MAXC OPERATIONS

Maxc 18.1

May 29, 1974

by

Edward R. Fiala
Charles M. Geschke
Paul Heckel
Edward Taft

Xerox Palo Alto Research Center
3180 Porter Drive
Palo Alto, CA 94304

1. Introduction

This document describes many of the commonly used procedures for Maxc operation, as well as a number of uncommon procedures used during system debugging and maintenance. Also included are the complete operating instructions for several pieces of software written at PARC and unique to Maxc; these include Midas, NVIO, DMPLD, and Micro-Exec.

"Maxc Operations" is intended primarily as a reference document for system personnel. However, in the absence of system personnel, any user should be able to restart Maxc from a Tenex crash using the procedure outlined in Section 14.

Conventions used in this document: In examples in which user typein is intermixed with machine typeout, the user typein is underlined. <cr>, <lf>, <sp>, and <esc> are used to stand for the carriage return, line feed, space, and escape keys.

2. Overview of the Maxc System

From a user terminal, Maxc looks like a PDP-10 running the Tenex operating system. However, this is mostly an illusion. Maxc is really a PARC-constructed microprocessor which emulates the complete user-mode PDP-10 instruction set, as well as many of the privileged instructions and the BBN pager operations. The Tenex system has been considerably modified to account for the many differences between Maxc and a PDP-10, particularly in the area of input/output.

The system consists of the following major components:

A. The Maxc microprocessor itself. Besides performing PDP-10 emulation, Maxc also directly controls the disks.

B. A 256K word by 48 bit MOS memory system, with hardware built in for correcting single bit errors in any word and detecting (though not correcting) double errors.

C. Eight disk drives, each connected to its own controller inside Maxc.

D. A Data General Nova, which has all the other Maxc peripherals connected to it, including the data line scanner (DLS), two magtape controllers with a total of four drives, the IMP interface supporting Maxc's connection to the ARPA Network, and two MCA interfaces supporting connections to all other Novas at PARC.

The Nova also performs a number of other important functions relating to system initialization and debugging. It has direct access to Maxc's main memory (through a separate memory port) and indirect access to all of the internal microprocessor registers and memories. The Nova can load microcode into Maxc's writeable control store, and can command Maxc to start and stop execution, single step, and perform a number of other useful operations.

The Maxc system operates under the control of a large complex of support software, most of which is described in this document. These programs are described here, in approximately the order they would be used if one were to bring up the system from scratch.

When first booted from switches, the Nova runs under the control of DOS, the Data General Disk Operating System, which is described in the pertinent Data General documentation. The Nova has its own fixed-head disk, of quite insufficient capacity, which is used for storage of Nova system programs and other important files such as the Maxc microcode. DOS is also able to perform I/O to magtape unit 0 (the one directly over the Nova itself).

Assuming the necessary files have been loaded onto the Nova disk, the next program to be run is Midas. Midas is the loader and debugger for Maxc microprograms. It has a large number of commands for examining and changing microprocessor registers and memories and for starting, stopping, and single-stepping the microprocessor. Most of its capabilities are used only during debugging; in the normal course of events, it is used only to load the PDP-10 microcode into Maxc, and it then passes control to NVIO.

NVIO is the program in control of the Nova while Tenex is running on Maxc, and contains all the necessary I/O drivers for communicating between Maxc and the peripherals connected to the Nova. NVIO also has a command decoder, by means of which the operator may examine and change memory, start and stop the Maxc microprocessor, and perform a number of other operations.

One important NVIO operation is to boot into Maxc main memory and start a PDP-10 program called Micro-Exec. Micro-Exec has commands for performing a large variety of stand-alone Maxc operations, such as setting up the disk configuration, copying disk packs, and (most important) loading Tenex from disk and starting it.

Most of the rest of this document is devoted to explaining the above operations in much more detail.

3. Power Up

A. First look over the system and make sure it is put together. Ignore scope probes dangling from pins but at any other appearance of inoperativeness, give up or get some knowledgeable person to help.

B. Make sure the proper disk packs are mounted. The labels are on the top of the pack. Compare them to the chart on the wall of the control room. 3-by-5 cards with the currently mounted pack numbers are supposed to be posted on each drive. Other packs are in a yellow cabinet in the machine room and should also have cards identifying them.

C. Turn on the disk units. Two switches need to be turned on for each unit. One of them is in back and inside the cabinet and is labelled "AC power on". It should be turned on first. (Normally it is left on.) Then turn on the obvious switch on the front panel. Two minutes will elapse before the green light goes on. The read-only switches on the control panel should also be off.

* Note that if a unit is selected, the read-only restriction will remain in force until it is deselected. This is indicated by the read-only light on the panel. Don't worry if the read-only light remains on, since the unit will be deselected by firmware later on and the read-only light will go off at that time. However, be sure the read-only switch is in the off position.

D. Turn on the Nova power, Nova magtape unit 0 power, and Nova disk power. (Normally these are left on so you won't have to do this.) The switch for the Nova disk unit is under the little table in front of the Nova.

E. Turn on the Maxc microprocessor and memory as follows: Plug in the memory fans in back of Maxc. Then turn on the microprocessor fan, two microprocessor power supplies, and four memory power supplies in front of Maxc.

4. Loading the Nova Disk

A. Get the current "10SYS" tape from the tape rack and mount it on unit 0. Note that the tape unit directly above the Nova should be set to unit 0 and the one to the left of that should be set to unit 1. Unit 1 is a 7-track drive appropriate for mounting any Tenex minidumper tapes you may have. Unit 0 is a 9-track drive used for Nova tapes. Set the Nova console switches to 100022. Make sure the tape drive(s) are in remote and at the load point(s).

* It is customary to hang the tape jacket on the door of unit 1, and for novices to remove the write ring from tapes.

B. Push RESET - then PROGRAMLOAD on the Nova front panel. It should ask:

```
FULL(0) or PARTIAL(1)?
```

and you should type 0 which will clear the Nova disk directory, read-in a fresh copy of Nova DOS from magtape unit 0, and rewind the tape.

*Note that the Nova tape on unit 0 must be at the load point for this to work.

C. There will be sufficient Nova disk storage to keep a copy of Nova DOS on the disk only if you don't plan on doing any microcode patches while running the system and if you aren't concerned with file storage for any other reason. To do this, type the following:

```
XFER MT0:1 NSYS.SV; CHATR NSYS.SV SPW; INSTALL NSYS.SV<cr>
```

Then set the Nova's console switches to 100020. The effect of this is to cause a quicker disk restart when you push RESET-PROGRAM LOAD-CONTINUE on the Nova front panel switches. If you don't do this, leave the switches at 100022. Both kinds of restart are described under the "warm starts" section later.

D. You will now need to load files from tape unit 0. Normally this is accomplished by loading all the files from the most recent SYSF and CHANGEF records, i.e. the ones with the highest numbers. The tape jacket label directs you to these records with some comment such as

```
0 TBOOT
1 NSYS
2 BASEF
3 SYSF
4 CHANGEF
 ...
7 CHANGEF
```

Loading is accomplished by typing:

```
LOAD/A/V MT0:(i,j) <cr>
```

where i and j are the numbers of the most recent (highest-numbered) SYSF and CHANGEF records as indicated on the tape jacket. This will result in loading onto the Nova disk all of the files used by the system. The names of the files will be typed out as they are loaded. Make sure there are no read errors typed out while loading.

When you are done loading type

```
RELEASE MT0<cr>
```

to rewind the tape. The tape must be at the load point for a subsequent warm start if NSYS.SV has not been loaded. Normally NSYS.SV is loaded and the tape should be replaced on the tape rack.

