A COMPARATIVE STUDY OF THE MICHIGAN TERMINAL SYSTEM (MTS) WITH OTHER TIME SHARING SYSTEMS FOR THE IBM 360/67 COMPUTER

Elbert Farrell Hinson

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A Comparative Study of the Michigan Terminal System (MTS) with Other Time Sharing Systems for the IBM 360/67 Computer

by

Elbert Farrell Hinson

Thesis Advisor: G.H. Syms

December 1971

Approved for public release; distribution unlimited.

A Comparative Study of the Michigan Terminal System (MTS) with Other Time Sharing Systems for the IBM 360/67 Computer

by

Elbert Farrell Hinson
Lieutenant Commander, United States Navy
B.S., The Tulane University of Louisiana, 1958

Submitted in partial fulfillment of the
requirement for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1971

ABSTRACT

The principle of benchmark measurement is applied to yet another time sharing system, the Michigan Terminal System (MTS), and thus complete the comparison of the three time sharing systems for the IBM 360/67 Computer. MTS proves to be the superior time sharing system.

MTS is also evaluated from both a user's view and a performance view against the CP/67 and OS/MVT combination now in use at the Naval Postgraduate School.

The concept of Load Factor and its use is expanded. Use of measurement techniques to determine load factors are discussed with extension of the load factor concept to measure system loads on a continuous basis.

An Operator's Manual for MTS is presented to aid beginners in the operation of MTS. It is believed that this manual, or a generalized version, should be distributed as part of the MTS documentation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The interest in evaluation of time sharing system performance at the Naval Postgraduate School was generated by the bad experience associated with TSS/360 as an operational time sharing system. Earlier this year the performance of TSS/360 and CP/67 were compared by Haines and Porterfield in their Master's Thesis [Reference 1]. A technique using a benchmark for terminals was developed that could be used to vary the system load for time sharing systems and thus provide some basic technique for evaluating the performance of the system.

The problems of maximizing the performance of a computer system, especially a time sharing system, are unique to each installation and for that matter to the system under evaluation. An educational institution such as the Naval Postgraduate School has an extremely variable load that depends upon the course scheduling during the academic year, the professor's ideas and student research projects. The system must have a wide range of programming languages available for users, should be rapid in response and should be easy to operate from a user standpoint. Even with all of these variables, some method of performance evaluation must be devised. The benchmark mentioned above can be used to apply a load to the system under evaluation that is representative of loads seen by the system. If the loads are classified as editing, compute bound, large paging and small-size fast-execution then the choice of programming languages used to program the benchmark is not important. The response and throughput obtained from the benchmark evaluation indicates the behavior of the system under load.

A Load Factor may now be calculated to provide continuous indication and monitoring of system performance.

The Naval Postgraduate School had obtained the Michigan Terminal System (MTS) in July 1970 from the University of Michigan but insufficient computer time was available during the period January to June 1971 to evaluate a third time sharing system. This paper describes the reactions of MTS to the same benchmark technique developed for CP/67 and TSS/360 and compares the results. The basic criteria for good performance is still response time and throughput.

This thesis presents an evaluation of MTS as a time sharing system from the standpoint of performance under a benchmark load, and from a user's view. MTS using all system resources, Figure 1, is compared with the system now in use at the Naval Postgraduate School consisting of CP/67 with one CPU, one core box, the drum, and 2311 disk drives as the time sharing system and OS/MVT with one CPU, two core boxes and 2314 disks as the batch system. Emphasis is placed on the ability to use a system for one's benefit rather than become a slave to the system.

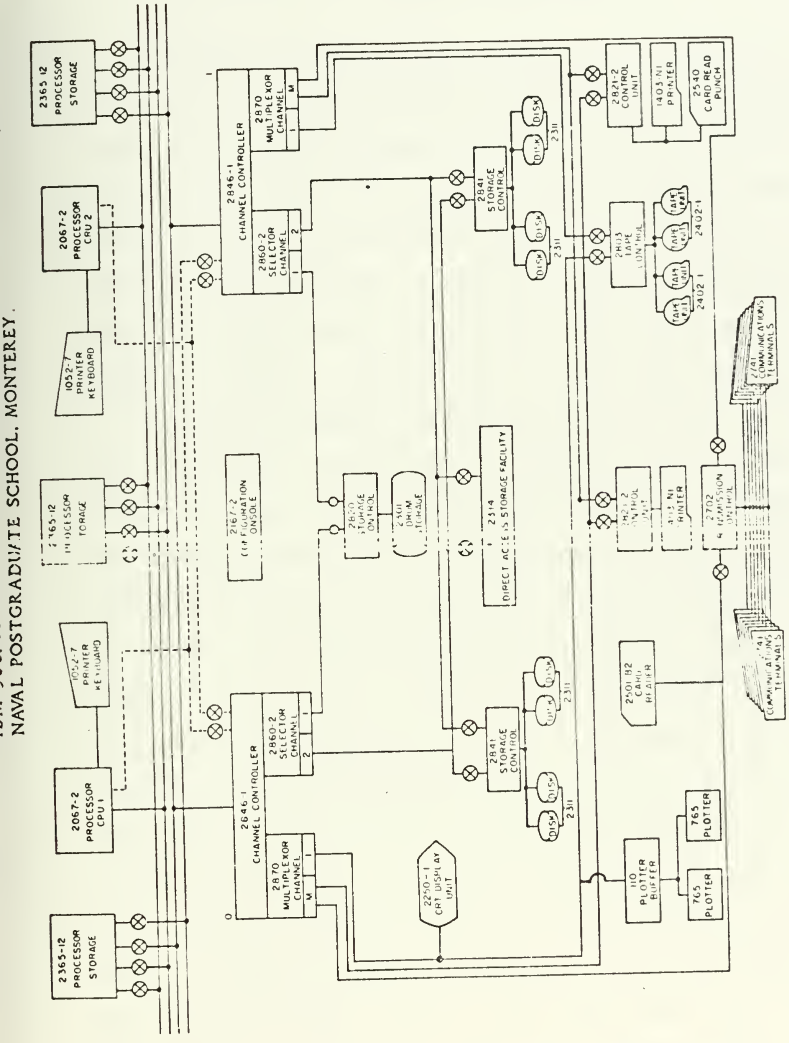
NAVAL POSTGRADUATE SCHOOL. MONTEREY
2365-12 PROCESSOR STORAGE
2067-2 PROCESSOR CPU 1
2067-2 PROCESSOR CPU 2
1052-7 PRINTER KEYBOARD
2846-1 CHANNEL CONTROLLER
2860-2 SELECTOR CHANNEL
2870 MULTIPLEXOR CHANNEL
2167-2 CONFIGURATION CONSOLE
2820 STORAGE CONTROL
2301 DRUM STORAGE
2314 DIRECT ACCESS STORAGE FACILITY
2841 STORAGE CONTROL
DISK
2311
2250-1 CRT DISPLAY UNIT
2821-2 CONTROL UNIT
1403-N1 PRINTER
2540 CARD READ PUNCH
2803 TAPE CONTROL
TAPE UNIT
2402-1
2702 TRANSMISSION CONTROL
2741 COMMUNICATIONS TERMINALS
2501-B2 CARD READER
110 PLOTTER BUFFER
765 PLOTTER

Figure 1.

II. SPECIFIC OBJECTIVES

The primary objective of this study was to evaluate the performance of MTS against the current systems in use at the Naval Postgraduate School (NPS), using the benchmark technique developed by Haines and Porterfield [Ref. 1]. This would then provide evaluation measurements for all three major time sharing systems currently available for the IBM 360/67 computer system. Management would then be presented with evaluations using a common base and allow a decision to use a particular system to be based on figures derived from the same technique.

In order to accomplish the primary objective it was first necessary to learn how to run MTS both as an operator and as a user. The use of MTS from a user's view is presented very clearly in Reference 2 but instructions to operators is very sparse and scattered throughout the literature provided by the University of Michigan [Refs. 2 - 11]. It was decided that a better evaluation could be made if the system was updated from Version 2.0 to Version 2.3. During attempts to update the system it was found that the scratch files generated by the assembler would overflow the disk space provided by the one 2314 disk pack being used for the system as soon as the program got about the size of the Supervisor. There were no other 2314 packs available at NPS and funding restrictions prevented obtaining another one. Since Computer time during academic quarter four at NPS is at a premium for other system evaluations because of the normal school load, and since the University of Michigan has stated that a complete update of MTS, Version 3.0, incorporating all changes to date should be issued early next year, these factors and the limited time available to complete the

research seemed to favor continuing with the research using the working version of the system instead of updating it. Future test should be conducted using the new version of MTS when it becomes available.

A secondary objective was to compare MTS with the current systems in use at NPS of CP/67 with one CPU, one core box, the drum and 2311 disk drives for time sharing and OS/MVT with one CPU, two core boxes, and 2314 disks as the batch system, from a user's view. Almost all of the operations and facilities of MTS were tried and compared against the equivalent operation or facility available in the other two systems. Some of the aspects of system maintenance that were acquired while attempting to update MTS from version 2.0 to version 2.3 are discussed. This evaluation is slanted to a student atmosphere, educational institution viewpoint rather than an industrial production viewpoint.

A byproduct of this work is presented in Appendix A as an MTS Operators Manual. This manual will hopefully be of use to neophytes, such as the author, for running MTS. The manual is presented so that a rank amateur can run MTS on an IBM 360/67 system. Operator functions from Memos to the operators at the University of Michigan, MTS Manuals, initial system distribution literature, University of Michigan CCMEMOS and CCNEWS [Refs. 2-11] and personal experience of the author have been condensed to one volume for ease of use.

