DIGITAL MICROSYSTEMS, INC.

HiNet Protocols for BIOS 2.249

Version 1.1

Notice:

Digital Microsystems, Inc. reserves the right to make improvements to the products described in this manual at any time without notice.

Release: 06/21/84

HiNet Protocols for BIOS 2.249

This document can be used by anyone desiring to write programs or design systems which interface with HiNet. HiNet is a CP/M and MS-DOS compatible local microcomputer network developed by Digital Microsystems. The features and commands of HiNet are presented in the DMS-3/4 HiNet System Manual. It is assumed that the reader of this document is familiar with the system manual, and well acquainted with HiNet.

Section 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:

Flag byte	User number	Data bytes (1 to 1024 bytes)	CRC bytes (2 bytes)	Flag byte

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 <u>always</u> 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 read from or to be 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 transaction 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.

Section 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 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
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

Each of these commands is described in detail in section 4.

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 4.

4. A user station that fails to respond to 256 consecutive polls in the following manner will get logged out: the user 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 user number and all of his 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 Z80 network station bootstrap code is transmitted periodically (once per polling loop <u>and</u> about once per second) to pseudo-user 254. The PROM at each Z80 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 first log in by responding to a poll of user 253 with a HiNet user name, password, binary serial number, and product type. The master consults the Machine table, Product Type table, and User table on the hard disk for a matching name, password, and product type. The master accepts the login request unconditionally and responds with a unique user number, the login time, and the binary serial number. Then it will immediately send boot phase 2, a loader for the BIOS or Login Please program. What is loaded depends on whether the name/password was found in the User table and 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 tables and buffers are maintained in the master:

Location	Name of table	Description
see Netinfo, currently at 0FF00h	Spool file table	Each spool file is described by a 16 byte entry. See the "Get who table" command in section 4 for more information.
see Netinfo, currently below spool table	Lock table	Each lock string 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 lock string length. The next 13 bytes are the lock string. The last byte is not used.
see Netinfo, currently below lock	Who table	Each user is described by a 16 byte entry. See the "Get who table" command in section 4 for more information.
400h bytes below who table	Master Buffer	General buffer for Master's use. Not all hard disk I/O operations use this buffer.
optional, 100h bytes below hard disk buffer	Floppy write buffer	This buffer is used for all double-density floppy write operations.
optional, 100h bytes below floppy write buffer	Floppy read buffer	This buffer is used for all double-density floppy read operations.

These tables can be manipulated while the network is running by using DDT or ZDTI. However, extreme caution should be taken.

Section 3: Login Procedure

The entire login procedure, from a station's point of view, is outlined below.

1. The Z80 PROM code programs the SIO-1 and the DMA chip to receive 380h bytes addressed to user 254. When this code is received, it is loaded into memory at location 9000h, and the PROM executes a "JMP 9000h" instruction. The code sent is known as Boot Phase 1. The 8086/8088 PROM contains Boot Phase 1 internally and therefore ignores this step.

2. An automatic login is attempted as follows: the eight character name field contains the eight ASCII hexadecimal digit machine serial number; a six-character password of blanks; binary machine serial number, in four bytes; product type, in one byte. The SIO chip is then programmed to receive a poll from user 253. When a poll is received (command byte 50h), the station will respond with a command byte of LoginNet (13h), followed by the automatic login data. The master will normally respond with a LogAck (4Ch) message. The LogAck response byte will be followed by: a unique user number and login time (i.e., the 7 bytes stored in locations 40h-46h on the Z80) and the machine's binary serial number (in four bytes).

The station will verify that the LogAck (4Ch) response was followed by the station's 4-byte serial number; this provides an additional level of error checking beyond that which the SIO provides in hardware.

The master will send a LogNack (4Eh) to pseudo-user 253 if no login request is received within approximately 1 mSec. Note, if two stations attempt to log in simultaneously, it is likely that CRC errors will be generated and each station will receive a response of LogNack (4Eh). If either a flawed network transmission or a LogNack (4Eh) is received, the Z80 station will use the refresh register to pause (from 400ms to 16 seconds), and will then attempt another automatic login. The 8086/8088 will use an algorithm based on its serial number to pause before retry.

If the ASCII serial number and password are not in the user table, the master will still respond with a LogAck (4Ch). However, when Boot Phase 2 (described in step 3 below) is sent, a Login Please program will be selected to run in the station. The user will be prompted for a login name and password, and the login procedure thru pseudo-user 253 will be repeated as above.

If no Boot Phase 2 can be found for the given product type, or any crucial table lookup failed, e.g., a Login Please program could not be found, the master will respond with a LogDeny (44h) and the same parameter list as LogAck.

3. Once a login request has been accepted, the master will follow the "login accepted" transmission with an immediate transmission (addressed to the number of the user who just logged in) of additional bootstrap code, Boot Phase 2. This procedure is necessary because the entire bootstrap code will not fit within 380h bytes: it must be transmitted in 2 parts. The second part of the bootstrap contains a loader (Boot Phase 2) and a load list. The loader is sent in one or more 1024 byte transmissions, depending on the value of the first byte sent. The load list contains the disk addresses, lengths, and transfer addresses of the BIOS/BDOS/CCP or Login Please program which the master has selected based on the Product Type and User name. The loader reads the station's BIOS or other appropriate program from the master hard disk. Note, once a user has been granted a unique user number, that user number is used for all following network transactions. Also note, the user is allowed to initiate a network transaction only after receiving a poll. If the loader reads in a Login Please program instead of a BIOS, that program will do an Instant Logout (20h) after receiving the first poll, prompt the user to enter a name and password, and repeat the login sequence similar to automatic login above.

```
--------------- Hard Disk Control Area Layout ------------------
                         (Partition 0)

Logical Address                 Contents
---------------                 --------
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:
                                     8 bytes:  user name or
                                               station serial no.
                                     6 bytes:  password
                                     1 byte:   OS code
                                     1 byte:   flags (incl
                                               big/small
                                               request)

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

                                Up to 64 16-byte entries:
                                   1 byte:  size (0-8)
                                   8 bytes: partition name
                                   6 bytes: password
                                   1 byte:  control byte
```

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	Password Table
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:	OS 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.

BOOT PHASE 2 program

Boot Phase 2 is the loader for the OS to follow.

Its first byte will be the number of 1024-byte blocks the Boot Phase 2 program occupies, enabling the receiving station to set up for another network transfer if necessary.

The next three bytes enable the host to jump to the beginning of the Boot Phase 2 code. In a Z80 this will simply be a jump instruction. In an 8086/8088 the first of these bytes will be null and the next two will be the offset from the beginning of Boot Phase 2 to the beginning of the executable code.

Next comes a data block, always in the following format:

```
Default partition assignments
    (four names of eight characters)           32 bytes
IOBYTE                                          1 byte
Default type-ahead buffer and pointer          32 bytes
Honor Flag (see description below)              1 byte
Load List
    (up to eight partial directory entries
    each 16 bytes, of the form:
        Disk Address (v,p,t,s)  5 bytes
        length (128-byte sects) 2 bytes
        RAM Load Address        4 bytes
        Relative Start Address  2 bytes
        Reserved                3 bytes)
                             total:           128 bytes
Product Type                                    1 byte
                                              -----------
                total length of data block    195 bytes
```

This data block is initialized by the Master before Boot Phase 2 is sent out. A description of each field follows:

Default partition assignments:
These are the assignments specified in the User Configuration Table entry (created by USERS), if a BIOS is to be loaded. If a Login Please or OS Menu is to be loaded, these fields are undefined.

IOBYTE:
This comes from the Machine Table entry for this machine. If there is no Machine Table entry, the IOBYTE will be 00 and the Boot Phase 2 program should attempt to determine the console type and set this field correctly. If this is done then a BIOS loaded by this Boot Phase 2 will be able to run in at least a minimal fashion, enabling the user to run MACHINE and put the machine in the Machine Table.