5. Contents of the 10SYS Tape

BASEF record (only needed for debugging -- files may vary)

DG01.MB	RDG01	DGMR.MB	RDGMR
DGP.MB	RDGP	DMPLD.SV	MTAPE.SV
DGIMH.MB	RDGIMH	MICRO.SV	TEST
DGIML.MB	RDGIML	OMICRO.SV	LDSYS
DGRL.MB	RDGRL	EMICRO	BASEF
DGI.MB	RDGI	DBEG.MB	MAINM
DGM.MB	RDGM	MAINI.MB	

Current SYSF record (must be loaded)

STOP.SV	GO.SV	MIDAS.SV	BRK.MB
EDIT.SV	TENLOAD	TENGO	UNBRK.MB
TCON	SYS.MB	SYS.ST	SYSF

Current CHANGEF record (must be loaded)

PATCHES	PATCHES.MB	NVIO.SV	CHANGEF

There may be other files of interest to you on Nova tapes. You can load a file FOO from any record n of the tape by typing

```
LOAD/A/V MT0:n FOO<cr>
```

However, there is a limited amount of storage on the Nova disk, so you should not load more than one or two small programs at a time.

The Nova command

```
DISK<cr>
```

should show 180 or more free disk blocks (300 or more if you plan to change the microcode) when you are done loading. This storage will be used by Nova DOS's program overlay feature when you run the system.

Note: Even though you begin operation with this number of free disk blocks, over 150 of them will disappear when you run the system. These blocks will be reused if you have to warm-start the system as described later, so don't worry about these disk storage numbers except during cold start.

6. Loading the PDP-10 Emulator

After loading all relevant files from the 10SYS tape, dismount and put away the "10SYS" tape. Then type

```
GO; MIDAS TENGO<cr>
```

Running the GO program may be omitted if the memory is already going (see below about memory lights), and it probably causes less wear and tear on the memory to avoid GO in this case.

The GO program turns on power supplies and configures the memory. It usually types out a moderate number of memory errors on the Maxc console. Do NOT worry about errors of type 1 or 4 since they are correctable. Any other error type may be serious and a hardware maintainer should be contacted immediately if a second attempt to run GO produces similar results. Periodic memory maintenance replaces bad chips in the memory before double errors develop. However, occasionally GO goes off the deep end and types reams of errors (more than 10). If it does this, type control-A. This gives control to the Nova debugger. Open location 411 and insert a -1 by typing:

```
411/ ISZ @MAINT+60 -1<cr>
```

Then type control-A and try again.

After GO runs, Midas will run through the TENGO command file which loads the microcode, displays some junk on the Infoton, and starts up the system. This takes several minutes.

After loading the microprocessor, TENGO resets the machine (21;G) and checks the microcode (25;G). The microprocessor may crash (IMA=20) if there were any failures detected during loading. LM 10 will contain the address of the clobbered word (if only one word was clobbered). Note the values of LM 10 and IMA in the log and call system maintainers if the microprocessor crashes. If you cannot get the system maintainers, get out of Midas by typing control-A and repeat MIDAS TENGO, since hardware flakiness has been known to cause loads to fail spuriously and you may be lucky on the second try. If the load succeeds, then it boots in Micro-Exec from one of the Maxc disks and the console terminal types out:

```
Micro-Exec, Ver <version identifying string>
*
```

Micro-Exec immediately executes an automatic "Go" command, which causes Tenex to be loaded from disk and started. Further procedures are described in the next section.

If it is desired to load the microcode without starting NVIO or Micro-Exec, use the TENLOAD command file instead of TENGO; i.e. type:

```
        GO; MIDAS TENLOAD <cr>
```

TENLOAD is identical to TENGO except that it does not start NVIO, Micro-Exec, or Tenex.

7. Starting Tenex

The current Micro-Exec normally has been setup to have the current disk configuration. If you doubt this, type:

```
        P D C <cr>
```

and it will print out the current disk configuration. Compare this to the chart on the wall of the control room. If you are confident, type:

```
        GO<cr>
```

which will boot in Tenex from the disk and start it running.

Tenex will ask for date and time. (The prompt specifies the format.) Please be careful to enter this correctly. If Tenex asks you to reconfirm your typein, you probably blew it and you should hit "Del" and try again. (But if you are really sure you typed the correct date and time, confirm with carriage return).

Tenex then runs the Bsys and Checkdsk programs to verify the consistency of the file system. This takes about ten minutes (the time is proportional to the number of files in the entire system), at the end of which is broadcast the message "Tenex in operation" if all is well. If any uncorrectable errors have been detected, the message "Tenex not available: Disk needs fixing" will be broadcast. If this occurs, you should attempt to notify system personnel. As a last resort, consult Section 22 for information on fixing errors of this sort.

8. NVIO and ODT

NVIO is a Nova Input-Output package which does all the Input-Output for Maxc except for the disk. Specifically it does Input-Output for the Data Line Scanners, the IMP (ARPA Network), the MCA (Multiprocessor Communications Adaptor), and the Tape Units. It also has an octal debugger called ODT (Octal Debugging Tool) which will allow the user to change memory locations in Maxc and the Nova, and perform other control functions.

8.1. Calling NVIO

NVIO can be called from Midas with the command:

```
        6,NVIO;T
or:
        !NVIO.SV/a/b ... <unitno><cr>
```

where /a/b etc. represents switches (of the usual Nova DOS type); and <unitno>, if it exists, is of the form 'Un' for n=0 to 7. The NVIO startup procedure is determined by the setting of these switches, the function of each of which is now defined -- in the order in which they are tested to avoid any ambiguity caused by multiple switch settings. Each of the switches has an analogous ODT command with the same (letter) name.

/B	Micro-Exec is Booted from Save Area 1 of disk unit n (from the 'Un' argument), where 0=A, 1=B, ... 7=H. Unit A is used if no unit is specified.
/S	Tenex is Started up, first booting Micro-Exec from disk as for /B and then executing an automatic "Go" command.
/E	The Maxc Micro-Exec is brought in from tape unit number n (from Un) or from unit 0 if no unit is specified.
/P	Protect mode in ODT is set. This means that ODT will allow only those commands which examine (but do not write into) memory and that do not change machine state.
/D	The Nova Debugger will be given control. ODT Control can be resumed by typing $P.
/H	Normally NVIO then sends a proceed to Maxc which was halted before NVIO was called. This proceed can be suppressed by using the /H switch.
/W	NVIO will Wait for an IORESET before proceeding.

Normally, NVIO is initially started up with the /B or /S switch set, in order to load Micro-Exec and possibly start Tenex. When it is desired to start NVIO without resuming Maxc (e.g. to examine Maxc main memory or start at an alternate address), the /H switch should be used.

8.2. ODT Commands

When NVIO is functioning, ODT (Octal Debugging Tool) is its highest priority process (only interrupts take precedence). It has two heralds, a number sign (#) which indicates that only commands that examine memory are legal, and a colon (:) which indicates all commands are legal.

If Tenex is running and no ODT commands have been typed for ten seconds, NVIO will continuously display and update the addresses and contents of five Maxc main memory variables. If you start typing on the Infoton, NVIO will stop doing this and will print the appropriate herald followed by your typein on the next line.

ODT commands consist of a single letter or other special character, optionally preceded by a numeric argument. The argument is an octal number of 16 or 40 bits (depending on the command) or an expression made up of such numbers and the operators "+" and "-". "." stands for the current (most recently displayed) memory address.

Many NVIO commands request confirmation with "...OK". The confirming character is a period.

n]	Change the current machine to the Nova; change the current location to n; and print out the current location.
n[	Change the current machine to Maxc; change the current location to n; and print out the current location in the format: aa bbbbbb cccccc where: aa (0 to 17) is the last 4 bits of the 40 bit Maxc memory word, it is not printed out if zero; and bbbbbb and cccccc are the left and right half 18 bits of the PDP-10 (Maxc) word.
n/	Change the current location to n; and print out the current location.
/	Print out the current location.
n<cr>	Put n into the current location. If the current machine is the Nova, a 16 bit number is permissible. If the current machine is Maxc, a 40 bit number is permissible. The first 4 bits input are the last 4 bits of the memory. Thus one can think of Maxc as a 36 bit machine (like the PDP-10), right adjusted in a 40 bit field, rather than a 36 bit machine left adjusted in a 40 bit word as is actually the case. This command is only legal after a command which prints out a Maxc or Nova location.
<lf>	(lf means linefeed). Change the current location to the current location plus 1, and print out the (new) current location.
n<lf>	Equivalent to n, <carriage return>, <linefeed>.
↑	Change the current location to the current location minus 1, and print out the (new) current location.
n↑	Equivalent to n, <carriage return>, <up arrow>.

