QM - NCS

OPERATIONS GUIDE

NANODATA COMPUTER CORPORATION

One Computer Park
Buffalo, New York 14203

TWX (710) 522-1180
(716) 845-6000

QM - NCS

OPERATIONS GUIDE

NANODATA COMPUTER CORPORATION
ONE COMPUTER PARK
BUFFALO, NEW YORK 14203

(716) 845-6000

Part Number #323-81000 Rev. 04

Date: 10/30/81

TABLE OF CONTENTS

LIST OF TABLES

SECTION 1: GENERAL

INTRODUCTION

The Nanodata Control System (NCS) is an elementary control system that operates on the NOVA emulator developed for the QM-1.

The NOVA emulator supports a card reader, line printer, card punch, cartridge unit, sixteen magnetic tape units, a teletype, a paper tape punch, and a disk with a fixed pack and removable pack.

NCS provides the user with a basic set of utilities using the QM-1 associated peripherals.

Information files on various media (cards, magnetic tape, disk, etc.) generally represent either binary files or sourcecode files. Internal to the computer, sourcecode uses a 7-bit ASCII representation with zero parity (leftmost) bit in an 8-bit byte. Some utilities will recognize "packed sourcecode" where a byte with MSB=1 represents a (negative) count of extra repetitions for the last true character. (Eighty asterisks would pack into "*" followed by -79.)

This document consists of a set of procedures for using the utilities provided by NCS. Familiarity with the operation of the QM-1 system hardware is assumed. Procedures for bootstrap loading and operation of the hardware switches are contained in the "QM-1 Console Reference Manual".

NOTATION

The following notation is used in this document to formalize the format of the procedures.

1. Literals are shown in upper case and are entered or displayed as written.
2. Variables are shown in lower case. The appropriate name or value is substituted before entry or display.
3. Letters supplied by NCS utilities are underscored (e.g. V$\underline{\text{OLUME}}$ means that V was keyed in and OLUME was supplied by the executing utility).
4. Brackets [ ] enclose optional parameters.
5. Braces { } enclose a set of parameters from which one must be selected.
6. Ellipses . . . indicate repetition.
7. ⓝ indicates the CONTROL character "n" and is the equivalent of CTRL n as used in the Tektronix 4023 Computer Display Terminal users Manual. It is entered by pressing a letter key while CTRL is held down.
8. Δ is any QM-1 delimiter (hyphen, comma, blank, carriage return, period).

NCS CONTROL COMMANDS

The NOVA* emulator responds to the ⓢ entry. ⓢ followed by certain letters forms commands to the emulator. These commands correspond to the switches on the front panel of the NOVA computer.

Table 1 contains the set of commands recognized by the NOVA emulator and their functions. (H) after the function description indicates that the emulator must be halted when the code is entered.

Command	Function
ESC	Escape from ⓢ entry
ⓢ H	Halt the running NOVA emulator
ⓢ Snnnnnn$_8$	Set the NOVA switches
ⓢ L	Load NCS by executing the bootstrap from disk (H)
ⓢ R	Restart at NOVA location indicated by the switch setting (H)
ⓢ G	Go. Continue from HALT (H)
ⓢ Pnnnnnn$_8$	Enter NOVA program counter (H)
ⓢ Bn$_8$	Enter new CRT Baud rate where $n=\frac{9600}{BAUD}$ (H)
ⓢ $X\begin{Bmatrix}A\\S\end{Bmatrix}$	Select print train (H) S = standard subset (64 char), A = ASCII (128 char)
CR space	Enter value just keyed for ⓢ S, ⓢ P, ⓢ B, or ⓢ X commands. Delimiters are not expected after other commands.

Table 1. NCS Control Commands.

*NOVA is a registered trademark of Data General Corporation

Any codes not in Table 1 are displayed on the CRT and are ignored by the emulator. Also, any command entered at the wrong time is ignored and displayed on the CRT. (e.g., Ⓢ H when the NOVA is already halted.)

SETTING F-SWITCH Ø (on the QM-1 front panel) to one will cause the emulator to treat the HALT instruction as a no op.

Setting emulated switch 17[Ⓢ S4XXXXX] "unlocks" control store allowing MLD, QMLD, and USER to function properly.

NCS DISK SYSTEM

The NOVA emulator supports a disk system containing a single fixed platter with a single removable disk platter. The N9755 disk has been partitioned into 10 equal sized partitions, numbered 0 through 9. Since the NOVA emulator only supports two disk platters, only two of the partitions (called USER's) are recognized at any given time. The first partition, USER 0, has been permanently assigned the function of the fixed pack. Any one of the other USER's (1 through 9) can be dynamically assigned the function of the removable pack. USER assignment is accomplished through the use of the USER utility, described in Section 2.

When the NOVA emulator is loaded, the removable pack assignment is defaulted to USER 1 and must be reassigned if any other USER is desired.

Throughout the rest of this document, the fixed pack is referred to as the SYSTEM pack, and the removable pack is referred to as the USER pack.

Each USER is divided into cylinders and sectors. There are 400_{10} cylinders each containing 24_{10} sectors, giving a total of 9600_{10} (22600_8) sectors. Each sector contains 256_{10} 18-bit words. The NOVA normally manipulates this data as 256_{10} 16-bit words. Thus, utilities such as SAVE and RESTORE will lose the top two bits of each 18-bit word, if attempting to handle 18-bit-word files (such as MULTI microcode).

There is no communication between USER's 1 through 9. To move a file from one USER to another, it must be copied from the currently assigned USER to tape or to a USER 0 disk file and then copied to a reassigned USER.

Each USER contains a file directory which contains the entries defining up to 192 files. Each definition consists of a file name plus the file block parameters describing the file. Each file name must be unique to that USER.

File declarations are made with the FILES utility and specify the starting address on the disk in terms of cylinders and sectors, the number of sectors to be allocated, the type of file, the date of creation, and the program file parameters for core image program files.

File declarations allow overlapping files. This is useful for such things as partitioning a large work file or for defining a file that is 22600_8 sectors long, beginning at cylinder 0 and sector 0 to declare the entire USER as one file.

SECTION 2: UTILITIES

NCS provides a set of utilities for use in creating, transferring, manipulating, and executing programs and to perform memory and disk housekeeping functions.

This section contains a description of the function and a procedure for using each utility. The following table lists the utilities and gives a brief description of their functions.

Utility	Function
Assemblers	Assembles nanocode, microcode, and NOVA code.
BOOT	Displays/modifies a mapped nanostore binary file.
BUILD	Copies an NCS file from cartridge tape to disk file.
COMPARE	Compares two files, line by line.
COPY	Copies an NCS file from one device to another.
COPYD	Fast copy between disk files alone.
DISCART	Copies an NCS file from disk to cartridge tape.
EX	Uses input parameters from a file instead of keyins.
EXEC	Uses input parameters from a file called COMMANDS.
FILES	Adds, deletes, changes and reports status of NCS files.
FREE	Displays the graphic FREE on the CRT.
LD	Loads a NOVA binary program into memory from a disk file.
MAP	Maps the binary output of the nanoassembler.
NOVA MEMORY DUMP	Dumps a specified range of NOVA memory to the line printer.
PREP	Places a loadable file on cartridge tape from disk files.
QMLD	Loads mainstore, control store and nanostore from disk files.
RESTORE	Copies an NCS file from magnetic tape to disk file.
REWIND	Rewinds the magnetic tape or cartridge tape.

.S	Sets the NOVA program counter for program execution.
SAVE	Copies an NCS file from disk to magnetic tape.
SYSINIT	Initializes an NCS USER.
XC	Cross copy of 16-bit/wd files between any users, any disks.
UPDATE	Line-editor for sourcecode files.
USER	Assigns a specified USER as the NOVA removable pack.
ZAP	Clears NOVA user memory.
* and $	Manipulates NOVA programs.

Table 2: NCS Utilities

ASSEMBLING PROGRAMS

Both the nanoassembler and the microassembler can be invoked under NCS. A detailed discussion of the use of the nanoassembler is included in the HARDWARE LEVEL USERS MANUAL. The microassembler is described in the QM MICROASSEMBLER REFERENCE MANUAL. This procedure presupposes a knowledge of the assemblers.

Assembling Nanoprograms

The nanoassembler is invoked to assemble only nanoprograms.

```
                                                  ⎧binary-file Δ⎫
Format:    !NP[NL] Δ INPT =source-file Δ BIN=⎨Δ            ⎬
                                                  ⎩             ⎭
```

NL	listing suppression options
source-file	file containing a source nanoprogram
binary-file	file containing the binary output of the nanoassembly. If binary-file is omitted, the binary is not saved.

To generate a definition file only, the following format is used:

```
Format:    !LD,*NASPC Δ

           !.S=1

           !.S=2 Δ  INPT =source-file Δ BIN =definition-binary file Δ (with listing)
                                              or
           !.S=4 Δ  INPT =source-file Δ BIN =definition-binary file Δ (no listing)
```

Setting switch 15 (Ⓢ S100000) will cause suppression of all printer output.

Setting switch 14 (Ⓢ S40000) will cause the listing to be printed on the console instead of the line printer. No symbol table listing is generated.

Assembling Microprograms

The microassembler is invoked to assemble either NOVA assembly language programs or microprograms. Microprograms require a definition file. If no definition file is specified, the microassembler assumes the source is NOVA assembly language program.

Format: !MA[-options] Δ DEF = $\left\{\begin{matrix}\text{definition-file}\Delta \\ \Delta\end{matrix}\right\}$ INPT = source file[...source-fil

BIN = $\left\{\begin{matrix}\text{binary-file}\Delta \\ \Delta\end{matrix}\right\}$

-options	[L][X][N][A] listing options as described in the QM MICROASSEMBLER MANUAL.
definition-file	file containing the instruction set definitions and symbolic names - output from a nanoassembly. This file is required for microprograms. If none is specified, a NOVA source program is assumed.
source-file	file containing the source program to be input to the microassembler.
binary-file	file containing the binary output of the microassembler. When not preceded by a binary-file name, Δ indicates that the binary is not to be saved.

All output is directed to the line printer unless otherwise specified. When using the microassembler, the A option will direct output to the console. However, this option suppresses the symbol table listing. Setting switch 15 (ⓈS100000) will cause all listing, including errors, to be suppressed.

BOOT

BOOT can be used to modify words or bytes in a nanoprogram residing in a disk file.

Format:

```
!BOOT △
FILENAME = filename △
*command
```

filename specifies the file to be modified. Command is one of the four BOOT commands. The BOOT commands and their functions are listed in Table 3.

Error Messages:

If the nanoword is not specified as octal digits, ? is displayed and a new address is expected. If the nanobyte exceeds 19, the ADDRESS = prompt is redisplayed.

If filename is not valid, the FILENAME= prompt is redisplayed. Commands X, Y, T, R, P, Q, F, G, H, and I provide special functions not available on the QM-1 and may cause BOOT to simply hang. Entering Ⓢ R or Ⓢ L is the only recovery.

Command	Function
*C ADDRESS = nanoword-address Δ	Clears the specified nanoword to zero. This process clears all 18 bits of all 20 nanobytes of the nanoword. Upon completion, the prompt character * is redisplayed.
*M ADDRESS = $\begin{bmatrix} \text{nanoword}_8\text{, nanobyte}_{10} \\ \text{ƀ nanobyte}_{10} \\ \text{+ displacement} \\ \text{- displacement} \end{bmatrix}$ Δ	Modifies the specified nanobyte. The nanoword and nanobyte entries are separated by a comma. The nanoword is specified in octal, but the nanobyte is a decimal integer from 0 through 19. As each address is entered, the specified nanobyte is retrieved from the disk file and displayed on the screen in a binary format. Each bit or set of bits is identified in the display. A blank (ƀ) in place of "nanoword" uses the last entered nanoword address as the current address. Entering a ± displacement causes retrieva of the nanobyte displaced by the specifie number of bytes from the last entered nanobyte. To modify the nanobyte, space to the bit to be modified and enter a 0 or 1. Spacing leaves a bit as is. Entering a carriage return specifies that no more bits are to be modified. The modified nanobyte is then written back to the file. The modified nanobyte will automatically be written back after 18 bits have been modified or spaced over. The ADDRESS = prompt is redisplaye and a new nanobyte may be specified.

