11.0Introduction

DNOS V1.1 is a Dragon Network Operating System, providing many of the features found on a DragonDOS compatible disk system. It enables a number of Dragons (32 or 64) to be linked together to share resources found on one of the machines (namely disks and optionally a printer). The system provides a set of DOS equivalent commands on each network station, as if you were using a single disk based machine. This manual provides information on how to set up and use the system and a summary of the commands with any differences between them and those found on a standard DOS. In addition to emulating DOS remotely, it also provides the facility for 'network printing' and allows almost unlimited expansion of the system. (An example is shown in Annex 5 'DCHAT.BIN')

11.1How It Works

Each Dragon required on the network is equipped with a network 'card' comprising of the operating system on EPROM and the serial hardware (for more details refer to the Hardware Setup Manual), with the exception of one machine. This will be the DOS based machine, and therefore will not have an EPROM on the card, the software is based on disk. It is this machine, termed the 'File Server' which will effectively run the network. All the Dragons are linked together via cables, and talk to the file server. Information is passed over the cables to and from the disks on the file server machine to the Dragon 'network stations' when requested.

12.0Setting Up the System

12.1Network Stations

Once each station is equipped with it's network card and is connected up to the file server very little additional setup is required. Turn on a network station, and the normally copyright message should be displayed, followed by:

```
DNOS V1.1 (C)1991, 1992
        (C) COPYRIGHT BY J.BIRD
NETWORK STATION #n
```

n will be a number from 0 to 255. Each station on the network needs to be identified with a number. This is achieved by setting a bank of DIP switches on the network card. Network stations are numbered from 1 to 255 in numerical order. Therefore, if you have 3 stations (not including the file server), they should be numbered 1, 2 and 3. Set the DIP switches accordingly, and when switched on the station should correctly display the number assigned to it.

12.2File Server

The File Server also requires the DIP switches to be set, however this machine MUST be set as station 0 (zero - all the DIP switches off). If this is not the case, the software will not run.

If you are using the software for the first time, make a back up immediatly of the Network Disk, and in future use the copy. Insert the Network disk in drive 1 of the file server, and type:

```
RUN "CONFIG" <RETURN>
```

The following message is displayed:

```
DNOS NETWORK CONFIGURATION
(C)1992 BY J.BIRD
WARNING: THIS PROGRAM OVERWRITES
THE CURRENT NETWORK
CONFIGURATION SETTINGS
ARE YOU SURE
YOU WISH TO PROCEED (Y)ES/(N)O?"
```

Enter 'Y' and press RETURN.

The program will now request the following information:

SERIAL PORT ADDR (HEX)?" - enter the IO location of the serial port base address in hexadecimal on the server (see Hardware Manual).

NO OF DRIVES ON SERVER? - enter the number of disk drives the server supports (minimum 1). Any Network access to drives above this number will return an error. This is used to prevent access to non-existent drives, or to prevent network access to a given drive (eg. the printer spool drive).

NO OF NETWORK STATIONS(1-255)? - enter the number of network stations currently on the network. Stations with ID numbers above this will not be addressed.

SPOOL FILE DRIVE(1-4)? - enter the drive on which network print spool files are to be stored on (see network printing section). This may be a drive higher than the specified number of server drives specified earlier if required.

HEADER ON PRINTOUTS(Y/N)? - enter 'Y' if a header is required on network printouts, identifying the station sending and spool file ID (see network printing section).

PRINTER RESET SEQUENCE - enter the escape sequence required to reset the printer connected to the file server. Press RETURN after each character has been entered (up to six characters may be entered). As an example, the reset sequence for an Epson printer is 27,64 so enter 27 and press RETURN, 64 then press RETURN twice.

The setup information is stored in the file 'CONFIG.NET' which is read every time the Server program is run.

12.3Starting the Network

Switch the Dragon designated as server on, insert the network disk into drive 1 and type BOOT. After a few seconds of disk activity, the screen is cleared and the following message displayed:

The server conducts a short test of the Network hardware, before loading the operating system off the disk. Once complete, the settings available for this version of

DNOS are displayed:

```
DRAGON NETWORK OPERATING SYSTEM
DNOS V1.0 (C)1991,1992
WRITTEN BY J.BIRD

64K SERVER VERSION
MACHINE ID: 0
DRIVES    : 2
STATIONS  : 2
SPOOL     : SPOOLXXX.PRN
SWITCHABLE: NOT SUPPORTED
RAM DRIVE : NOT SUPPORTED
SERVER(S) : NOT SUPPORTED
RUNNING.
```

No. of drives and stations will be dependant on the data entered in the config file. Server version will be 32 or 64K depending on machine type (Dragon 32 or 64).

The server is now fully functional.

If the server program fails in any way, an appropriate error message is displayed, and the program is aborted. Problems which can occur and possible solutions are outlined below:

CANNOT FIND CONFIG.NET FILE
CRITICAL ERROR, INIT FAILURE - CONFIG.NET file not present on disk, run the CONFIG program to create file.

?? **HARDWARE ERROR**
PROGRAM ABORTED - error on testing the ACIA network chip. Ensure the correct base address is entered in the CONFIG.NET file and refer to Hardware Manual to ensure your network card is fully functional.

CANNOT FIND SERVER.BIN FILE
CRITICAL ERROR, INIT FAILURE - The server operating system file is not present on the disk.

INCORRECT NOS VERSION
FOR THIS SERVER PROGRAM - A later version of the software has been identified within the CONFIG.NET file that this server does not support. Only use the new server program. Possible CONFIG.NET file corruption - run the config program to create a new one.

THIS VERSION OF DNOS DOES NOT
SUPPORT MULTIPLE FILE SERVERS
THE SERVER MUST BE CONFIGURED
AS MACHINE 0 - An attempt to set the DIP switches on the server network card to anything but 0. Ensure DIP switch setting is set to 0 and a network card fault does not exist (Refer to Hardware Manual)

Switch on the Dragon network stations, and the network should now be fully operational.

13.0Using DNOS

DNOS adds a set of BASIC commands to the Dragon, which are almost identical to those found under DragonDOS or compatible system. Programs which use DOS commands (including machine code files) should run under DNOS without alteration. A full list of DNOS commands are listed in Annex 1. There are, however a number of commands which are not supported. These fall into two catagories:

13.1Network Security

Certain standard DOS commands can be considered non-viable on a network. An example of this is the DSKINIT command. Where multiple users are sharing a disk, and one user has the ability to re-format the whole disk, the result would be disastrous. Therefore the following commands cannot be performed from a network station (or from machine code calls):

DSKINIT, BACKUP, SWRITE

When executed, these commands will return an ?FC ERROR.