Default type-ahead buffer:
As specified in the User Configuration Table (set up by USERS), if the user name/password is found. Otherwise, undefined.

Honor Flag:
This flag is set to show the extent to which the login request was honored, i.e. what sort of program will be loaded next. The lower nibble will be:

0 - An OS exactly matching the request will be sent, i.e. its Option Map will match the machine's (unless a high-TPA BIOS was requested.)
1 - A default OS will be sent, of the general type as the requested one but with a different option map. Implies that there is no OS available with the exact option configuration of the machine.
2 - A "login please" program will be sent. Implies that the name/psw were not found in the User Table.
3 - An "OS Menu" program will be sent. Implies that no OS of the requested type could be found that would stand a chance of running on the machine logging in. In the initial release this will probably be a message rather than a real menu.

4 - Total failure, no program will be loaded.

The high bit of the honor flag is used to show whether the machine is in the Machine Table, viz.:

0 - Normal (the machine is in the Machine Table.)
1 - Not found. Without a Machine Table entry, Boot Phase 2 will have to try to figure out the correct IOBYTE for this machine, probably by examining the PROM.

Depending on the value of the Honor Flag, Boot Phase 2 may print a message (using PROM I/O routines) notifying the user of an unusual situation (e.g. a default OS or a Machine tTble lookup failure.)

Load List:
This list consists of entries from the System Directory, with the eight-byte names removed, making each entry sixteen bytes long. Each entry describes a file to be read from the net, including information as to where it is to be loaded and where to begin executing it. If fewer than eight files are specified (actually two will be a more common number), the length field in the entry after the last one used is set to zero by the master (in fact, the block is filled with zeros on the end, but these might be legal values in other fields.)

Product Type:
Included in Boot Phase 2 because Z80 machines do not know their own product type and so must find out from the Master.

Thus there are at least 199 bytes in Boot Phase 2 before the real executable code.

"LOGIN PLEASE" program

This program will be loaded in lieu of an OS if the name/psw in the login request do not match any entry in the User Table. (This will often be true on auto-login by serial number.) Its first action will be to request an "instant logout" of the current user number. The program will then ask the user for a name and password. After appending the binary serial number, and product type, the program will wait for a poll of the login pseudo-user (253), issue a login request, and prepare for the log-ack.

The Master's response will be handled the same by this program as it would be in the above boot procedure; i.e., the serial number in the response will be matched against our own, and the login request will be retried if the serial numbers don't match. If they do match, and the response is an ack, the program notes the new HiNet user number and waits for a new Boot Phase 2 to be sent to that user number. It then executes Boot Phase 2. If the serial number matches but the response is a nack, this is a fatal error. A diagnostic message will be sent to the screen.

The login program will have access to the PROM I/O facilities, to the information passed in Boot Phase 2, and to the HiNet user number.

"OS MENU" program

This program loads in lieu of an OS. It uses PROM I/O facilities and has access to the information passed in the Boot Phase 2 code and to the HiNet user number.

The program reads the OS table from the master and presents to the user a menu of the OS's capable of running on the machine logging in. (Presenting the option map in a reasonable way may be difficult.) When the user selects one, the program loads it -- probably by reinitializing the still-resident Boot Phase 2 with the address of the selected OS, and re-running Boot Phase 2.

Since appearance of the "OS MENU" program implies that no system of the type specified in the User Option Table could be found for the machine, it is unlikely that the default partitions will be suitable for the OS eventually selected. Therefore the re-initialization of Boot Phase 2 must include new default partitions as specified by the user from the console. Program must check that the requested partitions can be used under the requested OS. Any TPA utility version of this program will also have to ask for the default partitions, for the same reason.

Note that this program is required to know the location of the OS table in the master.

Until completion of the OS menu program, the system would benefit from a skeletal program that tells the user why the login request could not be honored.

OS numbers

0 not to be used - can be returned if search fails, etc.

1 CP/M

- 11 = CP/M 2
- 12 = CP/M 86
- 13 = HiDOS
- 14 through 1F = other CP/M compatible OS's

2 MS-DOS

- 21 = MS-DOS vers. 2.11
- 22 = MS-DOS vers. 2.x
- 23 = MS-DOS vers. 3.x
- 24 through 2F = other MS-DOS compatible OS's

3 Reserved

4 through F not yet assigned

Product Numbers

The highest bit of the Product Number specifies the console type (0 for serial, 1 for parallel). This leaves 128 possible product numbers which specify the type of CPU board as follows:

0	Not to be used	
1	ZSBC-3 CPU's:	
	DMS-3	"DSC-3"
	DMS-3/A25	"Smart ADDS"
	DMS-3/4004	
	DMS-3/4008	
	DMS-3/101	
	DMS-3/102	
	DMS-3/103	
	DMS-3/B	
	DMS-3/F	"Fox"
	DMS-501/15	
	DMS-5080	
2	DMS-4	"DSC-4"
3	DMS-1280	
4	DMS-3/C	"Killer Bee"
5	DMS-5086 and HNS-86	
6	reserved	
7	DMS-816	
8	PC Adapter	

(Product numbers 9 through 127 not yet assigned)

Optional Device Drivers

This is a bit-map with the exception of the block describing the number of logical drives supported. An exact match with the Machine Table is required. Since it is possible to configure an arbitrary number of logical drives (up to 16), the only bit-mapped way of doing it would use 16 bits or one-third of the bit map.

N.B. There is no way of knowing, a priori, whether a driver named "custom printer" will run on a given machine.

bit	device driver
0	8-inch floppies (SD and DMS DD)
1	5-inch Fox-floppies
2	5-inch IBM-format floppies
3	8-inch HD with DMS controller
4	5-inch HD with Xebec controller
5	5-inch HD with Adaptec controller
6	Port 0 type-ahead (console)
7	Port 0 polled (")
8	Port 2 polled (printer)
9	Port 3 type-ahead (aux comm.)
10	Port 3 polled (" ")
11	Parallel Port 1 (console)
12	Parallel Port 1 (printer)
13	Parallel Port 2 (Fox 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

Section 4: HiNet Commands

Each of the HiNet command protocols is described in this section. A diagram showing the exchange of information between the master station and the user station is included for each command. The direction of information flow is shown by the arrows, and the format of the information is given in parentheses above the arrow. The approximate time required to send the information is given below the arrow. The time between exchanges of information (i.e., turnaround) is usually about 0.5ms, excluding disk access time. A disk access may require anywhere from 2ms to 150ms to complete. For example, the following diagram shows the poll-acknowledgment sequence:

```
                        poll (50h)
  master      ------------------------->      user
  station             0.16ms                  station

                    acknowledge (41h)
             <------------------------
                      0.16ms
```

Total time (including turnaround): 0.82ms

A poll is a one-byte message sent <u>from</u> the master station to user stations around the network; an acknowledgment is a one-byte message sent <u>to</u> the master from the network user stations. The BIOS in a HiNet station normally answers all polls in this fashion; this acknowledgment is considered to be one of the network commands-- its function is to tell the master that the station is active, but does not have any request for the master. The basic polling rate is approximately 62 times a second, with stations which read consecutive records from the master hard disk being polled continuously until they cease to make consecutive read requests. Network loading and the preferential polling given to stations making consecutive read requests will affect the frequency with which any given station is polled.

Error Detection and Recovery

When the network master receives either a command or response that is illegal or unrecognizable it will print the message shown below. But note, this applies only when the error message display flag is true.

```
"*** User ww xxxxxxxx yyyy Net error at zzzz"
"MASTbuf: Mst Usr To  Frm Dsk Track  Sec Vol DmaAdr"
"         aa  bb  cc  dd  ee  ff  gg  hh  ii  jj  kk"
```

ww	=	user number in hexadecimal
xxxxxxxx	=	user name
yyyy	=	error type [LATE, CRC, OVR, SYNC]
zzzz	=	error address in hexadecimal
aa - kk	=	command specific parameters in hexadecimal

The master will return immediately to the main polling loop, regardless of the error message display flag state; the user should be polled during the next invocation of the master.