Example:

```
*M
ADDRESS = 1,0
E B N M D H F A
N R A 1 M L S L      ----KN----
T N N C S P H U

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

            or

ADDRESS = 1,13
X R L G T U R G         O
I I E E X N 3 S I AUX-1 U F-SEL-1
O O S S X C 1 H N       T
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Command	Function
*N FILENAME = filename Δ	specifies a new file to be modified.
*K	causes entry to PREP.
*E	End of Job. Control returns to NCS.
*ESC	cancels the current command and redisplay the prompt character.

Table 3: BOOT Commands

BUILD

BUILD copies a file that has been stored on cartridge tape with the DISCART utility to a file on the disk.

```
  Format:  !BUILD △

      CART TAPE UNIT NUMBER = unit number△

              TRACK NUMBER = track-number△
                                                                   * 
              START AT FILE = [ Ⓢ S100400] file-marks-to-be-skipped₈ { + }
                                                                   -
                                                                   △

                  LOAD FILE = filename

                DATE OF HDR = date

                 VOL NUMBER = volume

              START CYLINDER = cylinder

                START SECTOR = sector

                 FILE LENGTH = length

        HIT (RETURN) TO WRITE DISK CR
```

unit-number	cartridge tape unit number first unit = 0 second unit = 1
track-number	track number 0,1,2, or 3
Ⓢ S100400	switch setting to disable header comparison between the saved file and the file definition in the destination directory. The file is written at the address specified in the header.
** CAUTION **	misuse of this option can cause file overwriting.

file-marks-to-be-skipped$_8$	number of file marks to be skipped for positioning. There are two file marks after each file. To read multiple files, the starting file numbers would be 0, 2, 4 .... * skip past specified number of file marks - do not rewind before or after the read. After the file is read, the next starting file is requested. + rewind, skip past specified number of file marks - do not rewind after the read. After the file is read, the next starting file is requested. △ rewind and skip past the specified number of file marks. Rewind after the file is read and return to NCS. - return directly to NCS.
filename, date volume, cylinder, sector, length	file attributes obtained from header record of the DISCART's file.

BUILD will write to an uninitialized USER if the ⓢ S100400 option is used. Otherwise, the USER must be initialized with the SYSINIT utility or a disk error message will be displayed.

If the ⓢ S100400 option is not used, the header parameters are examined and the destination directory is searched for the filename. If the filename has not been defined, or if the length is too short, or if the file is write protected, BUILD will ask for an alternate filename. The alternate filename must have been defined with appropriate parameters. The parameter list for the alternate file is displayed followed by

HIT (RETURN) TO WRITE DISK

Error message;

DISK ERROR

Disk is not ready.

COMPARE

COMPARE is a utility which reads two sourcecode files to determine whether they are identical, or at what point they differ. If the two files contain identical sourcecode, COMPARE quits after reporting

MATCHED THRU E.O.F.

The files are compared on a line-by-line basis. If they differ at some pairs of lines, COMPARE first displays the contents of the two line buffers, each followed by its (implicit) line number, then pauses for user response after displaying the user-prompt,

MORE? (Y, N, 1, 2, ESC)

This user-prompt alone (with the following program pause) also can be elicited during a run, if the user creates an interrupt by pressing any key on the keyboard. Further operation of COMPARE then depends on the user's choice of response.

RESPONSES	MEANING
Y	Yes, begin comparisons on the next pair of lines, regardless of whether the current pair had matched or not.
N	No, terminate comparisons on this pair of files. If not in A-mode, perform as for ESC (below). If in A-mode, go on to the next pair of files (if any).
1	File#1 advances by one line; both line buffers redisplay.
2	File#2 advances by one line; both line buffers redisplay.
ESC	Escape unconditionally from COMPARE, back to NCS.
(other)	(repeats the same user-prompt, and again pauses)

INVOCATION

Invocation of the COMPARE utility requires user entry of six parameters and allows user entry of certain options. These are two modes of operation, depending on whether option "A" is chosen.

Non-A Mode Invocation

The comparisons will be on a single pair of files, whose names generally will be different. The required parameters come as two sets of three each.

Format: !COMPARE [-options not including A] Δ

1ST FILE:

DISK-UNIT # (0, 1): u_1 Δ

VOL.# (OCTAL, 0-max): v_1 Δ

FILENAME = $name_1$ Δ

2ND FILE:

DISK-UNIT # (0,1): u_2 Δ

VOL.#(OCTAL, 0-max): v_2 Δ

FILENAME = $name_2$ Δ

Parameters:	u_1, u_2 -	single digit, currently limited to 0 or 1.
	v_1, v_2 -	a one-digit or two-digit octal number; max is determined individually for the separate disk drives, and will be 11 for full NCS disks, or 04 for "split packs".
	$name_1$, $name_2$ -	a one- to ten-character name, existing in the directory of the specified Unit and Volume; no slash (/) precedes these names.

Options: L - Logging: the line printer will log all items which appear on the console CRT, after the "L".

U - Unpack: the disk-read subroutine will attempt to recognize and unpack, from either or both files, any packed sourcecode which may be present, before copying the sourcecode into the line buffer.

Q - Quantify: if the files succeed in being read all the way to their ends, then tallies will be displayed for the "used" space (non-null bytes tally), and for the "unused" space (null-bytes tally), separately for each file. These reports will be in terms of full sectors, plus number of bytes in the last (partial) sector, used/unused.

A-Mode Invocation

Under option "A", the comparisons may continue through many pairs of files, each pair having the same name, with names automatically obtained from some directory. The purpose is to validate whether all files of a whole USER volume match all files of another such volume by name and by content. In this mode, new options become available, while those of non-A mode are retained. The required parameters, in this case, comprise three sets of two each.

Format: !COMPARE-[options] A [other options]$\triangle$

DIRECTORY:

DISK-UNIT # (0,1): $u_d\triangle$

VOL.# (OCTAL, 0-max): $v_d\triangle$

1ST FILE:

DISK-UNIT # (0,1): $u_1\triangle$

VOL.# (OCTAL, 0-max): $v_1\triangle$

2ND FILE:

DISK-UNIT # (0,1): $u_2\triangle$

VOL. # (OCTAL, 0-max): $v_2\triangle$

Parameters:	u_d, u_1, u_2 -	single digit, currently limited to 0 or 1.
	v_d, v_1, v_2 -	a one-digit or two-digit octal number; max is determined individually for the separate disk drives, and will be 11 for full NCS disks, or 04 for "split packs".

Options: A - Automatic filenames: each name will be taken from the directory found on unit/volume indicated here as u_d, v_d. A file of this name then will be looked for in the directories of both the other units/volumes.

L - Logging
U - Unpack
Q - Quantify

$\Big\}$ as above (see non-A mode)

N - N-response automatically generated in case of mismatches so that the run can proceed without stopping. The prompt "MORE (Y, N, 1, 2, ESC)" will not be displayed, under option "AN", for mismatches. COMPARE will go on to the next filename in the master directory, or quit if the end of the directory is reached. (Without option "N" a mismatch would cause this prompt to be displayed, followed by a pause for user response; the user's answer of ESC would then terminate the whole run, while an "N" response would force it to go on to the next filename.)

: - Restart at specified name. This colon must be followed immediately by a filename, also existing in the specified master directory. The purpose is to restart an interrupted run without having to recompare all preceding files. When execution begins, under options "A:", all names from the specified directory are passed over if they do not match the given name. This name, being a separate input parameter, requires its own terminator (△). Thus the input sequence could be of forms

```
!COMPARE-A[options not :] :name△[other options]△
...etc...
```

```
!COMPARE-A[all options not :] :name△△
...etc...
```

The resulting full display, for instance under the first of these input sequences, would have the form

```
!COMPARE-A[options not :] :
FILENAME =  name△[other options]△
DIRECTORY:
...etc...
```

REMARKS

1. This utility program violates the constraint honored by most NCS utilities, that the only accessible files are on the SYSTEM pack and on the single USER pack previously selected, on a single disk-drive Unit. The program accomplishes this switching among various Units and various Volumes via internal calls to the NCS utilities UN and USER.

2. Neither NCS nor COMPARE has any way of recording what Unit/Volume were current at the time that COMPARE was invoked, so a return to that assignment cannot be guaranteed. Thus, when control finally returns to NCS, the current Unit/Volume assignment will remain at that which was last accessed -- which may not be where the user wishes to be.

3. Each line buffer currently is sized at 82 characters, to allow the normal (edited) sourcecode format: LF, 80 printables, CR. Used with normal sourcecode files, COMPARE computes line numbers which should agree with those printed under execution of the MA and UPDATE utilities. However, if used with binary files, or with long-line printable files (e.g., assembly listings), COMPARE will display line numbers which have little meaning, except as a reminder of whether the user had advanced one file relative to the other.

4. Files to be compared are read as 16-bit-per-word records. Thus 18-bit-per-word files (such as MULTI microcode) cannot be guaranteed to compare correctly. Binary files, even if containing 16-bit-per-word data (such as the emulated-NOVA invokable files), may well contain embedded null-bytes which are meaningful. Since COMPARE drops null-bytes, comparisons of even these files also should not be considered wholly reliable.

5. The Q (quantify) option, intended to show how much space is needed and how much is still available in an existing file, also would not give unambiguous results for binary files. This is because the embedded null-bytes (if any), before being dropped, are tallied in the same counter as are the final padding null-bytes.

6. Under option A (automatic filenames generation), a name from the specified master directory may be found missing from either or both Unit/Volume directories where the two comparand files are required to be located. The remark

<u>NO FILE BY THAT NAME ON THIS VOL.</u>

is then displayed for each such file, and COMPARE skips to the next filename in the master directory.

7. The ESC character can be used to terminate input of parameters, and return to NCS. This is true during the option-string input, and also

during input of either the u paramater or the v parameter. However it is not true during input of each filename.

8. Under options AN (automatic No), a pair of files whose first lines mismatch will not cause the comparisons to terminate. It will take mismatch of a later pair of lines to force the termination of comparisons on the pair of files. This feature was installed because many sourcecode files differ in some minor way (e.g., revision number) in the title-lines, whereas what we really wish to display is the first line of operable code by which they differ. Thus a first-lines mismatch still displays the mismatch (line buffers, followed by line numbers both equalling one), but then takes an "automatic Yes".

COPY

This utility copies byte-oriented data from one file or device to another file or device. All files (if any) must exist on a single disk drive.

Format: !COPY[-options]Δ I=sourcecode-inputΔ O=sourcecode-outputΔ

Parameters: sourcecode-input = name of the input file, or device.

sourcecode-output= name of the output file, or device.

Options:	U -	Unpack the input file contents (meaningful only if reading from disk).
	P -	Pack the output file contents (meaningful only if writing to disk).
	F -	Formfeeds are to be made visible (usually for verification purposes only). Here COPY will use the four characters < FF > to display on the CRT (if O=*CRT), or to print on the lineprinter (if O=*LPT), at each occurrence of a formfeed character on the input device/file.

Disk file names on the USER pack (USER n > 0) are prefixed by a slash (/).

Disk file names on the SYSTEM pack (USER 0) are not prefixed.

Device names consist of an asterisk (*) followed by the device mnemonic.

The recognized input device names are:

*TTCΔ	Console keyboard with parity removed
*TTIΔ	Console keyboard with parity Intact.
*CDRΔ	Card reader. For formfeed (FF), punch a 0,2,8 combination in any single column. For end-file (EOF) indication, put FF in both columns 1 and 2.
*TP(unit)	Magnetic tape. Can be either 9-track, unblocked, EBCDIC format, or 7-track, unblocked, BCD format. The "unit" parameter is 0 for the first tape drive, 1 for the second, etc. Close parenthesis replaces the terminatorΔ .

The recognized output device names are:

*CRT△	Console display.
*LPT△	Line printer.
*PCH△	Card punch.
*TTY△	Teletype (printer or paper-tape punch).
*TP(unit)	Magnetic tape; same conditions as for input device.
*NUL△	Dummy device; useful if advancing a multiple-file medium (usually magnetic tape) past one file out of many.

Examples:

To read cards from the card reader and output to the line printer as 80-column card images, stopping only on encountering EOF card in the card deck:

```
!COPY△I=*CDR△O=*LPT△
```

To move tape past four end of file marks, guaranteeing a start from the BOT (if so desired), for tape unit zero:

```
!REWIND 0
!COPY△I=*TP(0) O=*NUL△
!COPY△I=*TP(0) O=*NUL△
!COPY△I=*TP(0) O=*NUL△
!COPY△I=*TP(0) O=*NUL△
```

To assemble some sourcefile, put the listing on disk (packed), then retrieve the same disk file with unpacking of the packed code, putting two readable copies on lineprinter and one on tape:

```
!MA-AD△LISTING FILE=lp.filename△DEF={def.filename△}
                                     {     △      }
    INP= sourcefile△BIN={bin.filename△}
                        {     △      }
!COPY-U△I= lp.filename△O= *LPT△

!COPY-U△I= lp.filename△O= *LPT△

!COPY-U△I= lp.filename△O= *TP(0)
```

NOTE: COPY does not support Cartridge Tape drives as input or as output devices.

COPYD

COPYD copies data from one NCS disk file to another disk file. The transfer is done on a sector by sector basis making it much faster than COPY.

Format:	!COPYD△ I=source-filename△ O=destination-filename
source-filename	the NCS disk input file
destination-filename	the NCS disk output file

Disk file names on the USER pack are prefixed by a slash (/). Disk file names on the SYSTEM pack are not prefixed.

Example:

```
!COPYD△ I=WI△  O=/30A△
```

DISCART

DISCART copies NCS disk files to cartridge tape. These files can be copied back to disk using the BUILD utility.