13.2DNOS V1.1

This version of DNOS does not provide the following commands:

CHAIN, CREATE, FLREAD, FREAD, FROM, FWRITE

When executed, these commands will return an ?FC ERROR.

Later versions of DNOS should provide these commands, and ensure full compatability with DOS of existing commands.

13.3DNOS Workspace and Graphics RAM

Like DragonDOS, DNOS workspace occupies page 1 of graphics memory at 1536 ($600) to 3071 ($BFF) on network stations.
The file server utilises the second graphics page for workspace, a full memory map of file server and network station is listed in Annex 3.

14.0Data Files

Data files are handled in much the same way as DragonDOS 3.0+ handles them, through the OPEN, PRINT & INPUT (similar to the tape system) commands. However, only 3 file channels are available (1-3), and one is used if you have a network print spool file open (see Network Printing section). The CLOSE command always refers to a stream number, not a drive or last file open. These file channel restrictions are further reduced since the file server has only ten channels available in total for all network users. The LOC command for manipulating the read pointer, can now only reference a stream number.

14.1Machine Code Character I/O

Single character input and output can be performed using the standard BASIC character handling calls. Location 111 ($6F) must be set to the file control block number. To output a byte, load 'A' with the byte and call JSR [$A002]. To input a byte call JSR $B50A and the byte is returned in 'A'.

15.0Network Printing

DNOS provides the ability for any network station to use a printer attached to the file server machine as if it were attached to the station itself. Standard BASIC commands PRINT #-2 and LLIST are used as normal, but data is sent to the server instead of out the local parallel port. When DNOS is first switched on, printing is in local mode ie. PRINT #-2 data is sent to the parallel port (or serial port if configured on a D64). In order to 'connect' to a network printer type:

NPRN <Enter>

The machine is now effectivly re-routed to the file server. In order to revert back to normal use the LPRN command. The first effect of this, is that by typing:

PRINT #-2,"DNOS"

will return a ?NO ERROR. This is because, data is not sent to the server and out to the printer, it is stored on disk in a spool file ready for printing later. Like normal files, you must OPEN and CLOSE the spool file for printing. So, by entering:

OPEN "O",#-2,"" (no filename is necessary)

will open up a spool file on the server. Subsequent PRINT #-2 or LLIST commands will have the data stored in the spool file on the server. The drive it is stored on must have enough capacity to cope with the files (particularly graphics screen dumps, which can be quite lengthy) and is set by the user through the server CONFIG program Spool Drive option. Printing via the network also obeys the standard printer locations (ie. End of Line Sequence at 330, LF flag etc.)

When you wish to commence printing, type CLOSE #-2 (or CLOSE to close all files), and the server will add your spool file to the printer queue. Note, that a side effect of the LLIST command is that it CLOSEs the print channel before returning to BASIC.

Note that when you open a spool file, you lose one of the file streams (1-3) available. You will no longer be able to directly access this file stream through the use of PRINT#, INPUT# etc.

15.1Print Queues

When a print spool file is closed by the user, it is added to the print queue. This is a list of files awaiting printing. The first file will commence printing as soon as it enters the queue, subsequent files must wait until the printer is free. Since there may be a number of queue entries, an option through the server CONFIG program is provided to add a small header to each of the printouts to enable identification of the sender. There is no method of assigning priority to print spool files within the queue, they are sent in the order they were closed by the user.

15.2Network Printing from Machine Code

Printing to a network printer can be achieved through the normal BASIC character out routine, provided the channel has been first opened by a BASIC OPEN command. The machine code routine should not attempt to OPEN or CLOSE a print channel directly or data will be lost (see Annex 2 for DNOS assembler calls). When printing a character, set location 111 ($6F) to 254, and call JSR [$A002]. Do not use the JSR $800F call to print a character, as this accesses the hardware directly and DNOS has no way of intercepting this call. This may cause problems with a number of machine code programs (eg. screen dumps) which use the $800F call. A possible way to correct this is to search through the file for $7E, $80,$0F (mnemonics for JSR $800F) and replace them with a jump to your own code:

```
ie. LDB #254
   STB >$6F
   JSR [$A002]
           CLR >$6F
```

This will output the character correctly to either the network or local printer depending on which is selected.
This will also remain compatible when running on a non-network machine

If it is necessary to OPEN and CLOSE a print spool directly, then it can be accomplished with normal assembler calls by opening a file named 'SPOOL.PRN'. Refer to the File Control Block number assigned to this file when sending data to print.

16.0Additional Errors

In addition to the full set of DOS errors (listed in Annex 6), a new error (code $A8 - ?NT Error) may be returned by network routines. This indicates Network Timeout (or Not There!), communication between the server was interrupted unexpectedly or could not be established. Network 'requests' to and from the server have a time out figure attached which is decremented every time the hardware tries to fetch a byte from a machine. If this figure reaches zero, then an NT error is returned. If DNOS routines continually return this error, then it is likely that a hardware fault exists. Refer to the Hardware setup manual.

16.1Getting Used to working on a Network

DNOS is designed to make the network station feel as if it is a standard DragonDOS Dragon. However because you are sharing a set of disks with other users their actions can effect you. For example, when trying to open a file you may get an ?AO error (file already open). There are two instances when this could occur. In the first, another user is accessing this file and you cannot. The second is slightly more rare, the request to open a file has 'retried'. On any command issued to the server, DNOS allows 5 retries if any transmission error or timeout should occur. If a request to open a file is successful on the server, but fails on the returned status block DNOS will retry. In this case, the server will flag the file already open and a ?AO error will be returned. It is important to ensure files are properly closed on a DOS machine, even more so on a network. DNOS maintains the status of a file (OPEN or CLOSEd) on both the server and network station. If one of these is corrupted, you may be in a position where your station believes you have files open, when in fact they are shut. A worse situation is that the station believes the file is closed, when in fact it is open. You should therefore ensure that ALL files are closed before switching a machine off, and entering a CLOSE command after an error has occured through a network access.

ANNEX 1
Dragon Network Operating System
Extensions to BASIC

The following lists the commands and functions provided by DNOS 1.1, through which BASIC accesses the network, with a definition and description of each. It also describes any differences to standard DOS counterparts. Square brackets are used to denote optional parameters, and DEFD is used to indicate the default drive number. The drive number can be between 1 & 4, but may be limited by the CONFIG.NET NumDrvs setting on the server. Drives specified by Network Stations above this will return a ?DN error.

