nascom

NASCOM DISC OPERATING SYSTEM

NAS-DOS1

SERIAL NO.D.....

NAS-SYS/NAS-DOS UTILITIES

Application Note No AN - 007

Nascom Microcomputers
Division of Lucas Logic Limited
Welton Road Wedgnock Industrial Estate
Warwick CV34 5PZ
Tel: 0926 497733 Telex: 312333

Lucas Logic

IMPLEMENTATION NOTES

1. Whenever reference is made to a drive number, the logical drive and not the physical drive number should be given. In a double sided system one side is logical drive 0 and the other logical drive 1. This method will allow compatibility between single and double sided disks, and allow mixed single and double sided drives on the system. A system which contains two single sided drives will therefore have logical drive numbers 0 and 2 if the drives have been configured for physical drives 0 and 1.

2. The prefix to disk commands in conversational mode has been chosen as], which is conveniently placed in the top right-hand corner of the keyboard of the Nascom 2. Nascom 1 owners will probably find it more convenient to select another character. The character chosen should be one which is not normally output to the first column of the display screen, due to the way in which NAS-DOS commands are detected. The ASCII value of the character chosen should be placed in memory location 0D06 hex using the NAS-SYS M command whenever NAS-DOS is cold-started.

3. Owners of ZEAP or other utility programs which are on tape can load these programs into memory normally from tape. The programs can then be saved on disc using the NAS-DOS commands described in this manual. Owners of ZEAP (the assembler/editor package) in EPROM must remove the ZEAP EPROM'S since these occupy the same memory addresses as NAS-DOS. A conversion pack is available to allow registered EPROM ZEAP owners to convert these EPROM's to allow ZEAP to be relocated in RAM and loaded onto disc for future use.

4. NAS-DOS is normally supplied located in memory at D000. Special versions located at alternative memory locations are available on application to your dealer. An additional charge is made for these special versions. We strongly recommend that these alternatives should only be used by OEM's with special requirements, since all future applications software will be based only on standard versions of NAS-DOS.

Contents

1. Introduction

NAS-DOS has been specifically designed to enable Nascom users to add disk drives to their system with the minimum of alterations to existing software. Enhancements to existing standard products are made, such as the 4.7 ROM Basic, NAS-PEN and ZEAP, so that a two byte command may be entered to save any of the sources. There are two modes of operation:

1. Conversational mode, which allows the user to type commands from NAS-SYS or Basic.

2. Program mode, so that the user may perform disk operations from programs. The screen display is not affected whilst in program mode.

All necessary software to operate the Disk Operating System is in the 4K of EPROM supplied. This allows fast operation.

With the exception of the data file read/write and format routines, no additional RAM from the standard machine is required as all workspace is contained within the areas 0D00H to 0EFFH. This will allow a small machine code program to operate from 0C80H to 0CFFH. The system monitor should be NAS-SYS 1 or later, compatible, versions.

NAS-DOS may be considered to be in three distinct parts:

a) D000H to D3FFH - Floppy Disk Controller software (FDC)

b) D400H to D7FFH - Conversational mode

c) D800H to DFFFH - Intelligent Disk Operating System

As can be seen each section falls upon a 1K boundary, so that users may alter any part to to suit their special requirements.

2. Installation

1. NAS-DOS is supplied normally in 4 off 2708 EPROM's, which must be located in memory addresses D000 to DFFF.

2. In a Nascom 2 these EPROM's can be located in bank A or bank B of the on-board sockets or on a RAM-A board or on an EPROM card. In a Nascom 1 the EPROM's must be located on either a RAM-A or EPROM card. Please refer to the appropriate manuals in order to set the links and switches correctly to set the addresses of these four EPROM's at D000 - DFFF.

3. Special versions of NAS-DOS are available contained in 2 off 2716 EPROM's and/or located at different memory addresses. The remainder of this manual refers to the standard version located at D000 - if your copy is located elsewhere you must adjust addresses referred to in this manual accordingly.

4. When fitting EPROM's please handle them with care, and avoid touching any of the pins. Make sure that each EPROM is in the correct position (particularly if you use bank A of Nascom 2 memory), that it is inserted the correct way round and that ALL the links and switches are correctly set.

5. When fitting NAS-DOS, the I/O switch LSW2 (SW8) on the NASCOM2 board should be UP, (or if links are fitted a link should be made between pins 18 and 28).

3. Execution

NAS-DOS is executed by typing 'ED000' (or just 'D' from NAS-SYS 3) for a cold start, or by typing 'ED003' for a warm start. It may also be cold started by altering the restart address so that NAS-DOS is initialised whenever the Reset button is depressed. This can be done by placing switched 1,3 and 4 on LSW1 in the down position. It is important to note that during a cold start all NAS-DOS workspace is initialised. During a warm start only some is set. A warm start should not be attemeted if a cold start has not been executed since power-up or if it is thought that the workspace may have been corrupted.

Before it is possible to use a disk with NAS-DOS, it is necessary to format a disk. Please refer to the format routine in Conversational mode.

It is strongly advised that back-up's (i.e. the copying of one disk to another) are done on a regular basis to prevent the loss of data.

4. Conversational Mode

List of commands

]B(d)(:filename:)	Save Basic program
]C(d)	Clean directory (Remove deleted entries)
]D(d)	Display the directory
]E(d):filename:	Execute a program
]F(d):diskname:	Format disk
]L(d):filename:	Load program/file, but do not execute
]O(d):filename:llll hhhh eeee	Save object code program
]P(d)(:filename:)	Save NAS-PEN word processor file
]Rllll nnnn tt ss d	Read directly from disk
]S(d):filename:	Scratch (delete) a file
]U(d)	Load user boot
]V(d):filename:	Verify that a file can be re-read
]Wllll hhhh tt ss d	Write directly to disk
]Z(d)(:filename:)	Save ZEAP source file

All values shown in brackets are optional.

If no drive number is input, zero will be assumed.

(d) shows the drive number as optional, drive zero assumed.

Description of Commands

The following description of commands shows the optional fields in brackets where, for example, '(d)' would mean that the drive number need not be input (drive zero will be assumed).

JB(d)(:filename:) - Save Basic program

This command will save the current Nascom 4.7 ROM Basic source program from 1000H to the contents of memory locations 10D6H and 10D7H so that when the program is re-executed using the JE command or whenever the program is chained to, the program will commence to run immediately. All variables are lost during the saving process. If the program name to be given to this source file is not specified on the parameter line, the first line of the Basic source program is searched. If the first byte of this line is a 'REM' instruction, the rest of the line is searched for the file name within colons.

As the contents of memory locations 1000H to 10D5H are saved, it is not necessary to cold start Basic prior to executing the program, unlike the CLOAD command. However, due to this fact, the amount of memory available to the Basic program must not decrease between the save and the reload process of the program. If it has decreased, reload the program with the 'JL' command from NAS-SYS. Inspect the contents of memory locations 10D6H and 10D7H which contains the address of the last byte in the source file. Call this address 'x' (remember least significant byte first). Using the 'JD' command save the memory locations 10D6H to 'x' having an execution address of FFFDH. Cold start Basic. Execute the program just saved. Resave it with the 'JB' command.

