IMP

Integrated Monitor Package for DEMON/II

Description and Instructions for

Disk Editor
Disk Command Interpreter
Disk Assembler
Disk Loader
Disk Move Program
Disk I/O Supervisor

Revised December 1974

NICOLET INSTRUMENT CORPORATION
5225 Verona Road
Madison, Wisconsin 53711
Telephone: 608/271-3333

TABLE OF CONTENTS

I. Introduction

The Integrated Monitor Package (IMP) is a collection of programs for disk file handling using DEMON/II with the Nicolet 1080 data systems. It includes the ability to create, assemble, edit, load and run programs as well as the ability to transfer ASCII and binary information from high and low speed paper tape devices. Using IMP, paper tape can be used only for back-up copies, as no paper tape intermediates are required during assembly. Indeed, listings generated by the assembler can be written onto disk and then the relevant portions examined using the editor rather than listing the entire file.

Further, the disk command interpreter and disk I/O supervisor allow very sophisticated disk file handling capability from user programs with only minimal programming complexity. This allows both program segments and data to be swapped in and out during program operation.

The beginning user will need to become familiar with the Disk Editor and the DCI before proceeding to the other programs as all remaining programs utilize the command decoding capabilities of the DCI.

II. DISKED

Disk Based Text Editor For the Nicolet 1080 system

(NIC-28-40605)

Introduction

DISKED is a text editor which operates on files stored on disk and places the resulting edited file back on disk. It is intended to be used with the disk assembler, but can be used to punch out the text onto paper tape as well.

DISKED operates in conjunction with DEMON/II and must not be used with older disk monitors. Each file which is to be edited is stored on disk and given the extension .A (for ASCII), so as not to be confused with binary data or programs. The files consist of tracks having packed 8-bit ASCII characters, 5 per two words, with all characters less than 240 ignored except 215 (return), 214 (Form feed), 211 (Tab) and 212 (Line feed). The Form character is used as a delimiter within each track's text, indicating the end of text within that track. Thus, all tracks are stored as 1536_{10} word blocks, regardless of how many meaningful characters they contain.

During operation, DISKED reads in one track at a time from the Input File keeping count of the line numbers. It unpacks each track to one character per word for ease of editing and when editing is complete, packs the information back into the 5-character per 2 word format and writes it into a second file, called the Output File. The minimum 1080 system which will support DISKED consists of a 12K system with one disk. The current version assumes that all operations are performed on disk 1.

Loading and Storage

DISKED is supplied as a binary tape and is loaded using the standard Binary Loader by typing BIN. To store the program type

```
STORE DISKED 0-4300;0;P
```

Be sure to restart the DEMON monitor after loading and before starting DISKED. The program starts at location zero, prints a number sign (#) and allows the commands described below.

Command Conventions

The length of each command varies with the amount of information which is required for its completion. For this reason, the entire command is typed, followed by a Return before it is interpreted for execution. This feature also allows character by character modification of the line until the terminating Return is typed.

The actual commands are all one letter in length. They may be followed by a number of characters describing the line number or the file name in various cases.

This modifying information is relatively "freeform." It may or may not be preceeded by a space. It may have as many spaces between components as are desired. Any character in the command line may be deleted by typing a Rubout. The Rubout convention of DISKED is that it will print a backslash and echo the rubbed out character. Additional rubbed out characters will be printed without additional backslashes until some character other than Rubout is struck. At this time, a second backslash is printed, thus bracketing the rubbed out text between two backslashes. For example, if you had typed

```
F FQRT" FORT3
```

but had really meant to type FQRT2 instead of FORT", you could correct this by striking the Rubout key 7 times, which would produce

```
F FORT" FQRT3\3TRQF "
```

Then, to correct the error, simply type in the 2 followed by the remaining text:

```
F FQRT" FCRT3\3TRCF "\2 FQRT3
```

It is also occasionally useful to delete the entire command line. This can be done by typing CTRL/O. The program will print ^0, type a carriage return-line feed and allow entry of a new command. Typing a Return with no command in the string will cause an ILLEGAL COMMAND error message, as will typing any letter which is not a command. Spaces are not required except between file names, but may be added to improve legibility.

All characters produced by holding down the CTRL key and typing the character are represented in this manual as CTRL/(char); for instance, CTRL/A. These characters are printed as ^A. The TAB character (CTRL/I) has the functions of spacing over to the next column divisible by 8. This allows easy tabulation of text. It can be used in the F, I and S commands.

DSKED Commands

A -- Add text to the beginning of a line. Use CTRL/R to finish the line.

CTRL/A — Append tape in reader to currently open file. Asks MORE TAPE? Answer Y or N.

B Print out bottom line number and text of the bottom line in the current buffer.

CTRL/B Write out current buffer and load next one.

C nnnn Change line to new text. Exits at first CR. Equivalent to D followed by I.

CTRL/C Close current Output file. Writes out all remaining buffers. This must be done before the output file actually exists in the disk directory.

D mmmm-nnnn -- Delete lines mmmm-nnnn.

E FILE1 FILE2--Edit Input file FILE1 and place result in Output file FILE2.

F FILE1— Create a new file named FILE1. The user can enter text until a dollar sign is typed. This closes the file.

G — Get and print the next line having a label followed by a comma.

CTRL/G FILE1 FILE2 -- Compress FILE1, combining tracks to occupy minimum disk space and place the result in FILE2.

H FILE1 -- Punch out FILE1 on the high speed punch.

I nnnn -- Insert text before line nnnn. Exit with CTRL/D.

K FILN FILE1 FILE2 FILE3... — Combine FILE1, FILE2, and FILE3 into one large file named FILN. Note that the output file is first in the list .

L FILE1 -- List FILE1 on the Teletype.

M -- Jump to DEMON. This is not allowed if the file has not been closed.

N -- Print out the next sequential line in this buffer.

CTRL/N — Jump to Nicobug II at 4700.

P mmmm-nnnn— Print lines mmmm-nnnn. Must all be in same buffer.

R FILE1 -- Read in source tape and name it FILE1. Asks MORE TAPE? when $-sign is found.

SABCDEF —Search for text string ABCDEF starting at current line. String may be up to 72 characters if desired and may contain spaces. Terminated with Return.

CTRL/R - Print out rest of line in search string.
CTRL/N - Search for next occurrence of string.
Rubout - Rubs out characters back to beginning of line.
Return - Delete all characters in the rest of the line.
CTRL/O - Delete entire line and allow entry of new text.
ALT MODE - Split line into two at this point in the line.

T — Print out the top line number and top line of the current buffer.

V mmmm-nnnn/dddd -- Move lines mmmm-nnnn to before line dddd. Old position is not deleted. Lines mmmm-nnnn must all be in the same buffer.

W FILENN -- List out text of FILENN on low speed punch. Equivalent to L with leader and trailer added.

CTRL/W — Write out all text above current line as a separate buffer and put remaining text in a new buffer to allow more room for inserting. Should be used whenever a Bell signals a full buffer.

X mmmm-nnnn FILEN — Extract lines mmmm-nnnn and put them in a disk file named FILEN. Previous Output file lost.

CTRL/Y -- Yank in next buffer, deleting current one.

+nnnn — Print out the line nnnn below the current location.

-nnnn — Print out the line nnnn above the current location.

: — Print out the current line number.

DESCRIPTION OF COMMANDS

INPUT COMMANDS

R FILENN Read in Tape

The command R causes DISKED to select the high speed reader if it exists and has tape in it and if not, the low speed reader, and read in an ASCII source tape until a dollar sign is found. During readin it pauses after every 3584 characters and writes them onto the disk. When the dollar sign is found it asks the question

MORE TAPE?

If this is all there is, type N and the program will make an entry in the disk directory of a file having the name FILENN.A (where any 6-character file name can be used). If you wish to append several tapes together, type Y, place the new tape in the reader and type Return. The dollar sign of the first tape will be deleted and the tape will read in and be appended to the end of the first tape. The program will, of course, ask MORE TAPE? again at the end of this and all successive tapes. If the high speed reader is used, the program will type out the message UNLOAD CATCHER AND TYPE RETURN after every four tracks stored on disk.

E FILE1 FILE2 - Edit

Editing of a given file begins by specifying the input and output files to be used. The input file to *be* edited is FILE1.A and the output file will be FILE2.A. The command also reads in the first disk text buffer and initializes the line pointer to the first line. FILE1 is now said to be "open."

It should be emphasized that FILE2 does not exist at this point, no matter how much editing has been done, until it is closed using the CTRL/C command. Before this time data has been written on the disk, but no entry is made in the disk directory until the file is closed, since until that time its size is not known. The error messages NO INPUT FILE FOUND and OUTPUT FILE ALREADY EXISTS are self-explanatory. To delete an old output file, exit to the monitor, delete the file including the .A extension and restart DISKED.

CTRL/A - Append

While a file is open, additional paper tapes can be added to the end of the file by typing CTRL/A The program will delete the terminating dollar sign at the end of the file, select the proper reader and read in the tape. At the end of the file the program will print MORE TAPE?. Proceed as during read-in.

F ABC2 - Start a new file named ABC2

The F command allows a new file to be created at the Teletype. It may be of any length and the Rubout conventions apply but no other commands can be accessed until the text is terminated with a dollar sign. This causes the "MORE TAPE" question to be printed. If it is answered Y, this closes the file. It can then be editied using the E command. During this command the TAB character can be used.

EDITING COMMANDS

T, B - Print out the top or bottom lines

The editor divides the text into blocks of approximately 3840 characters each, and only one such block is in memory at one time. The line numbers and the text of the actual first and last lines can be found by typing T or B followed by a return. This information is only of use when a block of data is to be operated upon during a Move command.

P, P mmmm, P mmmm-nnnn Print

The Print command will print line mmmm if only one decimal number is entered. It will print the last line referenced if no line number is entered and will print lines mmmm-nnnn if two numbers are entered separated by a dash. If line nnnn is not in the buffer, the command will print all lines in the current buffer and then go on to the next buffer. If a line or group of lines is requested which lie before the beginning of the current buffer, the program will close and reopen the file, find that line number and perform the indicated operation.

D, D mmmm, D mmmm-nnnn Delete

The Delete command will delete the current line if no number is entered, one line in any buffer if one number is entered, and all lines from mmmm-nnnn if two numbers are entered.

I nnnn - Insert

This command allows insertion of lines before line nnnn. As many lines as desired may be inserted here, with the exception that when the current text buffer is full, the Teletype bell will ring after each character, indicating that some action must be taken immediately. When all lines have been inserted, exit from the Insert mode by typing CTRL/D. The TAB character can be used to tabulate lines.

CTRL/W - Write out the text above the current line

When the core text buffer is full, the user must make a decision as to how it is to be subdivided for storage. The total core text buffer will hold 6656_{10} words, or enough for nearly two 3840 word disk tracks. The Teletype bell will begin to ring when 6528 characters have been entered. The CTRL/W command will print the line number, write out all of the text above the current line into a separate disk track, and move all text starting at the current line into a separate disk track, and move all text starting at the current line to the top of the text buffer. This allows room for additional insertions if necessary.

This operation is only necessary when the Teletype bell rings after every character of text, and the editor program will normally handle small overflows by moving that text to the top of the next disk buffer as it goes.

: Print out the current line number

The : command prints out the number of the current line in decimal.

SANCDEF - Search

The Search command is the most powerful one in DISKED as it can be used for highly sophisticated line modification. The string of text (in this case ANCDEF) entered following the S command is searched for starting at the current line number and continuing throughout all buffers to the end of the text. Spaces are allowed, but the correct number of spaces must be specified. For instance, the S command would not find

```
        MEMA @TEMP
    if
        MEMA @ TEMP
```

were specified as the search string. The string may be up to 72 characters long and is terminated with a Return. If no match is found, a question mark will be typed and the line pointer will be pointing to the last line in the last buffer. It is, of course, possible that a search will miss a string, if it occurs <u>before</u> the current line number. If you feel that this has occurred, reset the line number to 1 by typing P1, and when the first line has been printed, try the search again.

Once the search has found its match, the program will print out the line up to the end of the search string and await modifications. These modifications can be the insertion or deletion of characters here by typing new characters or Rubouts. The following commands are also available:

```
CTRL/R _  Print out the rest of the line and leave the Search mode.
CTRL/N -  Look for the next occurrence of the search string.
Return -  Terminate the line at this point.
CTRL/O -  Delete the entire line and allow entry of new characters.
          ^0 is printed.
ALT MODE- Divides line into two at this point.
```

For example if the command STEM is given, the result might be the finding of the line MEMA @ TEMP and the printing out of

```
MEMA @ TEM
```

The CTRL/R command would cause the P to be printed and no modification to be made on the line. However, the M could be deleted by typing a Rubout, an R inserted by typing an R and the remaining letters printed out by typing CTRL/R. The Teletype would show the following for these operations:

```
          MEMA @ TEM\M\RP
      #P
          MEMA @ TERP
      #
```

The TAB character can be searched for or inserted or deleted in the Search mode.

A nnnn - Add Text at the beginning of line nnnn

This command allows code to be inserted at the beignning of a line of text without using the search mode to find it. The usual method would be to print the line and then type A followed by a Return, enter the required text and finish the line with CTRL/R. For example, to add a label to line 15, we would type:

```
P15
MEMA ABCD
A
LABEL, (type CTRL/R) MEMA ABCD (line is finished and the A command exits)
P
LABEL, MEMA ABCD (This is the revised text line).
```

N - Next

This prints the next line in the same buffer. It advances the current line counter to that line.

+nnnn, -nnnn - Print lines + and - nnnn lines from current line

The + and - commands allow jumping through text within a given buffer by causing the printing of lines + or - nnnn lines from the current line. The current line then becomes that printed. Lines outside the current buffer will cause the NOT IN THIS BUFFER message to be printed. These lines can be accessed by the P command or by reading in a new buffer.