AUTO

Format: AUTO [start[,incre]]

Defaults: start = 100, incre = 10

AUTO enables automatic program numbering when entering a program. When executed, by default it starts numbering from line 100 and increments in steps of 10. If a line already exists, a question mark will be displayed in front of the line number.

BACKUP

Format: N/A

Defaults: N/A

The BACKUP command has not been implemented, since it violates network security. Refer to section 3.2 on security restrictions.

BACKUP DIR

Format: N/A

Defaults: N/A

The BACKUP DIR command has not been implemented, since it violates network security. Refer to section 3.2 on security restrictions.

BEEP

Format: BEEP [number]

Defaults: number = 1

BEEP outputs the specified number of 'beeps' (max 255) from the Dragon.

BOOT

Format: BOOT [drive]

Defaults: drive = DEFD

BOOT loads sectors 3 to 18 of track 0 of the disk in the specified drive into memory from 9728 onwards; sets DEFD to 'drive' and executes the program from address 9730. The first two bytes of sector 3 **must** be OS.

NOTE: The BOOT command is primarily used to load another operating system (eg. OS9). Use on a network station will crash the machine since such operating systems require direct access of the disk system. The command should only be used for loading programs which remain in the BASIC enviroment eg. auto BASIC loaders.

CHAIN

Format: T.B.A.

Defaults: T.B.A.

The CHAIN command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

CLOSE

Format: CLOSE [stream [,stream[, ...]]]

Default: All streams

The file allocated to the specified control block number is closed. If the stream is -2 then then (if open) the print spool file is closed and queued for printing.

eg. CLOSE - closes all streams including print spools

CLOSE #-2 - close the print spool file, and queue for printing

COPY

Format: COPY source file TO destination file

Default: drive is DEFD unless specified

COPY creates a duplicate of the source file specified and names it destination file.

CREATE

Format: T.B.A.

Defaults: T.B.A.

The CREATE command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

DIR

Format: DIR [/] [drive]
DIR [/][#stream [,drive]]

Defaults: stream = 0
drive = DEFD

DIR lists the files on a disk in the specified drive. The output gives the files with their protection (a lower case **'P'** if write protected), file length, followed by the number of files on the disk and the amount of free space available in bytes. If the stream is 0 (default = the screen), then the output scrolls slowly. Use of the '/' parameter enables 12 line paging, pausing for a keypress. Output to the printer will be in 3 columns, unless the '/' parameter is specified, in which case data is output in a single column. The '/' parameter is ignored for all other stream numbers. BREAK can be used to abort the listing.

DRIVE

Format: DRIVE drive

The DRIVE command sets the DEFD number (location 1577) in the range 1 to 4. This will not check any restrictions of drive access on the server. At power on the default is 1.

DSKINIT

Format: N/A

Defaults: N/A

The DSKINIT command has not been implemented, since it violates network security. Refer to section 3.2 on security restrictions.

EOF

Format: E = EOF(stream)

The EOF function returns the status of an open file 'stream'. EOF returns 0 (false) if the file has more data to be read or +/-1 (depending on location EOFLAG =1635) (true) if the end of file has been reached. The file must have been opened by an OPEN command eg:

```
IF EOF(2) THEN CLOSE #2
```

closes file channel 2 when the end of file is reached

NOTE: DNOS V1.0 does not support with format E=EOF(filespec) since the FREAD/WRITE commands have not been implemented in this version. Refer to section 3.2 on non-implemented commands. The value returned by EOF is set by the EOFLAG location. In the default mode, EOF returns -1 (for cassette compatibility), in order to return +1 (for DragonDOS mode), use POKE 1635,254 (and 0 to reset back to cassette mode).

ERL

Format: E = ERL

The function ERL returns the BASIC line number which the last error occured (-1 if it occured within command mode). This is normally used when setting up an error trap (see ERROR GOTO and ERR)

ERR

Format: E = ERR

The function ERR returns the error code of the last BASIC error. This is normally used when setting up an error trap (see ERROR GOTO and ERL)

A full list of error codes is listed in Annex 6

ERROR GOTO

Format: ERROR GOTO line number

ERROR GOTO sets up an error trap routine. If a BASIC error occurs within the program, control is passed to the 'line number' within the last ERROR GOTO statement. Multiple ERROR GOTO statements can be used within a program. Error trapping is turned of by the RUN command, by specifying a line number of 0 or when an error occurs. FOR/NEXT loop information is cleared when an error occurs, so control cannot be passed back to inside the loop since an ?NF error will occur. (See also ERR and ERL).

FLREAD

Format: T.B.A.

Defaults: T.B.A.

The FLREAD command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

FREAD

Format: T.B.A.

Defaults: T.B.A.

The FREAD command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

FRE$

Format: F = FRE$

The function FRE$ returns the number of bytes of stringspace unused. This is increased by the CLEAR command.

FREE

Format: F = FREE [drive]

Default: drive = DEFD

The FREE function returns the number of unused bytes on the disk specified. This value is in multiples of 256 bytes (complete sectors) and does not include unused space in sectors at the end of files.

FROM

Format: T.B.A.

Defaults: T.B.A.

The FROM command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

FWRITE

Format: T.B.A.

Defaults: T.B.A.

The FWRITE command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

HIMEM

Format: T = HIMEM

HIMEM returns the highest memory location available to BASIC (at power on this is 32766). This can be altered via the CLEAR command to move the top of memory down.

INPUT

Format: INPUT #stream, variable list

Default: stream = 0

INPUT is used to read data in from disk in the same way as from a cassette file. The stream must be opened via the OPEN command. Data are separated by 'end of line' characters and except when between string quotes, commas and colons. Like cassette and keyboard, INPUT reads at least one complete record (ie. up to an 'end of line' character) from disc and discards and unused data after reading in the variable list. The LOC command can be used to manipulate the read pointer.

Example:

INPUT #2,B$

KILL

Format: KILL filespec

KILL is used to delete files from the a disk, releasing space used by the file for re-use by another file.

NOTE: The KILL / global delete is not implemented in this version of DNOS.

LINE INPUT

Format: LINE INPUT #stream, string variable

Default: stream = 0

LINE INPUT reads complete records from disk into a string variable. Input is completed by an 'end of line character'. The stream number must have been opened explicitly via an OPEN command. The LOC command can be used to manipulate the read pointer.