III. DESCRIPTION OF MTS

The Michigan Terminal System (MTS) is a multiprogramming, multi-processing, time sharing system that utilizes the special features of the IBM 360/67 computer system to create a virtual memory space for the jobs being processed. The overall scheduling of system resources is done by the University of Michigan Multi-programming Supervisor (UMMPS). All time sharing is done by a reentrant process called MTS. The actual task of moving pages in/out of core from/to drum or disk is done by the Paging Drum Processor (PDP). A batch stream is also available that is controlled by the Houston Automatic Spooling and Priority System (HASP). Each task controlled by UMMPS is assigned a name corresponding to the process required to execute it and a five digit job number so that UMMPS can keep track of tasks assigned to the system. The virtual memory available to a task is limited to a maximum of 256 pages (1024K bytes). The task name specifies the entry point of the program the task is to execute and specified to the system whether the relocation hardware is to be on or off. HASP does not actually execute any jobs on its own but rather passes the job to MTS when the program is ready for execution. UMMPS also provides the interface between all input/output equipment and schedules jobs for execution on an available CPU.

In order to explain some of the results found during testing, an explaination of the scheduling algorithm used by UMMPS is in order. One queue is maintained for tasks awaiting a processor. The task may be in any of four states [Ref. 12]:

1. Running - currently running on some processor;
2. Ready - could use a processor if one were available;

3. Wait - waiting on some event; and

4. Page wait - waiting for a page to be brought into main storage. Tasks are always added to the top of the processor queue so very quick service is given to interrupts and tasks which have just had a page brought to main storage. A task is removed from the processor queue during a wait for only the following reasons:

1. The wait was initiated by the supervisor itself for an input/output operation or a page read.

2. The byte defining the wait is in the tasks private virtual storage.

3. The task specifically requests to be removed from the queue during wait.

In other cases, such as waiting for a byte in shared storage to be cleared to zero without notification, tasks remain on the processor queue while waiting. Each task is allocated a fixed time slice to start and when this time slice is used up the task is placed at the bottom of the processor queue, after being assigned a new time slice. A new time slice is not assigned each time the task goes into a wait state so eventually the assigned time slice will be used up and the task will go to the bottom of the queue. UMMPS will select the first ready task starting from the top of the processor queue.

Another mechanism of interest is the privileged/non-privileged task assignment. This works as follows:

1. Whenever a task is initially added to the processor queue it is added as a "neutral" task.

2. When a task accumulates more than a fixed maximum allowed number of blocks (pages) of main storage a decision point is reached.

The next time it requests a main storage block it is either made privileged or non-privileged, depending on other tasks in the system.

3. If the task reaches the decision point and the number of main storage blocks allocated to privileged tasks is less than the maximum allowed then the following things are done:

a. The task is made privileged, meaning that it is allowed to get as many blocks of main storage as it wants.

b. The task is given an extra long time slice.

otherwise the task is made non-privileged and is not allowed to have a processor again until some privileged task leaves that state.

4. A task that is privileged remains so until either it uses up its extended time slice, it voluntarily asks to be placed at the end of the queue or it enters a wait state other than page wait. A task leaving privileged state is made neutral and now a non-privileged task can be made privileged.

5. A non-privileged task maintains its place on the processor queue relative to other non-privileged tasks and is made privileged, vice neutral, when started again. The privileged and non-privileged assignments are a very effective mechanism for handling overloads, as will be discussed later.

Other queues, operation of the system, etc. are not germaine to the understanding of this paper. Reference 12 contains a detailed explaination of all aspects of MTS.

The MTS system in use at NPS is Version 2.0 with PTF1 and PTF2 entered. Changes to update the system to Version 2.3 could not be entered in time for testing. The following hardware was used on the IBM 360/67 computer configured as shown in Figure 1.:

3	2365	core boxes (768K)
2	2067	central processing units
1	2314	direct access storage
8	2311	disk storage units
30	2741	terminal units
1	2301	drum

IV. EVALUATION OF MTS FROM A USERS VIEW

The requirement for some form of batch processing on a continuous basis at NPS and because CP/67 does not support batch processing has caused a deemphasis of time sharing at NPS. Only four hours of time sharing are available daily, and then a batch stream is still maintained with OS/MVT. One CPU, one core box, the 2311 disks and the drum are used for CP/67, while the other CPU, two core boxes and 2314 disks are used for OS/MVT. What the school needs is a system that is a good time sharing system with a batch capability. The remainder of this section will discribe how MTS can fulfill this need and, in some respects, surpass the existing systems.

A. SOFTWARE SUPPORT

The following compilers and interpreters are available: [Ref. 3]

ALGOL-360	PL/I-F
ALGOL-W	PL360
APL	REDUCE
ASSEMBLER-G	SLIP
BASIC	SNAP
CSMP	SNOBOL-4
FORTRAN-G	SPL
FORTRAN-H	STASS 360(Student Assembler)
GPSS	SWAT(Student Watfor)
DCALC	UMIST
LISP 1.5	WATFOR
MAD/I	WATFIV
PIL	XPL
PLC	*1 (An L6 type language)

thus, MTS supports all the languages that are available under CP/67 and OS/MVT except COBOL and BRUIN. The installation and interface of COBOL to MTS should not be difficult since MTS now has OS/MVT compatible files. As a comparison, CP/67 supports only FORTRAN-G, ASSEMBLER-F, LISP, BASIC, ALGOL-W, BRUIN, and PL/I-F. On the major languages above, OS/MVT

does not support (at this school) APL, PIL, PL360, MAD/I and an L6 type. MTS REDUCE and DCALC is equivalent to BRUIN.

A full range of system utilities are available to the users. Also, many of the operations in OS/MVT that require utilities can be performed in MTS by the normal command language. Some good examples of this are:

"$LIST *SOURCE*" to produce a listing of a card deck;

"$COPY *SOURCE* TO *PUNCH*" to duplicate a card deck; etc.

The full FORTRAN scientific subroutine package including BIMED is supported. Subroutine packages are also available for ALGOL, PL/I, and PL360.

Software support is also available for the 2250 Graphics Display Unit.

B. DEVICE SUPPORT

MTS supports all devices currently attached to the IBM 360 system at NPS. Most devices can be either directly addressed or indirectly addressed by the user. Pseudo device names [Ref. 2] can and probably should be used to allow the system to assign any available unit rather than chance that a particular one will be available. Examples are:

SOURCE	- typewriter for terminal or card reader for batch is the default. The command $SOURCE SOMENAME, where SOMENAME is a file or device name, will change the input source to almost any file or device.
SINK	- typewriter for terminal or line printer for batch is the default. The command $SINK ANYTHING, where ANYTHING is a file or device name, will change it to another file or device.
PUNCH	- the punch.

Another use is for tape usage where a pseudo device name is made up by the user and assigned to the tape to be mounted. This allows use of

any available tape drive without having to change user programs for a particular drive. Example: *TAPE* or *IN* could be assigned.

Terminal users can initiate CALCOMP plot jobs or line printer plot jobs with equal ease. Of course if you don't like terminals the job may be submitted in batch from the terminal or from a card reader.

C. COMMAND LANGUAGE AND FILES

Although a complete description of the MTS command language is contained in Reference 2, a few of the more important ones will be enumerated and explained below. Commands are normally preceded by a "$" to distinguish them from files or devices. In the examples to follow, "filename" will refer to any valid name given to a file by the user or a system library file. System library or public files are normally preceded by an "*". "FDname" will mean that either a file or a device may be specified. "par" will indicate a parameter is to be passed by the user. Examples: [] indicates optional requirements

CONTROL FDname control-command	- used for tape drivers and disks to accomplish functions such as rewind, forward space files, etc.
CREATE filename [keywords] SIZE LOC TYPE VOLUME	- used to create a file. If no keywords are specified a linefile of 100 lines on any available disk will be created. Size may be specified in number of 50 byte lines, pages or tracks. Loc can be disk or datacell. Type can be line, sequential or sequential with line numbers. Volume specifies that the file is to be created on the volume of that name.
DESTROY filename	- Does just that.
EMPTY filename	- Removes contents of file but doesn't destroy it.

COPY FDname1 [TO FDname2]	- Takes contents of file1 and puts in file2. If a device instead of a file then data is read from it or transferred to it.
EDIT filename/command	- Invokes the context editor.
LIST FDname1 [ON FDname2]	- List contents. If ON FDname2 is not specified, the default is *SINK*.
RUN objectFDname [MAP] [NOMAP] [XREF] [I/OFDnames] [limits] [PAR=parameters]	- Load and execute the object module specified. All other parameters optional. Details enumerated in References 2 and 3.
RESTART [location]	- To continue execution after an attention interrupt. If location is not specified it begins where it was interrupted.
SIGNON userid [keyword] ['comment']	- Like the OS jobcard and LOGIN for CP. The keywords can be the password, amount of time to run, number of pages of output desired, number of cards desired, the number of copies of your output desired, or a combination of keywords.
SIGNOFF	- To say you are finished for now.

Anything that can be done in CP/CMS can also be done in MTS by some equivalent command or combination of commands. The real big advantage over CP/CMS is the wide range of software and resources that are available from the terminal.