CTRL/B - Write out the current buffer and read in the next one

This command allows the next buffer to be read into memory after a NOT IN THIS BUFFER error message has been given. The command changes the T and B counters, and sets the current line number to the top of the buffer.

G - Get the next labelled line

G causes the printing of the next line containing a comma before a slash in the current buffer. It does not go beyond the current buffer and it always starts at the line after the current one.

C nnnn - Change line nnnn

This combines the Delete and Insert commands into one command. Only one line can be changed, however, as exit from this command occurs when the first Return is typed.

CTRL/Y - Yank in the next buffer

This command reads in the next buffer without writing out the current one. This command should be used carefully as it effectively deletes the entire current buffer. It can be most useful when used in conjunction with the extract command.

Vmmmm-nnnn/dddd - Move

The MoVe command moves a block of text from one location to another. The initial line number mmmm need not be in the current buffer, but the second line number nnnn must be within the same buffer as mmmm. The block is moved to before the line that was numbered dddd. After the move, of course, the number will become (dddd + nnnn mmmm + 1). The current line number will be dddd. The previous copy of the moved text, that occupied locations mmmmnnnn remains in the text and must be deleted by the user.

CTRL/C - Close the output file

This command writes out the current buffer, reads in all further input buffers and writes them into the output file. It calculates the length of this file and enters this, along with the file name, in the directory. The output file does not exist in the directory until it is closed, so this command is of utmost importance. A partially edited copy of the output file will exist if a line in an early buffer is called for after a line in a later buffer.

OUTPUT COMMANDS

L FILEQ - List

This causes the file named FILEQ.A to be listed on the Teletype.

W FILEQ - Write on low speed punch

This is exactly the same as the L command except that leader and trailer are also punched.

H FILEQ - Punch

This causes the file FILEQ.A to be punched on the high speed punch.

MANIPULATION COMMANDS

X mmmm-nnnn FILE2 - Extract

This causes lines mmmmnnnn of the current input file to be extracted, and stored as a separate file named FILE2.A. The lines mmmm and nnnn need not be in the same buffer. An Edit must be in progress for this command to be allowed. However, the original output file specified in the Edit command is a dummy and is destroyed by the X command. The extracted file is closed and no further editing can be done on the input file. It must be reopened with a new E command.

K FILEX FILE1 FILE7 TEMP5 - Combine

This command combines files FILE1, FILE7 and TEMP5 into one new file called FILEX. Error messages are printed if FILEX already exists or if the input files do not. Dollar signs at the end of all but the last file are deleted. As many files can be specified in the list as can be typed on one line.

M - Jump to DEMON

Causes a jump to 7600 and starts the disk monitor. This will not be allowed if an Edit is in progress. If you wish to abort an edit and restart the monitor, you must use the switch register.

CTRL/N - Jump to Nicobug II

This causes a jump to 4700. If Nicobug is loaded there, fine; otherwise disaster may strike.

CTRL/G FILE1 FILE2 - Garbage Collection

This command compresses FILE1 to use disk space more efficiently and writes the result in FILE2. This can be useful if FILE1 was subjected to extensive deletions or if it was produced by combining short files using the K command. Note that CTRL/G is the BELL on most terminals and the bell will ring in this case.

III. DEMON/II Disk Command Interpreter

(included in DEMON/II NIC-26-40614)

The DEMON/II Disk Command Interpreter (DCI) is a routine located on track 11 of the DEMON/II Monitor which accepts input from the teleprinter and sets up as output tables of input and output files and devices which then can be used by any calling program. These tables can then be used with the Disk I/O Supervisor. The following is a general description of the DCI. This description is of use since a number of Nicolet programs, including the Assembler and Loader utilize commands from the DCI.

When the DCI is called into core and started, it types a carriage return-line feed and then prints a commercial sign (@) on the Teletype. The general format of a command string appears as follows:

```
@INPUT1,INPUT2/OUTPUT [maximum Filelength ] :OPTIONS
```

The commercial is printed by the DCI. The slash (/) separates the input files from the output files and the comma (,) separates the individual files from each other. If no slash is present, all files are regarded as input files. For example, in the command

```
(@INPUT1, INPUT2 , OUTPUT
```

all three of the files would be regarded as input files as there was no slash. In the next example, all the files in the command string are regarded as output files.

```
@/INPUTl, INPUT2, OUTPUT
```

Whether both input and/or output files are needed depends, of course, on the requirements of the program that calls the DCI.

Devices and Filenames

The general format of a file is as follows:

```
FILENAme.Extension-Device
```

where FILENAme is the name of the file, Extension is a one letter extension to the filename and DEvice is the logical name of the device which the filename is on. Presently, the DCI accepts the following devices:

Logical Name	Device	Software Device Number
Dl	Disk Unit 1	1
D2	Disk Unit 2	2
D3	Disk Unit 3	3
D4	Disk Unit 4	4
HT	High Speed Paper Tape	5
LT	Low Speed Paper Tape	6

The Logical Device Name is separated from the filename and extension by a dash (-). A space is not permitted. If no device is specified, Dl is assumed. In the case of the paper tape devices (HT,LT), a filename can be given but is ignored. The dash, however, still must proceed the Logical Device Name (ie. -HT is legal whereas HT is not). A filename can be any number of letters but only the first six are significant and the remainder are disregarded. The extension, which is separated from the filename by a period, should be either a A,B or C. Whether the extension needs to be included depends on the individual program. If included, only a period should separate the filename and extension.

The following extensions are meaningful to all IMP programs.

blank - core image file. This is a copy of a memory region stored on disk.

.A - ASCII file. This is the text produced by DISKED or ASM and contains 8-bit ASCII characters, packed 5 per 2 words.

.B - BASIC file. Produced by Nicolet BASIC. Maybe either a program or a data file.

.C - Binary paper tape image file. This is a disk representation of a binary tape which can only be loaded using the DISK LOADER program.

Special Characters

The Disk Command Interpreter regards the following characters as special characters and the following action will be taken whenever they are encountered.

Rubout

Typing a Rubout will delete one character to the left for each time it is struck. The deleted characters will be enclosed in the back slashes (\). For example, if

```
©ABCDEF
```

was typed and the F and the E were to be deleted and a Z added, the rubout key would be struck twice producing the following output:

```
@ABCDEF\FE\Z
```

Internally, the string becomes

```
ABCDZ
```

Line Feed

The Line Feed key will cause the DCI to print the command string as it appears internally with all deleted characters missing. For example, if Line Feed is typed after

```
@ABCDEF\FE\Z
```

the DCI prints

```
ABCDZ
```

and await more input which is then appended to the string after the Z.

CTRL/O

CTRL/O prints ^0 and deletes the entire line and allows the user to type a new command.

CTRL/Q

CTRL/Q causes the Disk Command Interpreter to exit to the DEMON/II Monitor.

Return

Return causes the DCI to start building the tables derived from the command string. If no errors occur, it will exit to the program from which it was called. If an error occurs, the DCI will print another commercial and await a new command string.

Options

Options consist of ASCII printing characters which are preceded by a colon and followed by a space (or carriage return). Options can appear on either side of the input/output delimiter and can appear more than one in a command string. For instance,

```
INPUT:B /OUTPUT:FG
```

is legal. However it is usually convenient to group the options at the end of the line. The meaning of each option is decoded by the calling program.

Optional File Length

The Optional File Length is an octal number enclosed in brackets ([ ]) which is the maximum number of tracks which an output file will occupy. This is useful in optimizing storage on a file structured device since an empty space large enough to hold the file will be selected rather than the largest empty. On input files and non-file structured devices this number is disregarded. Below is an example of usage.

```
@INPUT:A/OUT1 [3D ,OUT2,OUT3C103
```

The first output file will have a maximum length of three tracks, the second is unspecified and the third output file has a minimum length of ten octal or eight decimal tracks.

Error Messages

All errors are fatal. An error free line must be processed before a return to the user program can be made.

SYNTAX ERROR

The command interpreter encountered a mistake in the syntax of the command string.

ILLEGAL DEVICE

There is no Logical Device Name for this device.

NAME.X NOT FOUND

The filename NAME with the assumed extension X was not found on the device specified. The extension may not be the one typed in as each program has the capability of giving the DCI an assumed extension which is used for a search if the search with the original one failed.

Programming Using the Disk Command Interpreter

This section describes programming using the DCI. It can be disregarded by those only interested in responding to it.

The Disk Command Interpreter resides on Track 11 of the DEMON/II Monitor and is 1000(8) words long. It must be called in at 6000 and 3000-7577 should be stored on tracks 1 and 2 of the Monitor. Below is an acceptable call in of the DCI.

```
               ONEA                /WRITE
               JMS @ DISK          /FIRST STORE 3000-7577
               100001              /ON TRACKS 1 AND 2
               4600                /STORE 4600 WORDS
               3000                /START AT 3000
               ZERA                /READ
               JMS @ DISK
               100011              /FROM TRACK 11
               1000                /1000 WORDS LONG
               6000                /LOAD AT 6000
               ZERM @ DEVDIR       /SET SWITCH TO INDICATE THAT CORE SEGMENT
                                   /3000-5777 IS IN CORE (RATHER THAN ON DISK)

DISK,          7612                /ENTRY POINT TO DISK HANDLER
DEVDIR,        7764                /CORE SEGMENT SWITCH
```

Once the DCI is in core, it is started by performing a JMS to location 6000. After the JMS there should be three arguments which are used by the DCI. The first argument is a pointer to the Input/Output table buffer. The second argument is a pointer to the Option table buffer. The third argument is the ASCII value of the assumed extension which is used if the initial search for an input file fails. The following is an acceptable

call to the DCI.

```
          JMS @ DCI
          IOPNT               /POINTER TO INPUT/OUTPUT TABLE
          OPNT                /POINTER TO OPTION TABLE
          301                 /ASSUMED EXTENSION (A)
                              /RETURN HERE

IOPNT,    BLOCK 20            /RESERVE 20 LOCATIONS FOR INPUT/OUTPUT TABLE
OPNT,     BLOCK 10            /RESERVE 10 LOCATIONS FOR OPTION TABLE
DCI,      6000                /ENTRY POINT OF DISK COMMAND INTERPRETER
```

Format of Input and Output Tables

An input table entry is three locations long and has the following format:

```
     IENTRY1   /DEVICE #
     IENTRY2   /STARTING TRACK.  ZERO IF NON-FILE STRUCTURED

     IENTRY3   /WORD COUNT.  ZERO IF NON-FILE STRUCTURED
```

The end of input entries is designated by a 3777777 (-1).

The output table entries are four locations long.

```
     OENTRY1   /DEVICE #
     OENTRY2   /FIRST THREE CHAR. OF FILENAME, ZERO IF NFS
     OENTRY3   /SECOND THREE CHAR. OF FILENAME AND EXTENSION
     OENTRY4   /MAXIMUM WORD COUNT IF SPECIFIED, ZERO OTHERWISE
```

The output file entries are terminated by a 0.

If there are no input files, the start of the table will contain a -1 and if there are no output files, a zero will follow the input terminator of -1.

Format of the Option Table

The option table simply contains the ASCII values of the option character, one character per word. After each string (one or more characters) a zero is stored to indicate the end of the string for that file. The option table is terminated with a 3777777.

Example of Usage

```
@INP1.B:D /0UT1 [1O],-HT:ZS
```

The above command string would be parsed as follows by the DCI.

```
        JMS @ DCI           /CALL DCI
        IOPNT               /POINTER TO INPUT/OUTPUT TABLE
        OPNT                /POINTER TO OPTION TABLE
        303                 /ASSUMED EXTENSION (C)
                            /RETURN

IOPNT,  BLOCK 20            /RESERVE 20 LOCATIONS
OPNT,   BLOCK 5             /RESERVE 5 LOCATIONS
DCI,    6000
```

After execution of this routine, the following tables would be set up.

```
IOPNT,  0000001        /DEVICE #
        0000300        /STARTING TRACK
        0007600        /WORD COUNT
        3777777        /INPUT ENTRIES TERMINATOR
        0000001        /DEVICE #
        0576564        /FIRST THREE CHAR OF FILENAME
        0210000        /SECOND THREE CHAR AND EXTENSION(NONE)
        0030000        /MAXIMUM FILE LENGTH IN WORDS
        0000005        /DEVICE # OF SECOND OUTPUT ENTRY
        0000000        /NO FILENAME (NON-FILE STRUCTURED)
        0000000
        0000000        /NO WORD COUNT SPECIFIED
        0000000        /TERMINATES OUTPUT ENTRIES
              0
              0
              0
OPNT,       304        /OPTION D
              0        /STRING TERMINATOR
            332        /OPTION Z
            323        /OPTION S
              0        /STRING TERMINATOR
             -1        /END OF TABLE
```

Note that if input file INP1.B was not found a search for INP1.C would be performed.

Cautionary Notes

It is possible for the command string to overflow your buffers. It is a good idea to check the addresses of the terminators to check for overflow. Do not call the DCI from or have your table pointers in this 6000-7577 region. The DCI does not dismiss itself but does leave 3000-5777 in core when it returns to the user program. The scratch area in the Monitor Head is used since the DCI calls DIRFUN to look up the input files to see if they exist.

IV. Disk Assembler (ASM)

(NIC - 29-40515)

The Disk Assembler is a program which translates the Nicolet 1080 mnemonic codes into a binary format which can then be loaded into memory and executed. ASM has the capability of storing up to 1710 user and permanent symbols on a 12K system. The large size of the symbol table allows a user to assemble extremely large assembly language programs and thus let the assembler resolve addressing problems at assembly time instead of having the programmer do it with smaller sections. Also ASM has a large number of Pseudo-operators which also ease the burden of programming. ASM is a three pass assembler. In its first pass through the text, it creates a symbol table which is stored in memory. During its second pass, it produces a binary tape or disk file and during its third pass a listing. These three functions are commonly referred to as Pass 1, Pass 2 and Pass 3.