LLIST

Format: LLIST [startline[-endline]]

Default: entire BASIC program

LLIST is used for outputting a BASIC program to the printer. In 'local mode' the default at power up, or set by the LPRN command data is sent to the parallel port. If an NPRN command has been issued, and a print channel OPENed data is sent to the printer on the server. If the print channel is not opened an ?NO error is returned.

NOTE: LLIST closes the print channel on completion.

LPRN

Format: LPRN

LPRN is used to set the printer to 'local mode'. BASIC print commands then revert back to the parallel (or serial port if a Dragon 64 is in use). This is the default setting at power up, and need only be used after an NPRN command.

NOTE: LPRN is a new command. Therefore it should be used in command mode only, in order to maintain compatability with programs that may be run on a DOS machine. Use of the OPEN #-2 and CLOSE #-2 can be used within programs which will be run on a non-network station, since they are ignored by BASIC.

LOAD

Format: LOAD filespec [,binary load addr]

Defaults: file extension = .BAS
binary load addr = start address used when the program was SAVEd

LOAD transfers the contents of a BASIC or binary file into memory from disk. If a BASIC program is being loaded, any existing program will be erased.

If the file is a binary program, the EXEC address is set to the value when the file was saved. The optional offset is used to override the load address, in which case the EXEC address is adjusted by the same offset. Segmented file addresses are also offset by the binary load addr. The EXEC address is taken from the terminating null segment.

NOTE: DNOS V1.0 does not support the LOAD/ command for loading ASCII BASIC files.

LOC

Format: L = LOC(stream)
LOC(stream) = value

LOC returns the value of the read pointer or sets the read pointer to the specified value. The stream must be opened by the OPEN command.

NOTE: DNOS V1.0 does not support with format LOC(filespec) since the FREAD/WRITE commands have not been implemented in this version. Refer to section 3.2 on non-implemented commands.

LOF

Format: L = LOF(filespec)
L = LOF filespec

LOF returns the length of the file in bytes. If the file is not opened, then it will be opened by this command but NOT closed.

MERGE

Format: T.B.A.

Defaults: T.B.A.

The MERGE command has not been implemented in this version of DNOS. Refer to section 3.2 on non-implemented commands.

NPRN

Format: NPRN

NPRN is used to set the printer to 'network mode'. BASIC print commands are sent through the network card and output on the printer attached to the server. However, the print channel must be OPENed before printing can commence, and CLOSEd once complete.

NOTE: NPRN is a new command. Therefore it should be used in command mode only, in order to maintain compatability with programs that may be run on a DOS machine. Use of the OPEN #-2 and CLOSE #-2 can be used within programs which will be run on a non-network station, since they are ignored by BASIC.

OPEN

Format: OPEN mode,#stream,filespec

Default: file extension = .DAT

OPEN is used to allocate a stream number to a disk file, and allow file manipulation by standard BASIC commands (PRINT#, INPUT# etc.). The format is similar to that used by cassette files:

MODE	NAME	ACTION
'I'	Input	The file must exist, but is not restricted to read
'O'	Output	A new file is creaed, and any existing file is backed up
'A'	Append	An existing file is opened, or a new one created
'E'	Empty	Any existing file is killed and a new file created

Example: OPEN "O",#2,"DATA"

creates a new file named 'DATA.DAT', and renames any existing file to 'DATA.BAK'. Stream number must be in the range -2 to 3.

NOTE: This command is also used to open a print channel to the network printer:

OPEN "O",#-2,""

The print channel is closed by the CLOSE #-2 command.

PRINT

Format: PRINT [#stream,][USING string;][output list]

Default: stream = 0

PRINT is used to output data to a disk file in the same way as cassette. The file must be expressly opened via an OPEN command, and terminators are not normally written between the variables output. If **CMFLG** (location 1634) is set to the stream number of the file then 'cassette mode' is enabled and 'end of line' characters inserted between variables output.

Example: PRINT #3,A$,B$

writes the strings out to the disk file associated with stream 3.

NOTE: PRINT#-2 is used for outputting a data to the printer. In 'local mode' the default at power up, or set by the LPRN command data is sent to the parallel port. If an NPRN command has been issued, and a print channel OPENed data is sent to the printer on the server. If the print channel is not opened an ?NO error is returned.

PROTECT

Format: PROTECT [option] filespec

Default: option = ON

PROTECT alters the write protect setting of a given file. When set by the option ON, the file cannot be written to or deleted. A lower case 'P' is displayed against files that have this setting in the directory listing. The setting OFF clears the write protect setting.

RENAME

Format: RENAME filespec1 TO filespec2

RENAME will change the name of the file in 'filespec1' to that in 'filspec2'. Both files must refer to the same disk drive, and the second must not exist.

NOTE: DNOS does not support the RENAME #drive option, since the network disks are public to all, and should be set locally on the DOS machine.

RESTORE

Format: RESTORE [line number]

Default: line number = first program line

RESTORE allows the setting of the pointer used by the READ command to the specified line. Future READ statements will take data from the first DATA statement occuring on or after this line. The line need not actually contain a DATA statement, but must exist. Line numbers referred to by RESTORE will be renumbered by RENUM.

RUN

Format: RUN [filespec[,number]]

Default: File extension = .BAS
Number = first line (BASIC) or
default load address (Binary)

RUN allows a program to be loaded from disk and automatically executed. The option number is used to signify the line number of the program to start execution from or the load address for a binary program. The program is run from the default EXEC address plus any offset calculated by the optional load address.

SAVE

Format: SAVE filespec[start,end,entry]

Defaults: File extension = .BAS (BASIC)
.BIN (Binary)

The SAVE command is used to write a BASIC program or area of memory to disk. If no parameters are supplied, the BASIC area is saved as file type 1, else the memory block is saved as type 2 file. The SAVE command cannot write a type 3 segmented file.

SREAD

Format: SREAD drive,track,sector,string1,string2

SREAD transfers a 256 byte sector specified by the track and sector parameters into the two strings specified with 128 bytes in each.

SWAP

Format: SWAP variable1,variable2

SWAP exchanges the contents of two variables (of the same type). They can be strings, numeric or array elements.

SWRITE

Format: N/A

Defaults: N/A

The SWRITE command has not been implemented, since it violates network security. Refer to section 3.2 on security restrictions.

VERIFY

Format: VERIFY [option]

Default: option = ON

This command is used under DragonDOS to turn on and off disk verification. However, this cannot be set by a network user and therefore is ignored. It will, not return an error though and will check for valid options (ON or OFF).