```
    Format:     !DISCART△

                CART TAPE UNIT NUMBER = unit-number△

                         TRACK NUMBER = track-number△
                                                           / - \
                        START AT FILE = file-marks-to-be-skipped{ * }
                                                           { + }
                                                           ( △ )
                             FILENAME = filename△

                ENTER DATE (mmddyy) = date△
                                                 / - \
    (* or + entered) START AT FILE = file-number₈{ * }
       ( △ entered) !                            { + }
                                                 ( △ )
```

unit-number	cartridge tape unit number 0 or 1
track-number	track number 0, 1, 2, or 3
file-number$_8$	number of physical file marks to be skipped over to position the tape for writing. Each file is terminated with two file marks. To write multiple files, the starting file numbers would be 0, 2, 4 ..... * skip past specified file marks - do not rewind before or after the write. After the file is written, the next starting file is requested. + Rewind, skip past specified file marks - do not rewind after the write. After the file is written, the next starting file is requested. △ Rewind and skip past specified file marks. Rewind after the file is written and return to NCS. - Rewind and return to NCS.
filename	name of file to be written. USER file names are prefixed with a slash (/). SYSTEM file names are not prefixed.
date	the date. May be any six digits.

Any entry can be cancelled by entering ESC. The prompt is then re-displayed requesting the entry.

DISCART writes the address and size parameters of the file into a header record which is read and used by the BUILD utility.

EXEC

EXEC replaces operator key-ins with characters from a unique disk file, named COMMANDS (pseudonym \\). This permits automatic entry of prerecorded commands and responses, at maximum speed.

Format: $\underline{!}$ EXEC Δ

EXEC copies characters from the COMMANDS file into upper memory, as one complete block. Then it switches the NCS character-inputter, from console-keyboard reading, to reading of this buffer, and turns control back to NCS. When NCS next obtains commands and responses, these will therefore come from the memory-buffer copy of what had been in disk file COMMANDS. When EXEC's buffer-reader can read no more characters for NCS, it restores NCS such that reading of command characters is switched back to the console keyboard.

RESTRICTIONS: EXEC code and the memory buffer appear at the emulated-NOVA memory locations 40000-44177 octal. Any program that writes in this area can clobber the action of EXEC; e.g. as with symbol tables for large assemblies under the microassembler utility program MA. The file named COMMANDS can have maximum length of 10_8 disk sectors, i.e. at most 512_{10} *8. $=4096_{10}$ characters will be copied to the memory buffer. Finally, EXEC expects this file to begin on an exact cylinder boundary, i.e. sector address of zero. Any installation finding that EXEC malfunctions, should investigate whether COMMANDS has been moved--as often done to disk files during compacting of unused directory space. Use the FILES utility program to verify location and, if necessary, relocate the file named COMMANDS via "Change" parameters

$\underline{\text{ADR=}}$ cyl Δ sect Δ length Δ

with length $\leq 10_8$, sect=0 (always), and cyl=any convenient position.

EX

EX performs somewhat like EXEC, but with certain improvements. The disk file which is to be read in, and which must already contain commands and responses for execution, is named in the initial dialog.

Format: !̲ EX△ filename△

The named disk file again has a length restriction; however, this is offset by the fact that several files can chain-link in succession. Example:

!̲ EX△ file1△	(console command)
EX △ file2△	(last command in "file1")
EX △ file3△	(last command in "file2")

Because program EX and its buffer are located higher in the emulated-NOVA memory, there are fewer occasions where another program clobbers the functioning of EX.

RESTRICTIONS: EX code and the memory buffer are placed at fixed memory locations 74400-77734 octal. The named disk file must not overflow a track on disk. Its length must be six (6) disk sectors or less; i.e. a maximum of 512_{10} *$6=3072_{10}$ characters will be copied from disk to the memory buffer.

FILES

FILES, the file-maintenance utility, reports the status of file directories; and it adds, changes, or deletes individual file name-blocks within any directory. FILES is directed by keyed-in commands; to prompt for these it displays "*". Some commands have one or two levels of subcommands. Each command and subcommand is activated by entry of a single-letter mnemonic; the suffix characters are filled in by FILES. Other (unabbreviated) parameters are required after some commands and subcommands. When invoked, but before accepting the first single-letter command, FILES demands entry of a six-digit date code.

```
    Format:     ! FILES
                FILE DIRECTORY UTILITY, ENTER DATE (MMDDYY) = date
                * command [ ... subcommand ... ]
```

Table 4 summarizes the primary commands by initial letter, their completion (filled in by FILES), and functioning. For A, C, and D, after filename parameters the ellipsis "..." indicates subcommands. Table 5 specifies secondary and tertiary commands; these are the subcommands needed under primary commands A, C, and D. Only A and C allow multiple subcommands; only under secondary command TYPE are there tertiary commands. Termination of a multiple-subcommand sequence depends on level. Enter ":" to terminate the tertiary command sequence; enter "G" (GO) to terminate secondary ones. Table 6 names all parameters of the file name-block. Under commands PRINT and REPORT, parameters are displayed for all files. For commands A, C, and D, file-parameters display before the user can command the finalization. The checksum parameter is created by FILES; all other parameters are enterable under command A.

Table 4. Single-letter (*) Commands, accepted by FILES:

*	Display after Completion	Notes
E	END OF JOB	1.
V	VOLUME NR. (OCTAL,0-11): nn Δ	2.
R	REPORT ON mmddyy , VOLUME NUMBER nnnn	3.
P	PRINTING REPORT	4.
A	ADD (NEW FILENAME:) cccccccccc Δ ...	5.
D	DELETE (ENTER FILENAME:) cccccccccc Δ ...	6.
C	CHANGE (ENTER FILENAME:) cccccccccc Δ ...	7.
U	UNUSED ON THIS VOL.=	8.
L	L.CUTOFF= nnnn Δ	9.
N	NAME-SUMMARY	10.
S	SEARCH (ENTER SUBSTRING OR FULL NAME): cccccccccc Δ	11.
T	TOTAL-DISK SEARCH (ENTER SUBSTRING OR FULL NAME): cc...c Δ	12.

Notes: If instead of the initials above, a "control" key (low-valued ASCII codes, including standard "separator" characters) is entered, it is ignored and FILES restarts the command entry. Exceptions are initial CR (outputs an extra LF and then restarts); and an ESC character anywhere in the string (terminates current command or subcommand, restarts entry of another complete command). Ellipses "..." above indicate where subcommands are expected.

1. Forces an exit. FILES terminates; N.C.S. obtains control. User number remains set to value for last Volume command, or to its value at entry to FILES if no Volume command was given.

2. User is to enter "nn" as a one or two digit octal number, where nn=1 up to upper limit implies some "user volume", while nn=0 implies the "system volume". Upper limit, indicated here as eleven, becomes 4 in the case of "split pack" configurations.

3. Here FILES itself displays the date "mmddyy" (entered by user when FILES was invoked), and the octal volume number "nnnn" (last entered by Volume command). After the line shown, there follows a line of column

headings, and then full directory display on successive lines. To stop at the end of a line currently being output, hit any key; when thus stopped hit ESC to terminate the Report command at that point, or hit any other key to continue.

4. This display, on the console CRT, is followed immediately by output to lineprinter of items under command R and Note 3.

5. The symbols cccccccccc indicate that a 1-10 character filename parameter must be entered. It should not exist already in the directory of the selected Volume (else an error message will be displayed). If no error, after the filename entry FILES waits for entry of subcommands and parameters per Table 5.

6. Again a 1-10 character filename parameter must be entered. This filename must already exist in the directory (else an error message will be displayed). After the filename, to confirm deletion enter subcommand Y (Table 5); to avoid deletion enter any other character (usually CR).

7. Conditions as per Note 6. Parameters as per Note 5.

8. The U command displays only the "unused" areas, abstracted from the display produced under the Report command.

9. Determination of what is meant by an "unused" area, under the R, P and U commands, is complicated by the fact that large pseudo-files can be defined so as to overlap several smaller genuine files, including unused spaces between the small files. A "cutoff" value of file length is used, for decision as to whether to display an unused area. The default cutoff length is 2200 octal. Any area will be prevented from reporting as unused, if it is overlapped by any existing file of length in sectors less than this cutoff value. Use the L command to enter a new cutoff length (any octal value from 1 to 22600).

10. Displays the filenames of the current directory, packed eight per line of the 80-character CRT line. All other parameters from the directory are dropped, in an attempt to display all or most names from one directory on a 24-line screen.

11. The symbols cc...c indicate that the user is to enter 1-10 characters representing all or part of a filename whose existence is in question. Directory of current Volume is scanned, and each full filename compared to see whether it fully matches the inputted name, or the inputted substring can be embedded in the full name. Any successful matches are displayed, seven per line on the CRT.

12. Same parameter entry as for command S, Note 11. Here all volumes are scanned in descending (wraparound) order, so as to end at the current Volume Number. For each volume, the output displayed is that described in Note 11.

Table 5. Subcommands (*) accepted after certain Commands:

*	Display after Completion	Notes
	...for DELETE command...	
Y	Y	1.
	...for ADD or CHANGE command...	
A	ADR = cyl Δ sect Δ length Δ :	2.
T	TYPE = $code_1$ [Δ $code_2$ [Δ $code_3$]] :	3.
P	PROGR = load-pt. Δ pgm.length Δ entry-pt Δ :	4.
G	GO	5.
:	:	6.
CR	(Carriage Return) LF (Line Feed)	7.

NOTES: If instead of the above subcommands (secondary commands, in leftmost column, and tertiary commands, under the Type secondary command), what is entered for the initial letter is a "control" key (low-valued ASCII codes including the standard "separator" characters), then this initial character is ignored and FILES restarts the reading of the particular secondary or tertiary command. Exceptions are: initial CR (first produces an extra LF output, then restarts); and an ESC key (given anywhere in the string, this terminates the current secondary or tertiary command, and restarts entry of another complete primary command).

1. Unlike all other subcommands, here a CR will not give approval for completion to take place (deletion of filename from the directory). To make it more difficult to err, only if an upper-case Y is entered will deletion be approved.

2. Subcommand A needs three octal numbers as parameters; after the third standard terminator character, FILES outputs a ":" to indicate acceptance of the input. The first two parameters must represent a legal starting location for the desired file (for sect use octal values 0-27). The third parameter must represent a legal file length in sectors (octal values in range 1-22600 for the N9755 disk).

3. Subcommand T is a secondary command and the codes are treated as tertiary commands; each is input as a single letter, chosen from this set of six characters:

R , W ; D , P ; N , E ;

where R,W are mutally cancelling, likewise for D,P and for N,E.

Completions and meanings for these tertiary commands are:

READ	File is to be write-inhibited (default).
WRITE	File is to be write-enabled, under N.C.S. utilities
DATA	File is to contain "data", i.e. not a program (default).
PROGR	File is to contain an invokable program under N.C.S.
NO-EXEC	File not executable if a program; i.e. N.C.S. after loading, will not obey the entry-pt parameter (default).
EXEC	Invocation starts execution after loading; uses entry-pt parameter (entered under secondary command P of Table 5). The N.C.S. invocation function (!filename) uses this address regardless of any start-address given during assembly of the program itself.

Since these tertiary commands can be given in any amount and any sequence, the standard separator merely prompts FILES to expect another such command. Thus, to escape from this string of tertiary commands back to secondary commands, a different character, ":" , is needed. This is one of two cases where the user enters a colon.

4. Subcommand P needs three octal numbers as parameters; after the third standard separator character, FILES outputs a ":" to indicate acceptance of the input. If elsewhere the user has specified Type=Data, then the three parameters are not interpreted by N.C.S. If however Type=Progr, then per Table 5 the three parameters will represent two absolute locations and one program length (words), of a presumed NOVA core image program defined in 16-bit binary words. In this case the "!filename" invocation under N.C.S. is allowed to load the specified number of words, starting at the specified absolute address in emulated NOVA memory. If it is true not only that Type=Progr, but also that Type=Exec, then this invocation function will, after completion of loading, begin execution at the indicated entry-point (absolute address).

5. GO indicates that the user has finished specifying the file's description or alteration. On receipt of Go, FILES will display the complete set of tentative file name-block parameters (see Table 6), with warning messages (if any), for approval. Warning formats are:

EXTENT OVERLAP ON FILE filename

6. The user may enter a colon, ":" , after the Note 5 display has completed, if he has observed a mistake and wishes to change the parameters. This is the second of two cases where users enter colons.

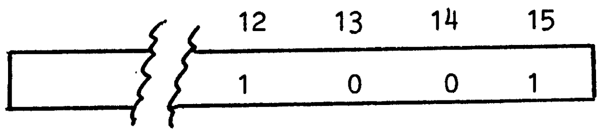
7. The user may enter the CR (carriage return) after the display of Note 5 has completed, to command FILES to accept the alteration of parameters of an old file, or the installation of a new file.

Table 6 File-block Parameters:

Parameter	Definition
filename	a name of one to ten characters, unique to this volume's directory, and not embedding any of the standard separator characters (period, space, etc).
date	Any six digits; recommended as latest upgrade date.
cyl	Starting disk-cylinder address (octal) for the file.
sect	Starting disk-sector address (octal, 0-27).
length	Octal length of file in sectors, where each sector can hold (in emulated NOVA as used under N.C.S.) 400_8 words or 1000_8 characters.
type	File type/access information (per Table 7).
load-pt.	Meaningful only for Progr files, an absolute address load point (octal) for the program.
pgm.length	Octal number of 16-bit words in the program, used along with load-pt when loading, and meaningful only for Progr files.
entry-pt.	Octal entry point, as an absolute word address, used by N.C.S. as the next point to which to transfer control after completion of loading, but only if the file has been declared both Progr and Exec type.
checksum	A file-block validation word (octal sum of all other parameters), made up automatically by FILES.

Comments: These parameters are displayed in a single row, for one file, after a "G" (G<u>O</u>) subcommand is entered under A<u>DD</u>, C<u>HANGE</u>, or D<u>ELETE</u>. Multiple rows of these parameters, representing all files in the current volume's directory, are output under R<u>EPORT</u> and P<u>RINT</u>. Numeric fields (all parameters except filename) are displayed in octal, as either 6, 4, 2 or no digits; i.e. length of 22654 is displayed as 022654, while value zero is displayed as a blank field. The "type" code, reported as a two-digit octal number, represents the pattern of four bits described in Table 7.

These bits are retained in a separate 16-bit word within the file name-block group, as bits 12-15 (NOVA convention). All other bits of this word are reserved.



This example shows a bit pattern, displayed as octal 11 by FILES, with meainings: write-enabled; contains data; is on User 0.

Table 7. Type-parameter Bit Assignments:

Binary-pattern	Bit-nr.	Function
0001	15	1 = File is write-enabled.
		0 = File is read-only; any attempt to write on it should produce an error condition.
0010	14	1 = File contains program, loadable into emulated NOVA memory (QM1 mainstore). User key-in of filename, after the "!" prompt, activates the N.C.S. invocation function, which accepts parameters described under PROGR subcommand. The load-pt tells it where the destination area begins in memory. The length tells it how many 16-bit words to copy into one contiguous region. These parameters are also needed by the "$" utility which must have been invoked (!$filename) to have copied the absolute core-image program, from mainstore to disk file in the first place.
		0 = File contains data only (non-program). The absolute address and length fields are meaningless to the N.C.S. invocation function (although they may be user defined for other purposes).
0100	13	1 = Allow the N.C.S. invoker to transfer control, after loading, to the absolute address given as entry-pt parameter under the PROGR subcommand. This bit is set by EXEC, cleared by NO-EXEC tertiary commands. For N.C.S. to obey this transfer of control, the previous bit (pattern 0010) must also have been set.

Binary-pattern	Bit-nr.	Function
		0 = Prevent the invocation from transferring control after loading; instead, control returns to N.C.S.
1000	12	1 = Cause file to be obtained from SYSTEM pack.
		0 = Cause file to be obtained from the USER pack (whichever volume last selected for Volume command or, external to FILES, by !USER.)

FREE

FREE displays the graphic FREE on the CRT.

Format: !FREE △

<u>LD</u>

LD loads a NOVA binary program into mainstore from a disk file, and optionally starts execution of the program.

Format: !LD△ [*]filename[$\underline{+}$nnnnnn$_8$]△

filename	the disk-file holding microassembler output of a NOVA program. The output is received in a QM-1 format and LD reformats it into NOVA format binary as it is loaded into mainstore.
*	indicates that the file is not to be executed.
$\underline{+}$nnnnnn$_8$	offset the load address by this value.

If the file does not contain a start address block or if the * option is used, control is returned to NCS; otherwise control is passed to the loaded program at the address in the start block.

MAP

MAP prepares an executable NANO binary program from the nanoassembler output of a nanoprogram source.

Format: ```
!MAP△
INPUT FILE= input-filename△
OUTPUT FILE= output-filename△
nnnnnn8 SECTORS IN, nnnnnn8 SECTORS OUT
```