Loading Procedure

ASM must be used with Demon/II Disk Monitor. When the monitor is in residence, place the ASM binary tape in the appropriate reader and type

```
BINLDR
```

and Return. In the case of the Teletype, turn the reader to start. After the processor and reader stop, remove the tape from the reader and restart the monitor at 7600. To store the program type

```
STORE ASM 0-7577;0 :P
```

and Return. The program is now stored on disk for future use.

Using the Program

To run the program, type

```
RUN ASM
```

and Return. The program will start and move the permanent symbol to 106000 and then call in the Disk Command Interpreter which will then print a commercial (@). ASM will assemble up to four input files and can create a binary output file, and also a listing output file compatible with the Disk Editor (DISKED). The input files must all have .A extensions. Input is not allowed from a paper tape device (ie. high speed reader or Teletype). The following options are available.

E Error Analysis. No output files are needed for this operation. The assembler will look for errors in the source file(s) and if found print them on the Teletype or terminal.

B Binary. The source is assembled and a binary output file is created. If this file is written onto disk it will have a .C extension.

L Listing. The source is assembled and a listing file will be created. As mentioned previously, if a listing is stored on disk, it can be listed and searched by using the Disk Editor. One must be careful not to use the same name for the listing file as the source file(s).

T Tabulate. This option forces the assembler to insert eight leading spaces on non-labeled lines. This feature enhances the format of the listing, especially for lazy programmers who do not indent their non-labeled code. This option can be used with the L and F options.

F Full options. This option causes the assembler to perform both the binary and listing passes. Two output files must be specified with the binary being the first of the two.

X Convert tabs to spaces. Whenever a tab is encountered, a space is printed instead of a tabulation.

ASM always returns to the monitor when finished. The program is not restartable. It must be reRUN every time it is used. Control can be transferred to the monitor by typing CRTL/Q during execution.

Special Characters

Legal characters consist of the numbers 0-9 and the letters A-Z and the special characters listed below. Symbols can only be formed from the alphanumeric set with the exception of A-M, A+M, M+A and M-A. Conversely the symbols M and A are illegal since they have special meaning in the Nicolet mnemonic codes.

, comma — The comma defines a label. ex:

```
*1000
TEMP, 0 /The comma defines TEMP to 1000.
```

+ plus — Adds symbols or numbers. Arithmetic is performed in the order of occurence. ex:

```
MEMA TEMP+1 /Load the contents of the address
following TEMP.
```

minus — Negates symbols or numbers. ex:

```
MEMA TEMP-1 /Load the contents of the address
            /preceeding TEMP.
```

! exclamation — Multiplies symbols or numbers. ex:

```
MEMA TEMP!2 /Load the contents of the address of
           /twice TEMP.
```

	space	Combines an delimits symbols and numbers. Spaces should not be imbedded between other items of syntax.
*	asterisk	Set current location counter. ex: *200 /Set PC to 200.
	Return	Terminate line.
	Tab	Same function as a space
=	equals	Define parameters. ex: TEMP=1000 /Set TEMP to 1000 MEMA TEMP /Equivalent to MEMA 1000 Note, do not imbed spaces either before or after the equal sign.
/	slash	Indicates start of comment.
	quote	Obtain ASCII value of following character. ex: MEMA ("A /Equivalent to a MEMA (301
@	commercial	Set indirect bit.
(	left parens.	Set immediate mode.
#	number sign	Indicates value of current location counter. ex: *200 TEMP, # /Location 200 contains 200
$	dollar sign	Terminates pass.
;	semi-colon	Floating point constant. Must be used with label.
<	less than	Delimit conditional assembly.
>	greater than	Limit conditional assembly.

Description of the Pseudo-Operators

Pseudo Ops are special assembler instructions for performing special tasks that generally make programming easier. Use of the Pseudo Op name in a manner other than described will often cause the assembler to crash. Therefore, do not use their names as labels!

TEXT

The TEXT Pseudo Op packs a character string into a stripped ASCII format. The general format of this Pseudo Op is as follows,

```
TEXT ZNNNNNZ
```

where Z is a delimiting character and N is any printing character except «-. A space must separate TEXT from the delimiting character. When the second delimiting character is encountered, a 77, which is the stripped ASCII terminating code, is inserted in the binary. For example,

```
TEXT %HELLO THERE%
```

would be assembled as follows

```
504554    TEXT %HEL
545700    LO
645045    THE
624577    RE%
```

Notice that in this example the percent sign (%) was used as the delimiting character and caused a 77 to be inserted at the end of the string.

PAGSKP

The PAGSKP Pseudo Op forces the listing to skip to the top of the next page. This is useful in separating sections of code. This command is also given internally by the TITLE Pseudo Op.

BLOCK

BLOCK is used to reserve storage with zeroes. The general format of the BLOCK is

```
BLOCK n
```

where n is the number of sequential locations to be filled with zeroes. The number can be an octal or decimal constant or alternatively it can be an expression. If so, all labels used in the expression must be defined previous to that point or an assembly error will occur. If the value of the expression is minus, an IR error message will be printed and the Pseudo Op will be aborted.

DECIMAL or DECIMA

One bothersome programming detail is the searching for an octal equivalent of a decimal number. The DECIMAL Pseudo Op causes all numbers encountered after it to be treated as decimal instead of octal.

Ex.

```
               *0
0    100   C100,     100  /100 OCTAL
           DECIMAL
1    144   D100,     100  /100 DECIMAL
2   1750   D1000,   1000  /1000 DECIMAL
```

OCTAL

The OCTAL Pseudo Op forces the number radix of the assembler back to octal. Since the radix of the assembler is normally in octal, this command is only needed after the DECIMAL Pseudo Op.

FIXTAB

This Pseudo Op appends all symbols previously encountered to the permanent symbol table. They will not be printed on the symbol table listing. This Pseudo Op should only be used after EXPUNGE or before actual program coding.

EXPUNGE

EXPUNGE zeroes the permanent symbol table excluding the Pseude Op section. Therefore, the symbols A+M, M+A, M-A and A-M are not affected.

NOLIST

In a large number of cases, only a small section of coding is changed in an assembly. In some assemblers, all the source must be listed in order to view a certain section. The NOLIST Pseudo Op suppresses listing. Coupled with the LIST Pseudo Op, it can be used to list a section of code. If NOLIST is still set at the end of the listing pass, no symbol table will be printed.

LIST

The Pseudo Op enables pass 3 output. This is the default listing mode.

NOPUNCH or NOPUNC

NOPUNCH halts binary output on pass 2. Used in conjunction with STPUNCH, it can be used for generating overlays and relocatable code. Below is an example of what is meant by relocatable.

```
          *0
          NOPUNCH
          *100          /CHANGE THE ORGIN BUT DON'T PUNCH IT
          STPUNCH       /ENABLE PUNCHING
          MEMA TEMP     /THIS IS LOADED AT 0, BUT ASSEMBLED AS IF
TEMP,     0             /AT 100
```

STPUNCH or STPUNC

STPUNCH enables binary output on pass 2. This is the default mode.

TITLE

The pass 3 page heading is generated from the first line in the source. The TITLE Pseudo Op allows the user to change the heading during the listing. It has the general format,

```
TITLE XZZZZX
```

where X is a delimiting character and Z is a printing character. For instance,

```
TITLE %CHANGE THE HEADING!%
```

would cause the heading

```
CHANGE THE HEADING!
```

to appear on succeeding page headings. TITLE also causes a PAGSKP.

TAPEND

ASM can assemble more than one input file. TAPEND causes the assembler to terminate the current file and fetch the next one. If TAPEND is not present, a PH (phase) error occurs when more than one file is assembled.

ASMIFZ

ASMIFZ stands for ASseMble IF Zero. The general format is as follows:

```
ASMIFZ expression or symbol
<code
>
```

If the value of the expression or symbol following ASMIFZ is zero, then the code delimited by the less than (<) and greater than (>) character will be assembled. If the expression is not zero, then the code enclosed will be ignored. This Pseudo Op can be nested. Below is a example of how conditional assembly can be used.

```
SWTCH=0
ASMIFZ SWTCH
<
MEMA TEMPI          /ASSEMBLE IF SWTCH=0
          >
ASMINZ SWTCH
<
MEMA TEMP2          /ASSEMBLE IF SWTCH=1
          >
```

If the symbol SWTCH is set to zero as shown here, the line MEMA TEMPI is assembled and the line MEMA TEMP2 is ignored. If the symbol SWTCH was defined as non-zero by SWTCH=1, the line MEMA TEMP2 would be assembled. The MEMA TEMPI line is then ignored. Assembler instructions such as NOLIST or DECIMAL within conditional assemblies are ignored if that section is not assembled.

ASMINZ

ASMINZ stands for ASseMble if Not Zero. This Pseudo Op is the complement of the ASMIFZ in that the delimited code is assembled if the expression is not zero. ASMIFZ and ASMINZ can be nested together.

Address Arithmetic

One programming mistake that is commonly made is overstructuring of the program. For instance, lists have no provision for additional entries, starting points are fixed, etc. Below are two examples of lists, one using the assembler's arithmetic capabilities and another which could be coded by hand with little difficulty.

```
/THIS COULD BE CODED BY HAND
ACLIST,        1000     /STARTING ADDRESS
LCNT,             5
              *1000
             303240     /100000
              23420     /10000
               1750     /1000
                144     /100
                 12     /10
                  $
```

```
/THE ASSEMBLER COULD DO THIS ONE
ACLIST,   XLIST          /STARTING ADDRESS OF LIST
LCNT,     CNT            /# OF LOCATIONS IN LIST
DECIMAL                  /SET RADIX TO DECIMAL
XLIST,    100000
           10000
            1000
             100
              10
     CNT=#-XLIST         /CALCULATE LENGTH OF LIST
          OCTAL          /RETURN TO OCTAL RADIX
               $
```

Notice in the second example that the origin setting is unimportant and that in the other it is fixed. The second list could be assembled on any page and still function correctly, but the first list would require changes in the origin setting and pointer to the starting address. Also, if the number of items in the list changed, the first example would require a change to LCNT but in the second example, the assembler would automatically compensate for length changes. Since the symbol table is so large, one should not hesitate to use these features. However, when used in an expression, such symbols must be defined previous to that point.

Error Messages

Error messages have the general format

NN XXXXXX AT ZZZZ

where NN is the error code, XXXXXX is either the symbol name or octal value of the expression that caused the error and ZZZZ is the value of the current location counter. All error messages are printed on the Teletype during the first two passes and are printed on the listing on the third pass.

Error Codes

IS Illegal suffix. The suffices used are the same or one was used where it shouldn't have been.

NL No label. The label has not been defined on the first pass. The address in the instruction contains the local address of where the label was first encountered.

DL Duplicate label. This label has been previously defined. It is not redefined.

SE Symbol table exceeded. More than 1706_{10} permanent and user symbols are used.

IC Illegal Character. A character which the assembler considers illegal has been encountered.

IR Illegal reference. The page of the address and current page are not the same or a minus BLOCK size has been specified.

PO Pushdown overflow during parsing. The expression is too complex.

PU Pushup underflow. This is usually a machine error.

RD Redefinition of an expression.

IM Illegal immediate. There was no instruction present, the value of the immediate expression was greater than 2000 or the M suffix was used.

NO No output file.

PH Phase error. The number of input files used and the number specified do not agree.

II Illegal input. The high or low speed reader was specified as the input device.

HD Hardware error. An unrecoverable disk read error occured.

NR No room on disk for output.

NO, PH, II, HD and NR return to monitor. The PO and PU errors cause the current pass to be terminated and the next one initiated.

Examples of Usage

The following examples deal only with the setting up of the Input/Output specifications. The following example assembles one source file FT74.A on disk 1, creates a binary file FT74.C on disk unit 2 and puts the pass 3 listing on the low speed paper tape device (Teletype or terminal).

```
@FT74.A/FT74.C-D2,-LT:F
```

The F option was used since both the binary and listing were created. If the A extension was not used on the source file, first a file with no extension would be searched and if this was not found, a search for a file with the A extension would be performed before a FILE NOT FOUND error message is printed.

Below is an example of an error analysis of FILE1 on disk unit 2 and FILE2 on disk unit 1.

```
@FILE1-D2,FILE2:E
```

Notice that no output files were needed and that disk unit 1 is the default disk if no disk is specified.

Below is an example of a forced tabulated listing of FT74.A which would go on disk for examination by the Disk Editor. If LIST did not have an A extension, ASM would force the extension on.

```
@FT74.A/LIST.A:TL
```

If T was not used the listing would be non-tabulated unless tabs were used in the source.

V. Disk Loader Program (LOADER)

(NIC-30-40514)

Files having the .C extension are generally produced by the Disk Assembler and are simply images of what would have been put on paper tape if binary output to paper tape had been specified. They contain starting addresses, checksums and rubouts much as a binary tape would. They cannot therefore be loaded using the DEMON LOAD command, as this command expects a copy of a memory region called a core image file. The Disk Loader program has been designed to load these .C files into memory. Thus, it is really a Binary Loader for disk files that look like paper tape. Once these files have been loaded once by the Disk Loader, they can be STOREd using the DEMON STORE command as core image files which could be LOADed or RUN using DEMON. For versatility, the Disk Loader program also allows loading of core image files (those having no extension) but this feature is of somewhat lesser use.