JC(d) - Clean Directory

Whenever a file is deleted or over-written, the directory entry is flagged and not actually removed from the directory. The directory will, therefore, require to be 'cleaned' periodically to remove the deleted entries otherwise a 'Directory full' error will result. This command should not be used for any program which resides or uses areas 1000H to 3000H as this area is used as work area for the clean. A 'Sure' message accompanies this command.

*D(d) - Directory display

Initially, the directory header is accessed to display the name allocated to the disk during the last format. This is known as the Identification or ' Id'. NAS-DOS will then display each directory entry in the sequence created (unless the directory has been sorted). All deleted directory entries are skipped. The file name is displayed together with the load address (where the program is re-read), the the execute address (where control is passed to), the starting track and sector and finally the length of the file in sectors. All values are in hexadecimal. Twelve entries are displayed at a time after which the ESCAPE key (shift and enter simultaneously) may be depressed to end the command. Any other key will display the next twelve entries until the whole of the directory is displayed at which point the number of free sectors are displayed, again in hexadecimal.

The directory is displayed in such a way that the execute command and drive may be inserted in front of the entry to enable easy execution of the program.

*E(d):filename: - Execute a Program

A program held on disk may be reloaded and executed with this command. The contents of ARGS are set in the NAS-SYS workspace so that the program will behave as if one argument, the execute address, had been input.

*F(d):diskname: - Format a disk

This command will format a disk so that the disk may be used with NAS-DOS. A directory is created on the disk together with a directory header which will contain the name of the disk specified within the colons. A skew factor can be specified to optimise disc response in special applications - normally a value of 1 is used.

*L(d):filename: - Load program/file (but do not execute)

The specified file name is loaded into memory and then control is returned to the calling program. Ensure that the program to be loaded does not overwrite the memory locations used by the program initiating the command. If in doubt return to NAS-SYS. It is possible to load a basic program from within Basic.

*O(d):filename:llll hhhh eeee - Save object code

This command will save the contents of memory from locations llll to hhhh-1 having an execution address of eeee with the specified file name. Address eeee may be D003H, the warm start address of NAS-DOS, for example.

JP(d)(:filename:) - Save NAS-PEN source

The current NAS-PEN source file is saved. If the file name to be given to this source file is not specified on the parameter line, the first line of the source file is searched for a colon. If found the source file is saved with the file name following the colon. If not control is passed to the error routine. The execution address will be B806H. It is not necessary to cold start NAS-PEN prior to re-loading this file.

JR1111 nnnn tt ss d - Read into memory

The read command will read nnnn sectors into memory location 1111 from track tt, sector ss from drive d. The area of disk that is being read need not be under the control of a NAS-DOS file.

JS(d):filename: - Scratch a file

This command scratches (deletes) the specified file name from the directory, so that the disk space may be reallocated to other files. A 'Sure?' prompt accompanies this command.

JU(d) - Load User boot

This command loads the User Boot program that exists on track 0, sector 1. The program is loaded into memory location 1000H and executed. This program must not be longer than one sector in length. The 'W' command may be used to put the user boot on disk. A simple user boot example listing is provided at the end of the documentation.

JV(d):filename: - Verify file can be re-read

The file specified is read to ensure that the file can be re-read into memory. A scratchpad area below 1000H is used for this operation.

JW1111 hhhh tt ss d - Write to Disk

The contents of memory from 1111 to hhhh are written to track tt, sector ss on drive d. No check is made to ensure that the operation would overwrite a file under NAS-DOS control and no directory entry is created.

JZ(d)(:filename:) - ZEAP source file save

The current ZEAP source file is saved. If the file name is not specified in the parameter line, the first line of the ZEAP source file is examined. If the first character of this line is a semi-colon the rest of the line is searched. If a colon is found, the source file is saved on disk with the file name following this colon. It is necessary to cold start ZEAP prior to the reload of the source file, unless it is known that the ZEAP workspace area (0F00H on) is still intact.

Examples

1) Save a Basic program on drive 1 called 'PORTSTAT'. The first line of the Basic program is thus:-
1000 REM Port status display :PORTSTAT:
Type JB1 from NAS-SYS or Basic

2) Save above program with a different file name of 'PORTTEST' on drive zero
Type JB:PORTTEST:

3) Clean directory on drive 2
Type JC2 from NAS-SYS

4) Display directory on drive 0
Type JD

5) Execute program PORTSTAT from drive 3
Type JE3:PORTSTAT:

6) Format disk with an identification (Id.) of WORKDISK
Type JF:WORKDISK:
To have no name...
Type JF::

7) To save object code from 1000H to 2000H to execute at 1003
Type JO:ZEAP:1000 2000 1003

8) To write 1000H to 1003H to track 0 sector 1 drive 2
Type JW1000 1003 0 1 2
REMEMBER 256 bytes will actually be written
The command is identical to JW1000 1100 0 1 2

9) To scratch program 'ADDRMENU' on drive 0
Type JS:ADDRMENU

Method of operation

Conversational mode is located from D400H to D7FFH in the supplied firmware and therefore occupies 1K of ROM. Entry can be made into conversational mode by typing ED000 from NAS-SYS (or just D from NAS-SYS 3) to cold start or ED003 to warm start. It may also be entered by configuring the restart jump switches 1, 3 and 4 on LSW1 on the main board so that NAS-DOS is executed whenever the Reset button is depressed.

During a cold start to NAS-DOS the contents of the field 'USRMCP' are inspected. If the first byte of this three byte instruction is a Z-80 JUMP instruction, control will be passed to the address following the jump instruction. This field is reinitialised to point to the normal routine in NAS-DOS once examined.

The command identifier (normally ']') may be altered by inserting the new command character value at 0D06H. This field is known as CMDCHR.

To execute a command, simply type the command identifier followed by the command and the relevant arguments. Then type <ENTER>. The command identifier must be the first character on any line. Potentially destructive commands will cause a 'Sure' prompt to which the user must reply 'Y'. Any other key will abort the command.

NAS-DOS corrupts £UOUT in the NAS-SYS workspace so that all characters output to the screen first pass through conversational mode. If a printer routine does likewise and NAS-DOS output is required on the printer, the printer routine should be altered so that at the end of the routine it passes control to D405H instead of executing a return instruction.

Conversational mode can be disabled by typing 'N' from NAS-SYS.

Conversational mode workspace

Conversational mode requires some RAM workspace. This occupies memory locations 0D00H to 0D17H (24 bytes), some of which are defined below.