| | |
|---|---|
| input-filename | the input to MAP is the binary output of a NANO program assembly by the nanoassembler. |
| output-filename | the executable NANO binary program output by MAP. |

Error Message:

If an invalid file is specified, the following message is displayed.

```
FILE INCOMPLETE, RESTARTING
INPUT FILE=
```
```

NOVA MEMORY DUMP PROGRAM

The NOVA memory dump program is a location independent program that dumps NOVA memory to the line printer. Current invokable verisons of the NOVA dump program are called D1000 and D15100. The name incorporates the load point of the dump program. The dump program occupies just under 700_8 mainstore locations.

```
Format:     !D1000 Δ
              or
            !D15100 Δ
            DUMP FROM:  start-address Δ  TO:  end-address Δ
```

The contents of all locations from start-address to end-address are dumped to the line printer in octal, hexadecimal, and ASCII, as shown in Figure 1. In the octal dump, all-zero words are depicted by six dashes rather than six zeros. In the ASCII dump, lower case ASCII characters (greater than 140) are printed as upper case. Special symbols (less than 040) are printed as underlines.

When the dump has been printed, pressing any key causes the DUMP FROM: prompt to be displayed, and new bounds may be entered.

A file containing the dump program loadable at any other desired location in memory can be created using the procedure, HOW TO PREPARE AN INVOKABLE NOVA PROGRAM, described in Section 2.

The new file directory entry has a new file name, location parameters that overlap D1000, and program parameters specifying the new loadpoint and entrypoint (entrypoint equals loadpoint) and the length (700_8 words).

The NOVA dump program may be invoked by an executing NOVA program, as follows:

Load the start and end addresses into ACO and AC1, then store them in the address array in the dump program. START represents the load/entry point of the dump program in this procedure. The dump program is activated by entry to START + 2.

```
  LDA       2,ADRST          ;START ADDRESS OF DUMP PROGRAM
  LDA       3,3,2            ;FETCH ADDRESS ARRAY DISPLACEMENT FROM START
  ADD       2,3              ;GENERATE ADDRESS ARRAY LOCATION IN AC3
  STA       0,0,3            ;STORE FIRST ADDRESS TO BE DUMPED
  STA       1,1,3            ;STORE LAST ADDRESS TO BE DUMPED
  LDA       3,ADRST          ;START ADDRESS OF DUMP PROGRAM
  JMP       2,3              ;JMP TO START + 2
ADRST:  START                ;ADDRESS OF DUMP PROGRAM
```

When the specified dump has been produced, the dump program jumps to the keyboard interrupt routine. If control is to be retained by the calling program, location START must be modified to execute a jump back to the calling program. The following instruction sequence may be used:

```
        LDA     2,RETRN         ;RETURN ADDRESS
        STA     2,7             ;SAVE IN LOW CORE
        LDA     2,ADRST         ;ADDRESS OF DUMP PROGRAM
        LDA     3,INSTR         ;RETURN INSTRUCTION
        STA     3,0,2           ;PLACE AT START
        JMP     2,2             ;JUMP TO START + 2
RETRN:  GO.ON                   ;RETURN ADDRESS
INSTR:  JMP     @7              ;RETURN INSTRUCTION
ADRST:  START                   ;ADDRESS OF DUMP PROGRAM
GO.ON:  **                      ;RETURN HERE
```

```
  ADR      NOVA MEMORY DUMP -OCTAL-  (NANODATA CORP)                   -HE[illegible]ECI[illegible]-                        CHA[illegible]ER-