Loading Procedure

This program must be used in conjunction with Demon/II. When the Demon/II Keyboard Monitor is in residence, place the LOADER binary tape in the appropriate reader and type

```
BINLDR
```

followed by a Return. If a low speed reader is being used, turn it to start. When the reader and processor stop, remove the tape from the reader and restart the monitor at 7600. To store the program on disk, type

```
STORE LOADER 100000-101500;100000 :P
```

and Return. The program will now be stored on disk for future use.

Program Usage

To use the program, type

```
RUN LOADER
```

and Return. LOADER then calls in the DCI which responds with a commercial (@).

The general format for loading a disk file named ABCDE in .C format is

```
@ABCDE:opt
```

where the options are L, M and G. Several files can be strung together and loaded at once by typing

```
@ABCDE,AB,FZ,Al:opt
```

The options have the meaning

```
       L    -    load the files and return to the LOADER
       M    -    load the files and return to DEMON/II
       G    -    load the files and start at 0
   G=nnnn   -    load the files and start at address nnnn
       C    -    load the core image file
```

If no options are given, L is assumed.

While ordinary binary files, such as spectra or paper tape loaded programs are most easily loaded using the DEMON/II commands, the LOADER will allow combinations of all three. Only one such file per command line is allowed, however.

When the LOADER is run, it intially destroys 100000-102777. This is of little consequence since if that section was saved on disk before the LOADER program was run, it can be reloaded using the C option and overlay the LOADER. Whenever a G or M option is used, all core is restored. After using these two options, you cannot type GO 100000 to restart the LOADER since it overlays itself with that code which was loaded into 100000-102777 or if none was loaded, with what was last on tracks 14 and 15 (the scratch loading area on disk).

If no extension was given on the input file (output files are ignored), first a directory search will be made for that name and if the search fails, the name with a C extension will be used for the search. You must be careful not to load a core image file instead of a binary or *vice versa.*

Examples of Usage

To load the file FT74.C which was produced by the Disk Assembler, type after the commercial sign of the Disk Command Interpreter:

```
©FT74.C:L
```

or

```
©FT74 . C
```

In order to load this program and start it,

GPT74.C:G

This starts the program at location 0. If the program were to be started at 1000, the command would have the following format.

```
@FT74.C:G=1000
```

Mow, suppose the binary FT74 tape file produced by this assembly does not have the Floating Point Package included. To load the FPP from the high speed reader and the FT74 file from disk, and return to the Monitor, the following command could be used.

```
@-HT,FT74.C:M
```

When the paper tape file is to be read in, the LOADER prints either an ^ or on the Teletype or terminal and waits for any character to be struck on the keyboard. This initiates reading of the paper tape. Each time a new paper tape file is to be read, the ^ or ^ will be printed. In order to load the core image copy of FT74 (generated by the DEMON STORE command), type

```
@FT74:C
```

Only one core image file can be loaded at a time.

Error Messages

BAD BINARY CHECKSUM!

The file read in had a bad checksum. Control returns to the Disk Command Interpreter for new input specifications. This can also occur if a core image file was specified instead of a .C file.

MORE THAN ONE CORE IMAGE!

More than one file was used when using the C option. Control returns to the Disk Command Interpreter.

READ ERROR!

The disk hardware error flag was set during the last operation. Control returns to DEMON/II.

MONITOR CANNOT BE OVERLAYED!

LOADER will prevent any intrusion into the Monitor Head as it could prove potentially fatal. Control returns to DEMON/II.

VI. Examples of Assembly, Editing and Loading

The following Teletype output was produced during the assembly, editing and debugging of a simple program to print out the word "TEST." It illustrates simple uses of the Editor, Assembler and Loader. The process starts by the creating if a file named TEST using DISKED.

```
* RUN DSKFD                          DISKED is started from DEMON

#FTEST                              The F command is used to begin a file named TE
/TEST PROGRAM
*0
START,   MEMA ("T         /T     Note the use of the TAB character to tabulate
         JMS TYPE                labels, code and comments.  This greatly
         MEMA ("F         /F      improves legibility.
         JMS TYPE
         MEMA ("S         /S
         JMS TYPE
         MEMA "T          /T
         JMS TYPE
         JMP 9 K7600      /RETURN! TO DEMON
K7600,   7600

TYPE,    0
         TTYPE
         JMP #-1
         PRTTY
         JMP @ TYPE



$
MORE TAPE?N                         Answering N here closes the file and allows
#M                                  other DISKED commands.  M causes a return to
                                     DEMON.
*RUN  ASM                           The Disk Assembler is started.

@TEST.A:E                           An error analysis is performed on the file
                                     TEST.A
*RUN  ASM                           The Disk Assembler is restarted

@TFST.A/TEST.C:\:\,-LT:F            The program is told to assemble the file
TEST.A/TEST.C,-LT:F                 TEST.A, produce a binary file named TEST.C and
#                                   a listing on the low speed tape device
                                     (Teletype).  A Line Feed was struck after the
                                     first line to get a clean copy of the command
                                     string before executing it.
```

```
/TFST  PROGRAM                  Note that the title is the first printed line
                                unless the TITLE Pseudo-Op is used.



/TFST  PROGRAM
*0
     0   110324 START,  MEMA  ("T         /T
     1 2000012          JMS  TYPE
     2   110305     MEMA     ("E          /E
     3 2000012          JMS  TYPE
     4   110323     MEMA     ("S          /S
     5 20000 12         JMS  TYPE
     6 2110324          MEMA "T           /T
     7 2000012          JMS  TYPE
    10 1000011          JMP @ K7600   /RETURN   10 DEMON
    11     7600 K7600,  7600

    12        0 TYPE,   0
    13     6444         TTYPE
    14     13       JMP #-1
    15     4443         PRTTY
    16 1000012          JMP @ TYPE

                               CTRL/Q is typed to abort the listing after the
                               text and before the symbol table.



     *RUJ \ J\M LOADER   The loader is started.

     @TEST.C:G           The program TEST.C is loaded and started at 0.
     TES                 But only the characters TES are printed out.
```

Clearly there is a bug in the program TEST since it does not print out the final T as we wanted it to. Therefore we look back at the listing and discover that at location 6 the code MEMA "T is used rather than MEMA ("T. This loads the contents of address 324 instead of the number 324 into the AC.

Therefore, in order to get this program to work, we must generate a new file with this missing left parenthesis added. This is shown on the following page.

```
*RUN DSKED                           The Disk Editor is started

#ETEST TEST1                         Input file is TEST, output file is TEST1

#S"T                                 We search for "T

START,  MEMA ("T         /T          The first occurence is in a legal text line.

        MEMA "T\T"\("T                   /T

                                     But the second occurence is at location 6.  We
                                     rub out two characters, insert the parenthesis
                                     type the two characters back in and type CTRL/I
                                     to finish the line.



#↑C                                  The character CTRL/C is used to close the file
                                     writing the file TEST1.A onto the disk.
#M                                   We return to DEMON with the M command.

*RUN  ASM                            and re-run the assembler.

@TEST1/TEST1,-L↑O                    This line is in error and is aborted with CTRL/
TEST1/TEST1:B                        We produce a new binary file TEST1.C.  No list-
                                     ing is generated.

*RUN LOADER                          The Loader is started

@TEST1:G                             and the file TEST1.C is loaded and started at
TEST                                 It works this time.
*STO S\S\TEST1 0-16;0                We store it as a core image file.
DELETE:Y                             deleting an old version.
*RUN TEST1                           We then run the core image file from DEMON
TEST                                 It too works, of course.
*
```

VII. Disk Transfer Program (MOVE)

(NIC-31-40611)

MOVE can be used to transfer files from disk, to paper tape devices or vice versa using the DEMON/II monitor routines.

Loading Procedure

This program must be used in conjunction with the DEMON/II monitor. When the DEMON/II monitor is in residence, place the MOVE binary tape in the appropriate reader and type

```
BINLDR
```

and Return. If a low speed reader is being used, turn it to start. When the reader and processor stop, remove the tape from the reader and restart the monitor at 7600. To store the program on disk, type

```
STORE MOVE 0-1777;0 :P
```

and Return. The program will now be stored on disk for future use.

Using the Program

To use the program type

```
RUN MOVE
```

and type Return. MOVE will load and start and then call in the Disk Command Interpreter to process your input/output specifications. Any number of binary and ASCII input files can be combined, but MOVE makes no attempt to change the format. For instance, rubouts, dollar signs, leader and trailer are not trapped for and are passed on to the output file. If more than one output file is specified, only the first is used. If a core image file (null extension) is to be transfered, a C option must be used in your command string for the Disk Command Interpreter. The B option will convert a core image file to binary paper tape format. This is useful whenever a core image file is transfered to paper tape as a core image file has no meaning on paper tape. In addition, only one input file can be specified whenever a core image file is transfered. When transferring a file that is larger that 50 tracks, the H option must be used.

Examples of Usage

To transfer two ASCII paper tapes from the high speed reader to a file on disk unit 1 called SCR.A, the following command string could be used.

```
@-HT,-HT/SCR.A
```

For each paper tape which is read in, a ↑ or ^ will be printed on the Teletype or terminal and then the I/O routine will wait for a character to be struck on the keyboard before the tape reading is initiated. Notice also that the disk unit on SCR.A was not specified since it was unit 1.

To transfer the core image file FT74 from disk unit 2 to disk unit one would type

```
@FT74-D2/FT74-D3:C
```

Error Messages

NO OUTPUT FILE

An output file was not specified in your command string.

NO ROOM ON DISK

There isn't enough free space on disk to complete the transfer

MORE THAN ONE CORE IMAGE FILE

There was more than one input file when the C option was used.

HARDWARE ERROR

An unrecoverable disk read error occured.

ILLEGAL INPUT

The B option was used when the input file was a paper tape device.

E option [illegible] tape binary to core image

[illegible]

VIII. General Input-Output Handler (IOSUPER)

(part of DEMON/II - NIC-26-40614)

IOSUPER is a collection of routines that handle input and output from the disk, high and low speed paper tape devices. It is a powerful programming tool for disk swapping and transfers. The provision is made for additional devices to be assembled into the program in order to utilize other devices available for a given system. This program resides on Track 12 of the DEMON/II monitor and is $1000_{(8)}$ words long. When it is loaded into core, it must be loaded at 6000. Also, the core locations from 3000-7577 should be stored on tracks 1 and 2 if that core area is to be saved. Location 7764 should be set to zero if the core segment 3000-5777 is indeed in that area and to a -1 if the core segment 6000-7577 is in locations 3000-4577. Otherwise it is set to the number of the disk directory that is currently in core, which is set by DIRFUN. The program will exit after the I/O transfer with the core restored, if prior to the calling of the routine the locations 3000-7577 were saved on tracks 1 and 2.

Capabilities of IOSUPER

1. Perform a block read operation.
2. Perform a block write operation.
3. Search for a file and perform a block read.
4. Load a file using its directory information.
5. Store a file.

The latter three operations are of course only applicable to a file structured device such as a disk.

Calling Sequences

A typical calling routine for IOSUPER is as follows.

```
     ZERA=read           ONEA=write
          JMS @ IOSUPER          /JMS TO 6000
          DEVICE #               /DISKS 1-4 OR AS SHOWN
          WORD COUNT             /IGNORED ON DIRECTORY CONTROLLED READS
          TRACK #                /Ø MEANS FIND EMPTY ON WRITE OR USE DIRECTORY
                                 /ON READ
          ADDRESS                /-1 MEANS USE DIRECTORY ADDRESS
          FILPNT                 /POINT TO ZERO NAME MEANS BLOCK TRANSFER, NO
                                 /DIRECTORY INFO
          ERROR RETURN
          NORMAL RETURN
               .
               .
               .
               .
               .
IOSUPER,     6000
```

The entry point of the subroutine is 6000. After the subroutine call there are five arguments that the program uses to perform the transfer. Whether all of these arguments are used in a given operation depends upon the operation. The AC should be zero to indicate a read operation and non-zero to indicate a write operation. The first argument after the subroutin call is the Device number. Table I contains the correspondence between the Device number and the Logical Device Name. IOSUPER only allows numbers between 1 and 7 but as mentioned previously, this can be changed to reflect additional devices. If an illegal device is encountered, ID will be printe on the Teletype and control will be transfered to the Disk Monitor. The second argument is the word count of the transfer. This must be specified for a block transfer but may be left zero if desired in a

Search block read operation (3).
Load operation (4).

The third argument is the starting track of the transfer (ignored by paper tape devices). On a read operation if this argument is zero, IOSUPER will use the file name which is pointed to by argument five to perform a directory search and use that information for the starting track. For a block transfer this location should contain the actual starting track. The fourth argument contains the core address. For a load operation, this location should contain a -1 in order that the directory information is used for load. The fifth argument is a pointer to a two word filename which will be used in case a directory operation is needed. On a simple block transfer operation, this pointer should point to a zero.

TABLE I

NAME	LOGICAL DEVICE NAME	DEVICE NUMBER
DISK#1	D1	1
DISK#2	D2	2
DISK#3	D3	3
DISK#4	D4	4
High Speed Paper Tape	HT	5
Low Speed Paper Tape	LT	6
Optional	-	7

Use of the Individual Devices

The disk unit requires no interaction with the user other than being accessible. The tape readers, both high and low will print a ↑ or ^ on the Teletype or TI printer when accessed. The user should then type any key on the keyboard to initiate reading of the tape. On the Teletype simply turn the reader to start. The paper tape input devices sense the end of input by monitoring the time between characters. If the next character isn't read within a given time period, it knows that the tape has stopped reading. There is no user interaction with the paper tape output devices.

Table II contains the arguments and uses for IOSUPER.