Addr	Len	Name	Description
0D00H	3	USRMCP	User Master Control Program
0D03H	3	WFDC	Jump to cold start DOS
0D06H	1	CMDCHR	Command character (normally ']')
0D07H	2	WCMTAB	Start of command table. This field is similar to £STAB in NAS-SYS

5. Disk Operating System

The second part of NAS-DOS is the 'intelligent' part. Conversational mode accesses the disk totally via this software. Access from 'program mode' is activated by calling the Assembler Routine Handler (ARH) and following the call with a one byte call code. For example:-

```
DOSARH EQU  #D800
£CHAIN EQU  3
.
.
.
       CALL DOSARH
       DEFB £CHAIN
.
.
```

The Basic Routine Handler (BRH) is initiated by setting USRLOC (1004H, 4100) to point to DOSBRH (D806H, -10234) and then calling routines with the A=USR(n) command, where 'n' is the desired routine.

Workspace for this section of NAS-DOS runs from 0D20H to 0EFFH inclusive, which includes an area for the Dynamic Disk Space Manager (DDSM) and a directory buffer. As the number of sectors per track, and the number tracks per disk are bound to vary, only some of the workspace has been released. All of the stated addresses of these fields are fixed and shall remain compatible between future versions of NAS-DOS wherever possible.

Hex.	Dec.	Name	Description
0D20	3360	£WDRV	Drive number to be accessed
0D21	3361	£WFILE	File name address
0D23	3363	£WEXEC	Execution address
0D25	3365	£WSECT	Random sector number for BRH
0D27	3367	£WSIZE	Maximum number of sectors to allocate during £PREP
0D29	3369	£USDIO	User disk I/O error routine
0D2C	3372	£WDIRP	Directory pointer
0D2E	3374	£WDIRS	Next Directory sector number to be read
0D2F	3375	£WDDRV	Last Directory drive number to be accessed
0D30	3376	£WONME	Current output file name (if open)
0D38	3384	£WIBP	Input file pointer within buffer
		£WFPIN	Input file pointers
0D3A	3386		Start track
0D3B	3387		Start sector
0D3C	3388		Length of file at open
0D3E	3390		Drive number where file located
0D3F	3391		Current track pointer within file
0D40	3392		Current sector pointer within file
0D41	3393	£WOBP	Output buffer pointer within buffer
0D43	3395	£WFPOT	Output file pointers (as £WFPIN)
0D4A	3402	£WIBFL	Input buffer location
0D4C	3404	£WOBFL	Output buffer location
0D4E	3406	£WDTAB	Address of start of ARH table

Summary of NAS-DOS subroutines

```
  1 - £INITD - Initialise Drive 0
* 2 - £FORM  - Format a Disk
  3 - £CHAIN - Chain to another Program
* 4 - £CRATE - Create a Directory entry
  5 - £SCRAT - Scratch a Directory entry
* 6 - £SCAND - Scan Directory
*10 - £OPEND - Open Directory
 11 - £OPENI - Open Input File
 12 - £OPENO - Open Output File
 13 - £PREPO - Prepare new Output File
*20 - £READ  - Direct Read
*21 - £READD - Read Directory
 22 - £READS - Read Sequential
 23 - £READR - Read Random
*24 - £READF - Read Whole File
*25 - £READH - Read Directory Header
*30 - £SAVE  - Direct Save
*31 - £SAVED - Save Directory
 32 - £SAVES - Save sequential
 33 - £SAVER - Save Random
*34 - £SAVEF - Save Whole File
*35 - £SAVEH - Save Directory Header
 40 - £CLSEI - Close Input File
 41 - £CLSEO - Close Output File
```

'*' denotes that the subroutine is not available as a user call.

6. Basic Routine Handler

The BRH, as previously stated, is specifically designed to run with the standard Nascom 4.7 ROM Basic. Disk routines are initiated by user calls (e.g. A=USR(1)). The number specified in the brackets is the routine number to be called. This may be an expression, literal or numeric variable. It is sometimes necessary to follow the USR call with information. The fields following the call, in these cases, must be STRING VARIABLES. Prior to the execution of any of the following routines it is necessary to set USRLOC to point to the BRH. This is done with the DOKE4100,-10234 instruction.

As NAS-DOS will configure the I/O buffers at the top of RAM, the amount of memory allocated to the Basic Program should be specified as 512 bytes less than the top of RAM when Basic is cold started using the 'J' instruction, or the I/O buffers moved (£WIBFL and £WOBFL).

All of the following routine examples show 'nv' as being any numeric variable and 'sv£' as being any string variable:-

nv=USR(1) - Initialise Disk

It is necessary prior to any disk activity to use this routine to establish a link between NAS-DOS and the disk drive. The contents of £WDRV (3360D) are set to 0. The contents of the numeric variable return the status of drive zero.

For example :-

```
20 A=USR(1):REM FDC status now in 'A'
```

nv=USR(3),sv£ - Chain to new Program

This command may be used to pass control to another program. It should be noted that variables are not preserved. This command is particularly useful if the amount of free RAM is insufficient to execute the whole of a large program. The program may be split into 'phases' and control passed from one phase to another.

Place the eight byte file name to be chained to into a string variable and POKE the drive number into £WDRV and call this routine. Control is returned to the next instruction in the Basic program if the program name does not exist on disk.

For example:-

```
80 PG£="PRINTOUT"
90 A=USR(3),PG£
100 PRINT"Program '";PG£;"' does not exist"
110 STOP
```

nv=USR(5),sv$ - Scratch file from directory

This subroutine will scratch the file name specified in the string variable sv$ from the drive specified in $WDRV (3360 decimal).

Place the eight byte file name to be scratched from the directory in any string variable and place the drive number in $WDRV and call this routine. If the file name did not exist in the directory on that drive, the numeric variable nv returns with a non-zero result. This may be tested as follows:-

```
100 FL$="WORKFILE"
110 A=USR(5),FL$
120 IF A THEN PRINT "File not deleted"
130 REM Rest of Program
```

nv=USR(11),sv$ - Open input file

Before it is possible to read any file, sequentially or randomly, it is first necessary to open an input file specifying the name of the data file and also the drive number where that file should exist.

Place the eight byte file name in any string variable and the drive number in $WDRV (3360) and call this routine. If the file does not exist the numeric variable will return with a non-zero result.

For example:-

```
200 FILE$="DATAFILE"
210 A=USR(11),FILE$
220 IFATHENPRINTFILE$;" does not exist":STOP
230 REM Rest of Program
```

OR you could simply create it if it does not exist!

nv=USR(12),sv$ - Open output file

This command will open a file for write only access so that the user may append records to the end of the file. The file must already exist. The contents of the highest sector in that files allocation are brought into the output buffer. The output buffer pointers are then set to the end-of-file marker in that sector. If no-end-of-file marker exists, the pointer is set to the last byte.

Place the eight byte file name in any string variable, the drive number in $WDRV (3360) and call this routine. The numeric variable returns with a non-zero result if the file did not exist. This should be tested otherwise a ?FC error will result if the file is not open when a write instruction is executed.

For example:-

```
400 IA$="DATAFILE":A=USR(12),IA$
410 IFATHENPRINTIA$;" not found":STOP
420 REM Rest of Program
```

nv=USR(13),sv£ - Prepare new output file

This command is similar to the open output routine with the exception that the file must not exist and that the file pointers are set to the start of the file. If it does not already exist the contents of £WSIZE are examined and this field is used to determine the size of the file to be created. The contents of £WSIZE are normally set to 30H sectors. If this area is not available, the count is decremented and the routine retried. If this decremented count becomes zero, control is passed to the relevant error routine and the job abandoned.