The user station functions similarly, except:

1.) the user name is not printed;
2.) "NET buf:" is displayed instead of "MASTbuf:"; and
3.) whatever function the station was executing is retried automatically.

THESE MESSAGES INDICATE THAT AUTOMATICALLY CORRECTED TRANSIENT ERRORS OCCURRED - NETWORK INTEGRITY IS GUARANTEED. NO USER INTERVENTION IS REQUIRED.

Since MASTbuf is a general purpose buffer, the meaning of the command-specific hexadecimal parameters can vary greatly. In general, Mst represents a command that the master sends to a station, for example, a poll (50). Usr represents the command the station sends back to the master, for example, a read or write (11, 12 or 15). To and Frm generally represent user numbers that messages are sent to or received from. Dsk, Track, Sec, Vol, are usually meaningful for read and write, and DmaAdr may indicate an address in the Master's memory. Not every field may be meaningful even though it is displayed. Error types have the following meaning:

LATE	--	Master did not receive an expected trailing SDLC flag byte, which indicates that the station did not complete the transaction.
CRC	--	Master received data with transmission errors
OVR	--	Master received more data than was expected
SYNC	--	Master received unexpected data or command

Mimicking

Under HiNet, one can connect a second hard disk system to the network cable and duplicate all master disk writes on that system. The second system is called the "mimicker." The mimicker is brought on-line by answering the master's poll of pseudo-user 252. This pseudo-user is polled approximately 3 times per second. The mimicker prints an 'M' on its console whenever it is polled. The mimicker must answer each poll to remain on-line. While the mimicker is on-line, all writes will be mimicked.

The following diagram shows the master-mimic polling sequence:

```
                     poll (50h)
master      -------------------------->     mimic
station               0.16ms                station
                                           (user 252)
                 acknowledge (41h)
           <--------------------------
                      0.16ms
```

Total time (including turnaround): 0.8ms.

The other mimicking protocols are described under the "Write 128 bytes" and "Spool 128 bytes" commands.

Login Command

The login command is used to connect to the HiNet system. The entire login procedure is described in sequence. The login command sequence is summarized by the following diagram:

```
                         poll (50h)
    master       ------------------------->      user
    station               0.16ms                 station
                                                 (user 253)

          login (13h,name,psw,ser#,prod)
        <-------------------------------------
                          0.46ms


               login-nack (4Eh)  OR
         login-deny (44h,usr,time,ser#)  OR
           login-ack  (4Ch,usr,time,ser#)
        ------------------------------------->
                          0.35ms


       bootstrap code (1 or more 1024 byte groups)
       ----------------------------------->  to new
                          16.53ms            user number
```

Total time (including turnaround, excluding disk accesses for machine, product, and user table lookups): 19.00ms

The first byte of the bootstrap code sent indicates the number of 1024 byte transmissions necessary for the entire bootstrap reception.

Command Parameters	**Parameter Length**
nam = user name	8 bytes
psw = password	6 bytes
usr = user number	1 byte
time= login time (ticks,sec,min,hr,mth,day,year)	7 bytes
ser#= binary serial number	4 bytes
prod= machine product number	1 byte

Get Who and Spool Tables

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

```
                      poll (50h)
master     -------------------------->     user
station              0.16ms                station

                       who (10h)
          <--------------------------
                       0.16ms

          who table (513 or 1025 bytes)
          ---------------------------->
                8.34 or 16.53ms

              spool table (256 bytes)
           -------------------------->
                       4.24ms
```

Total time (including turnaround): 14.40ms or 22.59ms

The who table has either 32 or 64 entries, each 16 bytes long. The first byte of the who table sent indicates the 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 actual number of users. Each entry contains the following information:

```
byte 0        =  00   if entry not in use
              =  0FFh if user is logged in
              =  other if user is logging out (counting
                 down one for each missed poll)
bytes 1-8     =  user name
bytes 9-11    =  login time (secs, mins, hrs)
bytes 12-14   =  time of most recent HiNet transaction
byte 15       =  most recent HiNet command byte
```

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:

```
byte 0      =  00   if starting
               01   if spooling
               02   if ready to print
               03   if printing
               04   if finished
               05   if waiting
               0E5h if entry not in use
byte 1      =  user number
byte 2      =  spool job id
bytes 3-4   =  time of most recent spool request
bytes 5-6   =  current spool track
byte 7      =  current spool sector
bytes 8-15  =  user name
```

Read 128 Bytes

This command allows a station to read 128 bytes from a specified partition, track, and sector. The standard HiNet station BIOS does not use this command--the "Read 1024 bytes" command is used instead, because it is much more efficient. The standard station BIOS will process the read128 command as a 1024 byte read and will return the appropriate 128 byte data block to the application. Those applications which read or write to a multi-user partition need to guarantee sometimes that data is current and not outdated by other user modifications; therefore, a DMS-specific BIOS jump table entry is provided which allows an application to force the BIOS to throw away the 1k net buffer contents and perform a network transfer.

The BIOS may be assembled to allow true 128 byte network reads; however, in so doing, one sacrifices the ability to perform 1024 byte network reads.

```
                       poll (50h)
master       -------------------------->        user
station                 0.16ms                  station

            read (11h,dtn,src,dsk,trk,sec,vli)
        <-------------------------------------
                        0.27ms

                     data (128 bytes)
             -------------------------->
                        2.19ms

                      data-ack (44h)
             <--------------------------
                        0.16ms
```

Total time (incl. turnaround, excl. disk seek/read time): 4.28ms

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

Read 1024 Bytes

This command allows a station to read 1024 bytes starting at a specified partition, track, and sector. The sector number should be 1, 9, 17, ... or 121 (8n + 1). These sector numbers correspond to physical sector boundaries, i.e., each read operation will read a single 1K physical sector from the disk. The standard HiNet station BIOS sets aside a 1K buffer for all read operations.

```
                        poll (50h)
master      ------------------------->      user
station               0.16ms               station

           read (15h,dtn,src,dsk,trk,sec,vli)
       <-------------------------------------
                      0.27ms

                   data (1024 bytes)
            ------------------------->
                     16.53ms

                   data-ack (44h)
          <--------------------------
                      0.16ms
```

Total time (incl. turnaround, excl. disk seek/read): 18.62ms
(disk read time from 2 to 150ms)

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 (1,9,17,...121)	1 byte
vli = volume number (0-3)	1 byte

Write 128 Bytes

The write command is used to write 128 bytes to a specified partition, track, and sector. If the mimicker is on-line, the writes will also be mimicked.

```
                           poll (50h)
     master      ------------------------->      user
     station               0.16ms                station

            write (12h,dtn,src,dsk,trk,sec,vli)
          <-------------------------------------
                           0.27ms

                    cmd-deny (4Fh)  OR
                      msg-ack (4Dh)
                 ------------------------->
                           0.16ms

                    data (128 bytes)
               <-------------------------
                           2.19ms

                     data-ack (44h)
                 ------------------------->
                           0.16ms
```

Total time (including turnaround, excluding disk write): 4.94ms
(disk write occurs after data-ack)

If a mimicker is on-line, these protocols will also be used:

```
            write (12h,dtn,src,dsk,trk,sec,vli)
     master    ------------------------------------->    mimic
     station               0.27ms                        station

                      msg-ack (4Dh)
               <-------------------------
                           0.16ms

                    data (128 bytes)
                 ------------------------->
                           2.16ms

                     data-ack (44h)
               <-------------------------
                           0.16ms
```

Total time (including turnaround, excluding disk write): 4.25ms

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

* The high bit of the volume number is set automatically by the station if the station or local user is running HiDos (to verify that a shared partition is written to only by a HiDos OS and not a CPM-86 OS).