<tab>	Change the current location to the location pointed to by the current location. Print out its contents.
B	Boot Micro-Exec from Save Area 1 of physical disk unit A.
nB	Boot Micro-Exec from Save Area 1 of physical disk unit n (0=A, 1=B, etc.) The parameter n is remembered for subsequent "B" commands without arguments.
D	Go to the Nova Debugger. $P will resume ODT.
E	Startup the Maxc Micro-Exec, loading from Tape Unit 0.
nE	Startup the Maxc Micro-Exec, loading from Tape Unit n.
nG	Go to Maxc location n. Precisely: stop the Maxc microprocessor (cleanly if possible), then tell it to begin executing PDP-10 instructions at Maxc main memory location n.
H	Halt Maxc (cleanly if possible).
I	Initialize NVIO.
M	Halt Maxc (cleanly if possible) and return control to Midas.
nM	Go to Microprocessor location n. (If you use this command you had better know what you are doing.)
P	Protect NVIO. The herald becomes a number sign (#); only those commands that examine memory (and other state varibles) are legal.
nP	If the number typed is the Pass(word) number: ODT becomes unprotected; the herald becomes a colon (:) and all commands are legal. Currently the pass(word) number is 3301; this should be easy to remember as it is the smallest number which is the sum of two different cubes in two different ways. ($14^3 + 1^3 = 11^3 + 12^3$; octal of course.)
Q	Print out the current state of Maxc (running, stopped, or at a micro breakpoint).
R	Run Maxc (the microprocessor is told to proceed). If a reset was previously done ("21M" to ODT or "21;G" to Midas), PDP-10 interpretation will begin at the starting address pointed to by main memory location 7; otherwise, Maxc will resume from where it was halted.
nR	If n is zero, disable Maxc main memory error recovery; NVIO will halt instantly on any fatal Maxc error. If n is nonzero, reenable error recovery.
S	Startup Tenex, booting Micro-Exec from disk unit A.
nS	Startup Tenex, booting Micro-Exec from disk unit n (0=A, 1=B, etc.)
nT	Begin Testing output on DLS line n.
nU	Unhang Nova device number n by simulating a completion interrupt. If you use this command you had better know what you are doing.

nZ	Stop testing DLS line n.

8.3. Nova Locations of Interest

50:	Version number. This number is incremented by 100 for each assembly of NVIO.
51:	Normally this location has a -1. If it has the number n in it then NVIO will PUNT if it attempts to read or write into Maxc physical page n.
52:	The pass(word) number. Currently this is set ot 3301 as explained under the ODT command P.
53:	Date of most recent NVIO assembly, in the form MMDDYY, where MM and DD are octal but YY is decimal (at least until 1978).
55:	Save area to boot Micro-Exec from (usually 1).
56:	Default disk unit to boot Micro-Exec from (initially 0, but changed by arguments to B and S commands).

The following symbolic locations are of interest:

RTN:
M: location of code to return to Midas. The Nova debugger commands RTN$R or M$R are equivalent to the ODT command M.

BREAK: location to goto to make a save file of the current version of NVIO. After BREAK$R is executed, control will go to the Nova Exec and the command:

```
     SAVE name
```

should be executed to save NVIO on the specified file. This is normally done after NVIO is patched by the following sequence:

```
     DEB NVIO
     make patches
     BREAK$R
     R
     SAVE NVIO
```

If control goes to a debugger breakpoint after the BREAK$R is executed, this means that there is no disk space available. Exit via RTN$R; make disk space; and repeat patches. (Sorry!)

8.4. Memory Error Locations

The following locations are updated whenever a fatal Maxc error interrupt is processed by NVIO. They are listed in the order in which they occur.

SE	Single Error
DE	Double Error

The format of each of these words is wxyz where w can be a 0 or 4; x, a, 0 or 1; y a 0 or

2; and z a 0 or 3. This number specifies the quadrant that has had errors (4 meaning quadrant 0). Thus SE/ 4003 means that there were single errors in quadrants 0 and 3. Each fatal error interrupt causes the errors to be ored into the first six words, and stored into the last six.

DIP	Data Input Parity
APE	Address Parity Flag
PBF	Parity Bit Flag
TIME	Timeout
LSE	Last Single Errors
LDE	Last Double Errors
DIP	Last Data Input Parities
LAPE	Last Address Parity Flags
LPBF	Last Parity Bit Flags
LTIME	Last Timeouts

SE0C	Single Error Count for quadrant 0
DE0C	Double Error Count for quadrant 0
...	
TIME0C	Timeout Error Count for quadrant 0
SE1C	Single Error Count for quadrant 1
...	
SE2C	Single Error Count for quadrant 2
...	
TIME3C	Timeouts for quadrant 3.

There are (6 times 6 equals) 36 locations in this memory error block.

8.5. NVIO Punts

For each PUNT, the location at which the PUNT occurred is put into Nova register 3, and information particular to each PUNT is sometimes stored in register 0. Maxc is then halted (cleanly if possible) and "NVIO Punt" is printed on the Nova console.

Accumulators 0, 1, 2, and 3 are stored at PUNT0, PUNT1, PUNT2, and XPUNT respectively, and may be examined by going into the DOS debugger with the "D" command. After resuming NVIO with "$P", it is necessary to restart Tenex in one of the following (increasingly drastic) ways:

```
     1)  :R...OK.       (Simply tells Maxc to proceed).
```

```
    2)  :21M...OK.      (Resets Maxc microprocessor).
        :140G...OK.     (Attempts "soft" restart of Tenex).

    3)  :S...OK.        (Reloads and restarts Tenex).
```

More detailed information on handling NVIO punts may be found in Section 14.

9. Midas Commands

A. Conventions

The memories accessible to Midas are called:

MAIN	(system main memory - lower 64K words only)
LM	(left register bank 32 registers)
RM	(right register bank 32 registers)
SM	(scratch memory 512 registers)
DM	(dispatch memory 512 registers)
IM	(instruction memory 2048 registers)
MP	(map memory 1024 registers)

These names are consistent with all other microprocessor and microcode literature.

The Infoton display is completely controlled by Midas. The upper seven rows display 14 of the microprocessors internal registers. The lower 8 rows can display any memory words from the memories listed above.

A address, one of 3 forms

i) symbol identifier, possibly followed by blank and octal increment (increment may contain leading + or - sign)

ii) memory identifier followed by blank and octal address

iii) octal address, presumed to be in instruction memory

V octal value

P screen position

The letter L(left) or R(right) followed by a decimal integer from 1 to 15(row)

F a file name.

If it contains no period, a default extension will be added, as specified for each command.

Octal values, addresses and increments may contain embedded blanks.

Rubout deletes the previously typed character and backspaces the cursor.

B. Commands

In the following, some commands have two forms, with and without a screen position field. In the form without a screen position field, the current screen position is used. Unless otherwise noted, all commands with a screen position field reset the current position to that specified in the current command.