WAIT

Format: WAIT [delay]

WAIT delays the execution of a BASIC statement following the WAIT command for the specified number of milliseconds (1000ms = 1 sec). SHIFT @ can be used to pause the WAIT command indefinatly.

ANNEX 2
Dragon Network Operating System
Facilities for Assember Programmers

This Annex lists the primary routines used for access over the network. They take the form of an indirect jump table found on network stations and in most cases mimic the DOS routines found at these locations, thus providing compatability for programs which use these calls on DOS machines. Each routine is headed by its entry table address and function. The routines are called via an indirect jump eg.

JSR [$C004] - calls the NETCOMM command processor.

The routine also defines:

registers/locations for entry to the routine

registers/locations on exit of the routine

Assume ALL registers are modified by the routines.

Although most routines are the same as their DOS counterparts, there are some minor differences, where network security will not allow access of the command (eg. format) and in this case the routine returns code 8 (?FC error) in the B register when called.

At the end of the Annex, the server routines are listed for adding to the network command structure.

[$C004]
Network Command Processor (NETCOMM)

This routine replaces the Basic disk operation processor on DOS machines. It is used by all DNOS calls to issue a network request to the file server. It should be used by Network Users for setting up their own network commands. The routine sends up to 256 bytes of data stored in buffer at address 2048, along with certain control information (defined in the operation block pointed to via [$C006] and returns up to 256 bytes of data (and any relevent control information) back from the server in the same operation block. (See Annex 4 for examples and description on using this command).

Entry: Operation Block set up (see [$C006])

NETCOMD values 1-16 are reserved for system usage.

Exit: Operation Block set up (see [$C006])
B = Error code (0 if no error)
Z flag clear if an error occurs

[$C006]
Operation block address for [$C006] operations

+0=Network Command Byte (NETCOMD). Code issued by the network station for the command the server is to perform. 1-16 are reserved for system use, others are user definable.

+1=Network Error (NETERRO). Error code (if any) returned from server. Any error returned in the 'B' register from NETCOMM takes priority, since this data may be invalid if a NETCOMM error occurs.

+2=Network Stream Number (NETSTRM). Stream number currently in use. Set by the station.

+3=Network Length (NETLEN). Number of bytes in buffer at 2048 ($800). Set by the station on sending, and by the server to determine the number of bytes returned. (minimum 1 byte, 0=256 bytes).

+4=Network Pointer (NETPTR). 3 byte file pointer.

[$C008]
Copy File Details and Validate

This provides the facility to validate a file specification, and add a default drive number or extension if they are not included within the file specification.

Entry:B = File specification length (<15 characters)
X = Address of file specification
Y = Address of default file extension

Exit:B = Error code (0 if no error)
Z flag clear if error occurs

$650-$657	= Filename
$658-$65A	= File extension
$65B	= Drive number

This routine is identical to it's DOS counterpart, and does not access the network.

[$C00A]
Get Directory Entry and Copy to Control Block

This routine takes the file details stored within $650-$65B, and attempts to assign a file control block number to it (stream) (or ?TF Error occurs). If one is obtained, the directory is searched for the filename, and if found the control block is loaded with details from the file. ?NE Error will be returned if the file is a new one or could not be found.

Entry:File specification in $650-$65B

Exit:B = Error code (0 if no error)
X = Address of read pointer within control block
A, NETCBK = Control block number
   DSKDRV = Drive number
Z flag clear if an error occurs

NETCBK = $F1
DSKDRV = $EB

This routine is identical to it's DOS counterpart.

[$C00C]
Create Directory Entry

This routine creates a directory entry for the specified file, renaming an existing file to .BAK if necessary.

Entry:A = File Control Block Number (control block set up from [$C00A].

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

This routine is identical to it's DOS counterpart.

[$C00E]
Get File Length

This loads the logical sector number (no of sectors used for the file) and number of bytes used for the last file. This figure is equivalent to the 24 bit file length in registers U and A.

Entry:A = File Control Block Number

Exit:B = Error code (0 if no error)
U = Number of next sector to be written (upper 16 bits)
A = Number of bytes used in the sector (lower 8 bits)
DSKDRV   = Drive number in control block
Z flag clear if an error occurs

This routine is identical to it's DOS counterpart.

[$C010]
Close all files on a drive

This routine is not accessable from a network station, since files on the specified drive may be in use by another user.

Entry: N/A

Exit:B = 8 (?FC Error)
Z flag clear

[$C012]
Close a file

Closes the file specified, and if the last file on the drive update the directory bit map.

Entry:A = File Control Block Number

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

This routine is identical to it's DOS counterpart.

[$C014]
Load a file block into memory

Up to 64K bytes can be loaded into memory from the disk file specified. The start byte number + number of bytes to load must not exceed the file length.

Entry:A = File Control Block Number
U,B = 24 bit byte number from which loading is to start, B is least significant 8 bits
X = Memory start address
Y = Number of bytes to load

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

This routine is identical to it's DOS counterpart

[$C016]
Write buffer to file and verify

Up to 64K bytes can be written to a disk file from memory The start byte number must not exceed the file length.

Entry:A = File Control Block Number
Y,B = 24 bit byte number from which writing is to start, B is least significant 8 bits
X = Memory start address
U = Number of bytes to write

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

This routine is identical to it's DOS counterpart

[$C01A]
Kill a file and free sectors for re-use

The file associated with the control block number specified, will be removed from the directory and the sectors allocated, returned for re-use. The file control block will not be closed.

Entry: A = File Control Block Number

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

This routine is identical to it's DOS counterpart

[$C01C]
Set File Protection

The write protect setting for the file will be altered depending on the contents of the B register.

Entry:A = File Control Block Number
B = 0 (protect off), <>0 (protect on)

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

Note: This routine does not return the Directory Record address in the X register or the Buffer Details Block address in the Y register, since this information is not available within the file control blocks on a network station.

[$C01E]
Rename a file

The directory entry specified by the File Control Block number will be changed to the new file specification. No checks are made to ensure that the drive numbers match, nor that the new file does not exist. To avoid duplication of file names, [$C00A] should be called first with the new file details, and ensure that ?NE is returned.

Entry:A = File Control Block Number

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

Note: This routine does not return the Buffer Details Block address in the Y register, since this information is not available within the file control blocks on a network station.

[$C020]
Get Directory Record

This routine fetches the specified directory record (0 to 159), into memory.

Entry:B = Directory record entry number (0 to 159)

Exit:B = Error code (0 if no error)
X = Address of record
Z flag clear if an error occurs

Note: This routine does not return the Buffer Details Block address in the U register, since this information is not available within the file control blocks on a network station.