A write may be denied for several reasons:

1. A non-HiDos write is attempted to a shared partition.
2. The partition is marked read-only.
3. The partition is owned for writing by another user.

The write mode query function may be used to determine the source of the error.

Start Spool File

To start a spool file, a user station must execute the following command:

```
                          poll (50h)
master       ------------------------->      user
station                    0.16ms                 station

                     start-spool (14h,sid)
            <-------------------------
                           0.18ms

           response (04h,dsk,trk,sec,vli,size)
           ---------------------------------->
                           0.26ms
```

Total time (including turnaround): 1.6ms

The master returns:

1.) an AckSpStart (04h),
2.) the spool partition number (dsk),
3.) the track and sector at which spooling should start,
4.) a spool block size

If dsk = 0, then the spool request is denied. This will happen if there is no PRTSPOOL partition, or if the PRTSPOOL partition is full.

The station should maintain a spool job id byte whose high nibble is a job number (initialized to zero when the station boots, and incremented when a spool job ends), and whose low nibble is the spool block number (initialized to zero when a spool job starts and incremented when a spool block is filled).

The station should spool to consecutive sectors and tracks, beginning at the track and sector supplied by the master. Spooling should occur on sectors 3 to 128 on the first spool track, and 1 to 128 on subsequent spool tracks. The number of tracks is specified by the spool block size field. The station is responsible for enforcing this limit; if an attempt is made to spool beyond this limit, the space allocated to other spool files may be overwritten. Rather than surfeit the spool file , issue a "End spool file" command after writing the last sector, increment the low nibble of the spool job id byte and then issue a "Start spool file" command to resume spooling on a new spool block.

Command Parameters	**Parameter Length**
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
sid = spool job id number	1 byte
size= spool block size (1,2,4,8,16)	1 byte

Spool 128 Bytes

The spool command is used to write spool data to the PRTSPOOL partition. The spool command is identical to the write command, except that a different command byte is used.

The station should spool to consecutive sectors and tracks, beginning at the track and sector supplied by the master. Spooling should occur on sectors 3 to 128 on the first spool track, and 1 to 128 on subsequent spool tracks. The number of tracks is specified by the spool block size field. The station is responsible for enforcing this limit; if an attempt is made to spool beyond this limit, the space allocated to other spool blocks may be overwritten. Rather than surfeit the spool file , issue a "End spool file" command after writing the last sector, increment the low nibble of the spool job id byte and then issue a "Start spool file" command to resume spooling on a new spool block.

```
                           poll (50h)
      master      -------------------------->       user
      station               0.16ms                  station

                spool (1Ch,dtn,src,dsk,trk,sec,vli)
              <---------------------------------------
                            0.27ms

                          msg-ack (4Dh)
                  -------------------------->
                            0.16ms

                        data (128 bytes)
                  <-------------------------
                            2.19ms

                         data-ack (44h)
                  -------------------------->
                            0.16ms
```

Total time (incl. turnaround, excl. disk write): 4.94ms
(disk write occurs after data-ack)

If a mimicker is on-line, then these protocols will also be used:

```
                 spool (1Ch,dtn,src,dsk,trk,sec,vli)
  master         ------------------------------------>     mimic
  station                      0.27ms                      station

                        msg-ack (4Dh)
                  <-------------------------
                            0.16ms

                        data (128 bytes)
                   -------------------------->
                            2.19ms

                        data-ack (44h)
                  <-------------------------
                            0.16ms
```

Total time (incl. turnaround, excl. disk write): 4.28ms

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

* The high bit of the volume number is reserved, and will be set by HiDos stations, however, a spool partition cannot be marked HiDos shared.

End a Spool File

Use this command to end a spool file, or swap to the next spool block, making the current block available for printing.

The station should maintain a spool job id byte whose high nibble is a job number (initialized to zero when the station boots, and incremented when a spool job ends), and whose low nibble is the spool block number (initialized to zero when a spool job starts and incremented when a spool block is filled).

When the station reaches the end of a spool block (as specified by the spool block size field of the start spool protocol), the station should issue an "End spool file" command after writing the last sector, increment the low nibble of the spool job id byte and then issue a "Start spool file" command to resume spooling on a new spool file. The station is responsible for enforcing the spool block size limit; if an attempt is made to spool beyond this limit, the space allocated to other spool blockss may be overwritten.

```
                        poll (50h)
master        ------------------------->        user
station                  0.16ms                 station

                  stop-spool (16h,sid)
             <-------------------------
                         0.18ms

                     msg-ack (4Dh)
              ------------------------->
                         0.16ms
```

Total time (including turnaround): 1.5ms

Command Parameters	**Parameter Length**
sid = spool job id number	1 byte

Assign Partition

The assign command is used to determine the volume number, partition number and size of a specified partition on the HiNet shared disk. A station should not attempt to access a HiNet partition (except partition 0) without first using the assign command. The partition number should be used for all subsequent access to the HiNet disk; the partition size should be used to construct a disk parameter table required by the BDOS.

```
                        poll (50h)
master     ------------------------->     user
station              0.16ms               station

                   assign (17h,nam,psw)
          <-------------------------
                     0.38ms

                 response (siz,dsk,ctl,vli)
          ------------------------->
                     0.21ms
```

Total time (incl. turnaround, excl. alloc table lookup): 1.75ms

The master returns the partition's size, number, and control byte. A size of 0FFh indicates that the assignment was denied. This will happen if the partition name and password did not match any entry in the allocation table. To force the master to ignore the password, use a password of all zeroes.

Command Parameters	**Parameter Length**
nam = partition name	8 bytes
psw = password	6 bytes
siz = partition size (1-6)	1 byte
dsk = partition number (0-63)	1 byte
ctl = control byte	1 byte
vli = volume number this parition is on	1 byte

The contents of the disk parameter table for each possible partition size is shown below:

CP/M 2.2 Disk Parameters

	=1 * 256K	=2 512K	=3 1MEG	=4 2MEG
Sectors per track	128	128	128	128
Block shift, 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	0C0h,0	0C0h,0	0F0h,0	0FFh,0
Check vector size**	16	32	64	128
Op sys tracks	0	0	0	0

	=5* 4MEG	=6 8MEG	=7 16MEG	=8 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	0FFh,0	0F0h,0	0C0h,0
Check vector size**	256	256	256	256
Op sys tracks	0	0	0	0

* The numbers one through eight represent partition size.
** HiDos does not allocate check vectors, hence size = 0.

	8" Floppy Single Density	**8" Floppy Double Density**	**5" Double sided Mini 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	0C0h,0	0C0h,0	080h,0
Check vector size	16	32	32
Op sys tracks	2	2	3

MSDOS Disk Parameters

	=1* 256K	=2 512K	=2 1MEG	=4 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	64	128	256	256
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

	=5* 4MEG	=6 8MEG	=7 16MEG	=8 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	256	256	256	256
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 numbers one through eight represent partition size.

The first sector of each MSDOS 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 on page 2-14 of the MSDOS 2.0 Programmer's Reference Manual.

Lock Record

The lock record command is used to lock a record on the HiNet disk.

```
                        poll (50h)
  master     -------------------------->     user
  station              0.16ms               station

                   lock (19h,len,lck)
            <--------------------------
                       0.38ms

                    response (stat)
             -------------------------->
                       0.16ms
```

Total time (including turnaround): 1.7ms

The master returns a single status byte:

- 0h = accepted
- 1h = denied, lockstring already present
- 81h = already locked by current user
- 2h = denied, illegal string length (>13 or <1)
- 82h = denied, lock table full

The total number of lock strings available may be determined by the NetInfo protocol. This number is typically 64 for a 5" master and 128 for an 8" master.