For example:-

```
240 FL£="WORKFILE"
250 DOKE3367,100:REM Create a file with 100 Sectors
260 A=USR(13),FL£
270 IFATHENPRINT"File `";FL£;"' already exists":STOP
280 REM Rest of Program
```

OR, line 270 could read:-

```
270 IFATHENA=USR(5),FL£:GOTO260
```

which would delete the file and recreate it!

nv=USR(22),sv£,sv£,sv£..... - Read sequential

This routine will read a number of bytes from the previously opened input file into the specified string variables. These string variables must have been previously defined and set to the required length. If end-of-file is encountered the numeric variable will return with a non-zero value. The total length of all of the string variables is used to determine whether the whole of the record is to be read. If it is less than the record length the next read instruction will result in bringing the next n bytes from that same record. If it is greater, then the rest of the string and any following strings are filled with nulls (00H). These strings must also be set to point outside the Basic program area, otherwise modification of the Basic program may take place. This can done by using the command AB£="1"+"234567890" (which will set AB£ to a length of 10 bytes). Null records are automatically skipped.

For example:-

```
500 A1£="1"+"234567890123456":REM Now 15 Bytes
510 A2£=A1£:A3£=A1£:A4£=A1£
520 A=USR(22),A1£,A2£,A3£,A4£
530 IFAGOTO10000:REM Goto end-of-file routine
540 REM Rest of Program
```

nv=USR(23),sv£,sv£,sv£.... - Read random

This routine may be used to read a file randomly. The relative sector number (i.e. the first sector of a file is sector zero) should be placed in £WSECT (3365D) and this routine called. The BRH will bring the relevant sector into the input buffer and then pass control to the read sequential routines. Please refer to that section.

nv=USR(23),své,své,své.... continued

For example:-

```
360 DOKE3365,100:REM we want sector 100
370 A=USR(23),své,své,.....
380 IFATHENPRINT"Sector contains EOF":STOP
390 REM Rest of Program
```

nv=USR(32),své,své,své.... - Write sequential

The contents of all of the string variables in the list are written to the previously opened output file. The BRH will then write an end-of-record marker (00H) to that file so that NAS-DOS will be able to determine where one record ends and another begins.

nv=USR(33),své,své,své.... - Write random

The routine will write the list of variables to disk after setting the file pointers to the sector number specified in éWSECT (3365).

nv=USR(40) - Close input file

This command will sever the links between NAS-DOS and the input file so that further read input file commands cannot occur unless a file is reopened. No disk activity will occur. No error will result if a file was not open.

nv=USR(41) - Close output file

The links between NAS-DOS and the output file are severed after an end-of-file marker (FFH) and the output buffer are written to disk. The directory is then updated with the new length of the file. This should be remembered if the last write to the file was not at the end of the file as truncation will occur.

7. Assembler Routine Handler (ARH)

The Assembler Routine Handler is situated in NAS-DOS at D800H. Subroutine calls are made by calling this routine and following it with a one byte call type, for example:-

```
10FA CD00D8    3040          CALL DOSARH
10FD 14        3050          DEFB £READ
```

The ARH offers all of the facilities of the Basic Routine Handler (BRH) and more. The routine call numbers between the BRH and the ARH are identical.

It is suggested that the user becomes fully aware of the logic involved with the BRH whenever possible before attempting to use the ARH. TAKE COPIES OF DISKS BEFORE TESTING ARH TYPE PROGRAMS.

The routine numbers and their descriptions now follow. All of the following codes are expressed in DECIMAL.

It is important to note that registers B,C,D,E,H and L are always preserved, register AF is used to denote if the disk subroutine call was successful. Register IX is modified and register IY is unused. All alternate registers are used and their returned values will be indeterminate.

1 - £INITD - Initialise disk

It is necessary prior to any disk activity to use this routine to establish a link between NAS-DOS and the disk drive. The contents of £WDRV are set to the default drive, drive 0.

2 - £FORM - Format a disk

With this routine it is possible to format a disk. The program calling this routine is responsible for creating a directory header, which contains the name of the disk.

3 - £CHAIN - Chain to another program

This command enables a user to pass control from one program to another.

Place the address of the eight byte file name in £WFILE and the drive number in £WDRV. The directory is searched, and if the program name is found it is read into memory at the LOAD address and control is passed to the EXEC address. If the program does not exist on that disk, control is returned to the calling program. (Ensure that the program being chained to will not be read into the same area as the stack, else the contents of the stack will obviously be corrupted as well as the program itself because NAS-DOS will be doing PUSHes and POPs of its own.

4 - £CRATE - Create a directory entry

The £CRATE routine will enable a user to create a directory entry of his/her own. The drive number should be placed in £WDRV, the address of the file name in £WFILE, the execute address in £WEXEC, register HL set to the load address, register BC set to the size of the file, register D to the starting track and register E to the starting sector. Carry is set if the file name already exists.

5 - £SCRAT - Scratch a directory entry

This routine allows you scratch a directory entry so that the disk space may be used by other files. Place the address of the file name to be scratched in £WFILE and the drive number in £WDRV and call this routine. Carry flag is set if the file did not previously exist.

6 - £SCAND - Scan directory

The scan directory command may be used to determine whether a file is on disk. The address of the eight byte file name should be placed in £WFILE and the drive number for the scan in £WDRV. Carry is set if the file name did not exist. If it does exist, register IX returns pointing at the first character in the directory entry. Refer to directory layout specification.

10 - £OPEND - Open directory

Prior to any read directory commands the directory must be initialised so that the next £READD call will start from the first entry in the directory on the appropiate drive. Place the drive number in £WDRV and call this routine. No disk activity will occur.

11 - £OPENI - Open input file

Informs NAS-DOS that the file name whose address is in £WFILE on the drive which is specified in £WDRV should be opened for read only access. The directory is scanned and if found the file pointers are set to point at the first record in the file. If the file is not found, carry flag is set.

12 - £OPENO - Open output file

The open output command requests NAS-DOS to open the file name specified whose address was placed in £WFILE on the drive placed in £WDRV for write only access. The file must already exist. If it does not, the carry flag is set and control is returned to the calling program. Otherwise the contents of the last sector in that files allocation are brought into the output buffer, the file pointers set and the output buffer pointer set to point to the end-of-file (EOF) marker. If an EOF marker does not exist in that sector (typical if the file was not closed correctly) the buffer pointer is set to the last byte in the sector.

19 - £PREPO - Prepare new output file

This command is similar to the £OPENO command with the exception that the file must not already exist. If it does, the carry flag is set. If it does not, NAS-DOS examines the contents of £WSIZE which is used to determine the number of sectors to allocate. If this amount is not found the count is decremented and retried until it becomes zero when control is returned to the relevant error routine. When contiguous space is found, a Directory entry is created and the file pointers set to point to the first sector in that files allocation.