TABLE II

```
/BLOCK READ
     ZERA                /AC IS ZERO FOR READ
     JMS @ IOSUPER
     DEVICE
     WC                  /WORD COUNT MUST BE SPECIFIED
     STRACK              /STARTING TRACK MUST BE SPECIFIED
     BUFADD              /CORE ADDRESS MUST BE SPECIFIED
     ZPNT                /POINTS TO ZERO
     ....                /ERROR RETURN OR END OF TAPE

/LOAD FILE AT SPECIFIED CORE ADDRESS
     ZERA                /AC IS ZERO FOR READ
     JMS @ IOSUPER
     DEVICE
     0                   /WORD COUNT IS ZERO
     0                   /STARTING TRACK IS ZERO
     BUFADD              /ADDRESS WHERE FILE WILL BE LOADED
     NAMPNT              /POINTS TO FILENAME
     STOP                /ERROR RETURN

/LOAD FILE USING DIRECTORY INFORMATION
     ZERA                /AC IS ZERO FOR READ
     JMS @ IOSUPER
     DEVICE
     0                   /WORD COUNT IS ZERO
     0                   /STARTING TRACK IS ZERO
     3777777             /CORE ADDRESS MUST BE -1 TO USE DIRECTORY INFO
     NAMPNT              /POINTS TO FILENAME OF FILE
     STOP                /ERROR RETURN

/BLOCK WRITE
     ONEA                /AC IS NON-ZERO FOR WRITE
     JMS @ IOSUPER
     DEVICE
     WC                  /WORD COUNT MUST BE USED
     STRACK              /STARTING TRACK MUST BE SPECIFIED
     BUFADD              /BUFFER ADDRESS MUST BE SPECIFIED
     ZPNT                /POINTS TO ZERO
     STOP                /ERROR RETURN

/STORE FILE ON DISK AND ENTER IN THE DIRECTORY
     ONEA                /AC IS NON-ZERO FOR WRITE
     JMS @ IOSUPER
     DEVICE
     WC                  /WORD COUNT MUST BE SPECIFIED
     0                   /IOSUPER WILL FIND A BLOCK
     BUFADD              /BUFFER ADDRESS MUST BE SPECIFIED
     NAMPNT              /POINTER TO FILENAME OF FILE
     STOP                /ERROR RETURN
```

Examples of Usage

The following subroutine will load IOSUPER into core. Locations 3000-7577 are first stored on tracks 1 and 2.

```
IOSIN,    0
          ONEA            /WRITE OUT 3000-7577
          JMS @ DISK
          100001          /TRACKS 1 AND 2
          4600            /4600 WORDS
          3000            /AT ADDRESS 3000
          ZERA            /SIGNAL READ
          JMS @ DISK      /CALL MONITOR HEAD
          100012          /TRACK 12, DISK 1
          1000            /1000 WORDS LONG
          6000            /LOAD AT 6000
          JMP @ IOSIN
DISK,     7612
```

The following example performs a block read operation of 2000 words from track 130 of disk unit 1 into addresses 4500-6477.

```
          JMS IOSIN          /CALL IOSUPER INTO CORE
          ZERAM @ XDEVDIR    /CORE SEGMENT 3000-5777 IS IN CORE
          JMS @ IOSUPER      /AC IS ZERO FOR READ, CALL IOSUPER
          1                  /DEVICE #1
          2000               /WORD COUNT=2000 WORDS
          130                /STARTING TRACK
          4500               /CORE ADDRESS
          ZPNT               /POINTER TO A ZERO FILENAME
          STOP               /ERROR RETURN (HARDWARE AND SOFTWARE)
          ....               /NORMAL RETURN, ALL CORE RESTORED.
ZPNT,     0
IOSUPER,  6000               /IOSUPER ENTRY POINT
XDEVDIR,  7764               /CORE SWITCH
```

A disk write operation has the same format except the AC is non-zero on entry to IOSUPER. For a paper tape device the starting track and filename pointer are ignored.

The following example searches for the file TEMP on disk unit 4 and loads it in core at 110000. In this case, the directory word count is used and the word count specified by the user is ignored.

```
          JMS IOSIN           /CALL IOSUPER INTO CORE
          ZERAM @ XDEVDIR     /CLEAR CORE SWITCH AND SIGNAL READ
          JMS @ IOSUPER
          4                   /DISK UNIT 4
          0                   /WORD COUNT IS IGNORED
          0                   /TRACK NO. MUST BE ZERO TO USE DIRECTORY INFO
          110000              /LOAD HERE, OVERRIDES DIRECTORY INFO
          NAMPNT              /POINTER TO FILENAME TEMP
          STOP                /FILE NOT FOUND OR HARDWARE ERROR
          ....
NAMPNT,   644555              /TEMP IN PACKED ASCII
          600000
```

If the file was to be loaded using the directory information, the fifth argument, 110000, would have to be changed to a 3777777. In the load operation, it is important that the starting track be zero since if it was non-zero, a block read operation is performed using that track as a track address.

The following code stores a program ABCDEF on disk 1. The area saved is 0-7577. If an old copy exists on disk, it is deleted. The starting address stored in the directory is indeterminate.

```
          JMS IOSIN
          ZERM @ XDEVDIR      /CLEAR CORE SWITCH
          ONEA                /SET FOR WRITE
          JMS @ IOSUPER
          1                   /DISK 1
          7600                /WORD COUNT
          0                   /IF ZERO, IOSUPER WILL FIND EMPTY
          0                   /STARTING ADDRESS
          NPOINT              /POINTER TO FILENAME
          STOP                /NO ROOM, DIRECTORY ERROR OR HARD ERROR
          ....
NPOINT,   414243              /ABC
          444546              /DEF
```

Using Paper Tape Devices

Whenever IOSUPER accesses a paper tape device, it clears its flag by either reading or punching a character. While this is desirable the first time the device is accessed, succeeding accesses to the device would find this feature less than desirable since it reads or punches a character and thus perhaps invalidates the tape. To bypass this feature, the following code should be executed before the second access to the device.

```
/PREVENT PUNCH AND TTY INITALIZATION
          MEMA (563          /JMP HANDLW+1 (562 RESTORES INITIALIZATION)
          ACCM @ HDEVW
/PREVENT READER FROM READING FIRST CHAR
          MEMA (371          /JMP CRDTTY+1 (110336 RESTORES INITALIZATION)
          ACCM @ DEVR
          ....
HDEVW,    6341
LDEVW,    6362
DEVR,     6364
```

Locking IOSUPER in Core

As mentioned previously, IOSUPER will exit to the user program with all core restored. Many programs would find this feature undesirable due to speed and timing considerations. The following code will prevent IOSUPER from swapping itself out of core at the end of a transfer.

```
          MEMA NOP
          ACCM @ PIN
          ....
PIN,      6234
NOP,      ACCA
```

The user program then must swap IOSUPER and appropriate core segments in and out as needed. It is a good programming practice to leave the core segment 3000-5777 in the swapping area and zero the core switch at 7764 (DEVDIR).

Additional Notes

The error flag at location 7704 (ERRFLG) is used both as an error flag by the hardware and software. The Keyboard Monitor checks this flag and if it is non-zero, it prints DISK READ ERROR. In order to prevent unnecessary panic from excessive DISK READ ERROR messages, this error flag should be set to zero whenever it is set to a -1 by an end of tape condition, file not found, no room etc.

Due to restrictions in the directory for specifying the word count of a file, the maximum file length is $50_{(8)}$ tracks long. Consult the factory for information concerning interfacing additional devices into IOSUPER and the INPUT/OUTPUT DECODER.

Appendix A

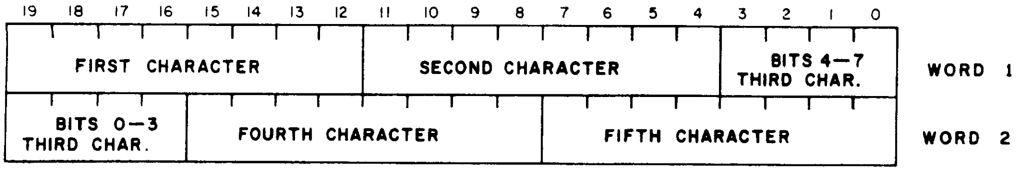
Core Information for DEMON/II

Routine	Length	Load Address	Track
Monitor Bootstrap	152	7600	0
Save Area	6000	not applicable	1-2
Directory	3000	not applicable	3
Keyboard Monitor	1600	6000	4
Binary Loader	152	7600	5
Monitor Head	160	7600	6
DIRFUN	600	7000	7
DIRLST	400	7200	10
INPUT/OUTPUT Decoder	1000	6000	11
IOSUPER	1000	6000	12

Appendix B

Character Packing Format

ASCII and binary charcters are stored on disk in a packed format. Five of these "paper tape" characters are stored in two disk words as show below:



Core image files are stored one 20 bit word per disk word.

Appendix C

Disk Assembler Mnemonics

Memory Reference

A+M
AMP
A-M
M-A
M+A
ACM
CAM
AND
MEM
MPO
MMO
MCP
MNG
ACC
APO
AMO
ACP
ANG
ZER
ONE
MON
MTO
JMP
JMS
CALL (=JMS)

Shift Instructions

LASH
RASH
LLSH
RLSH
RISH
VDSH

Test Instructions

SKIP
EXCT
ZAC
MOAC
POAC
ACØ
AC19
L

Miscellaneous Instructions

STOP
CLL
STL
TLAC
TACL

Display Instructions

TACXD
TACYD
INCXD
STATUS
INTENS

Multiply-Divide

MULT
DIVD
TACMQ
TMQAC
ZRAM
BITINV
ORMQAC

Input-Output Instructions

TTYRF
RDTTY
PRTTY
TTYPF
HSRF
RHSR
HSPF
PHSP
REDS
STDG
RDG
DWSK
ASRMP
RSWP
RDISK
WDISK
LTRACK
DSTAT

Appendix D

Listing of Disk MOVE Program

The attached listing contains examples of the use of the DCI, DIRFUN, and IOSUPER. Study them carefully before writing your own code.

/DISK MOVE PROGRAM

```
/DISK MOVE PROGRAM
*0
        0 2000066 IOSTRT,   JMS SAVE   /SAVE CORE
        1 2170000   ZERA
        2 3000025   JMS @ ZDISK   /READ CD IN
        3  100011          100011
        4     1000   1000  /WC
        5     6000   6000  /BUFFER
        6 3164562   ZERM @ ZDEVDIR
        7 3000543   JMS @ A6000   /ENTER CD
       10      443 ATABPNT,  TABPNT   /ADDRESS OF I/O TABLE
       11      513 AOPTPNT,  OPTPNT   /ADDRESS OF OPTION TABLE
       12        0   0  /NO ASSUMED EXTENSION
       13 2110010   MEMA ATABPNT
       14 2404531   ACCM ATEMP   /LET'S FIND NUMBER OF FILES
       15 2164532   ZERM NINPUT
       16 3122531 IINC,   MPOZ @ ATEMP
       17 2162000   ZERZ
       20       26   JMP INC10   /DONE
       21 2124532   MPOM NINPUT   /BUMP NUMBER OF FILES
       22  110003   MEMA (3
       23 2504531   A+MM ATEMP
       24       16   JMP IINC
       25     7612 ZDISK,   7612
       26 2124531 INC10,   MPOM ATEMP   /START OF OUTPUT FILES

       27 2000122   JMS FIRFLE   /READ IN FIRST BUFFER
       30 2000133   JMS OUTSET   /SET UP FOR OUTPUT
       31  110303   MEMA ("C
       32 2000104   JMS OPTEST
       33       42   JMP CORE   /CORE IMAGE FILE
       34  110302   MEMA ("B
       35 2000104   JMS OPTEST
       36     1106   JMP BIN  /CONVERT CORE IMAGE TO BINARY
       37 2000632   JMS FETMC   /JUST TRANSFER
       40 2000707   JMS PUTC
       41       37   JMP #-2

/CORE IMAGE FILE
       42 2702532 CORE,   MMOZ NINPUT
       43     1045   JMP TOOCOR   /MORE THAN 1
       44 3110567 COR100,   MEMA @ DBPNT   /GET A WORD
       45 3404553   ACCM @ OUTPNT   /STORE IT
       46 2124567   MPOM DBPNT
       47 2124553   MPOM OUTPNT
       50 2706277   MMOMZ IARG2
       51       44   JMP COR100   /GO AGAIN
       52 2102545   MEMZ DEVEND
       53       60   JMP COR200   /END OF FILE FLAG SET
       54 2000314   JMS OUTTRN   /SSTO FILE
       55 2000262   JMS IOTRN   /GET NEW ONE
       56 2000760   JMS OTSPNT   /SET UP OUTPUT POINTERS
       57       44   JMP COR100
```

```
---------
/DISK MOVE PROGRAM



      60 2110536 COR200,   MEMA 1ARG2A   /REMAINDER
      61 2404327   ACCM 0ARG2
      62 2330544   M-AA C3000
      63 2510523   A+MA TOTCNT  /MANIPULATE TOTAL COUNT
      64 2000314   JMS OUTTRN
      65     373   JMP CLS300  /CLOSE FILE

/SAVE 3000-7577
      66       0 SAVE, 0
      67 2030000   ONEA
      70 2000076   JMS DISTRN
      71 1000066   JMP @ SAVE
/RESTORE 3000-7577
      72       0 RESTORE,  0
      73 2170000   ZERA
      74 2000076   JMS DISTRN
      75 1000072   JMP @ RESTORE
/PERFORM SWAP OF SOME SORT
      76       0 DISTRN, 0
      77 3000025   JMS @ ZDISK
     100  100001         100001
     101    4600           4600    /WORD COUNT
     102    3000           3000    /BUFFER ADDRESS
     103 1000076   JMP @ DISTRN

/TEST FOR OPTIONS
/RETURN 1 IF IN TABLE, RETRUN 2 IF NOT
     104       0 OPTEST, 0
     105 2404066   ACCM SAVE  /SAVE CHAR
     106 2110011   MEMA AOPTPNT
     107 2404072   ACCM RESTORE
     110 3110072 OPTE10,  MEMA @ RESTORE  /GET CHAR
     111  425160   EXCT MOAC
     112     120   JMP OPTE20  /DONE WITH TABLE
     113 2462066   A-MZ SAVE  /COMPARE
     114 2162000   ZERZ
     115 1000104   JMP @ OPTEST  /FOUND MATCH
     116 2124072   MPOM RESTORE  /BUMP POINTER
     117     110   JMP OPTE10
     120 2124104 OPTE20,  MPOM OPTEST
     121 1000104   JMP @ OPTEST



/SET UP FIRST BUFFER
     122       0 FIRFLE, 0
     123 2000347   JMS IOFTCH  /DON'T BOTHER SAVING 6000-7577
     124 2110010   MEMA ATABPNT
     125 2404530   ACCM LSTADD
     126 2110532   MEMA NINPUT
     127 2404526   ACCM SNINPUT  /NUMBER OF FILES
     130 2000206   JMS DEVSET  /SET UP INPUT
     131 2000262   JMS IOTRN  /READ FIRST ONE IN
     132 1000122   JMP @ FIRFLE