[$C022]
Find a free buffer and read absolute sector

This call has not been implemented. Sector reads can be issued via the [$C026] Extended Absolute Sector call.

Entry:N/A

Exit:B = 8 (?FC Error)
Z flag clear

This call returns an error code, which the calling program should intercept appropriatly.

[$C024]
Copy directory sectors from track 20 to track 16

This call has not been implemented. Under normal DragonDOS this occurs automatically, later versions must be executed on the file server via this call or BACKUP DIR.

Entry:N/A

Exit:B = 8 (?FC Error)
Z flag clear

This call returns an error code, which the calling program should intercept appropriatly.

[$C026]
Extended Read Absolute Sector

This routine, under DOS compatibility returns the requested absolute sector specified in the Y register to a buffer address specified in the X register. On a network station, it also provides direct access to the Basic Disk Command Processor, Read/Seek physical sector calls to read a sector by specifying the track/sector numbers. By setting the X register to 0 (an invalid buffer address for the absolute sector call since it will point to system workspace) this format of the call can be used.

Entry:X = Buffer address
Y = Absolute sector address
DSKDRV = Drive number

or

Entry:X = 0000
Y = Buffer address
A = Track Number
B = Sector Number

Exit:B = Error code (0 if no error)
Z flag clear if an error occurs

Note: Read physical sector is an extension to this call, not found on DOS stations. If a program which makes use of this call, is run on a normal DOS machine, it will crash.

[$C028]
Write Absolute Sector

This call has not been implemented since it violates network security. Network users cannot write sectors to disk, since they may inadvertantly destroy another users data.

Entry:N/A

Exit:B = 8 (?FC Error)
Z flag clear

This call returns an error code, which the calling program should intercept appropriatly.

[$C02A]
Verify Absolute Sector

This call has not been implemented. Use [$C026] to verify a sector.

Entry:N/A

Exit:B = 8 (?FC Error)
Z flag clear

This call returns an error code, which the calling program should intercept appropriatly.

[$C02C]
Format Disk

This call has not been implemented, since it violates network security. Network users cannot format a shared media, thus destroying data that may be in use by another user.

Entry:N/A

Exit:B = 8 (?FC Error)
Z flag clear

This call returns an error code, which the calling program should intercept appropriatly.

[$C02E]
Base address table of disk error code character pairs

The two character NOS error code may be obtained by subtracting 128 from the code number (returned in the B register in the DNOS routines above) and adding to it the address at [$C02E]. This only applies to disk errors.

File Server Assembler Calls

When adding commands a network station via the NETCOMM call at [$C004], a routine must also exist on the server to deal with the data requested, and return a block back. (See Annex 4 for a full discussion on adding network commands).

When the file server boots up, it scans the disk in drive 1 for a file named 'NOSPLUS.BIN'. If found, the file is loaded and EXECed by the program. This should contain the routines necessary to implement the response to additional commands which can be issued by the network station. This routine should be split into two parts

1) Initialisation: Add the address of the response routine two the dispatch table (via [$1201])

2) Main program: Perform the function requested by the network station, and return via [$1203]

[$1201]
Address of User Dispatch Table

[$1201] points to the start of the user dispatch table. The address of the 1st server response routine should be stored at the address pointed to by this location, the next routine should use this address +2, the next +4 etc.

[$1203]
Return Network Control

The response command should leave via a jump to this address, after setting up the return block data:

$3EC - any error code you wish to return to NETCOMM from the server.

$3EE - number of data bytes being returned in the buffer at $C00 to the network station.(minimum 1 byte 0=256 bytes).

$C00 - 256 byte network return buffer. Any data the routine is to return to the network station is stored here.

ANNEX 3
Dragon Network Operating System Workspace

DNOS Workspace(All Stations)

Address		Length	Contents
Hex	Dec		
Page 0 Locations			
00E6	230	3	24 bit Network Block Timeout
00E9	233	1	Timeout flag
00EA	234	1	Disk Drive Number (DSKDRV)
00F1	241	1	Current Control Block Number (NETCBK or DSKCBK)

Page 3 Locations

```
03EB1003  68 byte Network Header Block
03EB1003  1Command Request Byte (COMRQST)
03EC1004  1Error number (ERROR)
03ED1005  1Current Control Block Number (STRMNO)
03EE1006  1Network Block Length (BLKLEN)
03EF1007  3File Pointer (FILEPTR)
03F21010  11 byte checksum value (CHKSUM)
```

Network Station only

```
06001536  1Network Command. (used by NETCOMM)
06011537  1Network Error. (used by NETCOMM)
06021538  1Network Stream (used by NETCOMM)
06031539  1Network Buffer length (used by NETCOMM)
06041540  3Network File Pointer (used by NETCOMM)
06071543  1Temporary 1 byte buffer
06081544  7File Control Block 1
060F1551  7File Control Block 2
06161558  7File Control Block 3
061D1565  1Stream status byte. Bottom 3 bits only: set = stream in use.
061E1566  4Temporary Use
06221570  2Current load address for LOAD/SAVE
06241572  2Current file length for LOAD/SAVE
06261574  2Pointer to current file control block
06281576  1Last command sent to server
06291577  1Default drive number (DEFD)
062A1578  1Number of retries used
062B1580  2Default block size for LOAD/SAVE blocks
062D1582 20USR vector table (moved from $0134)
06411603  1LOAD/RUN flag
06421604  2Count of sectors within a file
06441606  1Page count byte
06451607  1Directory record count number
06461608  1ERROR trap flag (0=off, $FF=on)
06471609  2ERROR destination line
06491611  2Line number in ERROR
064B1613  1ERROR type
06501616  8Current filename
06581624  3Current file extension
065B1627  1Current drive number

065C1628  1Temporary record pointer
065D1629  1Count of files in DIR
065E1630  1Network Printer Flag (0=local, <>0=network)
065F1631  1Stream number assigned to network printer
06601632  2Spool buffer pointer
06621633  1One bytes I/O buffer
06631634  1Cassette mode flag (CMFLG)
06341635  1EOF flag (EOFLG)

08002048256Network Buffer
0A002560256Network Print Spool buffer
```

File Control Blocks, 3*31 bytes from $608

+0/33 byte read pointer
+4Not used
+5/73 byte write pointer

File control block maintains DOS pointer compatability (write pointer remains 4 bytes forward from read pointer - address returned in X via [$C00C]).