20 - £READ - Direct read

This routine is used by all read accesses to the disk. Register HL should be set to your buffer, register BC set to the number of sectors to be read, register D set to the track number, E set to the sector number and A set to the drive number.

21 - £READD - Read directory

Prior to using this routine, a call should have been made to the £OPEND command which initialises the directory. Register IX will return with the address of the next directory entry. Deleted records are automatically skipped. If the first byte of the directory entry is FFH (end-of-directory marker) carry flag is set. Refer to the directory layout specification.

22 - £READS - Read sequential

This command informs NAS-DOS to transfer the next record or part record from the previously opened input file. Set register HL to point to your input buffer and register BC to the number of bytes to be read and call this routine. Carry flag is set when EOF is encountered.

23 - £READR - Read random

Place the relevant sector number (the first sector in a files allocation is sector 0) in register BC and set register HL to your input buffer and call this routine. The file pointers are set and then control is passed to the read sequential routine. Please refer to that section.

24 - £READF - Read whole file

This routine will read the whole of a file into memory. The load address in the directory is used to determine where this file should be read into. £WEXEC is set to the execution address in the directory. Place the address of the file name in £WFILE and the drive number in £WDRV and call this routine. Carry flag is set if the file did not exist. It is important to ensure that the stack pointer is not within the areas of the program that is to be loaded as the contents of the stack will be corrupted.

25 - £READH - Read directory header

The contents of the directory header are read into the directory buffer. Register IX returns with this address. Simply place the drive number in £WDRV and call this routine.

30 - £SAVE - Direct save

This routine is used during all write commands to disk. Register HL should be set to point to the output buffer, register BC to the number of sectors to be written, register D the starting track, register E to the starting sector and register A to the drive number where the data should be written.

31 - £SAVED - Save directory

The contents of the directory buffer are re-written to the directory sector from where the last £READD command occurred.

32 - £SAVES - Save sequential

Set register HL to point to your output buffer and BC to the length of the record to be written. NAS-DOS will write a sector to disk when the output buffer becomes full. An EOR marker is then appended to the end of the output buffer. If more disk space is required, NAS-DOS will attempt to allocate another sector and update the directory accordingly.

33 - £SAVER - Save random

The write random command will instruct NAS-DOS to write your sector image record (i.e. 256 bytes) to the relative sector number specified in register BC. The file pointers are set to point to the sector number and control is then passed to the write sequential routine.

34 - £SAVEF - Save Whole File

Set the contents of £WFILE to point to the eight byte file name to be allocated and the drive number in £WDRV, register HL to the low end of the file, register DE to the high end + 1, and BC to the execution address and call this routine. If the file already exists, an 'Overwrite' prompt will occur for the user to decide whether to delete the existing file. You will have seen this command in operation from conversational mode.

35 - £SAVEH - Save Directory header

The contents of the directory buffer are re-written to disk. The directory header must have been previously read with the £READH command. The directory header contains the name of the disk allocated during format.

40 - £CLSEI - Close input file

This command informs NAS-DOS that all links between the previously opened input file and NAS-DOS should be severed. No further £READS or £READR commands may occur unless a file is reopened for read only access. If the file was not open no error will occur.

41 - £CLSEO - Close output file

This command will write a single EOF marker and then write the contents of the output buffer to disk. The directory is then updated with the correct length of the file. If no output file was open, no error will occur.

It should be noted that the output file is automatically closed should any error occur.

8. Floppy Disk Controller software

The Floppy Disk Controller software (FDC) is located at D000H to D3FFH in the supplied firmware. This software is capable of reading or writing directly to disk but does not have the intelligence to determine where live data exists that is under NAS-DOS control. Entry is made to the routines via a jump table. It should be noted that every effort should be made to access these routines via NAS-DOS rather than directly to this software. This documentation is meant as a guide line only to enable users to adapt the software for their own requirements.

Jump table

COLD	D000	Jump to cold start routine in conversational mode
WARM	D003	Jump to warm start routine
INIT	D006	Initialise drive zero
FRMT	D009	Format a disk
RSCT	D00C	Read a sector
RTRK	D00F	Read a track
WSCT	D012	Write a sector
WTRK	D015	Write a track
CMND	D018	Output a command to the FDC
DSEL	D01A	Drive select

There then follow some values:

DLSPD	D024	Sides per disk
DLTPS	D025	Tracks per side
DLDPS	D026	Drives per system
WORKSP	D027	Workspace location (8 bytes)

Read/Write routines are called by setting register:

HL Buffer start (returns Buffer end if good IO)
D Track
E Sector
A Drive (returns with error code, 0=good IO)

To output a command to the FDC (CMND), load register A with the 1793 command and call this routine. Register A returns with the FDC status.

To format a disk, call FRMT. Register A returns with FFH in register A if the format failed. A Directory is created on drive 0.

To initialise the drives, call INIT. The FDC status returns in register A.

To select a drive call DSEL. Load register A with the logical drive number and call this routine. The FDC status returns in A.

It is important to note that NAS-DOS does not utilise the fields DLDPS and WORKSP in the FDC ROM due to space limitations. These fields may be altered as only the FDC software uses these fields.

9. Advice.....

* Take regular back-up's of disks.
* Keep disks in their covers when not in use.
* Remove disks from drives when not in use.
* Put write protect tabs on disks whenever practical.
* Remove disks from drives prior to power-up or power-down.
* Keep disks away from direct heat or sunlisht.
* Prevent disks being distorted.
* Never touch exposed sections of disk.
* Avoid using file names which contain colons or control characters.
* Use inactive disks during program development.

10. Directory layout

The whole of track zero will be unallocated by the Dynamic Disk Space Manager (DDSM). Sector one is used for the USERBOOT function, sector 2 contains the name of the disk allocated during the last format and sectors three to the end of track zero contains the directory entries. The layout of the entries are as follows:-

Description	Displacement	Length
File name	0	8
Execution address	8	2
Load address	10	2
Start track	12	1
Start sector	13	1
Length of file	14	2

11. Error messages

Due to space restrictions, with the exception of the I/O error trap, NAS-DOS does not output error messages as such. Instead an error code is displayed. These are as follows:-

00 Error

This message would indicate that the Assembler Routine Handler (ARH) or the Basic Routine Handler had been called with an invalid call type.

01 Error

Directory error - This error message will occur whenever the physical end of the directory is encountered. This will probably be due to the directory being full. All deleted entries can be removed with the 'JC' command in conversational mode which may leave space. If it does not, files will have to be moved to another disk.

02 Error

Indicates that a data file being accessed is not currently open.

03 Error

Whenever a file is accessed randomly, a check is made that the sector number does not exceed the files allocation. If it does this error message will occur. Remember that the sector number should be relative i.e. The first sector is sector ZERO.

04 Error

Indicates that the DDSM could not allocate another consecutive sector in the data files allocation or insufficient space has been found to save the current file. Reorganising the disk may prove successful.

Disk I/O error messages

The error code display by NAS-DOS is, in fact, the contents of the status register from the 1793 Floppy Disk Controller chip. The error code should be broken down to bit level and the following table used to ascertain the fault.