```

```
/SET UP FOR OUTPUT
     133         0 OUTSET,   0
     134 2164527   ZERM POUTFG  /CLEAR PAPER TAPE OUTPUT FLAG
     135 3110531   MEMA @ ATEMP  /GET OUTPUT DEVICE
     136  405160   EXCT ZAC
     137    1036   JMP NOUT  /NONE THERE
     140 2404326   ACCM OARG1
     141  470005   A-MA (5
     142    5104   SKIP AC19
     143     203   JMP OUTS10  /SET PAPER TAPE FLAG
     144  510005   A+MA (5  /RESTORE
     145 2404157   ACCM OUTS20  /FOR LOOKUP
     146 3144556   MONM @ ZDISOLVE
     147 2110531   MEMA ATEMP
     150  510003   A+MA (3
     151 2404066   ACCM SAVE
     152 3110066   MEMA @ SAVE  /GET WORD COUNT
     153 3404536   ACCM @ ZOARG2
     154 3164562   ZERM @ ZDEVDIR  /CLEAR CORE SWITCH
     155 2000360   JMS DIRIN  /READ IN DIRFUN
     156 3000565   JMS @ ZDIRFUN  /GO TO IT
     157         0 OUTS20,  0  /DEVICE
     160         2   2  /SEARCH
     161       534   ZPNT  /ZERO FILE NAME
     162 2162000   ZERZ
     163      5220   STOP  /IMPOSSIBLE RETURN
     164 2000072   JMS RESTORE  /RETURN CORE
     165 3174566   ZERAM @ ZERRFLG  /CLEAR ERROR FLAG
     166 3110536   MEMA @ ZOARG2  /GET WORD COUTN
     167    5144          EXCT AC19
     170 2230000          ANGA              /TAKE ABSOLUTE VALUE IF MINUS
     171 2404524   ACCM EMPCNT
     172 3110561   MEMA @ ZOARG1  /STARTING TRACK
     173 2404525   ACCM CLSTRK
     174 2404330   ACCM OARG3
     175 3164562   ZERM @ ZDEVDIR
     176 2164523   ZERM TOTCNT   \
     177 2110544   MEMA C3000
     200 2404327   ACCM OARG2
     201 2000760 OUTS30,  JMS OTSPNT  /SET UP OUTPUT POINTERS
     202 1000133   JMP @ OUTSET
     203 2144527 OUTS10,  MONM POUTFG  /SET PAPER TAPE FLAG
     204 2144535   MONM F1RFLG
     205     177   JMP OUTS30-2

/SET UP FOR INPUT TRANSFER
     206         0 DEVSET, 0
     207 3110530   MEMA @ LSTADD  /DEVICE
     210  425160   EXCT MOAC
     211     370   JMP CLSFLE
     212 2404276   ACCM IARG1  /DEVICE
     213  470005   A-MA (5
     214    5104   SKIP AC19
     215     236   JMP DEVPT  /SET UP FOR PAPER TAPE DEVICE
```

/DISK MOVE PROGRAM

```
      216  110310          MEMA ("H
      217 2000104          JMS UPTEST
      220 2000576          JMS TRKCAL      /FILE > THAN 50 TRACKS
      221 2124530    MPOM LSTADD
      222 3110530    MEMA @ LSTADD
      223 2404300    ACCM IARG3  /STARTING TRACK
      224 2124530    MPOM LSTADD
      225 3110530    MEMA @ LSTADD  /WORD COUNT
      226 2404533    ACCM IARG2A
      227 2124530    MPOM LSTADD  /BUMP TO NEXT ENTRY
/SET UP RETURN ROUTINES FOR DISK
      230  111033    MEMA (HARDER-IOSTRT
      231 2404303    ACCM ERRARG
      232 2110542    MEMA CBUMP
      233 2404304    ACCM ERRARG+1
      234 2164545    ZERM DEVEND  /CLEAR END OF FILE FLAG
      235 1000206    JMP @ DEVSET
/SET UP FOR PAPER TAPE DEVICES
      236  110003 DEVPT,   MEMA (3
      237 2504530    A+MM LSTADD
      240  110364    MEMA (364
      241 3404540    ACCM @ Q6333  /MAKE SURE IT INTIALIZES
      242 3404541    ACCM @ Q6354
      243 2164545    ZERM DEVEND
      244 2110301    MEMA DPFST  /LARGE EMPTY SPACE
      245 2404533    ACCM IARG2A
      246  110253    MEMA (IOTT10-IOSTRT  /SET UP RETURN FOR PAPER TAPE
      247 2404303    ACCM ERRARG
      250  110256    MEMA (IOTT20-IOSTRT
      251 2404304    ACCM ERRARG+1
      252 1000206    JMP @ DEVSET
/ERROR RETURN FOR IOTRN PAPER TAPE DEVICE (OUT OF TAPE)
      253 3164566 IOTT10,   ZERM @ ZERRFLG
      254 2144545    MONM DEVEND  /SET END OF DEVICE FLAG
      255     305    JMP ERRARG+2
/NORMAL RETURN
      256  110371 IOTT20,   MEMA (371  /BYPASS INITALIZATION
      257 3404540    ACCM @ Q6333
      260 3404541    ACCM @ Q6354
      261     305    JMP ERRARG+2

/PERFORM INPUT TRANSFER
      262       0 IOTRN,   0
      263 2102545    MEMZ DEVEND
      264 2000206    JMS DEVSET  /ACCESS NEXT FILE
      265 2110533    MEMA IARG2A  /SET UP A WORD COUNT
      266 2470544    A-MA C3000
      267  405164    EXCT AC19 ZAC
      270     311    JMP IOT100  /LAST ONE
      271 2404533    ACCM IARG2A  /REMAINDER
      272 2110544          MEMA C3000
      273 2404277 IOT200,   ACCM IARG2  /STORE WORD COUNT FOR TRANSFER
      274 2170000    ZERA
      275 3000543    JMS @ A6000  /GO TO IOSUPER
```

```
     276       0 IARG1, 0  /DEVICE
     277       0 IARG2,  0  /WORD COUNT
     300       0 IARG3,  0  /STARTING TRACK
     301  100000 DPFST, 100000  /BUFFER ADDRESS
     302       0   0  /FILENAME DISREGARDED
     303       0 ERRARG,  0  /SET UP ROUTINES PUT APPOPRATE CODE HERE
     304 2124300   MPOM IARG3  /BUMP TRACK ADDRESS
     305 2110301   MEMA DPFST  /START OF BUFFER
     306 2404567   ACCM DBPNT
     307 2164570   ZERM BCPNT  /CHAR POINTER
     310 1000262   JMP @ IOTRN
     311 2144545 IOT100,  MONM DEVEND
     312 2510544   A+MA C3000  /RESTOR VALUE
     313     273   JMP IOT200

/PERFORM OUTPUT TRANSFER
     314       0 OUTTRN, 0
     315 2102527   MEMZ POUTPG  /CHECK FOR PAPER TAPE OUTPUT
     316     336   JMP OUTPT
     317 2110544   MEMA C3000
     320 2514523   A+MMA TOTCNT  /UPDATE TOATL COUNT
     321 2330524   M-AA EMPCNT  /HAVE WE OVERFLOWED
     322    5144   EXCT AC19
     323    1042   JMP NOROOM
     324 2030000 OUTT10,  ONEA
     325 3000543   JMS @ A6000  /GO TO IOSUPER
     326       0 OARG1, 0  /DEVICE
     327       0 OARG2, 0  /WORD COUNT
     330       0 OARG3,  0  /STARTING TRACK
     331  103000 OARG4, 103000  /BUFFER ADDRESS
     332     534   ZFNT  /ZERO FILE NAME FOR BLOCK TRASNFERS
     333    5220   STOP  /CAN'T HAVE ERROR ON OUTPUT
     334 2124330   MPOM OARG3  /BUMP STARTING TRACK
     335 1000314   JMP @ OUTTRN
     336 2122535 OUTPT, MPOZ FIRFLG  /IS THIS FIRST TIME
     337     345   JMP OUTT20  /NO
     340 2164535   ZERM FIRFLG  /ZERO FLAG
     341  110562   MEMA (562  /JMP HANDLW
     342 3404550   ACCM @ JPHLW
     343 3404551   ACCM @ JPLLW
     344     324   JMP OUTT10
     345  110563 OUTT20,  MEMA (563  /JMP HANDLW+1
     346     342   JMP OUTT20-3

/CALL IN IOSUPER
     347       0 IOFTCH,. 0
     350 2170000   ZERA
     351 3000025   JMS @ ZDISK
     352  100012          100012
     353    1000   1000
     354    6000   6000
     355 2110546   MEMA NOP  /LOCK IN CORE
     356 3404547   ACCM @ PIN
     357 1000347   JMP @ IOFTCH
```

```


/CALL IN DIRFUN
      360        0 DIRIN, 0
      361 2000066         JMS SAVE
      362 2170000   ZERA
      363 3000025   JMS @ ZDISK
      364  100007         100007
      365      600   600
      366     7000   7000
      367 1000360   JMP @ DIRIN

/CLOSE OUTPUT FILE
      370 2102527 CLSFLE,   MEMZ POUTFG   /DON'T CLOSE PAPER TAPE
      371      422   JMP CLSPT   /FINISH OUT WHATEVER
      372 2000433   JMS FINBUF   /FILL BUFFER WITH ZEROES
      373 2110525 CLS300,   MEMA CLSTRK   /CLOSE FILE
      374 3404561   ACCM @ ZOARG1
      375 2110523   MEMA TOTCNT   /TOTAL NUMBER OF WORDS
      376 3404536   ACCM @ ZOARG2
      377 2110301   MEMA DPFST   /BUFFER ADDRESS
      400 3404563   ACCM @ ZOARG3
      401 2110537   MEMA Y7600
      402 3404564   ACCM @ ZSYSTRT
      403 3110531   MEMA @ ATEMP   /DEVICE
      404 2404412   ACCM CLS100
      405 2134531   MPOMA ATEMP   /ADDESS OF FILENAME
      406 2404414   ACCM CLS200
      407 3164562   ZERM @ ZDEVDIR
      410 2000360   JMS DIRIN
      411 3000565   JMS @ ZDIRFUN   /DO IT
      412        0 CLS100, 0   /DEVICE
      413        1   1   /CLOSE
      414        0 CLS200, 0   /POINTER TO FILENAME
      415     1042   JMP NOROOM
      416 2000072   JMS RESTORE   /RESTORE CORE
      417  110003   MEMA (3
      420 2504531   A+MM ATEMP   /FOR NEXT DEVICE
      421        0   JMP IOSTRT   ? ?
      422 2170000 CLSPT,  ZERA   /PUT A ZERO
      423 2000707   JMS PUTC
      424 2110552   MEMA OUTCNT
      425 2330544   M-AA C3000   /HOW MANY ARE THERE
      426 2406327   ACCMZ OARG2
      427 2000314   JMS OUTTRN   /OUTPUT LAST BUFFER
      430  110004   MEMA (4
      431 2504531   A+MM ATEMP   /BUMP TO NEXT ENTRY
      432        0   JMP IOSTRT   ? ?

/FINSIH BUFFER
      433        0 FINBUF, 0
      434 2070552   MNGA OUTCNT
      435 2510544   A+MA C3000   /# OF LOCATIONS LEFT
      436  405160   EXCT ZAC
```

/DISK MOVE PROGRAM

```
     437 1000433   JMP @ FINBUF
     440 2170000   ZERA
     441 2000707   JMS PUTC
     442     434   JMP FINBUF+1



/CONSTANTS
     443         0 TABPNT,   BLOCK 50
     513         0 OPTPNT,   BLOCK 10
     523         0 TOTCNT,   0
     524         0 EMPCNT,   0
     525         0 CLSTRK,   0
     526         0 SNINPUT,   0
     527         0 POUTFG,   0
     530         0 LSTADD,   0
     531         0 ATEMP,   0
     532         0 NINPUT,   0
     533         0 IARG2A,   0
     534         0 ZPNT,   0
     535         0 FIRFLG,   0
     536      7771 ZOARG2,   7771
     537      7600 Y7600,   7600
     540      6333 Q6333,   6333
     541      6354 Q6354,   6354
     542   2124300 CBUMP,   MPOM IARG3
     543      6000 A6000,   6000
     544      3000 C3000,   3000
     545         0 DEVEND,   0
     546      5020 NOP,   RASH
     547      6234 PIN,   6234
     550      6341 JPHLW,   6341
     551      6362 JPLLW,   6362
     552         0 OUTCNT,   0
     553         0 OUTPNT,   0
     554       714 CROUTO,   CRLSTO
     555         0 BCPNTO,   0
     556      7751 ZDISOLVE, 7751
     557         0 FLAG7,   0
     560         0 CKSM,   0
     561      7770 ZOARG1,7770
     562      7764 ZDEVDIR,  7764
     563      7772 ZOARG3,  7772
     564      7760 ZSYSTRT,  7760
     565      7000 ZDIRFUN,  7000
     566      7704 ZERRFLG,  7704
     567         0 DBPNT,  0
     570         0 BCPNT,  0
     571         0 PUT300,   0
     572         0 FETADD,  0
     573      7556 TRLOOK,  7556
     574      7143 TRCALC,  7143
     575      7136 K7136,   7136
/FIND WC OF > THAN 50 TRACKS
```