File Server only
Overview

00000256System Page 0
0100256256System Page 1
04001024512Text Screen
060015361536DOS workspace
0C0030724608DNOS workspace
120146097680DNOS Operating System
1E017681Server Runtime BASIC program
2400 9216 Free RAM
7918310001767Reserved for NOSPLUS.BIN server extender
80003276832768BASIC ROMS, DOS & IO

Workspace
Page 1 Location

03F51013 1Number of Network stations attached to file server

Graphics Page 2 Locatiions

0C003072256Network Buffer
0D003328 20Network Stream/DOS stream lookup table
0D143348 1Current Network Station number
0D153349 2Current file length for LOAD/SAVE or amount of RAM available for COPY
0D173351 1Temporary 1 byte buffer
0D183352 2Pointer to current read/write pointer
0D1A3354 2Default block size for LOAD/SAVE
0D1C3356 1Number of retries/Control Block number of file to COPY from
0D1D3357 1Command last sent/Control Block number of file to COPY to
0D1E3358 1Number of entries in print queue
0D1F3359 1DOS control block number of spool file
0D203360 1Not used

0D213361 1Next spool file ID
0D223362 2Pointer to spool file read pointer
0D243364 3Spool file length
0D273367 1Spool buffer pointer
0D283368 1Spool drive number
0D293369 1Number of network supported drives
0D2A3370 1Banner flag (0=no banner, <>0 banner)
0D2B3371 1Current spool file ID
0D2C3372 20Printer queue
0D423393 2Server ACIA base address
0D443395 2Length of file to COPY from
0D463398 2Pointer to write pointer of file to COPY to
11004352256Print spool buffer

Network stream/DOS lookup table, 2*10 bytes from $

Provides a DOS Control Block Number to Network Station Control Block Number/Network ID lookup table: (0 if not currently assigned)

+0Station ID associated with DOS Control Block 1
+1Station ID associated with DOS Control Block 2
...

+10Network stream associated with DOS Control Block 1
+11Network stream associated with DOS Control Block 2
...

Network Printer Queue, 2*10 bytes from $0D2C:

Lists up to 10 queue entry IDs, following by up to 10 network station IDs associated with the print job.

Network Card Port Addresses

The base address for the file server is determined by location $0D42, whilst on the stations it is at $FF40.

Base Address +

+0Data Register (MC6850)
+1Command (write) and Status (read) Registers (MC6850)
+2Network ID (read) byte and Data Direction (write) bit (bit 0)

Dragon Network Operating System
Command Expansion

Use of the NETCOMM Command Processor

This Annex details how a user may add network commands to the standard DOS equivalent commands. The program DCHAT utilises this method and is detailed fully in Annex 5.

DNOS as it is supplied usses the system to add DOS commands to a diskless station. However there are numerous usses that the system can be put to use as once a number of Dragons are connected together. A network command involves principally 3 stages:

1) Issue command to server

2) Server performs command required

3) Results of command sent back to station

As an example, this annex will detail how to obtain the number of entries currently in the network queue.

1) First, use the NETCOMM call at [$C004] to issue the command to the server. The parameter block is pointed to by [$C006]. In this instance only two parameters are relevent; Command ID byte & Network Block Length. The new command ID is 17 (1-16 are used by DNOS itself), and since no data needs to be passed to the server, the Block Length can be set to the smallest value 1 (0=256 bytes):

```
GETQUEUELDX $C006- X points to parameter block
LDA #17- A new command ID number
STA ,X
LEAX 3,X- move X to point to Block Length
LDA #1
STA ,X- Block length of 1
JSR [$C004]- issue the command
```

The server will return the number of entries in the queue in the Network Buffer at 2048 (assuming no error occured), which we can then deal with.

```
BNE ERROR- test for error
CLRA
LDB 2048- load the number of entries
JMP $????- & output the byte to screen
ERRORJMP $8344- system error routine
```

This routine provides all the necessary coding for a Network Station. By assembling, and EXECing (provided the server code has been implemented), a figure between 0-10 will be displayed corresponding to the number of entries in the network printer queue.

2) When the server program boots up it scans the boot drive for a file named 'NOSPLUS.BIN'. This file must contain your new code for adding to the server's command table, with the default exec address pointing to the install routine. The install routine simply adds the address of your new routine to the command

table, pointed to by [$1201].

```
INSTALLLDX #GETQUEUE- address of new routine
LDY $1201- point to next free slot in dispatch table
STX ,Y++- install the routine
STY $1201- ready for next command
RTS
```

The server routine 'GETQUEUE' must, when called by the system return the number of queue entries in the buffer, and leave via [$1203].

```
GETQUEUELDA 3358 - location for number of entries in queue
STA $C00 - put into buffer
LDA #1   - 1 byte to return
STA $3EE - number of bytes in buffer location
JMP [$1203] - return via return call
```

The routine is now complete. Note: When calling NETCOMM a timeout delay is initiated for response from the server (about 10s). If the routine you have written takes longer than these, then NETCOMM will timeout and return an error code $A8 (NT - Network Timeout) on the network station.

Network Errors: The above routine does not make use of the NETERRO location for returning server errors, since the server routine cannot fail. This location is primarily used when DOS routines fail, and the error code is passed back to the network station. Note, that the error code returned in the 'B' register (if any) from the NETCOMM processor is <u>not</u> the value stored in NETERRO, this code reflects any serial problems and takes priority over any returned codes since data may be invalid. Therefore in order to test for this at the station use code similar to:

```
JSR [$C004] - call NETCOMM
BNE ERROR   - test for NETCOMM error
LDB NETERRO - fetch any returned error code
BNE ERROR   - test for server return error
```

ANNEX 5
Dragon Chat
(DCHAT)

Dragon Network Chat
(DCHAT)

DCHAT is an add on program supplied on the network boot disc. It comprises 3 programs, DCHAT.BIN, DCHAT.BAS SCHAT.BIN. In order to use the program, SCHAT.BIN should be renamed to NOSPLUS.BIN so that the server program will automatically load the program. Network stations simply LOAD and EXEC the DCHAT.BIN program. The DCHAT.BAS program is an optionally CHAT enviroment making use of the extra commands DCHAT.BIN adds to network stations. In order to use one of your own server add-ons, rename the file back to SCHAT.BIN.