Bit	Description
7	Drive not ready
6	Write protected disk
5	Head loaded
4	Seek error
3	CRC error
2	Lost data
1	Data request
0	Busy.

In addition NAS-DOS will give these error codes:-

FD	Time-out. Device did not respond in time
FE	Invalid input data
FF	Format error

ZEAP Z80 Assembler - Source Listins

```
                0010 ;User Boot demonstration Program :USERBOOT
                0020 ;*****************************************
                0030 ;* Program name : USERBOOT               *
                0040 ;* Author       : Stephen W. Parrish     *
                0050 ;* Date written : 15th. November 1981    *
                0060 ;* Description  :                        *
                0070 ;* This program demonstrates the USERBOOT*
                0080 ;* function available in  NAS-DOS.  This *
                0090 ;* program must be  not greater than one *
                0100 ;* sector in length,  but may be used to *
                0110 ;* 'chain'  in a  much larger program as *
                0120 ;* this  program does.  Simply write the *
                0130 ;* program,  save  the source  and  then *
                0140 ;* save  the   object   code  with   the *
                0150 ;* JW1000 1001 0 1 0 command. To execute *
                0160 ;* this program, type JU(d).            *
                0170 ;*                                       *
                0180 ;*****************************************


                0210 ;NAS-DOS Equates
                0220 ;===============
0000            0230        ORG  0
0000 0003       0240 £CHAIN EQU  3      ;Chain to program
0000 0D21       0250 £WFILE EQU  #0D21 ;File name address
0000 D800       0260 DOSARH EQU  #D800 ;Routine handler


                0290 ;NAS-SYS Routines
                0300 ;================
0000 000D       0310 VIDCR  EQU  #0D ;Carriage return
0000 0028       0320 ZPRS   EQU  #28 ;Output a message
0000 005B       0330 ZMRET  EQU  #5B ;Return to NAS-SYS


                0360 ;*****************************************
                0370 ;*  P r o c e d u r e   D i v i s i o n  *
                0380 ;*****************************************


1000            0410 USERBT ORG  #1000
1000 210C10     0420        LD   HL,PROGGY   ;Inform DOS of..
1003 22210D     0430        LD   (£WFILE),HL ;file name
1006 CD00D8     0440        CALL DOSARH      ;Try to execute it
1009 03         0450        DEFB £CHAIN
100A EF         0460        RST  ZPRS        ;Tell op. program
100B 60         0470        DEFB "`
100C 100C       0480 PROGGY EQU  £           ;no on.disk
100C 532D444F   0490        DEFM "S-DOS   "  ;Program name
     53202020
1014 27206E6F   0500        DEFM "' not on disk"
     74206F6E
     20646973
     6B
1021 0D         0510        DEFB VIDCR
1022 00         0520        DEFB 0
1023 DF5B       0530        SCAL ZMRET       ;Return to NAS-SYS
```

13. NAS-DOS Utility Programs

The utility programs are provided to simplify disc back-up and house-keeping functions. All load into memory at 1000 hex.

COPYDD

The disk-to-disk copy utility enables a user to take a copy of disk providing a minimum of one double sided or two single sided drives are on-line. With the exception of the 'User-boot' and the directory header all tracks are copied from the source drive to the destination drive. Should an error occur during the copying process, the error is reported and processing continues at the next track.

Format of parameter:
s,d
where 's' is the source drive and 'd' the destination drive.

DIRSRT

Whenever a file is created or over-written, the new directory entry is written to the high end of the directory. The directory sort utility will sort the directory into alphabetical file name sequence.

Format of parameter:
d
where 'd' is the drive number.

FILCPY

The file copy utility will copy a given file on any drive to another drive. The destination file name need not be the same as the source file name and the copy can take place on the same drive. The amount of available RAM is calculated and used for the copy process. If any drive number is omitted drive 0 will be assumed.

Format of parameter:
s:xxxxxxxx
d:yyyyyyyy
where 'x' is the source file name, 's' is the source drive number, 'y' is the destination file name and 'd' is the destination drive.

REORG

The reorganisation utility will reorganise a disk so as to concatenate free disk space at the end of the disk. The source and destination drives must not be the same. For this reason, only a double sided or two single drives may use this utility. If a read/write error occurs, the error is reported and processing continues. The user should then attempt to copy the offending sectors at the end of the reorganisation.

Format of parameter:
s,d
where 's' is the source drive and 'd' is the destination drive.

VERIFY

This utility will read all sectors, one at a time, to verify that the disk can be totally re-read. If any cannot, the track and sector is displayed together with the error code and processing continues.

Format of parameter:
d
where 'd' is the drive number that is to be verified.

ID

During the format process of a disk, the disk is named to uniquely identify that disk. This utility will alter the current name and display the old name. The ID, or identification, is displayed during a directory display.

Format of parameter:
d:diskname
where 'd' is the drive that is to have its ID changed.

RENAME

This utility may be used to rename a file. The new file name must not already exist on that disk.

Format of parameter:
d:old-file
new-file
where d is the drive number for the rename. This field is optional (drive 0 assumed).

Nascom Microcomputers
Applications Note AN-0015

COMMON USER ERRORS IN NAS-DOS

We suggest that you read note A immediately, and the rest of these notes after reading the main body of the NAS-DOS manual, and whenever you may have problems. The problems noted here have been found to arise with first time users on a number of occassions.

A. Hardware and Installation

You must select EXTERNAL I/O (by means of link/switch LSW2/8 in the UP position on a Nascom 2 or 3) if this is not already implemented. This is required in order to access the disc controller card (FDC).

B. Conversational Mode

1. The] symbol must be typed in the first column.

2. Make sure that you have typed all the right arguments for the command concerned.

C. BASIC Routine Handler

1. Don't forget to set the link to the USR routine with the DOKE4100,-10234 statement.

2. The USR(1) function must be used once in the program (at least) before any disc access.

3. You MUST specify to BASIC when cold starting that the memory available is 512 byes less than the actual amount, to allow for disc buffer handling. Note that BASIC programs loaded from disc will carry their pointers from the last time they were run, so that the correct memory size must be specified when the program is first created or loaded from tape. A subsequent cold start followed by loading from disc with the JL command will not alter the available memory.

4. Filenames must be specified with EXACTLY eight character names, even though some of these may be trailing spaces.

5. To change the disc drive to be accessed you need to POKE the drive number into location 3360.

6. When creating a new output file you should DOKE the maximum file size into location 3367

7. When a file is closed an end of file marker is written at the last accessed point, and any remaining space will be freed. It is therefore esssential to use care in closing a random access file - you could easily close it in the middle of the file if this was the last point accessed in that file. The remainder of the file would then be lost. The solution is simple - don't bother to close the file after writing - no harm will be done, although unused space which you might have preferred to be freed will not be released.