A[	prints out numeric value of A, in octal
:	single step
A:	single step at address A
A/ A,P/	display contents of address A at given position
\<cr\> P\<cr\>	displays at given position the contents of next higher address than currently displayed there
↑ P↑	displays at given position the contents of next lower address than currently displayed there
\<lf\>	displays, at position below current position, contents of next higher address than currently displayed at current position. Resets current position one lower.
V← V,P←	store octal value into the memory word addressed or register named at the given screen position
A;B	insert a break point at given instruction memory address
A;K	remove a break point from given address
A;G	start processor at address A
;P	continue running the microprocessor at the current microaddress (in IMA)
;S	single step processor (same as :)
A;S	single step processor at address A (same as A:)
;C	repaint the screen
F[1],...,F[k];R	load files F[1] to F[k] (k≥1) (default extension ".MB")
1;T	"GO" (=6,GO;T)
2;T	"STOP" (=6,STOP;T)
3,F;T	(default extension ".XX") take commands from file F until exhausted. May not be nested.
4,F;T	(default extension ".ST") define /R file for subsequent patch calls to MICRO
5,F;T	(default extension ".MB") dump state of microprocessor onto file

F, for subsequent reload using F;R. State dumped consists of the complete IM, SM, DM, LM, and RM memories but does not include the MP memory or the registers. The address symbols are dumped also. Reloading, the dump file takes about one minute ten seconds and it occupies about 52,000 characters on the Nova disk.

6,F;T	(default extension ".SV") makes .EXEC DOS system call upon file F.
!F<sp><cr>	(null default extension) is the same as 6,F;T except that F can contain spaces and slashes for constructing more elaborate .EXEC calls (e.g. "!NVIO.SV/B").
<sp>TEXT<cr>	will carry out microassembly of the TEXT using the file specified by the last 4,F;T as a /R file for MICRO. The text of the patch is appended to DBGPTCH and the binary is loaded. Two garbage files PTCH$$ (text of last patch) and PTCH$$.ST (/R file created by assembly) are also left.

C. Special Information

The cursor on the Infoton display will rise above the line when a command is in progress and drop down again when it is done.

Control-A will stop the microprocessor if it is running, but crash Midas back to the DOS EXEC if the microprocessor is not running (in which case you will have to restart by reloading the microprocessor). Midas can take 15 seconds to evaluate a new symbol near the end of a big microprogram so don't get impatient and type control-A. (Reloading from a big file takes over two minutes.) The cursor will rise above the command line when any command is in progress. If the command is a ;G or ;P, then the microprocessor will be running and control-A can be used to stop it. However, if you say SYMBOL;G, be sure you have waited the required 15 seconds for symbol lookup before typing control-A.

The first time you reference a symbol, Midas takes up to 15 seconds. Subsequent references are very fast (about 1/2 second).

Instruction memory addresses can only be displayed in the left column of the display and take up the full width of the screen.

When the microprocessor is stopped by control-A it will sometimes be possible to continue by ;P safely, but don't count on it. It is possible to continue safely from breakpoints except when the break occurs during the read portion of a read-modify-write memory reference or on either of the two cycles following a write, if the memory data register has not been loaded.

When Midas runs, it creates two temporary files called $$DBGE and $$DGBS. If you use the patch feature, PTCH$$, PTCH$$.ST, and DBGPTCH also get created. If you wish you may delete these when you return to DOS.

Do not attempt to examine main memory locations >177777 with Midas. Midas will go through the motions of examining and changing the addressed cell, but in fact the address will be truncated to 16 bits. All main memory addresses may be examined using ODT, as follows:

```
!NVIO.SV/H <cr>               Starts NVIO but leaves Maxc halted.
NVIO
:nnnnnn/ mmmmmm mmmmmm        Examines main memory.
:M...OK.                      Returns to Midas.
```

10. Examining and Controlling Maxc from Midas

A. Reset is accomplished by typing

```
21;G
```

to Midas. The effect of this is to put the machine in monitor mode, to clear the interrupt system, the APR, the processor flags and the disk system. However, Nova peripherals aren't affected, nor are the accumulators or PC affected. If NVIO is called after RESET (by 6,NVIO;T) the processor will start at its starting address (pointed to by absolute location 7).

B. Starting at an arbitrary location n is accomplished by changing the contents of PC (which is on the display) to n and then typing

```
REMAPPC:        (Midas types out the value m of REMAPPC)
6,NVIO;T
```

C. The accumulators are LM 0, LM 1, ..., LM 17 and can be examined and changed in the usual way from Midas.

D. Flags are in the left-most 13 bits of the F-register on the display. If you change these from Midas, don't change the other 23 bits of F.

These and other interesting bits of F are as follows:

Bit	Definition
0	Overflow
1	Carry 0
2	Carry 1
3	Floating overflow
4	Byte interrupt
5	User mode
6	PARC mode (replaces PDP-10 user I/O mode)
7	Call from monitor (see JSYS, UMOVE, and Pager addendums to PDP-10 System Reference Manual)
8-10	Machine mode (0=PDP-10, 1=Byte Lisp)
11	Floating underflow
12	No divide
13	Pushdown overflow
14	XCT0
15	XCT1
16	XCT2
17	XCT3
18	Incompatible
19	PI system is active
20	PI cycle in progress

21	MONALT }	Temporary flags used by the map loading
22	THIRDPT}	microcode
23	MICRO	
24	TTYBSY	console teletype output busy
25	LOGF	enables instruction interpreter's optional feature
26	Unused	
27	CUM	
28	Unused	
29	IENABLE	
30	Unused	
31	NOVA	Nova has left an interrupt request
32	K	
33	J	
34	H	
35	G	

E. The interrupt system state is given by the following:

PISTAT[29,35] have 1's when interrupts are in progress on the corresponding PI levels[1,7]. PISTAT[22,28] have 1's when interrupts are disabled on the corresponding PI levels.

SM 600 to SM 643 contain in the right-most 24 bits the interrupt locations corresponding to devices 0 through 35. The device assignments are in Maxc document 11.

SM 644 to SM 652 are the interrupt-enabled bit tables for priority interrupt levels 1 through 7. 1's in each word indicate that the corresponding device (bit 0=device 35, bit 35=device 0) is enabled for interrupts at that level.

MICINTS contains 1's for each device which has requested an interrupt. (However, OVF, FOVF, and PDOVF are recomputed from F for each instruction, so their state in MICINTS isn't important.)

F. The microcode consistency CHECKER uses three variables CSUMD, CSUM0, and CSUM1 which contain checksums for all constants in the microprocessor's IM, DM, and SM memories. CSUMD is an overall checksum (for detection of errors) while CSUM0 and CSUM1 contain 12 6-bit bytes each of which is a checksum in a Hamming code. The checker computes the address which is clobbered assuming only a single word is wrong.

When newly assembled microcode is loaded for the first time, these checksums are computed by running CHECKER ("25;G" in the TENLOAD command file) and entering the values left in LM 11, LM 12, and LM 13 into the PATCHES file for CSUMD, CSUM0, and CSUM1. Subsequent runnings of the TENGO command file check the newly loaded system against these constants, and CRASH on errors. Also the CHECK JMC does this (executed during Tenex initialization).

G. Breakpoints are handled as follows:

Two versions of the emulator's main loop are standardly available. They are called BRK.MB and UNBRK.MB. Each version has an optional path executed when the LOGF flag in F is 1. When the microprocessor is loaded via MIDAS TENGO, the UNBRK version is loaded, and the optional path performs opcode logging in the 512-word table pointed to by LOGT. By typing to Midas BRK;R, you can load a version of the main loop which has facilities for single-instruction stepping and for breaking before executing an instruction at a single selected address. The register LOGT on the display controls the breakpoint feature as follows:

0	single-instruction execution
address	break address

To disable the optional path and resume full speed execution, simply turn off the LOGF flag.

Whenever you have broken or halted, you may switch between BRK and UNBRK. Note also that whenever the processor halts or breaks, the address of the next instruction is in P and the address of the last instruction +1 is in PC on the display.

However, note that the checksums which have been set up in PATCHES are for the UNBRK version, so you will have to recompute them to use the CHECKER with BRK.

11. Warm Starts

A. If for any reason you fall out of NVIO and end up in Midas, you can continue by restarting NVIO with

```
        6,NVIO;I
```

This is not recommended when Maxc is running Tenex because I/O communication (particularly DLS and MCA) is almost sure to get fouled up. Instead, a Tenex "soft restart" should be performed, as follows:

```
        !NVIO.SV/H <cr>        (Start NVIO but leave Maxc stopped)
        NVIO
        :140G...OK.            ("Soft restart" Tenex)
```

B. If you voluntarily or involuntarily exit back to Nova DOS, there is no easy way to resume Midas. You must reload the microprocessor with either the TENGO command file (complete reload of microcode, reset of the microprocessor, and disk boot of Micro-Exec) as described under cold start procedure or with the TENLOAD command file and attempt to "soft restart" Tenex, as described above.