001000 - 062677 004506 000574 000270 000001 006412 ------ 042125   65BF 0946 017C 00B8 0001 0D0A 00C0 4455   E?_F_\_@ _______GU
001010 - 046520 020106 051117 046472 020000 020124 047472 020000   4D50 2046 524F 4D3A 2000 2054 4F3A 2000   MP FROM:  _ TO: _
001020 - 054410 004401 175120 173020 004422 002403 004403 000775   5908 0901 FA50 F610 0912 0503 0903 01FD   Y___ZPV_ _______!
001030 - 001143 054414 034752 175035 000405 063551 000777 061151   0263 590C 39EA FA1D 0105 6769 01FF 6269   _CY_9JZ_ __GI__BI
001040 - 002405 063511 000777 061111 002401 001027 141220 151400   0505 6749 01FF 6249 0501 0217 C290 D300   __GI__BI ____E_S_
001050 - 040412 022411 050410 030410 101013 101300 143424 175400   410A 2509 5108 3108 820B 82C0 C714 FB00   A_%_Q_1_ ___@G_{_
001060 - 030402 001400 003337 000177 024720 125035 000410 063650   3102 0300 06DF 007F 29D0 AA1D 0108 67A8   1____2__ )P*___GI
001070 - 000777 064650 063551 000777 065151 000407 063610 000777   01FF 69A8 6769 01FF 6A69 C1C7 67E8 01FF   __I(GI__ JI__G___
001100 - 064610 063511 000777 065111 020757 107420 001400 152440   6988 6749 01FF 6A49 21EF 8F10 0300 D520   I_GI__JI !O_____U
001110 - 054435 050673 060210 060250 063710 000407 063650 000775   591D 51BB 6088 60A8 67C8 0107 67A8 01FD   Y_Q;@_@( GH__GI_!
001120 - 064650 151400 050662 000402 064610 030421 004672 030420   69A8 D300 51B2 0102 6988 3111 09BA 3110   I(S_Q2__ I_1__:1_
001130 - 004670 004420 024412 133420 050534 030413 004662 004412   09B8 0910 290A B710 515C 310B 09B2 090A   _8__)_7_ Q\1__2__
001140 - 050531 030405 004656 002402 177770 001002 177746 177752   5159 3105 09AE 0502 FFF8 0202 FFE6 FFEA   QY1___.____X___F_J
001150 - 177766 054420 152440 004711 034413 166442 002413 034411   FFF6 5910 D520 09C9 390B ED22 050B 3909   _VY_U _I 9_M"__9_
001160 - 166453 002410 151120 151120 151120 133020 000765 000060   ED2B 0508 D250 D250 D250 B610 01F5 0030   M+__RPRP RP6__U_0
001170 - 000010 001140 004001 014001 004011 004005 000512 000601   0008 0260 0801 1801 0809 0805 014A 0181   ___@_____________J_
001200 - 054435 024773 066217 063617 000777 064417 125223 000776   591D 29FB 6C8F 678F 01FF 690F AA93 01FE   Y_)@L_G_ __I_*__|
001210 - 024762 066217 002423 054422 024757 066217 000767 054416   29F2 6C8F 0513 5912 29EF 6C8F 01F7 590E   )RL___Y_ )OL__WY_
001220 - 024755 000761 054413 044413 004401 175120 173020 004617   29ED 01F1 590B 490B 0901 FA50 F610 098F   )M_QY_I_ ____ZPV_
001230 - 101010 004406 014404 000774 002401 001516 ------ 054421   B208 0906 19C4 01FC 0501 C34E 0CC0 5911   ________\ ___)__Y_
001240 - 024421 106032 000414 034417 116033 162440 136440 116033   2911 8C1A 010C 390F 9C1B E520 BD20 9C1B   )_______9_ __E = __
001250 - 162440 074417 175233 000776 061117 002403 020405 000772   E520 790F FA9B 01FE 624F 0503 2105 01FA   E Y_Z__I BO__!__Z
001260 - 001232 000040 000140 000037 000073 000015 000672 000063   029A 0020 0060 001F 003B C00D 01EA 0C33   ___ _@__ _;___:_3
001270 - 001270 001700 000756 000015 000774 000013 020055 ------   02B8 03C0 01FE 000D 01FC 000B 202D 0000   _8_@_N__ _\__ -__
001300 - 000122 000002 001010 000005 000043 001305 000020 000035   0052 0002 0208 0005 0023 02C5 0010 001D   _R______ _#_E____
001310 - 126520 044773 004701 024755 030755 132452 000661 014765   AD50 49FB 09C1 29ED 31ED B52A 01B1 19F5   -PI(_A)M 1M5*_1_U
001320 - 000423 024743 044762 004655 024743 030741 004674 024757   0113 29E3 49F2 09AD 29E3 31E1 09BC 29EF   __)CIR_- )C1A_<)O
001330 - 004561 024742 030740 004667 024753 004554 024737 030735   0971 29E2 31E0 09B7 29EB 096C 29CF 31DD   _Q)B1@_7 )K_L)O1]
001340 - 004662 004656 004655 024725 004511 000406 004671 000515   09B2 09AE 09AD 29D5 C949 010E 09B9 014D   _2_._-)U _I___9_M
001350 - 000652 000646 000665 024726 030724 004645 024712 044726   01AA 01A6 01B5 29D6 31D4 09A5 29CA 49D6   _*_&_5)V 1T_%)JIV
001360 - 024573 044722 126520 004526 026721 010720 125035 000405   297B 49D2 AD50 0956 2DD1 11D0 AA1D 0105   ){IR-P_V -Q_P*___
001370 - 004465 000406 004760 000471 024514 030705 004624 014704   0935 0106 09F0 0139 294C 31C5 0994 19C4   _5___P_9 )L1E___C
001400 - 000762 024546 004507 024665 044701 024546 044675 126520   01F2 2966 0947 29B5 49C1 2966 49BD AD50   _R)F_G)5 IA)FI=-P
001410 - 004501 026674 010673 004506 000403 004735 000510 014664   0941 2DBC 11BB 0946 0103 09DD 0148 19B4   _A-<_;_F ____H__4
001420 - 000767 024540 004467 030645 004401 024426 166040 133020   01F7 2960 0937 31A5 0901 2916 EC20 B61C   _W)@_71% __)_L 6_
001430 - 151120 024522 004716 126520 004455 030633 024451 133000   D250 2952 09CE AD50 092D 319B 2929 B600   RP)R_N-P _-1_))6_
001440 - 147000 044627 004401 024411 166040 133020 151120 024504   CE00 4997 0901 2909 EC2C B610 D250 2944   N_I___)_ L_6_RP)D
001450 - 004700 004700 000641 000201 000217 175420 054425 020427   09C0 09C0 01A1 0081 008F FB10 5915 2117   _@_@_!__ __{_Y_!_
001460 - 040425 152520 125120 000406 024420 030421 125120 125100   4115 D550 AA50 0106 2910 3111 AA50 AA40   A_UP*P__ )_1_*P*@
001470 - 125100 044413 125100 147400 020460 123020 014407 002404   AA40 4908 AA40 CF00 2130 A610 19C7 0504   *@I_*@O_ !0&_____
001500 - 014403 002402 000400 001371 001300 ------ 000007 000004   1903 0502 0100 030C 1060 0000 0007 0004   ________ _ ______
001510 - 000006 054444 044444 126520 030444 004633 014440 000774   0006 5924 4924 AD50 3124 099B 1920 01FC   __Y$I$-P 1$___ _\
001520 - 002435 175420 054761 020426 040761 044757 024756 125120   051D FB10 59F1 2116 41F1 49EF 29EE AA50   __{_YQ!_ AQIO)N*P
001530 - 125100 125100 125100 044751 125100 030415 147400 020415   AA40 AA40 AA40 49E9 AA40 310D CF00 210D   *@*@*@II *@1_O_!_
001540 - 123000 034407 166452 000733 024742 123000 000730 000002   A600 3907 ED2A 01DB 29E2 A600 01D8 0002   &_9_M*_{ )B&__X__
001550 - 000012 000005 000017 000010 000060 001364 ------ 020000   000A 0005 000F 0008 0030 0309 00C0 2C00   ________ _0_____ _
001560 - 000664 000003 020101 042122 020040 020040 020116 047526   01B4 0003 2041 4452 2020 2020 204E 4F56   _4__ ADR     NOV
001570 - 040440 046505 046517 051131 020104 052515 050040 026517   4120 4D45 4D4F 5259 2044 554D 5020 2D4F   A MEMORY  DUMP -O
001600 - 041524 040514 026440 020050 047101 047117 042101 052101   4354 414C 2D20 2028 4E41 4E4F 4441 5441   CTAL-  ( NANODATA
001610 - 020103 047522 050051 020000 026510 042530 040504 042503   2043 4F52 5029 2000 2D48 4558 4144 4543    CORP) _ -HEXADEC
001620 - 044515 040514 026400 026503 044101 051101 041524 042522   494D 414C 2D00 2D43 4841 5241 4354 4552   IMAL-_-C HARACTER
001630 - 026400 026455 026455 026455 ------ 000757 030037 132404   2D00 2D2D 2D2D 2D2D 0000 01EF 301F B504   -_------ ___O0_5_
001640 - 000404 014212 010207 000763 030444 132404 000410 006127   0104 188A 1087 01F3 3124 B504 0108 0C57   _______S 1$5____W
001650 - 024227 125225 000403 006126 000130 024233 020052 107400   2897 AA95 0103 0C56 0058 289B 202A 8F00   (_*____V _X(_ *__
001660 - 020426 101212 000403 107000 000417 176620 131221 151200   2116 828A 0103 8E00 010F FD90 B291 D280   !_______ __}_2_R_
001670 - 151112 173000 151004 000774 125302 167000 125100 101112   D24A F600 D204 01FC AAC2 EE00 AA40 824A   RJV_R__\ *BN_*@_J
001700 - 002406 030406 006110 014020 000776 002401 ------ ------   0506 3106 0C48 1810 01FE 0501 00C0 0000   __1__H__ _!______
```

SAMPLE NOVA DUMP

PREP

PREP prepares a loadable file on cartridge tape from program files on disk. The program files can be nanostore, control store, and/or mainstore files. The PREP'ed file can be loaded into the three storage regions by selecting the track and file with the F-switches. Automatic initialization and initiation can be performed by the IML function if the PREP'ed file is terminated by a start record. The start record specifies the initial conditions for execution and the disposition of the cart tape.

```
Format:     !PREP△
            CART TAPE UNIT NUMBER = unit-number△
                      TRACK NUMBER = track-number [*/△]
```

unit-number	cartridge unit number first unit = 0, second unit = 1
track-number	Track on which load file is to be PREP'ed
	* Do not rewind the cart before processing.

PREP is directed by a set of commands, each performed in the sequence in which they are entered. The commands are described in Table 8. Any string may be restarted by entering ESC.

Command	Definition
#P	skip over an existing file on the specified track. One file is skipped for each P command.

#H
PROCESS = PROGRAM = OWNER = YY.MM.DD =
HH.MM.SS = COMMENTS =
	enter header record. The header format is displayed but no entries are expected. This is currently a dummy record.

#C
PRESET CODE (NANOSTORE,CONTROLSTORE, MAINSTORE) = $\left\{\begin{matrix}M\\C\\N\end{matrix}\right\}$
FROM nnnnnn$_8$△ TO nnnnnn$_8$ △ VALUE: set-value△
	clear the specified memory region within the specified range and set each location to the specified value.
	M = mainstore
	C = control store
	N = nanostore
	Mainstore and control store addresses are word addresses. Nanostore addresses are byte values with the word address in the first digits and the byte address in the last two digits.

#N MAPPED NANOCODE FILE NAME = filename△	load one absolute mapped nanocode object file to the tape.
#M MICRO BINARY FILE = filename $\left[\begin{matrix}+\\-\text{offset}\\*\end{matrix}\right]$ △	load one control store file to the tape. Optionally, offset the loadpoint by the plus or minus value given, or set the loadpoint to the value if * is selected.
#V NOVA BINARY FILE = filename△	load one 16-bit format mainstore file to the tape.
#Q QM-1 MAINSTORE FILE = filename $\left[\begin{matrix}+\\-\text{offset}\\*\end{matrix}\right]$ △	load one 18-bit format mainstore file to the tape. Optionally offset the loadpoint by the plus or minus value given, or set the loadpoint to the value if * is selected.
#I	Interrupt the current PREP and return to NCS. (Usually used to change USER number.)

#START CODE (HARDSTOP,NANOSTART,MICROSTART = $\left\{\begin{matrix}H\\N\\M\end{matrix}\right\}$	prepare and load a start record on the tape. H = hardstop; no automatic entry - must use read-only program with F-switch selection to start. N = nanostart; direct transfer to nanostore. M = microstart; loads all QM-1 registers from the microstart array and directs transfer to control store.
REWIND? $\left\{\begin{matrix}Y\\N\end{matrix}\right\}$	Y = the read-only boot program will rewind the cartridge after loading. N = no rewind after loading.
SKIP TO EOF? $\left\{\begin{matrix}Y\\N\end{matrix}\right\}$	Y = causes a skip to EOF. N = no skip to EOF.
START ADDRESS = nanostart or microstart address△	address at which to start execution of the loaded program.
#W	write one end of file on the tape.
#R	rewind the tape. Another track may be selected. Press ESC if no track is desired.
#EXIT FROM CART PREP	terminate PREP. Write one end of file and rewind the cart tape.
*E	return to NCS. This command may only be given after #E.

Table 8. PREP Commands.

Error Messages:

If the specified file is not found, the request for filename will be repeated. If the file is not a valid binary file, the following message is displayed:

```
FILE INCOMPLETE, RESTARTING.
```

The prompt character # is displayed, requesting a new command.

PREPD

PREPD prepares a loadable file on NCS disk from program files on disk. The program files can be nanostore, control store, and/or mainstore files. The PREPD'ed file can be loaded into the three storage regions by selecting the file with the disk bootstrap. Automatic initialization and initiation can be performed by the IML function if the PREP'ed file is terminated by a start record. The start record specifies the initial conditions for execution.

NOTE- PREPD writes 18 bit output files on disk. These files cannot be saved or restored with nomal NCS utilities.

Format: !PREPD△

FILENAME = filename△

filename	The file to which the information will be written.

Disk bootstrap functions thus (if the QM-1 DLSP switch is on): Pressing buttons MC (master clear), START, produces the display

???LD

on the console CRT. Operator completes the command by entering the above filename without a prefixed slash if the file is on User 0, or by entering n/filename if the file is on User n.

PREPD is directed by a set of commands, each performed in the sequence in which they are entered. The commands are described in Table 8.1. Any string may be restarted by entering ESC.

Command	Definition
#P	skip over an existing file on the input. One file is skipped for each P command. (Not yet operational)

#H
PROCESS = PROGRAM = OWNER = YY.MM.DD=
HH.MM.SS = COMMENTS =
	enter header record. The header format is displayed but no entries are expected. This is currently a dummy record.

#C
PRESET CODE (NANOSTORE, CONTROLSTORE, MAINSTORE =
FROM $nnnnnn_8$ △ TO $nnnnnn_8$ △ VALUE: set-value△
	{M C N} clear the specified memory region within the specified range and set each location to the specified value.
	M = mainstore
	C = control store
	N = nanostore
	Mainstore and control store addresses are word addresses. Nanostore addresses are byte values with the word address in the first digits and the byte address in the last two digits.

#N MAPPED NANOCODE FILE NAME = filename△	load one absolute mapped nanocode object file to the output file.
#M MICRO BINARY FILE = filename [+ / -offset / *] △	load one control store file to the output file. Optionally, offset the loadpoint by the plus or minus value given, or set the loadpoint to the value if * is selected.
#V NOVA BINARY FILE = filename△	load one 16-bit format mainstore file to the output file.
#Q QM-1 MAINSTORE FILE = filename [+ / -offset / *] △	load one 18-bit format mainstore file to the output file. Optionally offset the loadpoint by the plus or minus value given, or set the loadpoint to the value if * is selected.

#L PROCESS = PROGRAM = OWNER = YY.MM.DD.=
HH.MM.SS = COMMENTS =
	enter library header record. This header will cause the start of a new library.

#T PROCESS = PROGRAM = OWNER = YY.MM.DD= HH.MM.SS = COMMENTS =	enter trailer block to end a library. Causes the boot loader to terminate the current library.
#U USER # = user-num△	change the input USER number to the specified USER. This is a temporary change, upon exit to NCS the USER is set to its value at initalization.
#F INCLUDE FILE NAME = filename△	specify a file to be included at micro-load time by the IML loader. If the specified file is a USER file, the current USER directory is searched to validate the files existance. If it is a SYSTEM file, the SYSTEM directory is searched. The file itself is not accessed until load time, PREPD only writes out the name of the file.
#S START CODE (HARDSTOP, NANOSTART, MICROSTART)= $\left\{\begin{array}{c}H\\N\\M\end{array}\right\}$	prepare and load a start record on the file H = hardstop; no automatic entry N = nanostart; direct transfer to nanostore M = microstart; loads all QM-1 registers from the microstart array and directs transfer to control store.
START ADDRESS = nanostart or microstart address△	address at which to start execution of the loaded program. If user enters address value 777777, a previously-specified START address is retrieved and used.
#W	write one end of file record on the file.
#R	remark all characters up to the next (CR) are ignored
#E	terminates PREPD. Guarantees that the file is terminated by an EOF, EOI, closes the output, and returns to NCS.

Table 8.1 PREPD Commands.

Error Messages:

If the specified file is not found, the request for filename will be repeated. If the input file is not a valid binary file, the following message is displayed:

```
FILE INCOMPLETE, PREPD ABORTED.
```

Control returns to NCS.

QMLD

QMLD is a QM-1 loader that runs under NCS. QMLD can load mainstore, control store, and nanostore from program files residing on the current USER or SYSTEM packs, thereby initializing and loading the QM-1 for new emulator operation. All files must be in absolute binary format.

The parameters in the QMLD parameter list are positional and are separated by NCS delimiters. As each delimiter is entered, the preceding files(s) is moved into a temporary region of mainstore. The final delimiter entry causes the three storage areas to be loaded from the temporary region of mainstore.

```
Format:  !QMLD△ [*] [$] mainstore-filename [+filename . . .]△
                [*] control store-filename [+filename . . .]△
                [*] nanostore-filename [+filename . . .] △ edit-function△
```

*	specifies clearing of the storage to zero area before loading.
$	specifies that the file to be loaded is in 16 bit binary format from the microassembler (i.e., NOVA format).
+(or &)	specifies that the next file named is to be loaded into the storage region in addition to the first file named. Files are loaded in the order specified.
mainstore-filename	mainstore program file to be placed at 20000_8-33777_8 of mainstore for residence at 0 to 13777_8.
control store-filename	control store program file to be placed at 60000_8-77777_8 of mainstore, for residence at 0 to 17777_8 of control store.
nanostore-filename	Mapped format nanostore file to be placed at 36474_8-57777_8 of mainstore (see MAP command), for residence at 000 to 67423 of nanostore.
edit-function	one of a set of functions that can alter the object code and select loading and execution. The edit functions are defined in Table 9.

edit-function	definition
LD△	load MS 00000_8-13777_8 from 20000_8-33777_8 of MS load CS 00000_8-17777_8 from 60000_8-77777_8 of MS load NS 00000_8-00674_8 from 36474_8-57777_8 of MS
ST△	performs LD function and MICROSTART's at CS(100)
ST($nnnnn_8$)△	performs LD function and MICROSTART's at CS($nnnnn_8$)
GO△	performs LD function and NANOSTART's at NS(0)
GO($nnnnn_8$)△	performs LD function and NANOSTART's at NS($nnnnn_8$)
RANGE nnn△<u>TO</u> nnn△	input NS address range for generating T-step stretch commands. The default is 0-777
T1 T2 T3 T4	stretch the specified T-steps(s) in all nanowords within the current RANGE.
END	terminate QMLD without loading or execution.

Table 9. Edit Functions of QMLD

Error Messages:

If the emulator is locked (see p.4) QMLD will halt instead of starting the loaded programs. Enter Ⓢ S417777 and reenter the QMLD command.

Switches:

If the "PROGRAM STOP" switch on the console is ON when the ST or GO command is given, the QM-1 will HALT before entering the user's program. SINGLE STEP, DOUBLE STEP or MICRO STEP can then be selected by the user.

RESTORE

RESTORE copies to disk a file that has been stored on magnetic tape with the SAVE utility.

```
  Format:   !RESTORE△

            MAG TAPE UNIT NUMBER = unit number△
                                                                        / * \
                           START AT FILE = [ (s) S100400]  file-marks to be skipped{  +  }
                               LOAD FILE = file-name                                -
                                                                        \ △ /
                             DATE OF HDR = date

                              VOL NUMBER = volume

                          START CYLINDER = cylinder

                            START SECTOR = sector

                             FILE LENGTH = length

           HIT (RETURN) TO WRITE DISK  CR
```

unit-number	magnetic tape unit number first unit = 0 second unit = 1
(s) S100400	switch setting to disable header comparison between the saved file and the file definition in the destination directory. The file is written at the address specified in the header.
** CAUTION **	misuse of this option can cause file overwriting.
file-marks-to-be-skipped	number of file marks to be skipped for positioning. There are two file marks after each file. To read multiple files, the starting file numbers would be 0, 2, 4 ....

	* skip past specified number of file marks - do not rewind before or after the read. After the file is read, the next starting file is requested. + rewind, skip past specified number of file marks - do not rewind after the read. After the file is read, the next starting file is requested. △ rewind and skip past the specified number of file marks. After the file is read, rewind again and return to NCS. - return directly to NCS.
filename, date volume, cylinder, sector, length	file attributes obtained from header record of the SAVE'd file.

RESTORE will write to an uninitialized USER if the Ⓢ S100400 option is used. Otherwise, the USER must be initialized with the SYSINIT utility or a disk error message will be displayed.

If the Ⓢ S 100400 option is not used, the header parameters are examined and the destination directory is searched for the filename. If the filename has not been defined, or if the length is too short, or if the file is write protected, RESTORE will ask for an alternate filename. The alternate filename must have been defined with appropriate parameters. The parameter list for the alternate file is displayed followed by

HIT (RETURN) TO WRITE DISK

Error message:

DISK ERROR

Disk is not ready, or has unrecoverable data error.

REWIND

REWIND positions the magnetic tape or the cartridge tape at the loadpoint.

Format: !REWIND△ unit-number△

unit-number	= 0	first magnetic tape unit
	= 1	second magnetic tape unit
	.	
	.	
	.	
	= n	nth magnetic tape unit
	= C0	first cartridge tape unit
	= C1	second cartridge tape unit

<u>.S</u>

.S directs execution of a preloaded NOVA program to begin at a specified location in mainstore.

Format: $\underline{!}$.S $\underline{=}$ location$_8$ $\triangle$

The dot symbol (.) tells the NCS nucleus to load and executes the transient program whose name follows. (Each NCS transient program occupies exactly one disk sector.) Symbol "S" is the name of one of these transient programs. When transient-S is in control, it in turn needs further information. Thus, it issues the equal-sign (=) as a prompter. What it needs is an octal address value followed by a standard terminator.

SAVE

SAVE outputs an NCS file to 9-track magnetic tape. The file can be retrieved and written back to disk by the RESTORE utility.

Format: !SAVE△

MAG TAPE UNIT NUMBER = unit-number△

START AT FILE = $\text{file-marks-to-be-skipped}_8\begin{Bmatrix}*\\+\\\triangle\\-\end{Bmatrix}$

FILENAME = filename△

ENTER DATE (mmddyy) = date△

START AT FILE = $\text{file-marks-to-be-skipped}_8\begin{Bmatrix}*\\+\\\triangle\\-\end{Bmatrix}$

or

!

unit-number	magnetic tape unit number 0 or 1
file-marks-to-be-skipped$_8$	number of physical file marks to be skipped over to position the tape for writing. Each file is terminated with two file marks. To write mulitiple files, the starting file numbers would be 0, 2, 4.... * skip past specified number of file marks - do not rewind before or after the write. After the file is written, the next starting file is requested. + rewind, skip past specified number of file marks - do not rewind after the write. After the file is written, the next starting file is requested. △ rewind and skip past specified number of file marks. After the file is written, rewind again and return to NCS. - return directly to NCS.
filename	name of file to be written. USER file names are prefixed with a slash (/). SYSTEM file names are not prefixed.
date	the date. May be any six characters except △ .

An entire USER can be output with SAVE by defining a file beginning at cylinder 0, sector 0, with a length that overlaps all other files in the USER. The maximum USER length is 22600_8 sectors.

Error Messages:

WRITE PARITY ERROR, RECOVERED - address

SAVE continues.

WRITE PARITY ERROR, UNRECOVERED - address

To continue, enter ⓢ G.

Any entry can be cancelled by entering ESC. The prompt is then re-displayed requesting the entry.

SAVE writes the address and size parameters of the file into a header record which is read and used by the RESTORE utility.

DISK PACK INITIATOR (SYSINIT)

SYSINIT is used to initialize or re-initialize a disk. A new disk must be formatted prior to initialization, as described in Section 3 (HOW TO FORMAT A NOVA DISK).

Initialization of a new disk prepares the directory. Individual files can then be defined using the ADD command of the FILES utility. Files may be placed on disk using the COPY utility. The contents of an entire disk platter (output of the SAVE utility) can be copied onto an initialized or uninitialized disk using the RESTORE utility.

Re-initialization of an existing disk erases all file names and pointers from the directory. Each USER is viewed by NCS as a separate disk and each is initialized separately.

C A U T I O N !

> Improper use of SYSINIT may cause permanent loss of files on the disk. Therefore, the use of SYSINIT is intentionally complicated.

This procedure assumes that the proper USER assignment has been made by invoking USER.

Format: <u>!</u>LD,GPO △

<u>!</u>LD,=SYSINIT=?△

<u>NOVA-CONTROL-SYSTEM-INITIATOR</u>

<u>*WARNING*(UNPROTECTED) CAN DESTROY ACTIVE DISK DIRECTORY</u>

SYSINIT is now in HALT state. Ⓢ L will allow escape without damage at this point.

Ⓢ G

<u>ENTER DEVICE TYPE--</u> $\left\{\begin{array}{l}\text{SYSTEM}\triangle \\ \text{USER}\triangle\end{array}\right\}$ SYSTEM specifies USER 0 initialized. USER specifies current user initialized.

If SYSTEM <u>SYSTEM DISK INITIALIZED</u>

If USER <u>ENTER VOLUME NUMBER (OCTAL)</u> any non-zero number (up to four digits)

<u>USER DISK INITIALIZED</u>

Ⓢ R

The disk platter is now initialized.

The user should now invoke FILES and add a file named DIRVn (where n is the USER#) whose address is 0,0,110. This is the space occupied by the volume header and the directory.

UPDATE

UPDATE, the line-editor utility, performs deletions/insertions of single lines or groups of lines, by line number, while copying sourcecode from one file (or device) to another. A "line" is a string of one to 132 characters containing exactly one CR which must be the final character; or else a string of exactly 132 characters containing no CR (carriage return).

UPDATE uses four files (or devices), functioning in part like the utility COPY with an additional two files (devices). One of these contains input for any "control" items (deletion/insertion commands) and also for any new sourcecode lines to be inserted. The other contains output for a "report" (logging) of all transactions. The old files ("source-in" and "source-out") from COPY are retained and line numbers in reference to either of those files represent implicit positions within each file (device), designated by positive decimal numbers less than 32768. (Zero implies "before first line".)