8. NAS-DOS does not maintain separators between data written to disc with a single USR(32) or USR(33) statement. This saves valuable disc space, but means that the length of the strings written must be fixed. You can easily normalise the length of strings via a subroutine of the form:

```
O$=LEFT$(O$+"                    ",L)
```

where L is the required length of the string.
On input it is essential that the variables referred to have been declared to the correct length. At first sight a statement of the form:

```
O$="12345"
```

seems the correct way to declare the variable O$ of length 5 characters. In practice this statement would cause BASIC to store the data read in to O$ in the PROGRAM space where the characters "12345" were originally placed - resulting in some peculiar listings! For this reason the declaration should involve BASIC in 'Calculating' a new string. Suitable forms of statement would be:

```
O$="1234"+"5"
or
S$="                    ":O$=LEFT$(S$,5)
```

```
1 REM:BRHDEMO:
2 CLEAR500
10 REM NAS-DOS FILE ACCESS DEMONSTRATION
20 REM ---------------------------------
30 REM
40 REM REV 1.1      21 FEBRUARY 1982
50 REM COPYRIGHT (C)1982 LUCAS LOGIC LIMITED
60 REM ----------------------------------------
100 REM FIRST DECLARE THE USR ROUTINE
110 DOKE 4100,-10234
120 REM NOW INITIALISE DISC
130 O=USR(1)
140 REM WE ARE GOING TO CREATE A SEQUENTIAL
150 REM FILE WITH THE NAME DATAF
160 REM
170 FI$="DATAF    "
180 REM NOTE THAT WE * M U S T * GIVE ALL 8
190 REM CHARACTERS OF THE NAME
200 REM
210 REM NOW WE DECLARE THE DISC NUMBER (0)
220 REM AND THE FILE SIZE (WE WILL ALLOW 50
230 REM SECTORS)
240 REM
250 POKE3360,0
260 DOKE3367,50
270 REM NOW CREATE THE FILE
280 O=USR(13),FI$
290 REM CHECK NO ERROR (FILE EXISTS?)
295 IFO=0THEN320
300 PRINT" File already exists"
302 REM DELETE THE OLD ONE AND TRY AGAIN
303 INPUT" Ok to delete";O$
304 IFLEFT$(O$,1)<>"Y"THENSTOP
305 O=USR(5),FI$
307 GOTO280
310 REM NOW CREATE SOME TEST DATA
320 INPUT" Christian name";CN$
330 INPUT" Surname";SN$
340 REM TO ALLOW ANY NAME WE SHOULD FIX THE
350 REM FIELD LENGTHS - SAY 12 CHARACTERS
360 REM FOR CHRISTIAN NAME AND 15 FOR
370 REM SURNAME.  THIS ENSURES WE CAN READ
380 REM THE DATA BACK CORRECTLY.
385 SP$="                                        "
390 CN$=LEFT$(CN$+SP$,12)
400 SN$=LEFT$(SN$+SP$,15)
410 REM NOW WRITE THIS DATA TO DISC
420 O=USR(32),CN$,SN$
430 REM WE ARE USING SEQUENTIAL ACCESS IN THIS
```

```
440 REM EXAMPLE
450 REM NOW CLOSE THE OUTPUT FILE.  THIS
460 REM CAUSE AN 'END-OF-FILE' MARKER TO BE
470 REM WRITTEN AND ANY UNUSED SPACE (MOST OF
480 REM THE 50 SECTORS IN THIS EXAMPLE!) TO
490 REM BE RELEASED.  NOTE THAT IN RANDOM
500 REM ACCESS IT IS USUALLY BETTER NOT TO
510 REM CLOSE THE FILE AS DATA STORED AFTER
520 REM THE LAST RECORD ACCESSED BEFORE THE
530 REM WAS CLOSED WILL BE FREED.
540 D=USR(41)
550 CN$="":SN$=""
560 REM THAT'S JUST TO PROVE NO TRICKERY!
570 REM OPEN AN INPUT FILE - THE SAME AS
580 REM BEFORE
600 D=USR(11),FI$
610 IFDTHENPRINT" File does not exist"
620 REM RATHER UN-NECESSARY HERE, AS WE JUST
630 REM CREATED IT'
640 REM NOW WE MUST SET UP THE STRINGS TO
650 REM CONTAIN THE INPUT DATA. THE LENGTH
660 REM MUST BE DECLARED BY AN EXECUTABLE
670 REM EXPRESSION. THE FORM
680 REM CN$="123456789012"
690 REM IS NOT ACCEPTABLE.
700 CN$=LEFT$(SP$,12)
710 SN$=LEFT$(SP$,15)
720 REM NOW INPUT THE DATA
730 D=USR(22),CN$,SN$
740 REM CHECK FOR END OF FILE
750 IFDTHENPRINT" END OF FILE":STOP
760 REM PROVE OK
770 PRINTCN$SN$
775 REM NOTE THAT WE COULD STRIP OFF THE
776 REM EXTRA SPACES
777 D$=CN$:GOSUB1000:PRINTD$" ";
778 D$=SN$:GOSUB1000:PRINTD$
780 REM CLOSE THE FILE AND FINISH
790 D=USR(40)
800 END
999 REM ROUTINE TO STRIP OFF TRAILING SPACES
1000 IFD$=""THENRETURN
1010 D1=0:FORD=LEN(D$)TO1STEP-1
1020 IFMID$(D$,D,1)<>" "THEND1=D:D=1
1030 NEXT:D$=LEFT$(D$,D1):RETURN
Ok
```

Nascom Microcomputers
Applications Note AN-0007

NAS-SYS/NAS-DOS UTILITIES

1. Introduction

Included on the NAS-DOS Utilities Disc are some utility functions stored in the files UTS8 - UTSC. Each of these files contains the same utilities, but located respectively at 8A00, 9A00, AA00, BA00 and CA00. These utilities do not form an integral part of the NAS-DOS system.

The utilities provide for the support of a printers - serial, serial with handshake, or parallel - turning the printer on and off by a single keystroke, screen dumps to the printer and a number of other useful functions. The operations are compatible with NAS-DOS.

2. Enabling the Utilities

You must first select the appropriate copy of the utilities to use with your system. This will depend on the amount of memory fitted to your computer:

32K RAM memory - UTS8
36K RAM memory - UTS9
40K RAM memory - UTSA
44K RAM memory - UTSB
48K RAM memory - UTSC

If you have NAS-DIS in EPROM your maximum amount of RAM memory is 44K; if you have NASPEN in EPROM your maximum RAM memory is 40K. In either case if this means that you cannot access some of your RAM memory (because the EPROM's have the same address as some RAM) we suggest you save the EPROM contents on disc and remove the EPROM's. This will mean you have more generally useful RAM, and the programs can be loaded very quickly from disc.

If you are using NAS-SYS rather than NAS-SYS 3 you should use the monitor to modify the location n909 (where n is the field number, as in the copy of the utilities selected) from 78 hex to 7E hex, and save the modified copy of the utilities for future use.

As will be described in the next section the utilities can be accessed both direct from the keyboard or from BASIC and assembler programs. If you require the keyboard functions the utilities are loaded and enabled by placing the disc in drive 0 and typing JE:UTSB (or whichever version you have selected). You could of course place the disc in drive 2 instead and type JE2:UTSB. If you only require access from programs you can use the JL command in place of JE.