The command MIDAS TCON may be used to reload the microcode and continue the microprocessor without otherwise disturbing its state; however, a new copy of NVIO will be running, so it is not possible to resume Tenex in this way. MIDAS TCON is useful only when the state of NVIO is unimportant when continuing, e.g. when running standalone PDP-10 diagnostics.

C. If you want to dump the 10 program you have created in core, you must exit back to DOS (by control-A or ;X to Midas). Then dump the first 32K of main memory by typing

```
        DFB DMPLD
        DMPLD%
        DUMP$R
```

The program will then ask DUMP TO FILE: and you should type a file name. If you make a mistake type rubout and try again.

D. To load a PDP-10 .SAV program when you are in Nova DOS type

```
        DMPLD<cr>
```

and answer the LOAD FROM FILE question. Type rubout and try again if you make a mistake. This zeroes the first 32K of main memory before loading.

If you are in Midas, type

```
        6,DMPLD;T
```

When DMPLD returns to Midas you may type

```
        ;C
```

to repaint the screen.

E. If the Nova disk gets cluttered you can DELETE files (see Nova DOS documentation). Note that the available disk storage will shrink when you run Midas due to usurption of disk blocks used by DOS's program overlay feature. These blocks will be reused by the program overlay feature, but aren't available to ordinary files. If for some reason you want to recover this storage, restart DOS by pushing RESET - PROGRAM LOAD - CONTINUE on the Nova front panel (tape must be at loadpoint and switches at 100022 as discussed earlier or else NSYS.SV must be INSTALLed on the disk and switches at 100020).

12. Stopping Tenex

In order to give users adequate notice and to protect files on disk, Tenex should ordinarily be stopped in the following manner.

As long as possible before the scheduled downtime (preferably 24 hours or more), notify users by putting a notice in the login message and by setting the system downtime cell.

A. To put a notice in the login message, login as yourself and:

```
         @SNDMSG <cr>
         Users: SYSTEM <cr>
         Subject: Scheduled downtime <cr>
         Message:
         (An appropriate message giving date, time, and reason)
         ↑Z
         SYSTEM -- ok
         @
```

B. To set the system downtime cell, you must be a wheel or a maintenance person:

```
         @ENABLE <cr>                  (only if you are a wheel)
         !HALT, <cr>                   (do not omit the comma)
         [Superpassword:] GUESS <cr>
         @@AT mm/dd/yy hh:mm <cr>      (date & time system going down)
         @@UNTIL mm/dd/yy hh:mm <cr>   (date & time system coming back up)
         @@ <cr>
         @
```

The system will automatically start notifying users of the impending downtime beginning one hour before it is to occur. When the zero hour arrives, all jobs will be forcibly logged out except those jobs logged in on either of the two machine room terminals, new logins will be prevented, and "Shutdown Complete" will type out on the Maxc console. If there are now no jobs logged in, Tenex will shortly hit a BUGCHK (EDDT breakpoint) at SWHLT1, at which point Tenex is properly halted.

If there are jobs logged in on machine room terminals, it will be necessary to halt Tenex manually. To do this (for which you must be a wheel):

```
         @ENABLE <cr>
         !QUIT <cr>
         .Halt Tenex.
```

Wait for the EDDT breakpoint at SWHLT1.

13. Power Down

You should not power down the processor or the memories by pulling plugs or throwing switches since this sudden shut down can damage components. Only resort to this drastic action in an emergency (e.g., fire).

Proper shut down is accomplished by running the STOP.SV Nova program first, then pulling plugs and turning off switches. This can be accomplished either by saying

```
         STOP <cr>
```

to Nova DOS or by 2;T (=6,STOP;T) to Midas. When you run STOP, the two green lights on the processor and the four red lights on each memory will shut off. The fans will continue to run and you may unplug them at the wall.

To turn off disks, turn off the front panel switches first. Do not turn off AC power (in back) until the pack has stopped turning, else the drive may stop with the heads still extended!

14. Tenex Crashes

If a Maxc-Tenex maintainer is available at PARC, inform him of the crash and normally he will take over. Otherwise, be brave and read on.

There are obvious problems in attempting to describe what to do when a system crashes. This section simply outlines a few simple procedures whose purposes are twofold: (1) to collect data about the crash for subsequent analysis and (2) to restart the machine as quickly as possible.

When system personnel are not present, Tenex is left in a mode in which it attempts recovery from Tenex-detected errors. Thus the majority of the time the crashes handled by non-system people will be of a more obscure (relative to Tenex) nature.

The following paragraphs describe some of the more common crashes and suggested recovery procedures. All pertinent information should be recorded in the Maxc Tenex logbook. Recent error typeouts (e.g. memory parity error messages) should be cut out and taped in the logbook.

A. BUGHLT or BUGCHK: A message of the form

```
        $8B>>BUGADR  BUGHLT/ CAI UUOH+4
```

types out on the Maxc console. This occurs when the monitor is in debug mode, which should never be the case unless system maintainers are present. However, if you cannot find one, set the DBUGSW and DCHKSW cells to zero and proceed from the breakpoint:

```
        DBUGSW/  1  0 <lf>
        DCHKSW/  1  0 <cr>
        <esc>P
```

Tenex will recover from the error if possible, else restart automatically with no further intervention required.

B. "Micro Breakpoint" printed out on the Nova console (Infoton). This and similar NVIO messages will usually appear above the lowest line of text and two lines of numbers usually displayed by NVIO. This message means that the microprocessor hit a breakpoint, which is usually caused by Tenex executing a HALT instruction but can also be caused by any of several microprogram-detected inconsistencies.

The only known HALT instructions in Tenex are associated with catastrophic disk errors, e.g. disk units offline, and a message such as

```
        Sector Interrupt Timeout Trouble with Disk Pack 111
```

is typed out on the Maxc console. If this was caused by accidentally turning a disk off or other obvious human error, the crash may be recovered by the following procedure.

1. Correct the cause of the error; i.e. turn on all disk units and wait 2 minutes for the green lights to come on. There should be no yellow (read-only) or red (select lock) lights on.

2. On the Nova console (Infoton), type:

```
        #3301P            (to "un-protect" NVIO)
        :140G...OK.       (to "soft-restart" Tenex)
```

All other disk errors (particularly "Error in Sensitive Page") should be handled by system personnel only. Attempting to restart could cause the file system to be wiped out!

A "Micro Breakpoint" not accompanied by a printed message on the Maxc console should be handled as in Paragraph E, "No Response".

C. "Fatal Memory Error, Maxc Stopped" on Nova console. This indicates that the memory is very sick, and a hardware maintainer should be notified.

D. "NVIO Punt" on Nova console. This indicates a serious inconsistency detected by NVIO. Crash data should be saved and the crash recovered as follows (type on the Nova console).

```
        #3301P                     ("un-protect" NVIO)
        :D...OK.                   (enter Nova debugger)
        XPUNT/ junk :sssss+n = mmmmmm <lf>
        PUNT0/ junk :sssss+n = mmmmmm <lf>
        PUNT1/ junk :sssss+n = mmmmmm <lf>
        PUNT2/ junk :sssss+n = mmmmmm <cr>
        <esc>P                     (Resume NVIO)
        :R...OK.                   (Resume Maxc)
```

If Tenex does not resume after this procedure, try a "soft restart" by typing:

```
        :140G...OK.
```

on the Nova console.

Write down in the logbook the data typed out by the debugger in response to the ":" and "=" characters you typed in.

E. No response from Tenex; i.e., no error messages have typed out, but nothing happens when you type control-C on the Maxc console. This type of crash is particularly hard to diagnose unless sufficient information is recorded.

First, note the numbers on the last line of the Nova console, and note whether any of them are changing over time.