```
Format:   ! UPDATE△ CTL.IN= control△ LOG.OUT= report△
                [additional dialog if "report" is a disk file]
                    I= source-in△ O= source-out△
                [additional dialog if "source-out" is a disk file]
```

When neither of the output names represents a disk file, there is no "additional" dialog, and the format may be abbreviated, in the way it would be entered into an EX-file:

```
UPDATE△ name 1△ name2△ name3△ name4△
```

where name1 and name3 represent inputs (devices or files) while name2 and name4 represent outputs (devices alone).

Any "name" if it is to represent a device (and not a disk file) must begin with an asterisk (*). Legal mnemonics are as follows:

-for-	Devices (same meanings as for the COPY utility)
name1	*CDR
name2	*CRT , *LPT , *NUL
name3	*CDR , *TTC , *TTI , *TP(u)
name4	*CRT , *LPT , *PCH , *TP(u) , *TTY , *NUL

The "control" file or device (name1) contains lines, each ending with a CR, which fall into two categories per contents of column 1:

-col.1-	Meaning
not ")"	Insertable-sourcecode line, to be copied to the "source-out" immediately upon being encountered.
is ")"	A control-line; three subcategories per column 2:

cols. 1 - 2	Later fields	Meaning (mmm,nnn = decimal integers)
) I	nnn	Copy "source-in" from its present postion up through its line # nnn , to "source-out". The expectation is that insertable lines will follow.

cols. 1 - 2	Later fields	Meaning (mmm,nnn = decimal integers)
) D	mmm-nnn	Copy "source-in" lines, up through its line # mmm-1 , to "source-out". Then read from "source-in" but do not copy (hence, delete) its lines # mmm through nnn. Separators between mmm and nnn can be one or more of any non-digit characters. If nnn=mmm then nnn is superfluous; the user may omit nnn (and the separators).
) *	Arbitrary.	The "*" can be any printable character other than "D" or "I". This control line is treated as a comment line, to be printed only in "report".

The "report" device (or file) receives three categories of line:

(a) Every control line is copied to it, i.e. of the forms

```
)D ....
)I ....
)* ....   .
```

(b) Every line deleted from "source-in" is recorded in "report", with an additional prefixing line number (implicit ordinal position which it has on the "source-in" file or device).

(c) Every inserted line, newly placed in "source-out", is duplicated in "report", with an additional prefixing line number (implicit ordinal position which it will occupy on the "source-out" file or device).

If "report" is a disk file, these items will be recorded in packed-sourcecode format, and may (user option) be appended to prior contents of this file.

As indicated in the initial format summary, if either of the two outputs is to be a disk file then the invoking command line requires additional dialog. (CAUTION: Also be sure that each such output disk-file is write-enabled.)

For the "report" disk file, the additional dialog allows the user to choose between starting a fresh file, or appending to an existing file; either

WANT TO APPEND NEW LOG.OUT TO OLD?(Y,N) N

or

WANT TO APPEND NEW LOG.OUT TO OLD?(Y,N) Y

LOG.OUT FILE PRESENT CONTENTS OCCUPIES ww SECTORS, LESS xx BYTES.

LOG.OUT FILE EMPTY SPACE AVAILABLE IS yy SECTORS, LESS zz BYTES.

DO YOU STILL WANT TO APPEND NEW LOG.OUT TO OLD? (Y,N) $\left\{\begin{matrix}\text{Y}\\\text{N}\end{matrix}\right\}$

For the "source-out" disk file, the additional dialog allows users to choose packing or non-packing of sourcecode:

COMPRESS THE OUTPUT? (Y,N) $\left\{\begin{matrix}\text{Y}\\\text{N}\end{matrix}\right\}$

Examples of special usage; here ZILCH = a totally-empty file:

(a) Pack an unpacked file, with no insertions/deletions.
UPDATE△ ZILCH△ *NUL△ <unpacked-file>△ <newfile>△ Y

(b) Unpack a packed file to view on lineprinter (CRT similarly).
UPDATE△ ZILCH△ *NUL△ <packed-file>△ *LPT△

(c) Merge two short files (one to totally precede the other).
UPDATE△ <firstfile>△ *NUL△ <secondfile>△ <mergedfile>△ $\left\{\begin{matrix}\text{Y}\\\text{N}\end{matrix}\right\}$

(d) Find out line numbers within existing file, for later run.
UPDATE△ <filename>△ *LPT△ ZILCH△ *NUL△

USER

USER assigns a specified user as the removable pack.

Format: !USER△n

n = 1 through 9	the corresponding user pack is assigned as the removable pack.
n = 0	The number of the currently assigned user pack is displayed.

To change USER assignments, the system sense switch setting must include bit 17 (the high order bit). If an attempt is made to change USER's while bit 17 is not set, the following message is displayed:

USER n LOCKED

where n is the currently assigned USER.

To unlock a USER, change the switch setting to set bit 17. For example:

Ⓢ S $4nnnnn_8$

Example:

!USER 4	assigns USER 4 as the removable pack.
!USER 0 IS 4	shows the current removable pack assignment.

XC

XC, the "crosscopy" utility, is restricted to disk files which contain an equivalent 16-bit-per-word format.

This utility program violates the restriction honored by most other utilities, of accessing only disk files on the SYSTEM pack and on the single USER pack most recently selected.

Format: !XC[-options]△

```
        CROSSCOPY (DISK TO DISK). -USE 16BIT/WD FILES ONLY-
        INPUT FILE:
         DISK-UNIT # (0,1):     u1△
         VOL.# (OCTAL,0-max):   v1△
         FILENAME =    name1△
        OUTPUT FILE:
         DISK-UNIT # (0,1):     u2△
         VOL.# (OCTAL,0-max):   v2△
         FILENAME =    name2△
        [further dialog before termination]
```

Parameters:

u_1 , u_2	-	single digit currently limited to 0 or 1.
v_1 , v_2	-	one-digit or two-digit octal value, in the range (0, 11) for full NCS packs, or in the range (0,4) for "split packs".

name_1, name_2	-	one to ten-character name; must exist in the directory of the respective Unit/Volume just selected. NOTE: There is no slash (/) preceding these names, as the Volume number suffices to determine whether the USER pack or the SYSTEM pack is intended.

Options:

U - Unpack the input file: the disk-read subroutine will look for any byte appearing to represent packed sourcecode character-repeat counts and will replicate the characters accordingly. Therefore, do not use option-U if copying binary files.

P - Pack the output file: the disk-write subroutine will look for any repeated bytes and will output instead to the disk the packed-sourcecode representation for such bytes. Also, any embedded null-bytes (but not the final padding null-bytes) will be dropped under this option. Therefore do not use option-P if copying binary.

V - View the copying process: the file contents, while being buffered line-by-line, is also displayed on the console screen. Binary files would probably garble the screen but should do not real damage here.

Further Dialog Before Termination:

1. The symbol [**], used below in two places, represents a report of byte-tallies which will occupy this 4-line format:

```
INP.FILE   USED:  ww1 SECTORS, PLUS  xx1 BYTES.
         UNUSED:  yy1 SECTORS, PLUS  zz1 BYTES.
OUTP.FILE  USED:  ww2 SECTORS, PLUS  xx2 BYTES.
         UNUSED:  yy2 SECTORS, PLUS  zz2 BYTES.
```

2. If copying completes successfully, the Final Dialog is of the form

```
        [**]
```

3. If the output file overflows, Final Dialog is of the form

```
?? ENDFILE ON OUTPUT -- COPY INCOMPLETE
        [**]
DO YOU WANT TO CONTINUE FOR FULL BYTE-COUNT?
(Y,N)?
```

The program pauses for response; any response other than "Y" or "N" causes the last "(Y,N)?" to be repeated. Response "N" causes immediate termination, and XC exits to NCS. Response "Y" causes a continuation of input-file reading in order to compute the full output-file size required. The above last line, and continuation thereafter, will have the format:

```
(Y,N)?   Y  ...STANDBY, PLEASE...
```

[here the input file resumes reading in; and if option-V was selected then the CRT screen continues to display sourcecode lines or other contents of the input file.]

```
TO AVOID OVERFLOWING,

  YOUR OUTP.FILE SIZE MUST BE (AT LEAST)..  ww2 SECTORS, PLUS  xx2 BYTES.
```

ZAP

ZAP clears NOVA user memory. The NCS nucleus remains intact.

Format: !ZAP△

* and $

* and $ are NCS program file manipulation utilities.

Format:

$$\underline{!}\left\{\begin{matrix} * \\ \$ \end{matrix}\right\} \text{filename}$$

* loads an invokable NOVA program file but inhibits execution. Execution is then specified by the .S utility.

$ writes a program core image from core to disk. The PROGR attributes of the file select the area of core to be moved to disk.

In both cases, control is returned to NCS.

SECTION 3: MISCELLANEOUS PROCEDURES

This section includes procedures for performing tasks that require the use of more than one of the utilities described in Section 2.

HOW TO PREPARE AN INVOKABLE NOVA PROGRAM

NOVA programs can be invoked and executed by typing the program file name once the program has been reformatted by LD to a NOVA format, and written to disk.

A NOVA source program is assembled by the microassembler to produce a binary output file. This binary output is in the QM-1 format and is not executable. The binary is loaded into mainstore by LD which reformats it as the loading takes place. The reformatted binary is now executable. The executable core image is then written to disk using the $ command.

The NOVA program can now be invoked by using the program file name.

Example:

```
!MA,DEF =,INPT =MYSOURCE,BIN = MYBIN.    (MYBIN must be write-enabled.)
!ZAP.
!LD,MYBIN.
!$MYPROG,Y                               (MYPROG must be write-enabled.)
!MYPROG. now will load and execute the program MYPROG.
```

MYPROG must have been declared using the FILES utility, specifying the appropriate program parameters.

Example:

```
!FILES△

 FILE DIRECTORY UTILITY, ENTER DATE (MMDDYY) = 041577△

*VOLUME 0△

*ADD MYPROG ADR = 600,0,10 : TYPE = WRITE EXEC PROGR : PROGR = 100,2400,300: GO

*END OF JOB
```

HOW TO FORMAT A NOVA DISK

Blank (or scratch) disks packs must be formatted for use in NOVA systems before they can be initialized.

1) Mount the pack to be formatted on the N9755 disk drive.
2) Mount the standard I/O diagnostic cartridge in the cartridge unit.
3) Load the N9755 disk diagnostic, using the F-switches to select the track and file on which it resides.
4) Respond as specified to the following prompts.

? CHn△	specifies the disk channel as n.
? HC24△	set head count to 24_8 to format entire disk pack.
? INT△	request initialization.
3 SECTOR MODE? N	N is for no. NOVA requires a 24 sector mode.
STAGGERED ADDRESSING? Y	Y for yes. Provides faster operation.

When formatting is complete, this message is displayed:

TEST COMPLETE

Verification of the surface can be done by writing the entire disk and reading it back. The writing is commanded thus:

? WTA△	requests the entire disk be written.
BYTE MODE? N	
ODD START? N	condition parameters
DATA CHAINING? N	
DMA? Y	

After the disk has been written, this message is displayed:

```
TEST COMPLETE
```

The reading-back function is commanded thus:

```
? RDA△                              requests the entire disk to be read.

BYTE MODE? N

ODD START? N

DATA VER? Y                         condition parameters

DATA CHAINING? N

DMA? Y
```

after the disk has been read, this message is displayed:

```
TEST COMPLETE
```

The disk has now been formatted for NOVA use, and all surfaces have been verified. The USER's can be initialized and established.

HOW TO LOAD USER 0 (SYSTEM PACK)

After a disk has been formatted, the system software must be loaded into USER 0. This is done by transferring the system from a disk containing the desired system to magnetic tape using the SAVE utility, and then transferring it to the new pack with RESTORE. Since the new pack does not yet have any system software, the old pack must remain mounted until the RESTORE utility is ready to write to the new pack.

The USER 0 filename is ALL.

To write this file to an uninitialized USER (such as USER 0), the NOVA switches must be set to 100400 as described in the RESTORE descriptions.

Example: ALL (which overlaps the entire system USER) was written on file 0 of a magnetic tape with the SAVE utility.

```
    !RESTORE
    MAG TAPE UNIT NUMBER = 0.
    START AT FILE = ⓢ S100400 0.
    file attributes are displayed followed by the
    HIT (RETURN) TO WRITE DISK
        *** before the carriage return, mount the new pack ***
```

The switches must be reset to NOVA setting (e.g. 417777) after the restore.

After USER 0 has been established, the other USER's can be established using the system software in USER 0.

APPENDIX A: STAND ALONE PROGRAMS

This appendix includes a description of the function and procedure for using several stand alone programs that are useful in conjunction with the operation of NCS.

The following table lists the programs and briefly describes their functions.

Program	Function
COPYDD	Copies one USER to another on the same or different pack.
LSP	Provides loader and primitive debug capabilities.
STAND ALONE DUMP	Dumps mainstore or control store selectively.
STATESAVE	Saves the state of the QM-1 in control store.

Table A-1: Stand Alone Programs.

COPYDD

COPYDD is a stand-alone program that copies one disk to another, one USER to another on the same disk, or one USER to the same or different USER on a different disk drive. COPYDD is booted into control store from track 3 of the standard I/O diagnostic cartridge or from a USER cartridge prepared as explained in the last half of this writeup. Or, from disk via MC, START, ???LD COPYDD.