Command Parameters	Parameter Length
len = length of lock string (1-13)	1 byte
lck = lock string	13 bytes
stat= lock status	1 byte

Unlock Record

The unlock command is used to unlock a record which has previously been locked.

```
                         poll (50h)
   master    ------------------------->      user
   station             0.16ms              station

                    unlock (1Ah,len,lck)
            <-------------------------
                       0.38ms

                     response (stat)
             ------------------------->
                       0.16ms
```

Total time (including turnaround): 1.7ms

A single status byte is returned:

0h	=	accepted
1h	=	denied, another user has it locked
2h	=	denied, illegal string length
82h	=	denied, lock string not locked

Command Parameters	Parameter Length
len = length of lock string (1-13)	1 byte
lck = lock string	13 bytes
stat= lock status	1 byte

Clear Locks

This command clears all locks which were created by this user. The "clear locks" command is executed whenever a station does a warm boot. The locks are also cleared if a station logs out. Exiting an MSDOS program will <u>not</u> clear locks.

```
                     poll (50h)
master     -------------------------->     user
station              0.16ms                station

                 clear-locks (1Bh)
           <--------------------------
                     0.16ms
```

Total time (including turnaround): 0.82ms

Hog the Network

The "hog" command allows a user station to gain temporary control of the network. Immediately after receiving a hog-acknowledgment, the user station may send or receive any number of messages on the network, as long as the total time is less than 16ms. This is enough time to send or receive approximately 1000 bytes. When the user station is finished, it should relinquish the network by sending an acknowledgment to the master station. If an acknowledgment is not received within 16ms, the master will assume that the station has failed, and will attempt to regain control of the network.

This command allows the user to set up his or her own network protocols. For example, this command can be used in conjunction with the WHO command to set up a direct interchange of information between any two stations on the network.

Note that the Hog protocol does not allow a station to talk to the master except to terminate the Hog sequence.

```
                          poll (50h)
   master         ------------------------->       user
   station                 0.16ms                 station

                           hog (18h)
                 <-------------------------
                            0.16ms

                      hog-deny (44h)  OR
                        hog-ack (48h)
                  ------------------------->
                            0.16ms

        (up to 16ms of station-station transmissions,)
           (such as Poll Prime, are inserted here)

                        poll-ack (41h)
                 <-------------------------
                            0.16ms
```

Total time (including turnaround): 2.14ms to 18.14ms

Poll Prime

Station to station transmission of 129 bytes of data is performed in the Z80 and 8086/8088 BIOS by means of the Poll Prime mechanism. Each user station wishing to receive poll prime data sets up a poll prime data block and informs the BIOS of its location (see "Poll Primes and BIOS Calls" document for more information). The BIOS automatically receives data from pseudo-user 251. Each user station wishing to send poll prime data must hog the network from the master, disguise itself as pseudo-user 251 while sending the data to another station, and relinquish the network back to the master with a poll-ack (41h).

```
          *** The hog prologue must be performed here ***

                             PPpoll (55h)
     sending     ---------------------------->      receiving
     station                   0.16ms               station

                             PPnack (57h)  OR
                             PPack  (56h)
     (user 251)  <----------------------------
                               0.16ms

                           PPdata (129 bytes)
                 ---------------------------->
                               2.21ms

                             PPnack (57h)  OR
                             PPack  (56h)
     (user 251)  <---------------------------
                               0.16ms

         *** The hog epilogue must be performed here ***
```

Total time (including turnaround): 4.19ms

Get HD Status

This HiNet protocol returns the status of a remote multiple hard disk (HD) subsystem. Capabilities for passing back status from both the master's remote hard disk and from a local hard disk have been provided.

See the accompanying "Poll Primes and BIOS Calls" document for Z80 and 8086/8088 BIOS calls to receive local and network hard disk status. The same format 136-byte status data is returned for all calls.

The protocol is as follows:

```
                      poll (50h)
  master      -------------------------->      user
  station               0.16ms                 station


                    HD status (1Dh)
                   <----------------
                        0.16ms

             136 bytes: 8 bytes + 128 bytes
             ------------------------------>
                        2.32ms

              HD status acknowledge (53h)
             <-----------------------------
                        0.16ms
```

Total time (including turnaround): 4.30ms

Get Date and Time

The current date and time maintained by the HiNet master may be obtained for those stations such as the DMS-1280 and 8086/8088 processors which do not have an internal real-time clock, and is available to all stations.

The protocol is as follows:

```
                           poll (50h)
  master      -------------------------->      user
  station               0.16ms                station


                       date/time (1Eh)
                      <----------------
                           0.16ms


       7 bytes: ticks, secs, mins, hrs, mon, day, year
       ---------------------------------------------->
                           0.26ms


                 date/time acknowledge (54h)
                <---------------------------
                           0.16ms
```

Total time (including turnaround): 1.58ms

Instant Logout

The "logout" command allows a user station to sign off from the master. The master ensures that the station does not request a user other than itself to logout, then logs out the station, releases the station's write ownership partitions and locks, and deletes any incomplete spool jobs.

```
                     poll (50h)
master   --------------------------->        user
station              0.16ms                  station


                logout (1Fh,usr)
         <---------------------------
                     0.17ms


                logout-deny (44h)  OR
                logout-ack  (41h)
         --------------------------->
                     0.16ms
```

Total time (including turnaround): 1.49ms

Write Mode

Stations must explicitly obtain write permission of an ownable partition to ensure that only one user on the network is capable of writing to that partition at any given moment. The write mode protocol allows a station to determine ownability of a partition, take and release that partition, check on its status and force a change if disaster should strike.

All write mode network calls should be invoked by the BIOS thru the BIOS jump vectors provided, because additional information regarding which logical drive letter owns a partition is stored and checked in each station.

```
                          poll (50h)
   master      ------------------------->      user
   station              0.16ms                 station

          write mode (20h,wmc,vol,dsk,val,usr)
         <-------------------------------------
                         0.24ms

                   cmddeny (4Fh)    OR
                 mesack (4Dh,stat,name)
         ------------------------------------>
                         0.30ms
```

Total time (incl. turnaround, excl. disk write): 1.7ms

Command Parameters	**Parameter Length**
wmc = write mode specific command	1 byte
vol = volume number	1 byte
dsk = unit number	1 byte
val = status value or logical user number	1 byte
usr = physical user number	1 byte
stat = returned write mode status	1 byte
name = user name	8 bytes

The write mode specific commands are as follows:

1	= grant	ownership to user <val>
2	= release	ownership from user <val>
3	= force	write status to <val>
4	= query	return current write status
5	= clear	all write ownership for user <val>

The returned write mode status values are:

```
FFh  = not owned
FEh  = unit is read/write
FDh  = unit is read-only
FCh  = unit is HiDos shared
FBh  = multi-grant (owned by you)
FAh  = generic error, illegal parameters
xx   = unit is owned by user xx
```

Network Information

The "netinfo" command allows a user station to receive up to 128 bytes of information about the configuration of the network from the master. Several fields are meaningful only when doing netinfo on the master.

```
                          poll (50h)
master        ------------------------->     user
station                0.16ms                station

                      netinfo (21h)
              <-------------------------
                       0.16ms

              netinfo table (128 bytes)
              ------------------------->
                       2.19ms
```

Total time (including turnaround): 3.51ms

Netinfo table format:

```
1 byte     table version number, (currently 1)
1 byte     number users max, (20h or 40h)
1 word*    address login user table
1 byte     currently 0
1 byte     number spool table entries, (currently 10h)
1 word*    address spool table
1 byte     number of lock strings max, (40h or 80h)
1 word*    address lock string table
remaining entries to 128 bytes reserved
```

* meaningful only when running on the master

Section 5: The HiNet BIOS Interface

The HiNet BIOS has several routines which can be called from a user program to send or receive data on the HiNet cable. Utility programs such as WHO, DIRNET, and ASSIGN use these routines to interface with HiNet. All four routines are accessible though jump vectors at fixed offsets from the base of the BIOS.

```
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 auto-
                    matic 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
```

The Jump vectors are continued on the next page.