DCHAT allows up to 10 users connected on the network to communicate with one another via eight extra BASIC commands. Messages can be sent & recieved by the commands, as well as automatically displayed on the screen as they arrive. The commands are summarized below:

1. MESSG ON/OFF

The MESSG command enables or disables the ability to send and recieve on-screen messages. By default, messages are disabled when the package is started. Typing MESSG on its own displays the current status on or off. When switched off the SHOUT and SAY commands are disabled. Other users attempting to contact you will be informed that you are no longer on the message system. Because of the way the network is setup, you may notice pauses every so often as the IRQ routine attempts to update any on-screen messages. If you are working on something important and this becomes irritable then turning MESSG OFF will stop the IRQ routines accessing the server.

2. CHAT ON/OFF

CHAT performs a similar function to MESSG, except it enables or disables the CHAT commands DROP & PICKUP. Typing CHAT on it's own displays the current status (defaults to OFF). Other users attemting to contact you will be informed that you are no longer on the CHAT system.

3. IAM <string>

The IAM command assigns yourself an alias of <string>. You are identified on the network by your station number (shown when a network station is switched on). You can send and recieve messages by use of this number. In addition to this you can give your station a name by use of this command. For example to call yourself 'FRED' use:

```
IAM "FRED"
```

Note: The alias string must be 10 characters or less, else an ?LS error is produced.

4. WHOIS <n>

WHOIS displays the alias of user <n>. If <n> is 0 then the entire known user list is displayed. If <n> is greater than 10 then a warning message is displayed.

eg. to find out who user 1 is:

```
WHOIS 1

1  FRED
```

and to list all users:

```
WHOIS 0

1  FRED
2  BORIS
4  NORMAN
```

5. DROP #<n>/<alias string>,<messg string>

The DROP command leaves a message for another user to PICKUP later. If there is insufficient room in their CHAT buffer, then a warning is displayed and the message is not sent. DROP can use a station ID or it's alias to identify a user. The station ID must be in the range 1-10, the alias string must be up to 10 characters, and the messg string up to 64 characters. If the station cannot be found or CHAT has been disabled via the CHAT ON/OFF command then a warning is displayed and the message is not sent.

eg. DROP #1,"WHAT'S HAPPENING FRED?"

leaves the message specified for user 1, likewise:

```
DROP "FRED","WHAT'S HAPPENING FRED?"
```

has the same effect.

6. PICKUP <messg string>

PICKUP retrieves the next message awaiting you on the server, and stores it in the string specified. PICKUP will display a warning message if you have disabled CHAT. If PICKUP is executed within a program and there are no messages waiting, a null string is returned else in command mode a warning message is displayed.

Example: PICKUP A$

```
PRINT A$
```

```
BORIS[2]:WHAT'S HAPPENING FRED?
```

7. SAY #<n>/<alias string>,<messg string>

SAY sends <messg string> to user <n> or alias <alias string> and displays it on their Dragon's screen. The alias string must be 10 characters or less, the messg string 64 characters or less. If the user cannot be found, MESSG has been disabled on the receiver's machine or the receiver's MESSG buffer is full, a warning is displayed and the message is not sent:

```
SAY "FRED","WHAT'S HAPPENING FRED?"
```

8. SHOUT <messg string>

SHOUT displays the message specified on all Dragon's connected to the network which have MESSG enabled and have free space in their MESSG buffer.

Messages sent via SAY and SHOUT will appear on the receiver's screen almost immediatly depending on what the machine is doing:

```
MESSG FROM BORIS[2]:WHAT'S HAPPENING FRED?
```

Chat Warning Messages:

The CHAT system does not force BASIC errors due to CHAT not being enabled, or user's not found so they can be incorporated into a BASIC program without aborting when a problem occurs. Errors generated through the serial hardware (such as I/O & NT errors can occur as normal). The CHAT software displays the following warning messages:

```
CHAT/MESSG BUFFER FULL FOR USER n
```

The storage space used for messages on the server for the specified user is full. The receiver has not issued a PICKUP recently, or has disconnected from the network without issuing a CHAT/MESSG OFF command.

```
CHAT/MESSG NOT ENABLED FOR USER n
```

The CHAT or MESSG system is not enabled for the user specified. If the user id refers to your station ID, turn on the system via a CHAT or MESSG command.

```
USER NOT FOUND ON NETWORK
```

The specified user requested is not identified as being attached to the network, or the alias specified could not be found.

```
NO MORE MESSAGES
```

Returned by PICKUP in command mode, when there are no more messages to read.

USER HAS NO ALIAS

Returned by WHOIS when the station identified has not been assigned an alias.

EXCEEDED MAXM NO OF CHAT USERS

This error occurs when attempting to communicate with a station with an ID of above 10, or when starting the DCHAT package on a station with an ID above 10.

DCHAT usses the method described in Annex 4 to add to the Network commands, and usses NETCOMD codes 17-22. The server overlay program requires 14K and resides at address 13000.

DCHAT.BAS

The DCHAT.BAS program is a chat enviroment program which makes use of the extra commands DCHAT adds to BASIC. Each station loads a copy of DCHAT which allows a full screen chat system via the DROP & PICKUP commands. If DCHAT.BIN is not already loaded, the program will do so before running.

ANNEX 6
Summary of Error Messages

HexDecCodeDescription

000NFNEXT without FOR
022SNSyntax error
044RGRETURN without GOSUB
066ODOut of DATA in READ
088FCIllegal Function
0A10OVMath overflow
0C12OMOut of memory
0E14ULUndefined line number
1016BSBad array subscript
1218DDAttempt to dimension an array twice
1420/0Divide by zero
1622IDIllegal direct statement
1824TMType mismatch
1A26OSOut of string space
1C28LSString too long
1E30STString formula to complex
2032CNCannot continue
2234UFUndefined function
2436FDFaulty data file
2638AOI/O device already open
2840DNDevice number outside limits
2A42IOI/O error (tape/serial/network)
2C44FMFile mode error
2E46NOFile not open
3048IEEnd of file reached on input
3250DSDirect statement
3452NEDLOAD file does not exist (D64)

DNOS error addtions:

80128NRDisk not ready
82130SKTrack seek error
84132WPDisk write protected
86134RTRecord type incorrect
88136RFRecord (track/sector) not found
8A138CCCyclic redundancy check error(disk/network)
8C140LDLost data, data not transferred to/from disk
8D141DBNo duplicate directory track
8E142BTBOOT failure
90144IVInvalid directory
92146FDDirectory full
94148DFDisk full
96150FSInvalid filename
98152PTFile write protected
9A154PEPast end of file
9C156FFFile not found
9E158FEFile already exists
9F159ENEntry number above directory limit
A0160NEDisk file does not exist
A2162TFMaxm number of files open (network or server)
A4164PRParameter error
A6168??Unknown error
A8170NTNetwork timeout