OS/360 Job Control Language (JCL) is probably the biggest thorn in the student computer user's side at NPS. JCL is not taught as a standard course at NPS and the literature is difficult for most users to understand. Usually if the catalog procedure does not cover the situation, the user will see the consultant. Reference 13 covers some of the common user changes to catalog procedure although special cases are not covered. On the other hand in MTS it is fairly simple for a user

to modify the resource requirements that are specified in the catalog procedures. For example, the program input sources can be changed by "SCARDS=NEWSOURCE", and the output area by "SPRINT=NEWPLACE". The best way to illustrate the differences in command languages and ease of use is by an example of a FORTRAN program that contains the source cards in a disk file named PROGRAM and the data for the program in a disk file named DATA.

```
CP/CMS:  LOGIN 1631p16
         HINSON                                     password
         0432CS04                                   job accounting
         IPL CMS
         FORTRAN PROGRAM
         ALTER DATA FILE * FILE FT05F001 *
         $ PROGRAM
         CP LOG

OS/360:  //HIN01631 JOB (1631,0432FT,CS04),'HINSON,E.F.-2756'
         // EXEC FORTCLG
         //FORT.FT05F001 DD UNIT=2314, VOL=SER=MARY,DSN=PROGRAM,
         //          DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=4000)
         /*
         //GO.FT05F001 DD UNIT=2314,VOL=SER=MARY,DSN=DATA,
         //     DISP=(OLD,KEEP),DCB=(RECFM=FB,LRECL=80,BLKSIZE=4000)
         /*

MTS:     $SIGNON 1631
         HINSON                                     password
         $RUN *FORTRAN  SCARDS=PROGRAM
         $RUN -LOAD#+*SSP  5=DATA
         $SIGNOFF
```

(The +*SSP added the SSP library to the load module from the FORTRAN program.)

If you wanted to run the job in MTS batch instead of at the terminal just keypunch the cards exactly like those entered at the terminal, pick up a pre-punched DECK card put it on top and read in via the card reader. If the above FORTRAN program needed more pages of output, OS would require "//GO.FT06F001 DD SPACE=(CYL,6)" to be added in the JCL where MTS requires only "PAGES=100" to be added on the SIGNON card.

File creation and use is a very simple and straightforward process in MTS. CP/CMS can probably hold its own in this respect except that it lacks the depth of file types and range of devices that MTS offers. The use of a line file in MTS provides the user with the capability of a addressing individual lines of a file. Suppose a subroutine is located between lines 35 and 50 in a source file SUB and you wish to use it when you compile another program. In MTS this is easily accomplished by "$RUN *FORTRAN SCARDS=PROGRAM+SUB(35,50)". Isolation of portions of files is not easily accomplished in either CP/CMS or OS. Temporary or work files are created in MTS by prefixing a "-" to a filename. A specific $CREATE command is not necessary to use a temporary file and the file is destroyed automatically when you sign off from the system. CP/CMS provides an equivalent capability but in OS/360 you must explicitly create temporary files by JCL statements and must specify in the disposition parameters that you want this file to be used in future job steps. Editing the contents of a file is about the same for MTS and CP/CMS, but becomes a rather difficult and involved problem in OS/360 requiring the use of a system utility and very explicit change locations.

Files in MTS can be linked together in two different ways. They can be explicitly linked by using "+" between file names or implicitly with the command "$CONTINUE WITH filename". This can be used internal to a file or within a program. Another option is to add "RETURN" after the filename to cause the program to pick up the next program line when it is finished with the linked file. The distinction is similar to a non-conditional transfer without RETURN and a subroutine call when RETURN is included. About the only way to accomplish file linking in CP/CMS is to create a new file containing the wanted files. Files may

be linked to a limited extent in OS/360 by JCL, but the easiest way is to create a new file.

A real advantage possessed by MTS is the ability to use the same files both for batch and terminal. Batch may be used to read in a large program deck and create a file and then the user can go to a terminal, manipulate the file almost at will, compile the program; execute it and still submit the program to the batch stream from the terminal. The only way to communicate between CP/CMS and OS/360 is via a card deck since magnetic tape and disk files are not compatible. Magnetic tapes and disks written in standard IBM format are directly transferrable from OS/360 to MTS.

D. USER PROTECTION

MTS file protection is better than CP/CMS or OS/360. CP/CMS provides user protection in the form of a password for system sign on and for total disk space for a private user, but not for individual files. Files may be shared only on a total disk basis and then only if programmed in by the computer center. OS/360 can provide sign on protection, individual file passwords and sharing of individual files but the average user doesn't know how to use the facility. On the other hand, MTS provides the user with easy to use facilities for password protecting a userid and files and easily sharing individual files with other users. The userid password for sign on may be changed at will by "$SET PW=NEWPASS" where NEWPASS may be any combination of up to twelve characters. If you really want to make sure that no one can read the contents of a file, a public file called *SCRAMBLE may be applied to the file. This randomly changes the contents of the file to make it completely unintelligible and requires the use of a password

before *UNSCRAMBLE may be applied. Programs may be read-only shared to all users or to specific project numbers by using the public file *PERMIT.

E. MAINTAINABILITY

The University of Michigan provides system updates to all MTS subscribers. These updates will maintain the system current to the version in use at the University of Michigan. Very few changes are required if your configuration is different from that at the University of Michigan. The MTS user group has a Newsletter that is used to circulate new ideas, new programs available and problems. MTS user manuals are updated frequently with interim documentation in the form of CCNEWS and CCMEMOS to bridge the gap between changes. A manual similar to the CP/67 Program Logic Manual is not currently available for MTS. System programmer and operator documentation is sparse but the programs are all written in relatively straight-forward IBM Assembler code (if IBM Assembler code can ever be straight-forward). Programs that need not be modified because of configuration are normally furnished as object modules for direct installation or update. Changes to programs that are effected by configuration are normally received in source form or in a form that allows application of a file called *UPDATE to produce a complete program. Easy to use public files and a powerful command language make entering changes a simple matter. The author encountered problems associated with computer availability, disk file storage space and time when trying to assemble large programs but this should not be a problem if MTS were in use on a continuous basis. One of the big disadvantages noted by the author was the cascading effect that was caused when a change to a copy section was received (Appendix A). Documentation received with change distributions

implies that the user of the changes has a good working knowledge of MTS operations and structures.

It is the opinion of the author that, if he could learn MTS and master the change system in the limited time available to him, then an experienced IBM assembly language programmer should have no problem at all in updating and maintaing MTS. The programming staff, especially Michael Alexander, at the University of Michigan was very helpful and responsive to questions and problems. Therefore, it is believed that the lack of formal maintenance support should not be the determining factor against MTS if it is determined to be the best system for NPS on its other merits. Also it is believed that the NPS computer staff have sufficient knowledge and background to become competent in the maintenance of MTS, although they may be reluctant to assume this responsibility.

F. COMMENT

In the opinion of the author, MTS is a much superior system to OS/MVT or CP/CMS from the users point of view. It is much easier to use and therefore a much more powerful and flexible system for the average user. OS/MVT is very inflexible and has an extremely difficult command language (JCL). CP/CMS lacks depth in its file capability and has limited software support. MTS has complete versitility in going from batch to terminal mode and vice versa whereas OS/MVT and CP/CMS are completely independent and, for that matter, not even compatible.

V. MEASUREMENT TECHNIQUES

The benchmark developed by Haines and Porterfield [Ref. 1] and further described by Syms, Haines and Porterfield [Ref. 14] was used as the primary load system during the measurements. The primary performance measurement was response time at the terminal and throughput for the various load conditions. The benchmark programs provided the real time for the commencement of a routine and the completion. Throughput in job completions per minute per terminal was calculated as follows:

$$TP_i = SS/(RD*NT_i)$$

where	SS	Sample size (number completed jobs)
	RD	Run duration in minutes
	NT_i	Number of terminals running program type i

The University of Michigan provided a statistics gathering program with the system that gathers onto magnetic tape all of the secondary performance measurements used by References 1 and 14 but analysis of the tape could not be completed because MTS is needed for the analysis program and insufficient computer time was available. It was decided to measure MTS as was done with TSS by primary performance alone, although statistics tapes were collected so that analysis could be done if computer time should become available.

The benchmark consisted of a set of six terminal scripts designed to simulate user conditions at a terminal. Besides printing the commencement and completion times, the scripts briefly consisted of the following:

EDIT	- A routine that performs several edit type functions such as locating a string in the program, moving the pointer up and down, typing 10 lines and then repeating itself.
FORTRAN	- A routine to compile an average size (75 card) Fortran program.
FORTEX	- A routine to simulate a compute-bound job. Execute 10,000 additions and then prints a line at the terminal.
PAGE	- A routine that uses a large array and accesses a different page for each operation.
PLILG	- A routine to compile a large (434 card) PL/I program.
PLISM	- A routine to compile a small (47 card) PL/I program.

The scripts were made up of a file of MTS commands linked back to itself so as to run in an infinite loop. A test was started by transferring the command source from the terminal to the file containing the routine. A different test could be started by merely changing the terminal command source to another file. A complete listing of the MTS command files and the benchmark programs are shown in Appendix B.

VI. TESTS CONDUCTED

A set of seven tests were run during the evaluation phase. These tests were designed to match as nearly as possible the test series of Reference 14 over the entire range of load conditions. The difficulty in obtaining computer time to run MTS precluded the running of all the tests. (Note: the tests in Reference 14 are also a sample of the original tests described in Reference 1.) The MTS test durations varied from 9 to 40 minutes depending on the load. Table I contains the number of terminals utilizing each of the scripts for the tests. Only 23 terminals were available for the tests, vice the 24 that were available for the CP/67 tests but it is felt that the lack of one EDIT program made little if any difference in the final results. The actual number of terminals does not appear to be as important to the evaluation as the actual load applied by the terminals especially if the extra terminal is a light load such as an EDIT.

Table I. Number of Terminals Using Scripts.

PROGRAM	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	RUN 6	RUN 7
EDIT	11	13	10	8	8	5	12
FORTEX	3	4	4	4	3	6	0
FORTRAN	1	1	1	1	3	5	3
PLISM	0	1	1	1	3	3	1
PLILG	1	4	7	4	4	4	7
PAGE	0	0	0	0	2	0	0
TOTALS	16	23	23	23	23	23	23
Ref 14 EQUIV. RUN NO.	R23	R43	R44	R42	R33	R31	-

Since References 1 and 14 indicated that the load on CP/67 and TSS was largely dependent on heavy paging from the PLILG and PAGE scripts, the test series for MTS was started with a light load and the number of PLILG scripts was increased until a PLILG compilation took greater than 20 minutes, then the load was reduced to provide some intermediate loads (actually intermediate overlaods) to determine the performance under other load conditions and different combinations of scripts. This allowed the measurement of the effect of the different scripts on the load. The original series was to consist of tests one through six but because of the entirely different results in MTS caused by the FORTEX routine, test seven was added to determine the effect PLILG has with no FORTEX scripts in the system.