Second, enter Midas by typing the following on the Nova console:

```
        #3301P            (to "un-protect" NVIO)
        :M...OK.          (to enter Midas)
```

(If the message "Unclean Micro Stop" prints out, note this as well.) Write down the contents of the following registers displayed by Midas:

```
        NPC    IMA    P    Q    STK 0    PC    PISTAT
```

Of particular importance is PC, the PDP-10 program counter.

Third, type the Midas commands:

```
        21;G
        25;G
```

to reset the microprocessor and check the microcode. Both commands should finish momentarily with IMA=27 or 30. (If IMA=20, the microcode has been clobbered and you will need to reload it, as explained in step F below.)

Fourth, re-boot Maxc by typing:

```
        !NVIO.SV/B <cr>
```

The Micro-Exec herald should type out on the Maxc console.

Fifth, pick a save area on which to dump the crashed Tenex. Save areas 10 through 17 are used for this purpose. Look through the last page or two of the Maxc Tenex logbook to see what areas already contain recent crash dumps (if any), and pick an area that hasn't been used. Type (on the Maxc console) the command:

```
        *Dump.Core.on.Area n <cr>
```

where n is the save area you have selected, and note in the log that the crash was saved on area n.

Finally, type:

```
        *Go <cr>
```

to reload and restart Tenex. After a few seconds, "Tenex restarting, wait..." should be typed out and Tenex will request the date and time, which you should type in very carefully. If Tenex asks you to reconfirm, you probably blew it and should hit the "Del" key and try again.

F. It may happen that a crash does not fall into one of the above categories or that the restart procedure fails. In this case, the following procedure will <u>always</u> succeed if all the hardware is working:

1. On the Nova front panel in the machine room, press, in sequence, "Reset", "Program Load", and "Continue". On the Nova console, you should see:

```
    DOS REV 04
    R
```

2. Type the command:

```
    GO; MIDAS TENGO <cr>
```

3. Wait about 2 minutes while the microcode loads and NVIO and Micro-Exec are started. Micro-Exec will execute an automatic "Go" command, after which you should enter date and time as explained above.

If this doesn't work, try to find any one of the people listed below at PARC. If none of them is around and the hour is between 9 AM and midnight, call one of the system maintainers. If between midnight and 9 AM, don't bother, but leave a message on the telephone recording (if you know how) saying that the machine will be down until morning.

People to notify: (use phone list on wall under phone)

Software and general system maintenance

Taft	Tenex and NVIO
Fiala	Microcode and Tenex
Geschke	General

Hardware

Lampson	Microprocessor
Clark	Memories
McCreight	Disks
Thacker	Almost anything

15. Using Micro-Exec

Micro-Exec is a stand-alone Maxc program used to maintain and start up the Maxc Tenex system. A few overall remarks on its structure will be helpful before describing the specific facilities available.

Micro-Exec is normally loaded by the TENGO command file (see section 4) from Maxc disk unit A. Micro-Exec loads into the high portion of the first 128K of memory, starting at 350000. All references to "core" in Micro-Exec commands refer to the range 20-to-347777. (The program which boots in Micro-Exec runs in 347000-347777 and saves the previous contents of the ACs in 347760-347777.)

Micro-Exec has a sophisticated command interpreter which allows editing, questioning, prompting for parameters -- what we in the biz call "Dwimified-Middle-English". Commands take the form of a sequence of words separated by periods. E.g.:

```
        initialize.disk.pack
or
        initialize.tape.
```

At any point in typing a string, you may type <escape> and the interpreter will complete as much as is unambiguous. Similarly, at any point you may type "?" and all the possible completions of that command will be listed. Once you are familiar with commands, you will find it convenient to abreviate them by entering the first letter of each word followed by a space. For example, I D P is an abbreviation for "initialize.disk.pack".

Many of the commands require parameters before the command execution begins. Typing an escape will prompt you with a brief description of the parameter (e.g., (octal number) or (pack number)). Typing escape a second time will furnish the default value of that parameter if there is one. Some commands require confirmation (with .) and some require the user to be ENABLEd. The ENABLE command requires a password.

Finally a few words on the disk structure of the Maxc-Tenex system will be helpful for the following command descriptions. A disk configuration is defined by establishing a correspondence between (1) logical units (Tenex), (2) pack numbers, and (3) disk drives (i.e., controllers). The current disk configuration is always posted on the control room wall and is (usually) the default configuration on the current Micro-Exec save area (usually area 1 on physical unit 0). Logical units are identified by digits 0 thru n-1 (for an n-pack Tenex). Packs are labeled with octal numbers starting at 100. Drives are labelled alphabetically starting with A. For example, the "print.disk.configuration" command might type:

```
        Number of units: 2
        --------Unit--------Pack Number
        0.         B          102
        1.         A          107
```

At present there are 20 AREAS allocated in the disk structure, but outside the Tenex file system. Each AREA consists of four track cylinders (i.e., 240 (decimal) pages). A directory of AREAS is posted on the control room wall.

In the following descriptions, a (C) means confirmation is required (E) means you must be enabled, and (C,E) means both are necessary. In commands that read from or write to disk, the number following the command indicates the error retry count used; "default" means that the retry count may be overridden by the count specified in the "set.disk.error.retry.count" command.

brief
Disables extended typeout of disk error status bits.

change.version.to <string> (E)
Changes the version description for Micro-Exec.

check.next.tape.file <magtape unit number>
Checksums next file on tape and types entry point on console.

clear.core (C)
Zeros locations 20 through 347777.

compare.disk.packs <pack$_1$> <pack$_2$> (20)
Compares the contents of the specified packs and reports discrepancies (usually used after a pack-to-pack copy).

copy.pack.to.pack <pack$_1$> <pack$_2$> (C,E) (default 20)
Does a bit-by-bit copy of data from first pack onto second pack.

copy.quadruple <pack$_1$> <pack$_2$> <pack$_3$> <pack$_4$> (C,E) (20)
Simultaneously copies the first pack onto the second pack and the third pack onto the fourth pack.

ddt (E)
Enters DDT for debugging Micro-Exec. Re-enter Micro-Exec by 20$G.

disable
Disables commands that require being enabled.

dismount.auxiliary.pack
Removes auxiliary pack from disk structure.

dump.core.on.area <area> (E) (20)
Dumps locations 20 through 347777 on <area>. Registers are saved at 347760 when Micro-Exec is booted in. This command is used to save crashes for later analysis.

eddt (E)
Enters Exec DDT if Tenex is in core. Re-enter Micro-Exec by typing 20$G.

enable <password>
Enables user to execute those commands which require it. Current password is SIEGFRIED. (Currently, when Micro-Exec is booted in, it is already enabled, so this command is necessary only if you have previously given a "Disable" command.)

go (20)
Loads Tenex from area 0 and starts it at SYSGO1.

help
Counsels you on the method of obtaining help.

initialize.disk.pack (C) (default 0)
Initializes the auxiliary pack by: (1) writing headers, using the pack number entered in the mount.auxiliary.pack command, (2) writing random data on the entire pack, and (3) checking the data and printing discrepancies. An error retry count of zero is used throughout.

initialize.tape <magtape unit> (C,E)
Rewinds tape to load point, writes a boot header, and waits at the end of the boot header.

load.dump.from.area <area> (20)
Loads core from specified area (converse of dump.core.on.area).

loop.on.specified.disk.page <pack> <track> <head> <sector> (E)
Continuously reads the specified disk page ignoring errors (useful for disk tune-ups).

mount.auxiliary.pack <unit> <pack>
Identifies auxiliary (i.e., extra-Tenex) pack to disk configuration. Certain commands such as initialize.disk.pack are valid only for the auxiliary pack.

print.disk.configuration
Prints disk configuration in the format illustrated above.

read.specified.disk.page <pack> <track> <head> <sector> (default 20)
Reads specified page into buffer at location BUF (377000).

read.tape.to.core <magtape unit>
Reads tape into core starting at location 20.