The utilities are now loaded and, if the JE command was used, the keyboard functions are active. The keyboard functions use the NAS-SYS UIN procedure. This means that if you press the 'Reset' button or use the NAS-SYS N command you can disable these keyboard functions. Providing you have not overwritten the utility program memory they can be re-enabled by typing E8A00 (or E8A00, E9A00 etc depending on the version used) in NAS-SYS.

The utilities use both the NAS-SYS UIN and UOUT procedures, the latter linking automatically to NAS-DOS. If you wish to use these routines yourself then you may do so, but on exit from your user routine you should transfer control to locations nA03 and nA06 for input and output respectively. The output routine is used purely to support the printer, and if you are using your own printer driver you can therefore transfer control direct to NAS-DOS rather than nA06, ie to the location D405.

Certain system programs use their own user input and output routines, or cold-start NAS-SYS on termination. This means that the utilities are disabled when these programs are used, and the utilities will need to be re-initialised after using these programs.

Remember that the utilities are loaded at the top of RAM memory, immediately below the disc handling buffers. Therefore if you use BASIC and you do not wish to overwrite these utilities you must specify the amount of memory when cold-starting BASIC. You should specify this as 2048 locations less than the amount of RAM actually available. To simplify this you may find it more convenient to perform this action once, and then to save the empty BASIC program using the command JB:BASIC. You can then cold-start BASIC in future, with the correct amount of memory specified, by means of the command JE:BASIC.

3. Functions provided

The functions provided by the utilities are available at two levels - direct keyboard entry and from BASIC and assembler programs. The two modes of use will be described separately.

3.1 Keyboard Operation

This mode of operation is only available when the utility initialiser has been executed. This is done as described above. Remember that, as described previously, performing a 'reset', the NAS-SYS 'N' command and certain special programs will disable the utilities.

The commands available are all performed by holding down the 'CTRL' key and pressing one other key. This is referred to as CTRL/I below, where 'I' is the letter to be pressed.

Command	Result
GRAPH/Q	Shift lock/shift unlock
CTRL/A	The tape led is switched on and off by successive operation of this sequence. This output can be used, with a suitable interface, to switch a tape recorder on and off.
CTRL/B	The utilities are disabled, and the input/output assignments are restored to the state before the utilities were enabled.
CTRL/D	The contents of the screen are dumped to the printer configured at initialisation of the utilities.
CTRL/P	Output to the screen is echoed to the printer by pressing this key. Pressing it again suspends printer echo. Note that this is only effective where the output statements use NAS-SYS - POKE's direct to the screen memory will not be echoed.
CTRL/Z	The flashing cursor is homed to the first position on line 1 of the display.

3.2 Program Mode

The program mode allow access to the utilities from either BASIC or assembly language programs. The utilities are accessed in BASIC by means of the USR function, and in assembler by means of a CALL. Each utility is allocated a number as follows:

Function number	Operation
64	Enable the utilities. Must be called before any other utility access.
65	Disable utilities and revert to normal operation.
66	Turn tape led on (next call turns it off)
67	Wait 1 second
68	Route subsequent output to printer only.
69	Cancel 68, and route output to VDU.
70	Dump screen contents to printer.
71	Blink cursor and wait for character.
72	Return current key pressed (zero if none).
73	BASIC input with edit (see below).

3.2.1 Access from BASIC

The BASIC USR function is used to access the utility functions. The utility USR function numbers are 64 - 73. Numbers below this are passed to the NAS-DOS USR function handler, so that only one USR vector is required. A special vector is additionally provided to USR(0), but this is reserved for future use. An extension is provided for the user to add further USR functions, from 74 - 87.

In the case of calls from BASIC the function number should be specified in the user call (ie USR(n), where n is the function number). The function returns a value of zero if there is no error, except for funtions 67 and 68, which return the ASCII value of the key pressed, and 73 which returns no particular value.

In order to enable the routines it is essential that the following statements are inserted in the BASIC program before any calls to the other routines:

```
DOKE4100,(4096*n+2060-65536)
0=USR(64)
```

where n is the field number of the version of the utilities in use.

Please note the need to specify the memory available when cold-starting BASIC, as described in section 2.

The function 73 blinks the cursor and awaits entry of a string of characters, terminated with the <ENTER> key. The string is returned in the FIRST VARIABLE DECLARED in the program. It is therefore essential that the first variable declared in the program is a 'scratchpad' string variable used for transferring string data into the program. The function limits the number of characters input to the length of this string variable. The length of the variable should be fixed to the required maximum length by a statement of the form

```
0£="      "+" "
```

This would fix the length as 6 characters. The string is set by a calculated value to ensure that the new string is stored in string space rather than in the program code itself. Note that the function starts with the string equal to the original value, so that it can easily be used as an edit function. The normal NAS-SYS cursor edit commands work, unlike normal BASIC input statements, except that the cursor cannot be moved up or down. The CS key has a special function - it returns an empty string, and can therefore be used as an abort key. This utility routine is used in a number of Nascom applications programs.

3.2.2 Access from assembler programs

The same numbers are used as in the case of access from BASIC, although calls to NAS-DOS should NOT be routed through this procedure due to the different method of passing parameters.

As in the case of BASIC it is essential to initialise the routines by a call to routine 64.

The call is made to the location n80F, with the D register containing zero and the E register containing the number of the routine to be accessed. The value of any key pressed when using routines 71 and 72 is contained in location nD00.

4. Routine organisation

4.1 Workspace

The routines use locations nD00 to nD42. These locations are allocated as follows:

Address	Function
nD00	Argument returned to calling program
nD01	Error number
nD03	Temporary store for stack pointer
nD05	Store for SP during printing
nD07	Calling program type (0=BASIC, 1=assembler)
nD08	Input table status before use of utilities
nD0A	Output table status before use of utilities
nD0C	Printer margin width
nD0D	Printer flag (1=on, 0=off)
nD0E-nD15	Temporary variables used for BASIC input
nD16	Shift flag
nD20-nD3F	Extension to user funtion table. Up to 14 more function addresses, terminated by 00,00. Function numbers are 74 upward.
nD40-nD90	Stack scratchpad area.

4.2 Jump Vectors

The utilities contain the following fixed position jump vectors to internal routines:

Vector location	Routine
nA00	Initialisation of utilities
nA03	Keyboard input intercept routine
nA06	Normal printer output routine
nA09	Printer output less linefeed (ZEAP)
nA0C	BASIC USR function interpreter
nA0F	Assembler routine entry
nA12	Vector for USR(0) (Reserved)

5. Postscript

As stated earlier this package is not part of NAS-DOS. It should therefore be treated as a free sample rather than part of the utilities package included in the purchase price of NAS-DOS. Those functions which have been tested are fully operational, but some parts, such as access from assembler programs, are not properly proved, so no doubt there are bugs at large! Some of the user routine numbers may change, so if you write programs using the utilities use a variable for these to simplify future changes.

Rev 1.1 20 February 1982