Since References 1 and 14 showed that there was little difference between using variable terminal scripts and fixed terminal scripts, and since the latter allowed the tests to be more easily controlled, the use of fixed terminal scripts should not prejudice the results of the MTS tests.

VII. DISCUSSION OF RESULTS

Responses and throughput for each type of job for each test run is included as Table II. The response and throughput figures for FORTEX are based on the number of lines that were printed over the duration of the test. The responses for the other jobs were obtained directly from the terminal printout. To provide a comparison of the loads caused by the benchmark during the tests, Table III presents the response to the scripts for a single job in the system and for a light load of 12 jobs (3 FORTEX, 3 PLILG, 1 FORTRAN and 5 EDIT).

An attempt was made to organize the data using the same load factor presented in Reference 14 but the response and throughput indicated loads different from those obtained by the formula. The load factor formula used by Reference 14 is as follows:

$$\text{Load Factor } L = N_{EDIT}+2N_{FORTRAN}+2N_{FORTEX}+3N_{PLISM}+4N_{PLILG}+8N_{PAGE}$$

The load factor equation indicated that test 5 should have been the heaviest load but response and throughput indicated that test 3 was the heaviest load. The load factor will be discussed further after some observations are made on the results in Table II.

The EDIT routines had very little effect on either throughput or response time. The throughput for FORTRAN had the greatest variation with a relatively low throughput for test 1, a peak occuring at the test 5 load and a minimum value at the test 3 load. The effect of the privileged and non-privileged task could be seen when a large paging job at a terminal typically took about 7 minutes while the second job took about 34 minutes. The actual load appeared more dependent on both the number of PLILG and FORTEX jobs in the system. Test 7 shows that

Table II. Response and Throughput for Tests.

SCRIPT	TEST NO	RESPONSE IN MIN.				THROUGHPUT IN COMPLETIONS/ MIN/TERMINAL
		MEAN	STD DEV	HIGH VALUE	LOW VALUE	
EDIT	1	2.68	0.26	2.95	2.18	0.29
	2	2.97	0.20	3.60	2.51	0.24
	3	3.43	0.33	4.54	2.97	0.22
	4	2.96	0.25	3.59	2.50	0.27
	5	3.21	0.58	4.74	2.48	0.25
	6	3.26	0.48	4.45	2.51	0.22
	7	3.36	0.46	4.62	2.91	0.23
FORTEX	1	0.0220	0	0.022	0.022	15.15
	2	0.0239	0.0001	0.0240	0.0238	10.05
	3	0.0300	0.0002	0.0302	0.0275	8.37
	4	0.0233	0.0003	0.0236	0.0230	13.80
	5	0.0234	0.0001	0.0235	0.0233	14.35
	6	0.0250	0.0007	0.0254	0.0241	6.65
	7	- -	- -	- -	- -	- -
PAGE	1	- -	- -	- -	- -	- -
	2	- -	- -	- -	- -	- -
	3	- -	- -	- -	- -	- -
	4	- -	- -	- -	- -	- -
	5	0.97	0.40	1.49	0.22	0.31
	6	- -	- -	- -	- -	- -
	7	- -	- -	- -	- -	- -
PLILG	1	2.86	0	2.86	2.86	0.31
	2	3.97	0.41	4.47	3.26	0.17
	3	21.00	14.54	37.22	6.71	0.025
	4	2.94	0.40	3.54	2.32	0.32
	5	6.13	1.08	8.01	4.83	0.18
	6	6.85	1.48	9.43	5.73	0.063
	7	6.70	1.12	9.08	4.77	0.13
PLISM	1	- -	- -	- -	- -	- -
	2	2.36	0.23	2.63	2.16	0.29
	3	7.24	4.40	11.67	2.94	0.11
	4	1.71	0.20	1.85	1.57	0.22
	5	3.55	0.94	4.75	2.52	0.17
	6	3.61	0.52	4.39	2.99	0.19
	7	3.53	1.26	4.45	2.10	0.24
FORTRAN	1	0.37	0.21	0.59	0.18	0.57
	2	0.41	0.21	0.82	0.20	1.49
	3	1.16	1.03	3.81	0.25	0.31
	4	0.36	0.26	0.89	0.20	1.67
	5	0.60	0.28	1.19	0.22	1.86
	6	0.67	0.39	2.70	0.20	1.06
	7	0.58	0.34	1.74	0.20	1.14

PLILG jobs alone are not the total load determination. The PAGE job did not have as much effect in MTS as it did in TSS and CP/67. This again is a function of the privileged task in MTS because the whole array was loaded in core and then paging was minimized until the job started over.

TABLE III. MTS Response in Min. to Single Job and Light Load.

SCRIPT	SINGLE JOB IN SYSTEM	12 JOBS IN SYSTEM
EDIT	2.84	2.86
FORTEX	-	0.0028
FORTRAN	0.18	0.18
PAGE	0.18	-
PLILG	1.64	1.93
PLISM	1.06	-

Since the response and throughput are the real indicators of load, the load factor described above for CP and TSS was not representative of the load on MTS. The response and throughput indicate that the load was minimum at test 1 and heaviest at test 3. The order of increasing load is then tests 1, 2 & 4, 5, 7, 6, and 3. EDIT and FORTRAN jobs have very little effect on the load whereas large paging jobs, PLILG, and compute-bound jobs, FORTEX, have the greatest effect on load. The effect of compute-bound jobs is more noticable in MTS because of the single CPU queue, the fact that all jobs, including the PDP are added to the top of the queue and must contend for CPU time. If both CPU's are tied up with compute-bound jobs the paging rate goes down since the PDP doesn't run. The tests indicate that a combination of compute-bound and paging jobs is the real load indicator. A larger than normal increase in load is caused if either FORTEX or PLILG is above 4. Because of the privileged task concept, jobs with a fairly high I/O rate or short

duration in the queue are given preferential service. Thus for these reasons the following load factor was defined for MTS.

$$L = N_{EDIT}+N_{FORTRAN}+6N_{PAGE}+6N_{PLISM}+8(N_{FORTEX\leq 4}+2.5N_{FORTEX>4})$$
$$+8(N_{PLILG\leq 4}+2.5N_{PLILG>4})$$

Load factors for MTS are listed in Table IV, and contrasted with CP load factors. In many cases the MTS load factor is about twice that for the CP load factor. The reason for this "change of scale" is discussed below.

Table IV. MTS and CP Load Factors.

	RUN 1	RUN 2	RUN 3	RUN 4	RUN 5	RUN 6	RUN 7
MTS Load Factor	44	84	142	84	98	132	114
CP Load Factor	24	43	52	43	57	55	-

Figure 2 is a plot of total throughput for FORTRAN, PLISM and PLILG jobs. It indicates a classical time sharing throughput curve with a peak at a load factor of about 100. At heavy loads it begins to asymptotically approach zero. The MTS throughput factor thus becomes in a sense an indicator of underload and overload. The load factor of 100 was scaled to represent the optimum system load or 100 percent load, above this is overload and below it the load factor represents the percentage of load.

Figure 3 is a plot of response to FORTRAN, EDIT, PLISM, PLILG and FORTEX jobs vs. load factor. This is a good indicator of the effects of the MTS privileged and non-privileged task as paging load is increased. PL/I response increases much faster with load than other jobs. FORTRAN, EDIT and FORTEX response is almost flat until a load factor of about 130. At a load factor of 130 response increases rapidly for all jobs with

paging jobs being much worse than the others. The large value for the PLILG response is the averages of the first PLILG which typically took seven minutes and the second PLILG which typically took 34 minutes. This large variation is due to the privileged and non-privileged task assignments. Table V contains the available figures for CP response and throughput during the equivalent tests. MTS throughput for the FORTEX job is as much as 30 times greater than CP for an equivalent test. These results seem to indicate that the test seven load for MTS was more nearly equal to the load seen by CP for test 3. The high MTS response for PLILG jobs during test 3 would not then be representative of any load seen by CP.

Table V. CP Response and Throughput for EDIT and FORTRAN.