read.tenex.from.area <area> (20)
Loads Tenex from specified area and checks its "fingerprint".

read.tenex.from.tape <magtape unit>
Loads Tenex from tape and checks its "fingerprint".

rewind.tape <magtape unit>

scan.disk.pack.for.errors <pack> (default 3)
Scans pack, counting soft errors, and reporting non-recoverable errors to console.

set.disk.configuration (E)
Sets disk configuration; asks for parameters via the same format used in print.disk.configuration.

set.disk.error.retry.count (E)
Overrides the default disk error retry count in certain commands.

silent (C,E)
Completely turns off disk error typeouts (useful for setting up scope loops for hardware debugging).

start.tenex.from.area <area> (C,E) (20)
Reads Tenex from the specified area, writes it on area 0, and starts at SYSGO1.

verbose
Reinstates full disk error typeout.

write.core.to.tape <magtape unit>
Writes core from 20 to 347777 onto tape.

write.micro.exec.to.area <area> (C,E) (20)
Writes a bootable (via NVIO "B" and "S" commands) copy of the currently running Micro-Exec to the specified area.

write.micro.exec.to.tape <magtape unit>

write.specified.disk.page <pack> <track> <head> <sector> (C,E) (default 20)
Writes page from BUF (377000) to specified disk page.

write.tenex.to.area <area> (C,E) (20)
After checking "fingerprint", writes Tenex core image to specified area.

write.tenex.to.tape <magtape unit> (C)
After checking "fingerprint", writes Tenex core image to specified tape.

16. Reading a .SAV File on Minidumper Tape into the Nova File System

PDP-10 Diagnostics and other stand alone PDP-10 programs which reside only in the low 32K of main memory may be read from minidumper tapes and written on Nova disk files using the MTAPE program. Subsequently they may be loaded into main memory and dumped back on disk files using DMPLD. The procedure is described below.

To load from minidumper tape, first mount the tape on the Nova's seven-track tape drive (unit 1) and put it in remote. Then give control to Nova DOS and type the following:

```
DELETE TENCODE.SV
MTAPE<cr>
```

which will ask "LOAD FROM FILE:". You should answer with a unique initial substring of the file name. For example, <SYSTEM>AUXD.SAV<cr>. MTAPE will copy this file onto TENCODE.SV which you can rename as follows

```
RENAME TENCODE.SV AUXD<cr>
```

Normally you will have to prepare locations 3, 4, and 7 as discussed in section 13 before running the program. To make these changes, proceed as follows:

```
DMPLD<cr>
LOAD FROM FILE:     AUXD<cr>
NVIO/H<cr>
3/ 0      NNNNN<Lf>
4/ 0      NNNNN<cr>
```

```
7/ 0      NNNNN<cr>
M...OK.
DEB DMPLD<cr>
DMPDLD%
DUMP$R
DUMP TO FILE:       AUXX<cr>
```

The file will normally grow by a substantial amount when this is done because DMPLD does not compress out zeroes when dumping.

```
DELETE AUXD<cr>
RENAME AUXX AUXD<cr>
```

17. Required Format for Standalone PDP-10 Programs

A. The loader (DMPLD) discussed in the previous section requires the program to be a .SAV file which resides in the low 32K of main memory. DMPLD currently zeroes the low 32K of core before loading the program.

B. The interprocessor communication locations must be set up with pointers as follows:

3/	MTBS	Magtape, IMP, MCA, etc. 144-word block (may grow later)
4/	DLSBS	Data line scanner 68-word block
7/	ADDRESS	Program starting address

Note that these are absolute main memory locations in the "shadow" of the accumulators, so they must be initialized in one of three ways:

(1) By hand from Midas or ODT
(2) By mapping page 0 to a different virtual address
(3) By using the JMC's for addressing absolute main memory addresses.

C. The program must perform an I/O reset (CONO APR,200000) before carrying out input/output to any Nova related peripherals and after initializing MTBS and DLSBS as described above.

18. Running Microprocessor Diagnostics

The microprocessor diagnostics are sometimes left on the disk and are on the BASEF record of the 10SYS tape (see Section 4). If they are not loaded, get the current uP DIAGNOSTICS tape from the tape rack and mount it on unit 0 at the load point. Push, in sequence, the Reset, Program Load, and continue buttons on the Nova front panel. You will probably have to free up some disk space by deleting extraneous Nova files before loading in the diagnostics. Then load the record labelled "*.MB" on the tape jacket:

```
        LOAD/A/V MTO:n <cr>
```

You must get the blue-covered, Velo-bound book labelled "Microprocessor Diagnostics" in order to do any useful debugging. However, the diagnostics all fall into a common pattern and they can usually be run in a simple-minded way without understanding too much about what the programs are really doing. The following generalizations may be helpful.

Associated with each of the diagnostics described below is a command file whose name is the concatenation of "R" and the diagnostic name. If you type to DOS:

```
        MIDAS Rdiagname <cr>
```

Midas will load the diagnostic and open registers pertinent to the diagnostic's operation on the Infoton display. You may wish to set some parameters before starting a diagnostic (for example, the low and high addresses for a memory test). However, the values as loaded are reasonable for thorough testing.

When you are ready, type START;G to start the diagnostic. (DGM has two alternate starting addresses for interrupt system testing.) The diagnostic will halt after one pass at IMA=20 (DGIML is the only exception to this rule). This means that no failures were detected. To repeat, type ;P to Midas. To loop indefinitely, delete the breakpoint at 20 by typing 20;K and then ;P. All of the diagnostics except DGM loop in less than two seconds, so something is wrong if the machine hangs in the loop.

A breakpoint at any location except 20 indicates that an error was detected. The most common error breakpoint is at 26, which is the comparison error breakpoint. The location in the diagnostic from which the comparison routine was called is displayed at STK 0. The .LS listing for each diagnostic gives address symbols and their values, sorted in order of value so that the symbol nearest the error address may be found readily (you normally are interested in IM memory addresses only). Following the .LS listing is the diagnostic listing, which contains general operating instructions in comments preceding the program itself. DBEGO is assembled or loaded ahead of some diagnostics, so its listing and .LS file may also be relevant. Some diagnostics INSERT other files or are assembled from several sources, so you may have to look a bit to find the relevant listings. When tracing an error, find the tag nearest the address in STK 0 (if the breakpoint is at IMA=26) or IMA (if the breakpoint is elsewhere) and read the comments there from the listing.

If the breakpoint is at IMA=26, the diagnostic may be resumed from the point of error by the command RETN;G, which will return to the caller of the compare routine (this usually works but not always).

One common problem that occurs when running diagnostics is that the microprocessor single steps and refuses to run. The usual cause of this is a memory interface hangup, which at present can be cured only by running the GO program (type "6,GO;T" or "1;T" to Midas). If the machine hangs, run GO and see if that cures the problem.

The diagnostic names and what they test are listed below:

DG01.MB	Most basic diagnostic. Does not use the main memory, the interrupt system or the P-register input multiplexors. Tests everything except the SM, DM, LM, and RM memories first, then tests these memories and the F-register. Cycle time < 1 second.
DGP.MB	Tests the P-register inputs and a few afterthoughts from DG01.MB. Cycle time < 1 second.
DGM.MB	Tests the memory interfaces. There are alternate entry points at START and XSTART for interrupt system testing. Be sure to read the listing comments before starting at XSTART. Cycle time ~ 40 minutes to test all 256K of main memory; correspondingly less if the address range is restricted.
DGI.MB	Tests the interrupt system and repeats parts of DG01 and DGP affected by interrupts. Cycle time < 1 second.
DGIML.MB/DGIMH.MB	Test the SM, DM, and MP memories using random numbers, then the instruction memory (IM) first by using each of the 72 patterns of a single 0 in a field of 1's, then the 72 patterns of a single 1 in a field of 0's, then random numbers. Cycle time < 3 seconds.
DGRL.MB	Tests the right register bank (RM) and the left register bank (LM) using random numbers. Cycle time < 1 second.
DGMR.MB	Tests the main memory using random numbers. Cycle time ~ 20 seconds.