/DISK MOVE PROGRAM

```
     576       0 TRKCAL, 0
     577 2000360         JMS DIRIN        /CALL IN DIRFUN
     600 2110276         MEMA 1ARG1
     601 2404604         ACCM TRK100
     602 3144556         MONM @ ZDISOLVE /DO DUMMY LOOKUP
     603 3000565         JMS @ ZDIRFUN
     604       0 TRK100, 0         /DEVICE
     605       2         2         /SEARCH
     606     534         ZPNT
     607 2162000         ZERZ     /ERROR RETURN ALWAYS TAKEN
     610    5220         STOP     /IMMPOSIBLE ERROR
     611 3174566         ZERAM @ ZERRFLG
     612 2130530         MPOA LSTADD
     613 2404604         ACCM TRK100      /GET STARTING TRACK
     614 3110604         MEMA @ TRK100
     615 3404575         ACCM @ K7136
     616 3000573         JMS @ TRLOOK     /FIND TRACK'S ENTRY IN DIRECTORY
     617 3000574         JMS @ TRCALC     /CALCULATE NUMBER OF TRACKS
     620 2124604         MPOM TRK100
     621    4354         TACMQ            /CONVERT TO WORDS
     622  505320         MULT
     623    3000         3000
     624  405120         SKIP ZAC
     625    1042         JMP NOROOM       />THAN 20 BITS
     626    4343         TMQAC
     627 3404604         ACCM @ TRK100    /REALISTIC WORD COUNT
     630 2000072         JMS RESTORE
     631 1000576         JMP @ TRKCAL
/FETCH CHAR ROUTINE
     632       0 FETMC, 0
     633 2110570   MEMA BCPNT   /CHAR. ROUTINE POINTER
     634 2510642   A+MA CROUT
     635 2404572 FET100,  ACCM FETADD  /CALCULATE ADDRESS OF ROUTINE
     636 3110572   MEMA @ FETADD  /GET ADDRESS OF ROUTINE
     637 2404572   ACCM FETADD
     640 2110571   MEMA PUT300
     641 1000572   JMP @ FETADD
     642     643 CROUT, CRLST
     643     650 CRLST, CHAR0
     644     653   CHAR1
     645     656   CHAR2
     646     670   CHAR3
     647     673   CHAR4
     650 3110567 CHAR0,  MEMA @ DBPNT  /GET WORD FROM DISK BUFFER
     651    5050   LLSH 10
     652     676   JMP FCHEK  /SEE IF FORM FEED
     653 3110567 CHAR1,  MEMA @ DBPNT
     654  405024   RISH 4
     655     676   JMP FCHEK
     656 3110567 CHAR2,  MEMA @ DBPNT
     657   10017   ANDA (17  /MASK FIRST PART
     660    5004   LASH 4
     661 2404572   ACCM FETADD  /TEMP STORAGE
     662 2124567   MPOM DBPNT  /ACCESS NEXT BUFFER WORD
```

```
  663 3110567    MEMA @ DBPNT
  664    5044    LLSH 4
  665   10017    ANDA (17
  666 2510572    A+MA FETADD
  667     676    JMP FCHEK  /CHEK FOR FORM FEED
  670 3110567 CHAR3,  MEMA @ DBPNT
  671  405030    RISH 10
  672     676    JMP FCHEK
  673 3110567 CHAR4,  MEMA @ DBPNT
  674 2124567    MPOM DBPNT  /ACCESS NEXT WORD
  675 2144570    MONM BCPNT
  676   10377 FCHEK, ANDA (377
  677 2404572    ACCM FETADD
  700 2134570 NOFORM,  MPOMA BCPNT
  701 2110567    MEMA DBPNT  /DONE?
  702 2462331    A-MZ OARG4  /DONE WITH BUFFER?
  703 2162000    ZERZ
  704 2000262    JMS IOTRN  /GET NEW ONE
  705 2110572    MEMA FETADD  /RETURN WITH CHAR IN AC
  706 1000632    JMP @ FETMC


/PUT CHARACTER INTO DISK BUFFER
  707       0 PUTC, 0
  710 2404571    ACCM PUT300  /SCR
  711 2110555    MEMA BCPNTO
  712 2510554    A+MA CROUNO
  713     635    JMP FET100  /LET FETMC DO REST OF WORK
  714     721 CRLSTO,  OCHAR0  /CHAR PACKING ROUTINES
  715     724    OCHAR1
  716     727    OCHAR2
  717     740    OCHAR3
  720     743    OCHAR4
  721    5070 OCHAR0,  RLSH 10  /FIRST CHAR
  722 3404553    ACCM @ OUTPNT
  723     746    JMP PUT200
  724    5004 OCHAR1,  LASH 4  /SECOND CHAR
  725 3504553    A+MM @ OUTPNT
  726     746    JMP PUT200
  727 2404572 OCHAR2,  ACCM FETADD  /THIRD
  730  405024    RISH 4
  731 3504553    A+MM @ OUTPNT
  732 2000751    JMS PUT100  /GET NEXT WORD
  733 2110572    MEMA FETADD
  734   10017    ANDA (17
  735    5064    RLSH 4
  736 3404553    ACCM @ OUTPNT  /STORE
  737     746    JMP PUT200
  740    5010 OCHAR3,  LASH 10  /FOURTH
  741 3504553    A+MM @ OUTPNT
  742     746    JMP PUT200
  743 3504553 OCHAR4,  A+MM @ OUTPNT  /FIFTH
  744 2000751    JMS PUT100  /ACCESS NEXT WORD
  745 2144555    MONM BCPNTO
```

/DISK MOVE PROGRAM

```
 746 2124555 PUT200,   MPOM BCPNTO
 747 2110571    MEMA PUT300
 750 1000707    JMP @ PUTC   /EXIT

/ACCESS NEXT BUFFER WORD
 751       0 PUT100,  0
 752 2124553    MPOM OUTPNT   /BUMP ADDRESS
 753 2706552    MMOMZ OUTCNT   /3000 WORDS YET?
 754 1000751    JMP @ PUT100
 755 2000314    JMS OUTTRN   /YES OUTPUT
 756 2000760    JMS OTSPNT   /RESET POINTERS
 757 1000751    JMP @ PUT100

/SET UP OUTPUT BUFFER POINTERS
 760       0 OTSPNT,  0
 761 2110331    MEMA OARG4
 762 2404553    ACCM OUTPNT   /ADDRESS
 763 2164555    ZERM BCPNTO
 764 2110544    MEMA C3000
 765 2404552    ACCM OUTCNT   /# OF WORDS
 766 1000760    JMP @ OTSPNT
/UNPCK PACKED STRING
 767       0 UNPCK,   0
 770 3110767    MEMA @ UNPCK   /ADDRESS OF STRING
 771 2404066    ACCM SAVE
 772 2124767    MPOM UNPCK
 773 3110066 UN1,   MEMA @ SAVE
 774 2000777    JMS UTYPE
 775 2124066    MPOM SAVE
 776     773    JMP UN1
 777       0 UTYPE,   0
1000 2404072    ACCM RESTORE
1001    5034    RASH 14
1002 2001007    JMS UNTYPE
1003    5026    RASH 6
1004 2001007    JMS UNTYPE
1005 2001007    JMS UNTYPE
1006 1000777    JMP @ UTYPE
1007       0 UNTYPE,   0
1010   10077    ANDA (77
1011  462077    A-MZ (77
1012 2162000    ZERZ
1013 1000767    JMP @ UNPCK   /FOUND TERMINATOR
1014  510240    A+MA (240
1015 2001020    JMS TYPE
1016 2110072    MEMA RESTORE
1017 1001007    JMP @ UNTYPE

/PRINT A CHAR
1020       0 TYPE,  0
1021    6444    TTYPF
1022    1021    JMP #-1
1023    4443    PRTTY
1024 1001020    JMP @ TYPE
```

```

/CR-LF
       1025            0 CRLF,  0
       1026   110215       MEMA (215
       1027  2001020       JMS TYPE
       1030   110212       MEMA (212
       1031  2001020       JMS TYPE
       1032  1001025       JMP @ CRLF

/ERROR MESSAGES
       1033  2000767 HARDER,   JMS UNPCK   /HARDWARE ERROR
       1034     1064       MHARD
       1035  1000537       JMP @ Y7600   /RETURN TO MONITOR
       1036  2001025 NOUT,   JMS CRLF
       1037  2000767       JMS UNPCK   /NO OUTPUT FILE
       1040     1072       MNOUT
       1041        0       JMP IOSTRT
       1042  2000767 NOROOM,   JMS UNPCK   /NO ROOM ON DISK
       1043     1100       MNOROOM
       1044  1000537       JMP @ Y7600
       1045  2001025 TOOCOR,   JMS CRLF
       1046  2000767       JMS UNPCK   /MORE THAN 1 CORE IMAGE FILE
       1047     1051       MTOOCOR
       1050        0       JMP IOSTRT
       1051   555762 MTOOCOR,   TEXT %MOR
       1052   450064   E T
       1053   504156   HAN
       1054     5756    ON
       1055   450043   E C
       1056   576245   ORE
       1057     5155    IM
       1060   414745   AGE
       1061     4651    FI
       1062   544501   LE!
       1063   770000   %
       1064   504162 MHARD,   TEXT %HAR
       1065   446741   DWA
       1066   624500   RE
       1067   456262   ERR
       1070   576201   OR!
       1071   770000   %
       1072   565700 MNOUT,   TEXT %NO
       1073   576564   OUT
       1074   606564   PUT
       1075     4651    FI
       1076   544537   LE?
       1077   770000   %
       1100   565700 MNOROOM,   TEXT %NO
       1101   625757   ROO
       1102   550057   M O
       1103   560044   N D
       1104   516353   ISK
       1105    17700   !%
/OUTPUT IN BINARY FORMAT
```

```
/DISK MOVE PROGRAM


1106 2702532 BIN,   MMOZ NINPUT
1107    1045   JMP TODCOR  /ONLY ONE CORE IMAGE FILE ALLOWED
1110 3164562   ZERM @ ZDEVDIR
1111 2000360   JMS DIRIN  /WE HAVE TO LOOK UP BUFFER ADDRESS
1112 2110010   MEMA AINBPNT
1113 2405201   ACCM LEADER
1114 3111201   MEMA @ LEADER  /GET DEVICE
1115 2405122   ACCM FAK100
1116  470005   A-MA (5  /CHECK FOR ILLEGAL INPUT
1117    5104   SKIP AC19
1120    1171   JMP ILLIN  /CAN'T READ CORE IMAGE IN FROM PAPER TAPE
1121 3000565   JMS @ ZDIRFUN
1122       0 FAK100,   0  /DEVICE
1123       2   2  /DUMMY SEARCH
1124     534   ZPNT  /ZERO FILE NAME
1125 2410000   ACCA  /PROBABLY RETURNS HERE
1126 3164566   ZERM @ ZERRFLG  /CLEAR ERROR FLGAG
1127 2125201   MPOM LEADER  /GET STARTING TRACK
1130 3111201   MEMA @ LEADER
1131 3405166   ACCM @ ZTRCK
1132 3001167   JMS @ ZTRLOOK  /FIND ADDRESS OF ST IN DIRECTORY
1133  110002   MEMA (2
1134 3325165   M-AM @ ZPOINT  /SET UP CORDEC
1135 3001170   JMS @ ZCORDEC  /DECODE DIRECTORY INFORMATION
1136 2001201   JMS LEADER  /PUNCH LEADER
1137 2001201   JMS LEADER
1140 2164560   ZERM CKSM  /ZERO CHECKSUM
1141 3110563   MEMA @ ZOARG3  /GET ORGIN ADDRESS
1142 2001213   JMS BPUN  /PUNCH IT
1143 3110567 BIN100,   MEMA @ DBPNT  /GET WORD
1144 2001213   JMS BPUN  /CONVERT TO BINARY
1145 2124567   MPOM DBPNT
1146 2706277   MMOMZ IARG2
1147    1143   JMP BIN100
1150 2102545   MEMZ DEVEND  /DONE?
1151    1154   JMP BIN200  /YES, PUNCH CHECKSUM AND TRAILER
1152 2000262   JMS IOTRN  /READ IN NEXT BUFFER
1153    1143   JMP BIN100
1154 2110560 BIN200,   MEMA CKSM
1155 2144557   MONM FLAG7  /PUNCH CHECKSUM
1156 2001213   JMS BPUN
1157 2001201   JMS LEADER  /PUNCH TRAILER
1160 2144557   MONM FLAG7
1161  110377   MEMA (377  /PUNCH RUBOYUT
1162 2001213   JMS BPUN
1163 2001201   JMS LEADER
1164     370   JMP CLSFLE  /CLSOE FILE

1165    7701 ZPOINT,   7701
1166    7136 ZTRCK,   7136  /LOCATION IN DIRFUN
1167    7556 ZTRLOOK,   7556
1170    7101 ZCORDEC,   7101
1171 2000767 ILLIN,   JMS UNPCK  /ILLEGAL PAPER TAPE INPUT
1172    1174      MILLIN
```

```
    1173          0    JMP IOSIRT
    1174  515454 FILLIN,   TEXT FILL
    1175  404741   EGA
    1176  540651   L 1
    1177  566065   NFU
    1200  642177   T1%
/"PUNCH" LEADER
    1201          0 LEADER,   0
    1202  110150    MEMA (150
    1203 2404066    ACCM SAVE
    1204 2144557    MONM FLAG7
    1205 2170000 LEA100,   ZERA
    1206 2001213    JMS BPUN
    1207 2706066    MMOMZ SAVE
    1210      1205    JMP LEA100
    1211 2164557    ZERM FLAG7
    1212 1001201    JMP @ LEADER

/BINARY PUNCH 20 BIT WORD
    1213          0 BPUN,   0
    1214 2404347    ACCM IOFTCH   /SCR
    1215 2504560    A+MM CKSM   /ADD TO CHECKSUM
    1216  405036    RISH 16
    1217 2001224    JMS HBINP   /PUNCH FIRST FRAME
    1220  405027    RISH 7
    1221 2001224    JMS HBINP
    1222 2001224    JMS HBINP
    1223 1001213    JMP @ BPUN
    1224          0 HBINP,   0
    1225   10177    ANDA (177   /MASK OFF 200 CODE
    1226 2102557    MEMZ FLAG7
    1227  510200    A+MA (200   /ADD 200 CODE
    1230 2000707    JMS PUTC
    1231 2110347    MEMA IOFTCH
    1232 1001224    JMP @ HBINP
```