(User station only)		
78h	ACKPOLL	Reactivate automatic poll acknowledgments.
(Master station only)		
78h	INTERRUPT	Process a one-second interrupt.

Offset	Name	Description
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 synchornize 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 pre-transmission delay will always be 500 microseconds while the timeout delay will always be about four seconds. The pre-transmission delay should give the intended recipient of a message more than enough time to re-program the DMA and SIO chips. So, 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 if error)
     ret
```

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 file 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 Z80 station: 47H. This number should be used to communicate with the master; it should never be changed.

```
          .
          .
;
; 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
```

Section 6: SENDNET And RECNET Listings

The SENDNET and RECNET subroutines are used to send or receive data on HiNet. Below is information on the subroutines. Listings of these programs follow.

1. The DMA chip is used to transmit or receive two or more data bytes, but is not used for one data byte. There are two reasons for this. 1: The DMA chip cannot be programmed to process only one byte. 2: Poll acknowledgments use only one byte because the DMA chip may be busy doing floppy disk operations when a poll interrupt is received.

2. The routines are written so that they will work properly on the DMS-3 and the DMS-4. The DMS-4 is approximately 20% slower than the DMS-3 due to Multibus access conventions. Some of the HiNet code is timing sensitive, and it is probably necessary to rewrite it if one wants to run these programs under any other timing conditions.

3. In SENDNET, an interrupt is generated when the DMA chip has finished transmitting all data bytes. The SIO chip cannot be turned off immediately because it still needs to transmit the two CRC bytes and several closing flags. The SENDNET routine waits for the CRCs to be transmitted, then it delays for a few dozen microseconds to force several closing flags. Now the SIO chip is turned off, stopping the transmission of flags.

4. In RECNET, an interrupt is generated when the first character of a message is received. If several data bytes are expected, the DMA chip, which will handle the reception of all remaining data bytes, is activated. Either the DMA chip or the SIO chip can cause an interrupt, at the end of a message. Both interrupts will vector to the same routine (RecLast), which will save the SIO status register and reset the SIO. If DMA is being used, the actual data length received by DMA will be saved (at which time the DMA chip is reset) for error checking later. If auto-poll acknowledgments are enabled, the poll is immediately acknowledged; otherwise, the status byte is set so that the RECNET code can know that a message has been received.

*Multibus is a trademark of Intel Corporation, Inc.
.page

```
 .sbttl        'SENDNET'

; ++++++++++++++++++++++++++++++++++++++++++++++
; +                                            +
; +      Network I/O Routines                  +
; +                                            +
; +    last modified>  12june84 jlw            +
; +                                            +
; ++++++++++++++++++++++++++++++++++++++++++++++

;
;----------
; Send a message to the master
;  Regs in:   A = message to be sent
;  Regs out:  none
;  Destroyed: A, BC, DE, HL
SENDMSG:
       lxi     H,NETmsg
       mov     M,A
       lxi     B,1
       sub     A         ; master number = 0
                         ; fall thru to SENDnet
;
;----------
; Transmit a block on the network
;  Regs in:  HL = block address
;            BC = byte count
;            E  = delay time (master only)
;            A  = user number
;  Regs out:  none
;  Destroyed: A, BC, DE, HL
; SendNet is extremely time-critical, particularly
; for DMS-4's, so modify it at your own risk.
SENDNET:

  .ifn MASTopt,[
       mvi    e,$halfms]; delay approx 1/2 mSecs

       inr     E         ; delay so that receiver has
..spin:dcr     E         ; time to prepare for message
       jrnz    ..spin

       mov     D,A       ; save user number
       shld    DMANSadr; store block address
       dcx     B         ; DMA chip wants real size - 1
       sbcd    DMANSsize
       mov     A,C
       ora     B
       di                ; no ints while reprogramming
       jrz     ..notDMA

; send a data block using DMA
```

```
..useDMA:
        lxi     H,DMATdone; set up the DMA vector
        shld    DMAvect
        mvi     A,1      ; multiplex SIO1B to DMA
        out     PIOAD

        lxi     H,DMANSprog; program the DMA chip
        lxi     B,DMAN$<8+DMA
        outir

        lxi     H,SENDprog
        lxi     B,SEND$<8+SIO1BC
        outir

        mvi     b,9
        djnz    .        ; force 3 leading flags

        lxi     h,sDMAflag
        mvi     m,0      ; reset dma done flag
        mov     a,d
        di
        out     sio1bd  ; send user number
        mvi     a,enaDMA
        out     DMA      ; enable dma chip
        mvi     a,rstEOM
        out     sio1bc  ; reset underrun flag (SIO)
        ei
..testDONE:
        mov     a,m      ; DMATdone sets sDMAflag
        ora     a
        jrz     ..testDONE
        jmpr    ..fin

; send one data byte so no need to use DMA
..notDMA:
        lxi     H,SENDprog
        lxi     B,SEND$<8+SIO1BC
        outir

        mvi     b,9
        djnz    .        ; force 3 leading flags

        lbcd    DMANSadr
        ldax    b
        mov     e,a      ; get message byte
        mvi     c,sio1bd
        di
        outp    D        ; send userno
        mvi     a,rstEOM
        out     sio1bc  ; reset under run flag
..wait:
        in      sio1bc
```

```
        bit     TxRdy,a
        jrz     ..wait
;
        outp    E        ; send message byte

; Handle the end of the transmission
..fin:
        mvi     B,20h    ; force CRC bytes
        djnz    .        ; and min 3 closing flags

        mvi     A,rstCHANNEL
        out     SIO1BC   ; reset the SIO chip
        ei
        ret
  .page
  .sbttl        'RecTime, RecMsg, RecNet, and NackPoll'
;
;---------------
; RECtime sets the timeout interval for RECNET
; Only assembled for a HiNet station
; Regs in:      BC = timeout constant
; Regs out:     none
; Destroyed:    none

  .ife station,[
RECtime:
        sbcd    Rtime
        ret

Rtime:
        .word   $4sec
        ]                ; end ' station '

;----------
; Receive a message from the master
;  Regs in:   none
;  Regs out:  A = xmission err status
;  NETmsg = receive message char
;  Destroyed: BC, DE, HL
RECMSG:
        lhld    USRadr
        mov     a,m      ; get user number
        lxi     H,NETmsg
        lxi     B,1      ; fall thru to RECnet

;----------
; Receive a block from the network
;  Regs in:  HL = block address
;            BC = maximum byte count
;            DE = timeout count (master only)
;            A  = user number
;  Regs out:  A  = error status
```

```
;                   bit 7 reset  = time-out
;                   bit 6 set    = CRC error
;                   bit 5 set    = receiver overrun
;                   bit 0 set    = poll only rcv'd
;  Destroyed: A, BC, DE, HL
RECNET:
        call    RECbegin  ; program the SIO chip

;----------
; Wait for next poll from network and don't ack
; ENTRY>        none
; EXIT>         see RECnet
;  Destroyed: BC, HL
NACKpoll:
        ei                ; make sure SIO can interrupt
        lxi     H,ACKflag ; point to ack/nack status
        mvi     M,1       ; don't ack polls (REClast)
        lxi     h,RECstat ; point to receiver status
        mvi     m,60h     ; init RECstat to errors