The diagnostics are assembled according to the instructions in the listings.

If you have to patch the microprograms to setup a scope loop, you will need familiarity with the microlanguage. The following pointers may help. The .ST files needed for patching are usually deleted after assembly because they won't all fit on the disk. If you need one, simply reassemble the microprogram. Since patching requires more disk storage, you may have to delete something from the file directory. Then reload the microprogram and do a 4,FILE;T to attach the .ST file and then use the E macro to make the patch. For example, to change the third instruction after LOOP type to Midas

```
        <sp>E[3,LOOP]; GOTO[.-2];<cr>
```

or whatever. Then run with START;G

Diagnostic sources are maintained in the Tenex <ERF> directory. There are Tenex command files for producing a cross reference listing and for dumping the sources onto a magtape for transport to the Nova file system where they are assembled and loaded. The contents of the command files are also printed in the "Microprocessor Diagnostics" book.

19. [Running PDP-10 Diagnostics]

19.1. [Standalone]

19.2. [Under Tenex]

20. [Hardware Maintenance Procedures]

20.1. [TMEM]

20.2. [TR]

21. Writing a New 10SYS Tape

If any system files have been changed, they should exist on the Nova disk prior to beginning the following procedure.

A. Mount the current 10SYS tape on unit 0. (You must use magtape unit 0 for 10SYS tapes since DOS will not handle any other drive.)

B. Type:

```
        XFER MT0:0 TBOOT.SV <cr>
```

C. If NSYS.SV is not already on the Nova disk, type:

```
        XFER MT0:1 NSYS.SV; CHATR NSYS.SV SPW; INSTALL NSYS.SV <cr>
```

D. Free up disk space consumed by the program overlay feature by re-booting DOS (push, in sequence, the "Reset", "Program Load", and "Continue" switches on the Nova front panel). Then load the most recent BASEF, SYSF, and CHANGEF records from the tape (consult the label on the tape jacket). E.g.,

```
        LOAD/A/V MT0:(2,7,13) <cr>
```

if the newest BASEF, SYSF, and CHANGEF files are on records 2, 7, and 13. Most of these files will undoubtedly already exist on the disk and will not be overwritten (The message "File already exists" will be printed out for these).

E. Mount the new tape, with a write ring in it.

F. Type:

```
        XFER TBOOT.SV MT0:0; XFER NSYS.SV MT0:1 <cr>
        DUMP/A/V MT0:2 @BASEF@ <cr>
        DUMP/A/V MT0:3 @SYSF@ <cr>
        DUMP/A/V MT0:4 @CHANGEF@ <cr>
```

G. Check that everything is written ok by typing:

```
        LOAD/V MT0:(2,3,4) <cr>
```

"File already exists" should be printed for every file.

H. Remove write ring, label tape jacket.

22. Recovery from Checkdsk Errors

As explained in Section 7, when Tenex is restarted, whether initially or due to an auto-restart after a crash, the programs Bsys and Checkdsk are run to verify the consistency of the file system. If either of these programs detects errors which cannot be corrected automatically, the message "Tenex not available: Disk needs fixing" is broadcast to all terminals instead of the usual "Tenex in operation" message, and logins are prohibited from all terminals except the two in the Maxc room.

The following procedures require wheel or operator status and are intended principally for reference by system personnel. With some assistance from a user in the Maxc room, a system maintainer can perform these procedures from a home terminal. Only in extreme circumstances should non-system personnel attempt any of these procedures.

Errors detected by Bsys (which are usually reflected by some further errors detected by Checkdsk) indicate inconsistencies in the structure of user file directories. Fixing these requires a fairly intimate knowledge of the Tenex directory structure; this should be left to system personnel.

Checkdsk errors come in a number of guises. For each file with errors, data will be printed as in the following example:

```
        40050172166 MDA 0        } List of errors
        140050172170 MDA 63      }
        <NEELY>MESSAGE.COPY;3   Filename
        1 PTE                    } Error
        2 MDA                    } summary
```

If there are many errors in a single file, Checkdsk will print out only the first few, followed by the filename and summary. Study the output carefully.

First, note that "NOT IN BT" errors have been corrected by Checkdsk, so don't worry about them. If these were the only errors that occurred, Checkdsk wouldn't have complained and the system would have flown on.

Next, note that you will have to use some judgement in discriminating between garbaged page tables and real MDA errors. A file with a garbaged page table will have an enormous error count (in the hundreds) with many categories of errors (IDA, MDA, PTE, etc). This is frequently caused by an untimely Tenex crash occurring between directory update and page table update during new file creation, so that the page table for the file will not have been written on the disk yet and whatever was on that page before will be interpreted as a page table. This type of error may result in many other files getting bogus MDA errors because some of the entries in the garbaged page table look like valid disk addresses that happen already to be assigned.

For further confirmation that the problem is a garbaged (unwritten) page table, a QFD of the filename should reveal that it was created within a minute or two before the time of the last system crash. Such a file should be deleted using the following procedure:

```
@ENABLE <cr>
!CONNECT <directory with bad file> <cr>
!DELETE <bad filename> <cr>
!EXPUNGE <cr>
!
```

This procedure causes the bad file to be expunged from the directory. A number of valid addresses possibly in use by other files may be deallocated, but don't worry about this. The system will generate a number of BUGCHKs for illegal disk addresses, but don't worry about this either. (Be sure DCHKSW is set to zero, however, to prevent the system from breakpointing on these errors). Run Checkdsk again after performing this surgery to make sure you did it right and that there is nothing else wrong. Checkdsk will reallocate pages incorrectly deallocated by the preceding procedure and will type out "NOT IN BT" for these.

```
!CONNECT SYSTEM <cr>
!CHECKDSK <cr>
REBUILD BIT TABLE? N
SCAN FOR DISK ADDRESSES? N
```

(This currently takes about 10 minutes).

After all files with garbaged page tables have been eliminated (if there were any), any further errors are considerably more serious, particularly MDA errors. MDA stands for multiply-allocated disk address, meaning that a particular page has somehow been assigned to more than one file. For each such error, Checkdsk has printed out the second file owning the page that it encountered in its scan of the file system; you do not yet know the name of the other owner of that page. Hence you should follow this procedure:

```
!CONNECT <directory containing file with MDA error> <cr>
!COPY <affected filename> GARBAGE <cr>
!DELETE <affected filename> <cr>
!EXPUNGE <cr>
!RENAME GARBAGE <affected filename> <cr>
```

Repeat this procedure for all affected files. You should be careful to type the <affected filename> in full, including version number, so you don't mistakenly fix up the wrong file.

Next, re-run Checkdsk as explained above. While running, Checkdsk will type out a number of NOT IN BT errors whose disk addresses correspond to the disk addresses in the original MDA error printouts; the filenames typed out will be those of the other owners of the pages that were multiply assigned. It may not be obvious which owner of a page has the correct copy (a QFD of the filenames will include the write dates, which may give some indication; i.e. the file with the newer write date is more likely to have the correct data), but you have done the best that can be done by giving non-conflicting copies to everybody involved. Use SNDMSG to notify all users who have (potentially) lost files. Both the original MDA and the final NOT IN BT files are involved in the loss.

When you have pieced the filesystem back together to what you believe is a reasonable state, you should open the system to users by the following procedure:

```
!QUIT <cr>
./
FACTSW/ 500000,,0   400000,,0 <cr>
<control-P>
ABORT
.↑
!LOGOUT <cr>
LOGOUT USER ......
<control-C>
```

After you type control-C, the auto-jobs should start logging in, and shortly thereafter "Tenex in operation" will be broadcast.

23. [Tenex File System Maintenance]

23.1. [Bsys Operation]

23.2. [Creating and Destroying Directories]

23.3. [Format of System Directory and Index]

23.4. [Fixing Smashed Directories]

24. [Assembly Procedures]

24.1. [Tenex]

24.2. [NVIO]

24.3. [Microcode]