MTS Run Number	CP Run Number	EDIT			FORTEX
		Response	Corrected Response	Throughput	Throughput
1	R23	-	-	-	-
2	R43	1.60	2.60	0.41	-
3	R44	1.96	2.96	0.36	-
4	R42	0.96	1.96	0.39	-
5	R33	0.98	1.98	0.48	0.45
6	R31	0.93	1.93	0.42	0.76
-	512K	-	-	-	1.73

If the MTS load factor of 98 is equated to CP load factor of 57, the compile time of jobs at the same load condition for each supervisor can be compared. Figure 4 compares the response to PLILG, PLISM and FORTRAN jobs using adjusted load factors. MTS response is at least twice as good as CP for equivalent load. Figures for CP with two core boxes indicate that response would be two thirds of the figure with one core box [Ref. 1] so MTS is better than CP with two core boxes. Figure 5 compares response of MTS and CP for just the equivalent test not

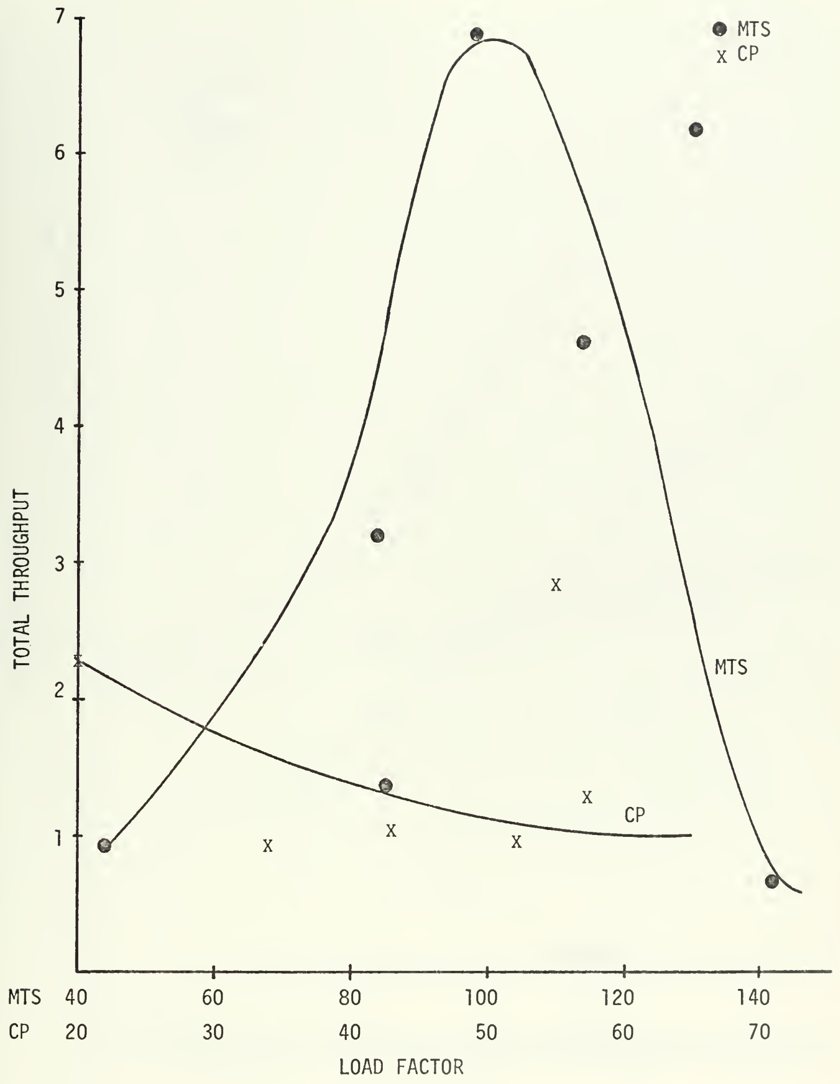

Figure 2. Total Throughput for CP and MTS FORTRAN, PLISM and PLILG Jobs vs. Load Factor.

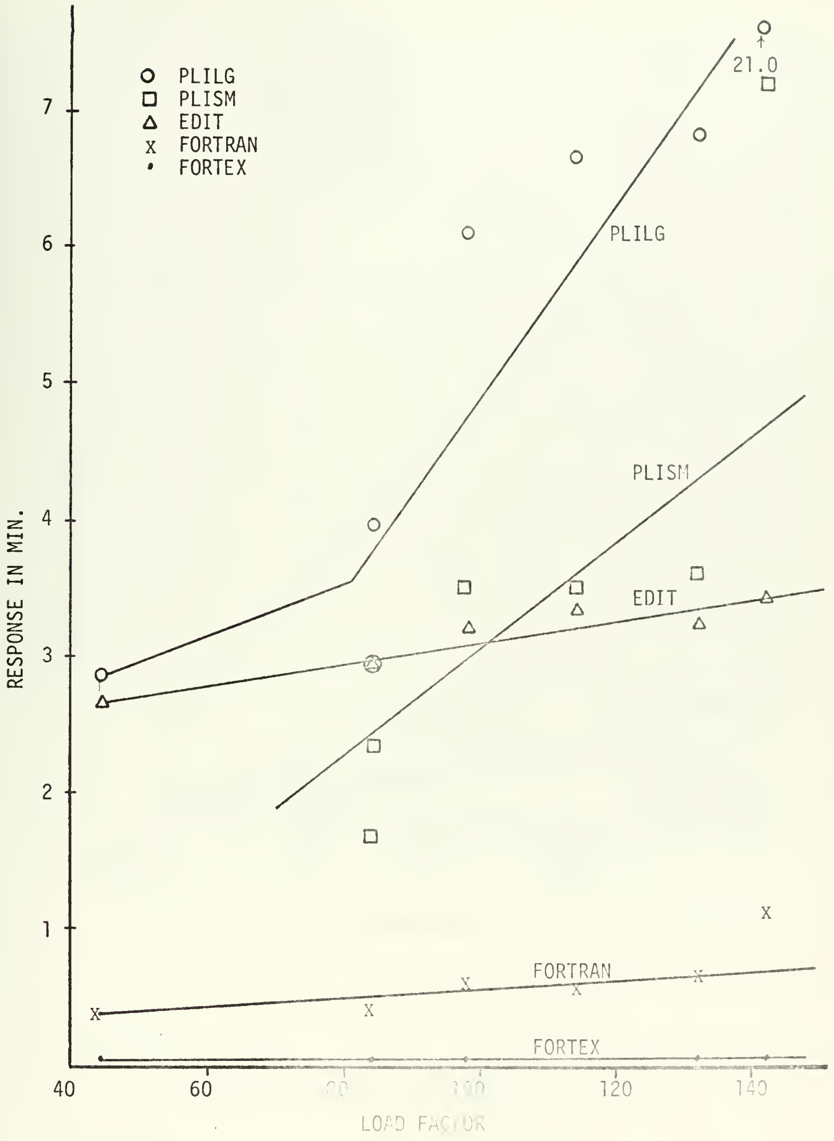

Figure 3. Response Time for MTS PLILG, PLISM, EDIT FORTRAN and FORTEX Scripts

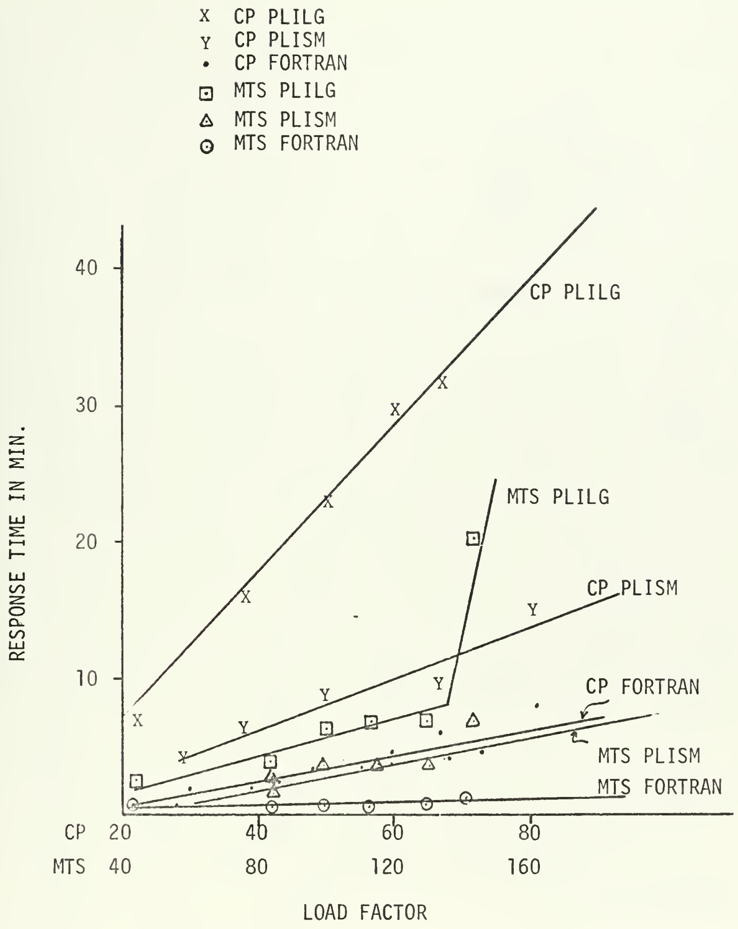

Figure 4. MTS and CP Response to PLILG, PLISM and FORTRAN Jobs vs. Load Factor.

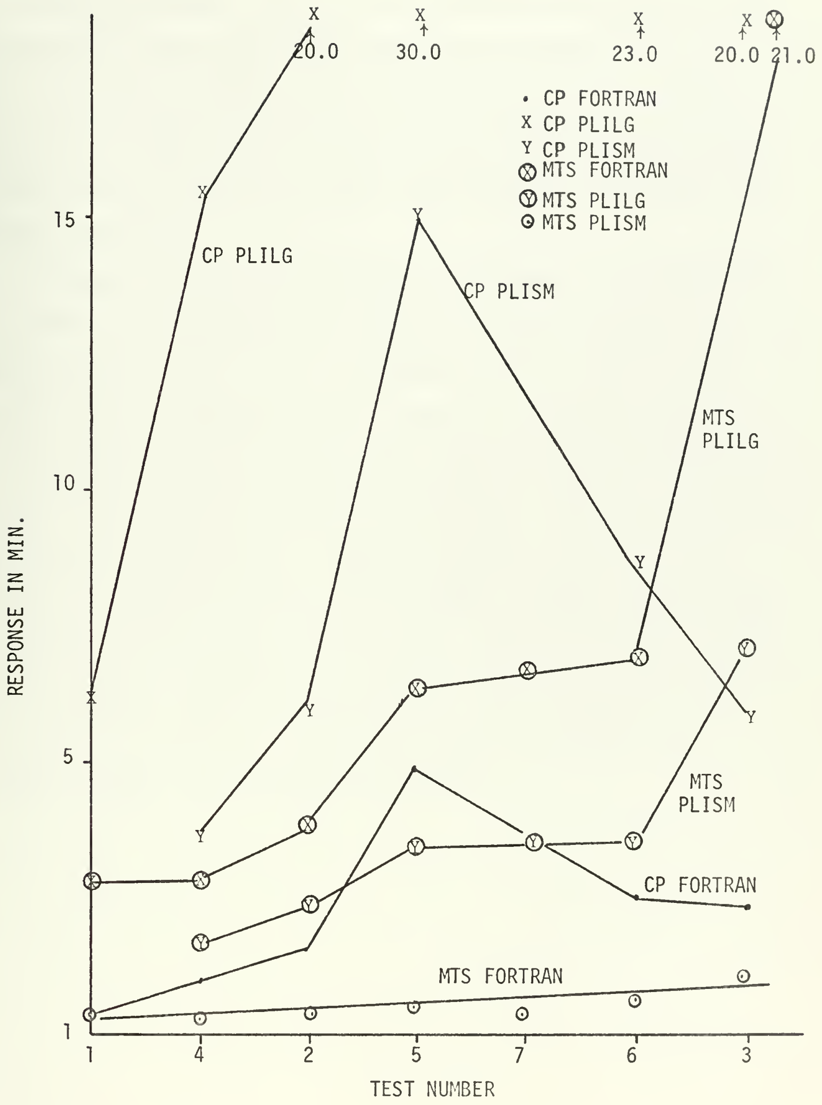

Figure 5. CP and MTS Response Time for PLILG, PLISM and FORTRAN vs Test Conducted