  .ifn MASTopt,[
        lded    Rtime     ; if slave, timeout is 4 sec
        ]
;
; Wait for block received from network cable
..wait:mvi      B,8      ; inner loop count
..loop:mov      A,M      ; check receiver status
        bit     4,A
        rnz              ; return if block received
        djnz    ..loop
;
        dcx     D        ; decrement timeout count
        mov     A,E
        ora     D
        jrnz    ..wait   ; fall through if timeout
;
        di               ; prevent real SIO int
        call    REClast  ; pretend we received a block
        res     pollRCV,m ; no valid poll rcv'd
        mov     a,m      ; return error status
        ret

;----------
; Program the SIO to acknowledge polls from the master
;  Regs in:   none
;  Regs out:  none
ACKpoll:
        xra     a
        sta     ACKflag ; let RecLast acknowledge polls
;
; SIO programming sometimes fails to take so inside
; ConStat keep a counter and force a call to AckPoll
```

```
; if it wraps around -  reset the counter upon every
; call to AckPoll.
;
  .ife station,[
       sta      failed2ack
       ]                ; end ' station '
                        ; fall into begAck:
;---------------------
; starts the poll acking sequence
begACK:
       lxi      h,NETmsg; put poll here
       lxi      b,1     ; receive one byte
       lda      NETusr  ; addressed to us

;----------
; Program the SIO chip to receive a block
RECbegin:
       sta      RECadr  ; station address
       shld     DMANRadr; DMA address
       dcx      B
       sbcd     DMANRsize; DMA rcv len = real len-1
       mov      A,C
       ora      B
       di               ; no ints while prog vects!
       jrz      ..prog
;
; Receive more than one byte
       lxi      H,REClast; set up the DMA vector
       shld     DMAvect
       mvi      A,1     ; multiplex SIO1B to DMA
       out      PIOAD
       lxi      H,DMANRprog; program the DMA chip
       lxi      B,DMAN$<8+DMA
       outir
..prog:
       lxi      H,RECfirst; set up interrupt vectors
       shld     SIO1vect+4
       lxi      H,REClast
       shld     SIO1vect+6
       lxi      H,RECprog; program the SIO chip
       lxi      B,REC$<8+SIO1BC
       outir
       ei               ; make sure interrupts enabled
       ret
  .page
;      Using the prom CONIN and CONOUT routines during
; the net boot prevents accessing location 47h (NETusr)
;      USRadr is initialized to NETusr because it
; assembles with station code as well.
;                       24MAR82MdG

USRadr:
```

```
        .word   NETusr

Nbootusr:
        .byte   0           ; User number is here
                            ; during net boot
ACKflag:
        .byte   0           ; ACKPOLL/NACKPOLL flag

sDMAflag:
        .byte   0           ; 0 = DMA not done

rcvDMAlen:
        .word   0           ; actual rcv'd block length

RECstat:
        .byte   60h         ; rcv status - init to timeout,
                            ; crc & ovr err - no poll rcv'd
REC.SP:
        .word   0           ; SP before interrupt

  .ife Station,[
RcvIntErrs:
        .byte   0           ; RecLast error count

failed2ack:
        .byte   0           ; counter for forced PollAck
        ]                   ; end ' not master '
  .page
  .sbttl        'SDLC interrupts'
;----------
; SDLC receive interrupt on first char
RECfirst:
        push    PSW
        push    h
        in      SIO1BD  ; flush user number
        lhld    DMANRsize; if len = 0, don't use DMA
        mov     a,l
        ora     h
        jrz     ..noDMA
;
; use the dma chip to receive a block of data
        mvi     A,enaDMA; enable DMA
        out     DMA
        jmpr    ..ret

..noDMA:
        in      sio1bd  ; get message byte
        lhld    DMANRadr; get buffer address
        mov     m,a     ; put message into buffer
        dad     h       ; done for timing
        in      sio1bd  ; flush first crc byte
```

```
..ret:
       pop     h
       pop     PSW
       ei
       reti
;
;----------
; SDLC receive interrupt on last char
RECLast:
       sspd     REC.SP
       lxi      SP,RECstck
       push     D
       push     PSW
       push     h
       push     b

       lxi      h,1<8+rstCHAN
       in       sio1bd  ; flush 1st crc byte if dma
                        ; 2nd if not dma
       mvi      c,sio1bc; hinet control port
       outp     h       ; get ready to read reg 1
       inp      a       ; read receive status
       outp     l       ; reset sio1b channel
       ani      0e0h    ; mask relevant bits
       set      4,a     ; no timeout
       sta      RECstat
;
  .ifn Master, [lxi    d,..begRet] ; in case of error

       lhld     DMANRsize; if len = 0, don't use DMA
       mov      a,l
       ora      h
       jrz      ..pastDMA
;
       lxi      h,ckDMAlen       ; program the DMA chip
       lxi      b,DMAlen$<8+DMA  ; to send actual rcv'd
       outir                     ; data length

       lxi      h,rcvDMAlen      ; store actual rcv'd
       lxi      b,2<8+DMA        ; data length at
       inir                      ; rcvDMAlen

       mvi      A,rstDMA         ; reset the DMA chip
       out      DMA

  .ifn Master, [lxi    d,..ret]; in case of error

..pastDMA:

    .ifn        MASTopt,[
; treat poll prime data block as special case
; - PPack or PPnack based on RecStat
```

```
        lhld    PPAadr
        mov     A,M     ; Get PP status byte
        cpi     2
        jrz     ..PPblk ; Jump if we got data blk
;
; if there was a network xmission error
; then don't respond to anything
        lda     RecStat
        cpi     90h     ; = EOM, no crc or ovr errors
        jrz     ..ckSize
;
        lxi     h,RcvIntErrs
        inr     m       ; errNet is used by NetErr
        xchg
        pchl
;
; If DMA size > 0, msg was neither a valid
; poll nor a valid poll-prime

..ckSize:
        lhld    DMANRsize
        mov     a,l     ; Non-zero if data block
        ora     h
        jnz     ..ret

; Check if msg. was a poll-prime
        lhld    DMANRadr
        mov     a,m     ; Get msg. byte
        cpi     POLLpr
        jrnz    ..ckPOLL; jump if not poll-prime

..isPrime:
; Reset internal time-out bit used by RECNET
        lxi     h,RECstat
        res     4,m
;
; Check to see if we are accepting poll-primes
        lhld    PPAadr
        mov     A,M     ; Get poll-prime status
        cpi     1
        lxi     b,1     ; len of msg (for SendNet)
        mvi     a,PPusr ; net adr (for SendNet)
        jrz     ..PrmAck; jump if acking poll-primes
;
        lxi     h,NetMsg
        mvi     m,PPnack
        call    SendNet ; PPnack the PPusr
        jmpr    ..begRET
;
; Update status byte
..PrmAck:
        mvi     M,2     ; Still pointing at PP status
```

```
; Send poll-prime acknowledge
        lxi     H,NETmsg
        mvi     M,PPack ; sending poll-prime ack
        call    SENDNET
;
; Call RecBegin to receive a PP data block
        lhld    PPAadr
        lxi     D,3
        dad     D       ; Point to PP data area
        lxi     B,PPmlen; PP msg. length
        lda     NETusr  ; Our user number
        call    RECbegin; Reprogram the SIO
        jmpr     ..ret

..ckPOLL:
; insure we received a poll
        cpi     poll    ; if poll, should we ack
        jrnz    ..begRET; ensure SIO is re-programmed

; Check to ack poll
..qACK:
        lxi     h,RECstat
        set     pollRCV,m ; flag for poll rcv'd
        lda     ACKflag
        rrc
        jrc     ..ret   ; Jump if no pollack wanted
;
; Ack the poll

    .ife        OptFlopAny,[
        lda     flopBUSY; don't answer if
        ora     a       ; flop is in use
        jrnz    ..skipACK
        ]               ; end ' OptFlopAny '
;
        mvi     A,pollack
        call    SENDMSG