```
Format:  ?ST△
           SOURCE DEVICE TYPE = device-type△
           DEVICE ID = channel-&-id△
           USER NUMBER (SOURCE) = user-number₁₀△
           USER NUMBER = unit△
           DESTNATION DEVICE TYPE = device-type△
           DEVICE ID = channel-&-id△
           USER NUMBER(DESTINATION) = user-number₁₀△
           UNIT NUMBER = unit△
```

device-type	type of disk format. 9750 2 USER NCS format 5550 9750 on a 9755 controller 9755 NCS split extents, 10 USER's (old format) 55 NCS non-split extents on a 9755 360 2-sector mode, 2 USER's Any other entry causes a redisplay of the prompt.
channel-&-id	disk channel number and ID (e.g. 312)
user-number_{10}	USER number for selective disk transfers. 0 through 9 transfer to/from specified USER } if type 9755; 0-1 } if type 360 > 10 transfer to/from an entire disk } if type 9755; > 1 } if type 360
unit-number	disk unit number (0-3)

When the destination device unit number has been entered, the transfer begins. When the transfer is complete, the prompt character ? is displayed.

Errors

If a disk transfer error occurs, there are two recovery options.

1) press the SPACE bar and the transfer will be tried again.
2) press any other key and the operation will terminate. The prompt character ? returns, and ST will start the request for transfer parameters again.

To discontinue any command string, press ESC.

Procedure for preparing a COPYDD file on cartridge tape.

COPYDD:S is the source program in microcode residing on USER 7 of the NCS disk.

1) Assign USER 7 as the removable pack.
2) Define a file for the binary form of COPYDD on the system pack, USER 0.
3) Assemble COPYDD:S with the microassembler using SYSDEF6 as the definition file.
4) PREP COPYDD onto the cartridge tape using MULTI6 as the nanosystem, and the binary file for COPYDD plus /M2DSPLAY:B as the micro binary files. Use 100_8 as the start address.

COPYDD can now be booted into control store using the F-switch settings to select the track and file.

LSP

LSP is an interactive Loader Services Program that provides selective loading and debug capabilities in a cartridge tape environment.

LSP resides in 2400_8 words of control store. It is relocatable and can be loaded at any user-specified loadpoint, allowing it to be loaded in addition to other programs already in control store or to be loaded by LSP.

LSP is capable of:

1) communicating with a user through the system console. It accepts and executes the various commands entered by the user.
2) loading programs from the cart tape unit.
3) stretching nanostore T's as nanostore is loaded from a cart tape file.
4) writing directly into control store, mainstore, and nanostore.
5) displaying control store and mainstore on the system console.
6) direct execution of a loaded program via a microstart or nanostart.

This writeup is devided into two procedures. The first explains how to use LSP, and the second explains how to prepare a cart tape file containing LSP.

How to Use LSP

LSP is booted into memory from the PREP'd cart tape file using the F-switch setting to select the track and file. When loaded, the LSP prompt ??? is displayed on the CRT.

Once started, LSP remains in control until the user gives a command which either passes control to an already loaded program or loads a program and passes control to it.

LSP retains control after loading a file from a cart tape when either of two conditions is encountered:

1) A physical end of file is read from the cart tape.
2) A start record is encountered on the cart tape and the G parameter of the LD or LU commands was not specified. In this case, LSP saves the address supplied in the START record for a future GO command.

The LSP command set consists of 21 commands. The following table lists the commands and their functions.

C	Display control store	LD	Load and modify
CC	Clear control store	LU	Load unmodified
CM	Clear mainstore	M	Display mainstore
CN	Clear nanostore	MS	Microstart
DC	Display control store	NS	Nanostart
DM	Display mainstore	R	Stretch T's in range
DS	Define stretch patch	SC	Set control store
EC	Set control store	SM	Set mainstore
EM	Set mainstore	SN	Set nanostore
EN	Set nanostore	T	Specify T's to be stretched
GO	Microstart		

Table A-2: LSP Command Set

The LSP command format consists of a command mnemonic followed by a set of parameters that selectively define the command. The parameters are separated by a delimiter which is any character other than an octal digit. In this document, the slash is used to indicate a delimiter. The first delimiter following a command mnemonic is optional, but if entered, must be a slash.

The command terminator is one of three characters: blank, carriage return, or period. The terminator is indicated by a $\triangle$ in this document.

Characters are entered into the command buffer as they are keyed in. Errors can be corrected by backspacing (pressing the backspace key) to the error position, making the correction, and spacing forward (pressing the space bar) to the end of the line. The space is a terminator only if entered at the end of a command. If entered in the middle of the command string, the space skips over the character already entered and leaves it unchanged in the buffer. ESC cancels the current command string and clears the display.

All addresses and numeric values are in octal, both on input and output.

After each command is executed, the LSP prompt ??? is redisplayed.

The following table lists the commands, their formats, and their functions.

Command	Function
???[D]C[/]start-address/number-of-words△	Display control store beginning with start-address and continuing until the number-of-words has been displayed, 8 words per line.
???[D]M[/]start-address/number-ofwords△	Display mainstore in the above format.
???CC[/]start-address/end-address/value△	Set all control store locations from start-address through end-address to the specified value.
???CM[/]start-address/end-address/value△	Set all mainstore locations from start-address through end-address to the specified value.
???CN[/]start-address/end-address/value△	Set all nanostore locations from start-address through end-address to the specified value.
???$\left\{\begin{matrix}\text{EC}\\\text{SC}\end{matrix}\right\}$[/] start-address/value$_0$ [..../value$_k$]△	Set consecutive control store locations beginning with start-address through start-address ± k to the values specified by value$_0$ through value$_k$. If the delimiter separating the address from the first value is a plus or minus sign, the EC or SC command is redisplayed with a new start-address ± k+1. The redisplayed command may be completed with new values to continue setting locations. This process continues until the delimiter is changed to something other than plus or minus or until the escape key is used to cancel the command. Using the minus sign as a de-

limiter causes the data to be stored at descending addresses. A total of 39 characters can be input for values and their delimiters. Any additional characters are ignored.

$\underline{???}\left\{\begin{matrix}\text{EM}\\\text{SM}\end{matrix}\right\}$[/] start-address/value$_0$ [..../value$_k$]△

Set mainstore locations as in the EC/SC commands.

$\underline{???}\left\{\begin{matrix}\text{EN}\\\text{SN}\end{matrix}\right\}$[/] start-address/value$_0$ [..../value$_k$]△

Set nanostore locations as in the EC/SC commands.

$\underline{???}$R[/]start-address$_1$/end-address$_1$ [..../start-address$_k$/end-address$_k$]△

Define the ranges of nanostore addresses in which the stretch bit will be set in selected T-vectors when the words are loaded. The addresses are nano <u>word</u> addresses, not word-byte addresses.

$\underline{???}$T[/]t$_1$ [..../t$_4$)△

Define the T-vectors in which the stretch bit is to be set in words loaded within the ranges specified by the R command. Only values of 1, 2, 3, or 4 are allowed for t_n.

$\underline{???}$DS[/]address$_1$[..../address$_n$]△

Define stretch patch. When any of the specified addresses are loaded from the cart tape, the stretch bit in the corresponding T-vector will be set. Only addresses xxxxØ7, xxxx13, xxx17, and xxxx23 are valid. Other addresses are not processed, and no error indication is given. This command sets stretch bits independently of the R and T commands.

$\underline{???}$LD[/]track/file/unit [/G]△

Load the cart tape file specified. Stretch bits will be set in all T-vectors specified by the R and T commands or by the DS command. If G is entered, pass control to the loaded program when a start record is encountered.

???LU[/]track/file/unit [/G]△	Load the cart tape file specified. No stretch bits are set. If G is entered, pass control to the loaded program when a start record is encountered.
???GO△	Perform a microstart as specified by the last microstart record read from the cart tape file.
???MS[/]address△	Perform microstart at address specified.
???NS[/]address/FIDX-setting△	Perform nanostart at address specified, with FIDX set as specified. Address is a nano word address, not a word-byte address.

Table A-3: LSP Command Description

How to Prepare a Cartridge Tape Containing LSP

LSP is a relocatable program that can be loaded at any loadpoint specified in the PREP utility. The microstart array starts at the loadpoint.

The following procedure uses a loadpoint of 20000_8 as an example of offsetting the loadpoint to an address other than zero.

1) Invoke PREP.

2) Using the H command, put the header record on the tape.

3) Using the M command, offset the LSP binary code to the required loadpoint.

```
#M MICRO BINARY FILE = LSP:B[+20000]△
```

Steps 4 and 5 are performed only if the loadpoint is offset.

4) Using the C command, store the value of the loadpoint + 100_8 in the location of the loadpoint + 33_8.

```
#C PRESET CODE (NANOSTORE, CONTROLSTORE, MAINSTORE = C

FROM 20033△ TO 20033△ VALUE 20100△
```

5) Using the C command, store the value of the loadpoint in the location of the loadpoint + 35_8. This serves as the base address of the program.

```
#C PRESET CODE (NANOSTORE CONTROLSTORE, MAINSTORE) = C

FROM 20035△ TO 20035△ VALUE 20000△
```

6) If a system console baud rate other than 9600 baud is desired, then using the C command, place the new baud rate (exactly as it occurs in the DCWA of the asynchronous communication controller) at the location of the loadpoint +17_8.

7) Using the S command, create a microstart record with a start address indicating the loadpoint.

```
# START CODE (HARDSTOP, NANOSTART, MICROSTART) = M

  START ADDRESS = 20000△
```

Any other programs may precede or succeed LSP on the cart file. However, no control store routines should overlap the address space of LSP. The microstart record should point to LSP's microstart array located at the loadpoint. Control can be transferred to the other routines via the MS (microstart command) of LSP.

STAND ALONE DUMP PROGRAM

The stand alone dump program dumps the contents of a specified range of control store or mainstore to the line printer. Memory addresses and contents of memory locations are output in octal format, six octal digits per word, 8 locations per line.

This dump program is stored on cartridge tape and is booted directly into memory. It usually occupies control store 5200_8-5777_8.

1) Insert the cartridge tape containing the stand alone dump program into the tape unit.

2) Set the F switches to select the track and file containing the dump program.

3) Press MASTER CLEAR and START to load the dump program. When loaded, the cursor moves to the beginning of the next line on the CRT and displays #.

4) Type in the boundary addresses (in octal) of the memory being dumped, followed by M or C to select either mainstore or control store. A carriage return instead of M or C causes a default to control store, or the last entered.

   Format: # $\text{nnnnnn}_8 \triangle \text{nnnnnn}_8 \left\{\begin{matrix} \text{M} \\ \text{C} \\ \triangle \end{matrix}\right\}$

   $\triangle$ is any QM-1 delimiter (hyphen, comma, blank, period).

   Leading zeros may be omitted.

After M, C, or $\triangle$ is entered, the contents of the specified memory range is output to the printer. A new set of boundary addresses may then be entered. As many dumps as desired may be taken from either mainstore or control store by typing in a new set of boundary addresses after each output to the printer. Printing may be interrupted at any time by entering an Escape character (ESC).

STATESAVE PROCEDURE

STATESAVE saves the state of registers in control store locations 200-312. Once saved, the stand-alone dump or LSP can be used to display the state.

1) Set the F-switches to 77.

2) Set the MODE switch to IML.

3) Press MC.

4) Press START.

The state is now saved. The following table shows the control store locations used to store each register. Reserving locations 200-312 of control store as non-program space, allows STATESAVE to be executed without loss of information. E-store 7 is destroyed and displays the contents of local store 37.

Controlstore location	Register		
200-237	local store		0-37
240-277	external store		0-37
300	FMIX	FMOD	FCIA
301	FAIL	FCID	FAIR
302	FCOD	FAOD	FSID
303	FSOD	FEID	FEOD
304	FEIA	FEOA	FACT
305	FUSR	FMPC	FIDX
306	FIST	FIPH	GØ
307	G1	G2	G3
310	G4	G5	G6
311	G7	G10	G11
312	G12	G13	Ø

Table A-4. Statesave Storage Locations