considering load factor. Test 3 is the only case in which MTS was not significantly better than CP but test 3 was the highest load seen by MTS. Test 5 was a higher load to CP but response by MTS was better than for a CP lighter load. It should be emphasized that even though response to heavy pagers is degraded, the response to small jobs still remains acceptable.

Although no tests were conducted in this research to compare MTS and OS/MVT, two observations were made that indicate MTS uses core memory more effectively. One, the resident portion of MTS is much smaller than the resident portion of OS/MVT. Two, under MTS the six FORTEX jobs each take three pages for a total of 18 pages (72K bytes), which is approximately one-tenth of useable core. On the other hand, OS/MVT requires 58K bytes for every job regardless of its size (in order to initiate it) and thus the six FORTEX jobs would require 348K bytes or over half of the available core. Thus MTS utilized the core memory much more effectively, especially if small jobs exist. Also, the very fast execution of FORTEX jobs indicates that MTS would probably execute a batch stream very quickly.

VIII. CONCLUSIONS

MTS response and throughput is better than CP/67 in all cases measured. In fact MTS response and throughput is better than CP/67 with two core boxes. A benchmark is an effective measurement tool, a good method of applying load on a system and an effective method of comparing the performance of different time sharing systems on the same computer. Load factors for a system can be easily determined using this loading method.

The concept of load factor is a valid and useful tool for measuring system performance. It is not generally feasible to apply the same load factor to any system and get consistent results. Load factors should be determined by measurement for each system under consideration since different scheduling algorithms for the supervisor will have a unique effect on the system load. Once the load factor equation is defined, the load factor could be calculated by the system and users could be informed of the current system load condition. For systems such as CP some method of preventing loads above 100 (new scale) is needed, such as holding jobs until later, so that the system performance will not be severely degraded. MTS already has a built-in holding technique for preventing overloads from degrading the response to all terminals with the privileged and non-privileged task assignment.

From the users view, MTS is better than the CP/67 and OS/MVT combination in use at Naval Postgraduate School for the following reasons:

1. Users need learn only one simple command language.
2. The same files may be used in both batch and terminal mode.

3. The use of virtual memory and time sharing make it possible to run any size job at any time with little regard to CPU time requested. Although some job control is still necessary to prevent very big jobs from dominating the system at prime times of the day, the privileged and non-privileged task assignments also prevents these jobs from clobbering the system. Printers and readers do not have to be secured for big jobs.

4. MTS can have many very small jobs in core because MTS can run jobs as small as 4K while OS/MVT requires at least 58K for its smallest job.

5. Since all resources, including both CPU's and the drum are used all of the time, better utilization of system resources will be obtained.

6. MTS has very easy to use and powerful file manipulation capability. Full utilization can be made of both 2314 and 2311 disk storage for files. The equivalent file capability does not exist in either CP or OS.

7. Better protection is afforded the user.

Since Haines and Porterfield [Refs. 1 and 14] showed that CP/67 provides equal performance under conditions of having much less hardware (only 1 CPU, only 256K bytes of core, and slower disks) and much better performance with equal hardware than TSS/360, then the demonstration in this research that MTS is better than CP/67 means that MTS provides the best performance, in terms of response time and throughput, of the three time sharing systems for the IBM 360/67 computer.

Although it has been shown in this research that MTS provides better performance than CP/67 as a time sharing system and although under some other test conditions (of dubious quality) MTS is better than OS/MVT in batch operation, it has still not been shown that MTS with both a batch and terminal load will outperform the CP/67-OS/MVT combination used at NPS. It is the authors belief that MTS would provide superior performance and the confirmation evaluation test should be made as soon as possible, so that the user could enjoy the benefits of MTS.

APPENDIX A

MTS OPERATORS MANUAL

This manual is intended to describe in detail the step-by-step procedure, with examples, for operating the IBM 360/67 under the Michigan Terminal System (MTS) and is intended for use by a machine operator or systems programmer who has some familiarity with the 360/67 operating procedures but none with MTS.

This manual is specifically applicable to MTS Version 2.0 with PTF1 and PTF2 as implemented at the Naval Postgraduate School. The procedures are generally applicable to all MTS versions that have not been significantly modified from the University of Michigan distributions. Computer configuration at Naval Postgraduate School is as follows:

2	2067	Central Processing Unit
3	2465	Core boxes (768K)
1	2301	Drum
1	2314	Direct Access Storage Facility
8	2311	Direct Access Storage Units
4	2400	Tape Drives

The material contained in this manual has been compiled from the MTS Manuals, CCMEMOS, CCNEWS, HASP MTS Operators Manual, Memos to the operators at the University of Michigan and personal experience of the author. More than anything else, pertinent procedures are put under one cover and organized into sections for different types of operations, [Refs. 2 - 11].

TABLE OF CONTENTS

I. GENERAL NOTES ON OPERATION FROM THE 1052 OPERATORS CONSOLE

A. INPUT FROM OPERATOR

1. Press REQUEST button.

2. The system will respond with a five digit job number and the time.

3. Type in your request.

   a. A " $ " as the first character means pass this line to HASP.

   b. Anything else means start the job with the name of the first word and pass the remaining part of the line as parameters to the job. This activates the job specified giving it the job number prefixed to the line. Example:

```
00012 08:57.57 mts LA01
```

Start the reentrant job MTS, pass device LA01 as a parameter.

The job will be assigned job number 12. Terminal LA01 will respond when activated "MTS (LA01-0012)".

4. The line is entered by pressing the carriage return, (except the first two questions on IPL and response to job dumps and superdumps must be entered by EOB). If other than carriage return is required, it will be specified in the procedure writeup.

5. If you make a mistake and wish to delete the entire line, type "?". The previous character may be deleted by typing double quote(") (logical backspace).

6. To cancel the request altogether enter a CANCEL. This is accomplished by pressing the ALTERNATE CODING key and then "0".

7. If EOB is specified as an entry, press the ALTERNATE CODING key and then "5".

B. OUTPUT FROM JOBS AND INPUT TO JOBS

1. Job output and input is prefixed by:

   a. Job number.

   b. Time.

   c. Job name.

   d. Two blanks if it is output.

   e. Two periods if input is required.

2. Type in the answer to an input request right after the periods.

C. FORM OF INPUT

1. Operator input can be either upper or lower case letters. If lower case letters are used, the system will automatically translate them to upper case.

2. Parameters are normally separated by a comma. Exceptions to this rule will be specified in the procedure.

3. An asterisk "*" as the first character of an input parameter specified the use of a public file.

D. CHANGING CONSOLES

The console at CPU01 is the primary console. Press "INTERRUPT" on CPU01 to shift the console to the one at CPU02. Shifting back is accomplished by pushing "INTERRUPT" on CPU02.

II. DEVICE NAMES TO BE USED AT NAVAL POSTGRADUATE SCHOOL

A. TERMINALS

LA01-LA11	-	Dial up lines. Terminals numberes 16-30. Hardware addresses 11-1B (All addresses are hexidecimal).
LA12-LA13	-	Not used. Hardware addresses 1C-1D.
LA14-LA27	-	Lines directly connected to the 2702 Transmission Control. Terminals numbered 1-13 and 15. Hardware Addresses 1E-2C.

B. 2250 GRAPHICS DISPLAY UNIT

DT1

C. CALCOMP PLOTTERS

PLOT

D. TAPE DRIVES

C0 - 7 track tape unit. Hardware address 0C0.

C1-C3 - 9 track tape units. Hardware addresses 0C1-0C3.

E. CARD READERS

RDR1 - Reader portion of 2540 Card Reader-Punch. Hardware address 031.

RDR2 - 2501 Card Reader

F. LINE PRINTERS

PTR1 - Line printer at hardware address 030.

PTR2 - Line printer at hardware address 070.

G. PUNCH

PCH1 - Punch portion of 2540 Card Reader-Punch. Hardware address 032.

H. DISKS

D230 - D237	-	2314 Direct Access Storage Facility. Hardware addresses 230-237.
D200 - D203	-	2311 Disk unit. Hardware addresses 200-203.
D290 - D293	-	2311 Disk unit. Hardware addresses 290-293.

I. 2301 DRUM

DRM1

III. SYSTEM STARTUP OR RESTART (IPL)

A. DISKS

1. Mount the MTS system disks on convenient drives. They are labeled MTS001 - MTS007. If two spool disks are to be used for HASP, no more than six 2314 disk packs should be used for the system disks, since spool packs can only be 2314 packs. 2311 disk packs can be used for the other system disk packs if they are desired.