..skipACK:
        call    ACKpoll
        jmpr    ..ret

..PPblk:
; Update PP status
        lhld    PPAadr
        lda     RECstat
        res     4,a     ; Pretend no poll rcvd
        ori     3       ; Indicate PP data blk rcvd
        sta     RECstat
        mov     M,A     ; Store PP status
;
```

```
; check for errrors on data block receive
        ani     80h+crc$ovr
        xri     80h     ; toggle timeout bit
        mvi     a,PPack ; ack reception
        jrz     ..blkResp
;
        mvi     m,1     ; ready to ack PPrimes
        mvi     a,PPnack; nack reception
;
..blkResp:
        lxi     h,NetMsg
        mov     m,a     ; send PPack or PPnack
        lxi     b,1     ; with msg len = 1
        mvi     a,PPusr ; to default PollPrime user
        call    SendNet
;
; Reprogram SIO for polls
..begRET:
        call    begACK
..ret:
        ]               ; end ' not MASTopt '

        pop     b
        pop     h
        pop     PSW
        pop     D
        lspd    REC.SP
        ei              ; ints ok now
        reti
;
;----------
; DMA transmit complete interrupt
DMATdone:
        push    psw
        mvi     A,rstDMA; reset the DMA chip
        out     DMA     ; and IEO line
        sta     sDMAflag
        pop     psw
        ret
  .page
  .sbttl        'pseudo numbers, network messages, commands'
;---------------
; ***  Pseudo-user numbers      ***
BOOTusr         ==      254     ; boot pseudo user number
LOGusr          ==      253     ; login pseudo user number
MIMICusr        ==      252     ; mimic pseudo user number
PPusr           ==      251     ; poll-prime pseudo user number

;----------
; Network messages and commands
poll            ==      'P'     ; poll from master
POLLpr          ==      'U'     ; poll-prime from user
```

```
pollack         ==      'A'     ; poll-received ack
PPack           ==      'V'     ; poll-prime-received ack
PPnack          ==      'W'     ; poll-prime received nack
datack          ==      'D'     ; data received ack
mesack          ==      'M'     ; message received ack
CmdDeny         ==      4fh     ; command deny
nack            ==      'N'     ; negative acknowledge
logack          ==      'L'     ; login acknowledge
lognack         ==      'N'     ; login conflict
logdeny         ==      'D'     ; login denied
lockack         ==      0       ; lock acknowledge
lockdeny        ==      1       ; lock denied
locknack        ==      2       ; lock error
notlocked       ==      82h     ; unlock not found
lockfull        ==      82h     ; lock table full
yourlock        ==      81h     ; lock already locked by you
hogack          ==      'H'     ; hog acknowledge
hogdeny         ==      'D'     ; hog denied
HSTack          ==      'S'     ; HD status rcve ack
DTMack          ==      'T'     ; date/time rcve ack
loutack         ==      'A'     ; instant logout ack
loutdeny        ==      'D'     ; instant logout deny
whoNET          ==      10h     ; who command
readNET         ==      11h     ; read command
writeNET        ==      12h     ; write command
loginNET        ==      13h     ; login command
startNET        ==      14h     ; start spool command
AckSpStart      ==      04h     ; Ack spool start req
read1NET        ==      15h     ; read 1K command
stopNET         ==      16h     ; stop spool command
assnNET         ==      17h     ; assign command
hogNET          ==      18h     ; hog command
lockNET         ==      19h     ; lock command
unlockNE        ==      1Ah     ; unlock command
clrlockN        ==      1Bh     ; clearlocks command
spoolNET        ==      1Ch     ; spool command
hdstNET         ==      1DH     ; hard disk volume status cmd
dttmNET         ==      1Eh     ; date/time status
loutNET         ==      1Fh     ; instant logout
WrtMode         ==      20h     ; generic Mode request
infoNet         ==      21h     ; request general net info
  .page
;
;-------------------------------------------------
; Network protocols:
;
;               MASTER      SLAVE
;               ------      -----
; Poll:         poll    -->
;                       <-- pollack
; ------------------------------------------------
; Write:        poll    -->
```

```
;                       <-- writeNET
;               mesack -->
;                       <-- DATA
;               datack -->
; --------------------------------------------
; Read:                  poll    -->
;                       <-- readNET
;               DATA    -->
;                       <-- datack
; --------------------------------------------
; Login:        poll    -->
;                       <-- loginNET
;               logack -->
;               (logdeny)
; --------------------------------------------
; Assign:        poll    -->
;                       <-- assignNET
;               DATA    -->
;---------------------------------------------
; Hog:          poll    -->
;                       <-- hogNET
;               hogack -->
;               (hogdeny).
;                       <-- pollack
;---------------------------------------------
  .page
  .sbttl        'sio and dma commands for network activity'
  .prntx        'made it to sio and dma commands'
; SIO commands
SENDprog:
        .byte      00011000b ; channel reset
        .byte   14h,00100000b ; SDLC mode
        .byte     3,00100000b ; auto enables
        .byte   11h,11000100b ; no interrupts
        .byte     5,11101011b ; transmitter enable
        .byte      10000000b ; reset CRC generator
SEND$           ==      .-SENDprog

RECprog:
        .byte      00011000b ; channel reset
        .byte     4,00100000b ; SDLC mode
        .byte     2,SIO1vect&0FFh ; interrupt vector
        .byte      01000000b ; reset crc check
        .byte   15h,10000000b ; transmit disable
        .byte     3,11111100b ; receiver info
        .byte   6
RECadr:.byte    0             ; user number
        .byte     7,01111110b ; flag byte
        .byte   11h,11101100b ; interrupt on first byte
        .byte   23h
RECable:.byte      11111101b ; enable receiver
REC$            ==      .-RECprog
```

```
;----------
; DMA commands
DMANSprog:
        .byte   0C3h     ; master reset              2D
        .byte   0C7h     ; reset port A              2D
        .byte   0CBh     ; reset port B              2D
        .byte   79h      ; write
DMANSadr:.word 0         ; filled by SENDNET or RECstart
DMANSsiz:.word 0         ; filled by SENDNET or RECstart
        .byte   14h      ; port A inc, memory        1B
        .byte   28h      ; port B fixed, I/O         1B
        .byte   95h      ; byte mode                 2B
        .byte   SIO1BD   ; port B
        .byte   12h      ; interrupt at end of block
        .byte   DMAvect&0FFh ; interrupt vector
        .byte   92h      ; stop at end of block
        .byte   0CFh     ; load starting address     2C
        .byte   writeDMA; write
        .byte   0CFh     ; load starting address     1A
        .byte   0ABh     ; enable interrupts         2D
DMAN$   ==      .-DMANSprog

DMANRprog:
        .byte   0C3h     ; master reset              2D
        .byte   0C7h     ; reset port A              2D
        .byte   0CBh     ; reset port B              2D
        .byte   7Dh      ; read
DMANRadr:.word 0         ; filled by SENDNET or RECstart
DMANRsiz:.word 0         ; filled by SENDNET or RECstart
        .byte   14h      ; port A inc, memory        1B
        .byte   28h      ; port B fixed, I/O         1B
        .byte   95h      ; byte mode                 2B
        .byte   SIO1BD   ; port B
        .byte   12h      ; interrupt at end of block
        .byte   DMAvect&0FFh ; interrupt vector
        .byte   92h      ; stop at end of block
        .byte   0CFh     ; load starting address     2C
        .byte   readDMA ; read
        .byte   0CFh     ; load starting address     1A
        .byte   0ABh     ; enable interrupts         2D

ckDMAlen:
        .byte   0bbh     ; set read status mask      2D
        .byte   06h      ; read the byte counter     2D
        .byte   0a7h     ; initiate read sequence    2D
DMAlen$         ==       .-ckDMAlen
```