[illegible] MOVE PROGRAM

```
A6000       543     ADFTPN       11     ATABPN       10     ATEMP       531
BCFNT       570     BCFNT0      505     BIN        1106     BIN100     1143
BIN200     1154     BPUN       1213     C3000       544     CBUMP       542
CHAR0       650     CHAR1       653     CHAR2       656     CHAR3       670
CHAR4       673     CKSM        560     CLS100      412     CLS200      414
CLS300      423     CLSFLE      370     CLSPT       422     CLSTRK      525
COR100       44     COR200       60     CORE         42     CRLF       1025
CRLST       643     CRLST0      714     CROUT       642     CROUT0      554
DBPNT       567     DEVEND      545     DEVPT       236     DEVSET      206
DIRIN       360     DISTRN       76     DFFST       301     EMPCNT      524
ERRARG      303     FAK100     1122     FCHEK       676     FET100      635
FETADD      502     FETINC      632     FINBUF      433     FIRFLE      122
FIRFLG      535     FLAG7       507     HARDER     1033     HBINP      1224
IARG1       274     IARG2       277     IARG2A      533     IARG3       300
IINC         16     ILLIN      1171     INC10        26     IOFTCH      347
IOSTRT        0     IOT100      311     IOT200      273     IOTRN       262
IOTT10      253     IOTT20      256     JPHLW       550     JPLLW       551
K7136       575     LEA100     1207     LEADER     1201     LSTADD      530
MHARD      1064     MILLIN     1174     MNOROO     1100     MNOUT      1072
MTOOCO     1051     NINPUT      532     NOFORM      700     NOP         546
NOROOM     1042     NOUT       1056     OARG1       326     OARG2       327
OARG3       330     OARG4       331     OCHAR0      721     OCHAR1      724
OCHAR2      727     OCHAR3      740     OCHAR4      743     OPTE10      110
OPTE20      120     OPTEST      104     OPTPNT      513     OTSPNT      760
OUTCNT      552     OUTPNT      553     OUTPT       336     OUTS10      203
OUTS20      157     OUTS30      201     OUTSET      133     OUTT10      324
OUTT20      345     OUTTRN      314     PIN         547     POUTFG      527
PUT100      751     PUT200      746     PUT300      571     PUTC        707
Q6333       540     Q6334       541     RESTOR       72     SAVE         66
SNINPU      526     TABPNT      443     TOOCOR     1045     TOTCNT      523
TRCALC      574     TRK100      604     TRKCAL      576     TRLOOK      573
TYPE       1020     UN1         773     UNPCK       767     UNTYPE     1007
UTYPE       777     Y7600       537     ZCORDE     1170     ZDEVDI      562
ZDIRFU      565     ZDISK        25     ZDISOL      556     ZERRFL      566
ZOARG1      561     ZOARG2      536     ZOARG3      563     ZPNT        534
ZPOINT     1165     ZSYSTR      564     ZTRCK      1166     ZTRLOO     1167
```

```
 684  --------
/DISK MOVE PROGRAM


A6000      10   215   241 # 350
AOPTPN  #  12    84
ATABPN  #  11    15   101   599
ATEMP      15    18    23    25   111   122   294   296   307   316 # 340
BCPNT     225 # 371   408   442   446
BCPNTO  # 360   458   486   487   504
BIN        34 # 594
BIN100  # 623   628   632
BIN200    629 # 632
BPUN      622   624   635   639   661 # 667   676
C3000      57   146   209   213   228   236   312   322 # 35.   505
CBUMP     176 # 349
CHAR0     416 # 421
CHAR1     417 # 424
CHAR2     418 # 427
CHAR3     419 # 437
CHAR4     420 # 440
CKSM    # 363   620   633   669
CLS100    296 # 301
CLS200    298 # 303
CLS300     59 # 286
CLSFLE    158 # 283   640
CLSPT     284 # 309
CLSTRK    142   286 # 336
COR100  #  42    47    54
COR200     49 #  54
CORE       31 #  40
CRLF    # 542   548   554   561
CRLST     415 # 416
CRLSTO    360 # 460
CROUT     409 # 415
CROUTO  # 359   459
DBPNT      42    45   225 # 370   421   425   428   431   433   438   441
          441   446   623   626
DEVEND     49   177   186   195   206   228 # 352   628
DEVPT     161 # 180
DEVSET    104 # 154   179   193   206
DIRIN     127 # 273   281   300   379   597
DISTRN     65    70 #  72    78
DPFST     186 # 219   223   290
EMPCNT    140   237 # 335
ERRARG    175   176   190   191   196   201 # 221
FAK100    602 # 606
FCHEK     423   427   436   440 # 443
FET100  # 410   459
FETADD  # 373   410   411   412   415   430   435   444   450   471   476
FETMC      35 # 407   452
FINBUF    285 # 320   325   327
FIRFLE     27 #  98   107
FIRFLG    151   250   252 # 344
FLAG7   # 362   633   637   659   664   679
HARDER    173 # 550
HBINP     671   674   675 # 676   683
```

ADISK MOVE PROGRAM

```
IARG1      158  # 216    381
IARG2       47    213  # 217    627
IARG2A      54    171    188    207    211  # 342
IARG3      167  # 218    222    350
IINC     #  17     24
ILLIN      604  # 646
INC10       19  #  25
IOFTCH      97  # 261    270    668    682
IOSTRT   #   3    174    188    191    209    318    557    564    649
IOT100     210  # 227
IOT200   # 213    230
IOTRN       51    105  # 204    227    449    630
IOTT10     198  # 194
IOTT20     190  # 198
JPHLW      255  # 355
JPLLW      256  # 356
K7136    # 376    394
LEA100   # 659    663
LEADER     600    600    611    613    618    620    635    640  # 655    665
LSTADD     102    155    166    167    169    169    171    182  # 339    391
MHARD      552  # 575
MILLIN     648  # 649
MNOROO     559  # 587
MNOUT      556  # 581
MTOOCO     563  # 564
NINPUT      17     20     41    103  # 341    595
NOFORM   # 445
NOP        267  # 353
NOROOM     240    305    401  # 557
NOUT       113  # 553
OARG1      115  # 242
OARG2       56    147  # 243    314
OARG3      143  # 244    248
OARG4    # 245    447    502
OCHAR0     460  # 465
OCHAR1     462  # 468
OCHAR2     463  # 471
OCHAR3     464  # 480
OCHAR4     465  # 483
OPTE10   #  85     93
OPTE20      87  #  93
OPTEST      31     34  #  81     90     94     95    164
OPTFNT      12  # 333
OTSPNT      52    147    496  # 500    507
OUTCNT     312    322  # 357    493    505
OUTPNT      43     46  # 358    467    470    474    478    482    483    492    502
OUTPT      235  # 250
OUTS10     117  # 149
OUTS20     119  # 129
OUTS30   # 147    151
OUTSET      28  # 109    149
OUTT10   # 240    257
OUTT20     251  # 257    258
OUTTRN      50     59  # 232    250    314    495
```

```
PIN       269 # 304
POUTFG    110   149   234   283 # 338
PUT100    174   181 # 491   495   498
PUT200    468   471   480   485 # 486
PUT300 #  372   414   406   408
PUTC       37   311   327 # 405   488   681
Q6333     183   200 # 347
Q6354     185   201 # 348
RESTOR #   67    71    85    85    91   134   305   405   518   532
SAVE        3 #  62    66    82    88   124   124   275   511   513   515
          658   662
SNINPU    103 # 337
TABPNT     11 # 332
TOOCOR     41 # 560   590
TOTCNT     57   145   236   266 # 334
TRCALC #  375   393
TRK100    382 # 384   391   393   397   403
TRKCAL    164 # 378   406
TRLOOK #  374   394
TYPE      531 # 535   540   545   547
UN1    #  512   516
UNPCK  #  508   509   512   528   550   554   557   561   646
UNTYPE    520   522   523 # 524   533
UTYPE     514 # 516   524
Y7600     293 # 346   552   560
ZCORDE    617 # 645
ZDEVDI     10   126   144   299 # 365   597
ZDIRFU    128   300 # 368   384   606
ZDISK       5 #  24    74   264   277
ZDISOL    121 # 361   382
ZERRFL    135   195 # 369   390   610
ZOARG1    140   288 # 364
ZOARG2    126   136   290 # 345
ZOARG3    292 # 366   621
ZPNT      131   246 # 343   387   608
ZPOINT    616 # 642
ZSYSTR    294 # 367
ZTRCK     614 # 643
ZTRLOO    614 # 644
```

SALES OFFICES

CT, MA, ME, NH, RI, VT, Upper NY State
Richard Monaco
Nicolet Instrument Corporation
2120 Commonwealth Avenue
Auburndale, MA 02166
(617) 969-7420
TWX: 710-335-1954

Eastern U.S. NMR Specialist
James Lappegaard
Nicolet Instrument Corporation
245 Livingston Street
Northvale, NJ 07647
(201) 767-7100
TWX: 710-991-9619

NY City Area, NJ, East PA
Bernard Conti
Nicolet Instrument Corporation
P.O. Box 36
Old Bethpage, NY 11804
(516) 249-6360

Washington DC, MD, VA, WV, DE
Brent Vitek
10750 Columbia Pike, Suite 205
Silver Spring, MD 20901
(301) 681-5800

Midwest Regional Manager
Frank Contratto
Nicolet Instrument Corporation
500 East Higgins Road
Elk Grove, IL 60007
(312) 956-0404
TWX: 910-222-5999

IA, MN, KS, ND, SD, NE, WI, IL, MO
Richard Bohn
Nicolet Instrument Corporation
500 East Higgins Road
Elk Grove, IL 60007
(312) 956-0404
TWX: 910-222-5999

OH, MI, IN, KY, West PA
Richard Hulburt
Nicolet Instrument Corporation
3570 Warrensville Center Road
Shaker Heights, OH 44122
(216) 921-6600

TX, LA, OK, AR
Jerry Meyer
Nicolet Instrument Corporation
701 - 15th Street
Plano, TX 75074
(214) 424-8611

NC, SC, GA, AL, TN, FL, MS
Dr. H. B. Evans
Nicolet Instrument Corporation
3796 North Decatur Road
Suite B-12
Decatur, GA 30033
(404) 292-2483

CO, NM, AZ, UT, MT, ID, WY
Bruce Lent
Nicolet Instrument Corporation
6023 South Lamar Drive
Littleton, CO 80123
(303) 798-3561

CA, NV, OR, WA, HI
Robert Olsen
Nicolet Instrument Corporation
145 East Dana Street
Mountain View, CA 94041
(415) 969-1258

CANADA
Allan Crawford Associates
3829 - 12th Street, N.E.
Calgary, Alberta T2E 6M5
(403) 276-9658

Allan Crawford Associates
1300 Marie Victorian Blvd, East
Lonqueuil, P.Q. J4G 1A2
(514) 670-1212
TWX: 610-422-3875

Allan Crawford Associates
6427 Northam Drive
Mississauga, Ontario, L4B 1J5
(416) 678-1500
TWX: 610-492-2119

Allan Crawford Associates
1299 Richmond Road
Ottawa, Ontario K2B 7Y4
(613) 829-9651
TWX: 610-562-1670

Allan Crawford Associates
Suite 201, Townsend Place
800 Windmill Road
Burnside Industrial Park
Dartmouth, N.S. B3B 1L1
(902) 469-7865
TWX: 610-271-1978

Allan Crawford Associates
Suite 203, 116 East 3rd Street
North Vancouver, B.C. V7L 1E6
(604) 980-4831

EUROPEAN COUNTRIES
Dr. Peter Langner
Nicolet Instrument GmbH
Goerdeler Strasse 48
D-605 Offenbach am Main
West Germany
0611/852028
Telex: 841/4185411

JAPAN
Takeda Riken Industry Co., Ltd.
1-32-1, Asahi-cho, Nerima-ku
Tokyo 176, Japan
930-4111
Telex: 781/272/2140

AUSTRALIA
ELMEASCO Instruments Pty. Limited
7 Chard Road
Brookvale, N.S.W. 2100
Australia

NEW ZEALAND
ELMEASCO Instruments Pty. Limited
P.O. Box 30515
Lower Hutt
New Zealand