2. Mount the HASP spool disks on convenient 2314 drives. They are labelled SPOOL1 and SPOOL2. Only two spools are defined to HASP so any number higher than SPOOL2 will not be recognized. 2311 packs cannot be used as HASP spools.

3. Insert address 230 in the place for the drive having MTS001 mounted. The card and tape disk writer expect MTS001 to be at address 230 but other addresses will work if IPL is from the disk pack.

4. Insert addresses in remaining drives. Location of the address for the remaining drives is not important.

5. Turn on all drives to be used.

B. 2167 CONFIGURATION CONTROL PANEL

1. Lineup the 2167 Configuration Control Panel as follows:

   a. Core storage Unit switches -

      A set to 0 TO 256K

      B set to 256 TO 512K

      C set to 512 TO 768K

   b. Channel Controller Compatibility Addressing switches -

      P1 set to CC-0/0-6

      P2 set to CC-1/0-6

c. Local/remote switch in REMOTE.

d. Processor switches -

01 - prefix and direct control both ON.

02 - prefix and direct control both ON.

e. Numbered switches 1-16 in the active "up" position.

f. I/O Control Unit switches - for both CC0 and CC1 - 2821^{I}, 2821^{II}, 2841^{I}-1, 2841^{I}-2, 2820^{I}, 2803^{I}-1, 2702^{I}-1, 2314 all in ON or "up" position.

g. Power ON light energized.

NOTE: This really amounts to all labelled toggle switches in "up" position.

2. If a piece of equipment is out of commission the switch for it may be left off and the computer will not try to access it.

C. CPU'S

Accomplish the following on both CPU's. If these procedures are not carried out at both CPU's strange results can occur at IPL.

1. Depress STOP button.

2. Disable INTERVAL TIMER.

3. Set bits 0, 21, 22 to OFF.

4. Depress START, SYSTEM RESET, CHECK RESET and ROS TRANSFER in that order.

5. Select POSITION #2 on top row of console and check for three blinking lights, (called rippling core). If lights are blinking continue otherwise accomplish step 4 again.

6. Depress LOCAL STORAGE toggle switch.

7. Set LOCAL STORAGE, INTERVAL TIMER, and bits 0,21,22 to center position.

8. Depress SYSTEM RESET.

9. Set PREFIX SELECT on one CPU to MAIN and on the other CPU to ALTERNATE.

D. ALTERNATIVE METHODS OF IPL

1. Using system disk - at CPU01 accomplish the following:

Select disk address on LOAD UNIT switches and press LOAD.

2. Using tape - Carry out the following steps:

a. Mount the system IPL tape on a convenient 9 track tape unit.

b. Load to rewind and READY the tape drive.

c. Turn PREFIX off for both CPU's on the 2167 Configuration Control panel.

d. Select tape drive hardware address on LOAD UNIT switches at CPU01.

e. Energize and READY at least one line printer.

f. Make sure address 230 is inserted for system disk MTS001.

g. Press LOAD at CPU01.

h. After the load map is printed on a line printer, press LOAD again. The system should respond "IPL WRITE FINISHED OK TO MTS001".

i. Push RESET on the tape drive containing the system IPL tape.

j. Turn PREFIX switches back on for both CPU's at the 2167 Configuration Control panel.

k. Select the address of MTS001 at CPU01 LOAD UNIT switches.

l. Press LOAD at CPU01.

m. If you make a mistake along the way, the system will respond with a message followed by "OH -- HELL.". At this point you must carry out the action required by the message, rewind the tape and start over.

3. Using cards - RDR1 is normally used for convenience (031).

Load system IPL deck in reader hopper, press END OF FILE and READY then continue with 2.c. above substituting reader address for tape unit address.

NOTE: CPU02 can be substituted for CPU01 in above.

E. CONSOLE ACTIONS AFTER "LOAD"

1. The first response of the system should be:

```
    MULTI-PROGRAMMING SUPERVISOR VERSION 01-12-70
DO YOU WANT HASP? (Y OR N)
```

2. Type "y" if you want HASP, "n" if you do not. <u>Terminate with EOB</u>.

3. System response should be:

```
ENTER TIME AND DATE
```

4. Type reply as hour, space, minute, space, am or pm, space, numeric month, space, day, space, last two digits of year. Example "03 52 am 11 15 71". <u>Terminate by EOB</u>.

5. Response by the system will be similar to following example:

```
    00001 03:52.26    INIT  TIME AND DATE HAVE BEEN SET TO 05:27.20
11-08-71
    00002 03:52.34     MTS   CONFIGURATION IS PROCESSORS: 1 2
 CCU'S: 0 1
    00002 03:52.40     MTS   STORAGE: A (FSA0) B (FSA1) C(FSA2)
    00002 03:52.45     MTS  ENTER REASON FOR RELOADING:

    00002 03:52.48     MTS..
```

6. You may type in anything you wish since the answer to this is only entered in an IPL log and has no effect on the system. Terminate with a carriage return. From now on a carriage return is the normal line entry method unless specified otherwise.

7. Response by the system will be similar to the following example:

```
00002 03:53.29    MTS  DRM2 NOT AVAILABLE
00003 03:53.32    PDP  NO PAGING DISK
00002 03:53.36    MTS        1 RECORDS RESTORED AT INITIALIZATION
00002 03:53.41    MTS  ...WELCOME TO THE U OF M AT MONTEREY
```

8. The system is now on the line except that you have no terminals or batch capability. See Sections IV and V.

IV. HASP STARTUP (BATCH PROCESSOR)

A. CARRY OUT THE FOLLOWING TO GET THE HASP JOB STARTED:

1. Request an entry and type "hasp", space, SPOOL1 device name, space, SPOOL2 device name. If only one spool is used the second device name is omitted. Example" "hasp d231 d233".

2. The system should respond as follows: (job number and time have been left off for simplicity)

```
HASP   $  SPECIFY SPOOL OPTIONS -- VERSION 2.1.2

HASP..
```

3. Reply should be of the form option1,option2,...,option(n).

Options 1-n can be any of the following:

COLD	- Jobs contained on the spool disks before the cold start are ignored. Only jobs entered after the cold start are processed.
<u>WARM</u>	- Jobs active at the time of last system termination are restarted.
FORMAT	- All SPOOL disks should be formatted (implies COLD). This should be used only for a new spool pack or if problems exist on a pack since it takes about 15 minutes per disk.
<u>NO FMT</u>	- Only un-formatted SPOOL disks should be formatted.
<u>REQ</u>	- SPOOL requests are to be entered before processing starts.
NO REQ	- No SPOOL requests are to be entered before processing starts.

All underlined options are implied or default and need not be entered.

If you make an error the system will reply:

```
HASP  $SYNTAX ERROR -- RESPECIFY OPTIONS
```

and you must start over.

4. If the NO REQ option was not specified the system will respond:

```
HASP  ENTER $SPOOL REQUESTS
```

5. The following is an example to start processing with RDR1, PTR1, and PCH1 also available to HASP. Lower case letters are operator entries, upper case letters are system replies. Anything in parenthesis is a comment related to the entry:

```
$start system         (Starts HASP processing jobs)

HASPLING  $*OK

$start more rdr1      (The more specifies that the reader should
                       continue to process jobs)

HASPLING $*OK

$start pn ptr1        (pn indicates a pn print train on the printer)

HASPLING $*OK

$start pch1

HASPLING $*OK
```

6. For a more detailed explanation of HASP and more detailed operator actions and commands see Reference 11 MTS HASP OPERATORS GUIDE. If your installation has the public file *HSP in existence the whole sequence above could have been accomplished by one entry:

```
mts *hsp
```

B. AFTER HASP IS RUNNING

Additional devices may be put on the line by appropriate commands. The MTS HASP OPERATORS GUIDE is excellent for running HASP so it is not duplicated in this manual.

V. TERMINAL STARTUP

A. 2741 TERMINALS

These terminals are LA01-LA27. Each must be started individually by a command of the following form:

```
mts la01
```

If you have the public file *LAS at your installation all terminals may be started by the single command:

```
mts *las
```

B. 2250 GRAPHICS DISPLAY UNIT

The 2250 is started by issuing the command:

```
mts dtl
```

C. CALCOMP PLOTTERS

The CALCOMP plotters can be started by entering:

```
mts plot
```

Names of devices at other installations may be different. Consult your device tables for the correct names to be used for a particular device.

VI. NORMAL SYSTEM OPERATIONS

A. CONSOLE REACTIONS TO A HASP BATCH JOB

Unless told to do otherwise [Ref. 11] a HASP job will cause the following responses at the console:

```
HASPLING $ JOB 001278  ON RDR1   -- B04.  TABLES ASSEMB
HASPLING $ JOB 001278  -B04.-EXEC AS JOB 00012
HASPLING $ JOB 001278  END EXECUTION.
HASPLING $ JOB 001278  PRINTING ON PTR1
HASPLING $ JOB 001278  PUNCHING ON PCH1
HASPLING $ JOB 001278  IS DONE.
```

B. TAPE MOUNT REQUESTS

If a magnetic tape mount is required for a job a message of the following form is received:

MTS HB or T (#of drives required type of drive) Rackname ON type of drive, RING IN or OUT, 'Comment from user'.

HB indicates HASP batch, T indicates terminal user.

Rackname is the storage rack location of the tape, normally the label.

Ring refers to the write protect ring.

A specific example for a HASP batch job requiring tape NPS275 follows:

```
MTS  HB (1 9TP) NPS275 ON 9TP, RING OUT, 'DIST 2.1' **************
```

1. Acceptable replies are as follows:

   a. Tape has been mounted on a drive with no problems reply "ok RACKNAME DEVICE". As an example for above tape:

```
ok nps275  c2
```

The system replies: "MTS ACCEPTED".

