off polling until enabled by ACKPOLL/SETPOL.

2. Tell the BIOS to start acking polls again. (ACKPOLL on the Z-80, SETPOL on the 8086/88).

3. Send a Network transmission (SENDNET on the Z-80, SNDNET on the 8086/88).

4. Receive a Network transmission (RECNET on the Z-80, RCVNET on the 8086/88).

5. Set the recieve time-out value (RECTIME).

The following describes the Z-80 calls in more detail. See the section on making BIOS calls for further information.

NACKPOLL (Z-80)

BIOS function number 38
no input registers
returns receive status
bit 7 off = timeout
bit 6 on = CRC error
bit 5 on = receiver overrun
bit 3 on = receiver underrun
bit 0 on = poll received
The remaining bits may be in any state
and should be masked prior to testing.

ACKPOLL (Z-80)

BIOS function number 39
no input registers
no registers returned

SENDNET (Z-80)

BIOS function number 36
A = user number of station to send to
BC = number of bytes to send
HL = address of data to send
no registers returned

RECNET (Z-80)

BIOS function number 37
A = receive user number
BC = number of bytes to receive
HL = address of receive buffer
returns receive status in A
bit 7 off = timeout
bit 6 on = CRC error

bit 5 on = receiver overrun
bit 3 on = receiver underrun
bit 0 on = poll received
The remaining bits may be in any state
and should be masked prior to testing.

RECTIME (Z-80)

BIOS Function Number 42.
BC should have timeout in number of
milliseconds.

Using the BIOS procedures defined, sending data from a Z-80 station to another station requires these steps:

1. call NACKPOLL if error go to step 9.

2. move a HOG request to a 1 byte send buffer call SENDNET - send the 1 byte send buffer to the Master (station 0).

3. call RECNET - use my user number. receive one byte from Master into 1 byte receive buffer. If net error or byte not hog ACK go to step 9.

4. call RECTIME to set the sending station's receive timeout to 4 msec.

5. move Poll-Prime to 1 byte send buffer call SENDNET - send 1 byte send buffer buffer to station n where n = receiving station.

6. call RECNET - use Poll-Prime user number. Receive into 1 byte receive buffer. If byte not Poll-Prime ACK, go to step 8.

7. call SENDNET - send 129 byte buffer to station n where n = receiving station's user number.

8. call RECNET - use Poll-Prime user number. Receive into 1 byte receive buffer.

9. move HOG Release to 1 byte send byte buffer call SENDNET to send 1 byte send buffer to Master (station 0).

10. call ACKPOLL.

11. Test the 1 byte receive buffer. If it is not a Poll-Prime ACK then the data transfer did not take place. Up to 5 retries (but in no case less than 2) are recommended.

-A-

-B-

-N-

-P-

-R-

-S-

-T,U-

-V-Z-