2. If there is no tape drive available reply "na RACKNAME COMMENT to say why" or "ab RACKNAME COMMENT"

   NA is normally used if only one drive is requested.

ABORT is normally used if more than one drive is requested since it cancels all mount requests from the same *mount job. Terminals must request again, HASP jobs are held.

The underlined portion is the minimum allowed entry. This notation applies to the remainder of the manual.

3. If for some reason you must have the request repeated, enter REPEAT. A terminal user will have to retype his request, HASP will request it again automatically.

4. When a drive becomes available entering RERUN BACKNAME for a HASP job will release the job from the hold status and rerun the job.

5. When the system is finished with the tape it will tell you to remove the tape from the device and unload the tape drive. Example:

```
MTS  REMOVE TAPE FROM     C2 ******************************
```

C. SYSTEM DISK AND PRIVATE DISK ENTRIES

1. If you don't mount all of the system disks as entered in the system tables (NPS has MTS001-MTS007 defined), the system will complain when a user asks to create a file. The following example assumes MTS007 was not mounted:

```
MTS  MOUNT VOLUME MTS007 AND RETURN, OR TYPE "CANCEL" IF NOT
AVAILABLE
MTS..
```

You must now either mount MTS007 on an available disk drive, ready the drive and press carriage return or type in "cancel". You could have avoided this complaint by carrying out procedures for removing a disk in Subsection 2. below.

2. To add a private user disk, add a system disk or remove a disk from the system tables you may execute a program named *DSK. This is

initiated by typing "mts *dsk" at the console. After the program is executing requests may be entered as follows:

a. ADD - Enter "add name(s)" where name(s) is a single volume label of the disk pack or a list of volume labels separated by commas or blanks.

b. REMOVE - Enter "remove name(s)". Comments of 2.a above.

c. FORGET - If an address of a volume previously known to the system is changed, it is necessary to tell the system to forget the old address and look for a new one. This is done by entering "forget name(s)".

d. LIST - To find out which disks are attached by volume name and location enter "list".

e. DONE - When you want to quit using *DSK then enter "done".

D. RUNNING JOBS SIMILAR TO TERMINAL AND BATCH FROM THE CONSOLE

1. Normal jobs - To run normal every day type jobs enter "mts oper". All that is necessary now is to sign on in the normal way except that a password is not required. Example: (lower case is operator entry and upper case is computer typeout)

```
mts oper

     MTS..sig mts

     MTS  **LAST SIGNON WAS:  04:27.51   10-05-71
     MTS    USER "MTS." SIGNED ON AT  04:56.27 ON 11-15-71

     MTS..
```

2. Special jobs - Certain special jobs are listed under a particular userid and must be run or changed using that userid.

a. System non-public programs are listed under userid MTS. This is a privileged userid to the system and can be used to modify protected files.

b. Programs for accounting files are listed under userid ACC. This userid should be used for system accounting file maintenance.

c. Some special operator utility programs are listed under userid SYS.

d. *SORT and *DEB are listed under userid DBS.

e. *FIX is listed under userid FIX. This is a privileged id.

f. *RST is listed under userid RSTR.

3. Library files - These are files that start with "*" (also called public files). Library files can only be changed from the operators console and then only with a privileged userid. MTS is normally a good userid for signon. Any library file can be used in normal programs from the console the same as from a terminal or in batch. An attempt to change a library file will result in the following message:

```
MTS  ENTER PASSWORD OR "CANCEL" TO CHANGE LIBR. FILE
MTS..
```

The password is composed of the first four characters of the date and the first four characters of the time for the "MTS.." followed by qqsv. Example: Date is 10-05-71, time is 05:22.34 so the password is 10-005:2qqsv.

E. MONITORED JOBS THAT REQUIRE OPERATOR PERMISSION TO CONTINUE

Some jobs require the operator's permission to accomplish. A good example is:

```
MTS    MTS ACCOUNTING FILE MAINTENANCE--ENTER VERIFICATION OR
"CANCEL"
```

To verify as all right to continue enter "ok" or "!", or stop the job by entering "cancel".

F. MESSAGES

1. To send a message to users enter "broadcst", wait for the response "BROADCST.." and then type in the desired message.

2. To change the message received by all users at signon, enter "signonm" wait for the response "SIGNONM.." and then type in the message. This message is also printed on the first page of all batch jobs.

G. MONITORING THE SYSTEM

1. To find out what jobs are currently being run by the system, enter "tasks". This will cause a printout at the console of JOB#, PAGES IN USE, NAME, JOB TABLE #, and PARAMETERS AND DEVICES BEING USED for each job currently in the system.

2. To display the HASP queue enter "mts *que".

3. Volume 2 of the MTS Manual lists several public files that will display information useful to the operator. You may signon with a convenient userid and issue a $run command for these files. As an example "*USERS" will display the number of batch and terminal users. There are several utility programs listed in Volume 2 of the MTS Manual [Ref. 3] that are of interest to the operator. You may wish to run a batch job for those jobs that do a lot of printing to reduce the output at the slow speed console. For instance, the disk volume table of contents printout, *LISTVTOC, should be run as a batch job.

VII. PROBLEMS

A. DEVICE PROBLEMS

1. If for some reason you wish a device to no longer be available to the system, enter "offline device-name". This modifies the system tables so that the device name is no longer available.

2. The device may be made available to the system again by entering "online device-name".

3. The switch at the configuration panel may be turned off to accomplish an "offline" as well.

B. SYSTEM PROBLEMS

1. Should a message similar to the following appear at the console:

```
00011 05:17.50     MTS

***************
HELP -- SNARK IN MTS!!
***************
```

A job has raised a system problem that requires operator action. Enter "stop job#" to clear it out of the system. Additionally if the job is a HASP job you must enter "$terminate Hasp job#".

2. If it is just one of those days you can't win and

```
***SUPERVISOR DUMP DO NOT CANCEL***
**DUMP** SPECIFY TAPE DRIVE...
```

should appear, you have a bad problem. Mount a tape on a convenient 9 track drive, ready it and type the device name after the periods. Terminate with EOB. You may be lucky and have the system recover itself but in all probability you will have to re-IPL.

C. TO GET RID OF A JOB

1. A job may be terminated and the job tables and queues cleared by entering "stop job#". If this was a terminal job you must again enter "mtslaxx", where xx is the number, to get the terminal back on the line. This is basically equivalent to a forced signoff.

2. To get rid of a job in a hurry enter "blast job#". This wipes out the job without even bothering to force a signoff. Batch jobs must be $terminated and terminals must be restarted.

3. To generate an attention interrupt for a job, enter "goose job#".

4. Sometimes a "stop", "blast" or a SNARK will lock out the userid that was associated with the job you killed. The userid may be reset by "mts *fix" from the console.

VIII. HINTS TO BEGINNING SYSTEM PROGRAMMERS

A. SYSTEM CONFIGURATION

If a change to one of the large system programs e.g., HASP, SUPERVISOR or MTS is to be entered, make sure at least two 2314 disk packs are available to the system. The scratch files from the assembler will overflow the available space if only one 2314 pack is used. For some reason, unexplained by the author at this time, 2311 packs will not work as scratch areas.

B. COPY SECTIONS

Several programs entered in the library or as user MTS are not in object form but rather in assembler source. These programs are copied and used in the assembly of other programs. A change to one of the copy sections will require reassembly of the programs that use it. The following is a list of the copy sections and the programs that use them:

1. *LLMPSEQU - used by:

SUPER	JBRP	BROADCST	LLXU
MTS	FBUF	TABLES	CHKSUM
GETFREE	FIOD	CONFIG	OLTSSUB
KWIC	TASKS	MOUNT	*CONFIG
USERS	JOBS	HASP	PLIMIT

2. EQU - used by:

MTS	CHKSUM
KWIC	LLUX
GETFREE	PLIMIT
GUINFO	

3. EQU2 - used by HASP.

4. CONTAB - used by:

MTS	GUINFO
HASP	CHKSUM
KWIC	LLUX
GETFREE	PLIMIT

5. RHTABLE - used by MTS and KWIC

6. PSA - used by SUPER, CONFIG, INITLOG and USERCONF.

C. HIDDEN INFORMATION

Some distribution tapes for system changes are received with the documentation hidden in the tape itself. If the tape is in *FSAVE format, run the program *FSAVE using the tape as input and list the files on the tape. The writeup should be the first file. Create a file by that name, restore the first file to it and list the file to obtain the information about the change.

Quite a bit of information about the contents and how to use the files on a tape is contained in the LEM:DDCOM listing of the tape contents. Items such as copy sections used, other files required with this one and good comments about the file are contained in this listing. A most valuable aid was the version 2.0 listing that came with the original NPS distribution.

D. GENERAL INFORMATION

1. The batch mode is probably the best method of assembling changed system programs mainly due to the heavy printer and punch requirements. Be careful of changes to TABLES, VTOC and HASP since the NPS version has been modified from the distributed version. Don't forget to attach the system macro library, *SYSMAC, to logical unit 0 when using *ASMG. Some programs don't use these macros and you may get away with it but it is rather frustrating to wait for a ten minute assembly and 20-30 minute printout of the supervisor to find many pages of errors because you forgot to attach the macro library.

2. Changes to library files must be made from the operators console. Using Batch will only result in failure.

3. In all probability changes will require that you build a new IPL tape. The public file *UPDATE is easy to use for this job. Its use is covered very nicely in Volume 2 of the MTS Manual. Use the old IPL tape as a source and enter change commands and new object decks via a batch job.

4. Volume 1 of the MTS Manual covers use of MTS in terminal and batch mode. This book is well written and is a must to have around when working with the system.

APPENDIX B

SCRIPTS USED BY THE BENCHMARK

EDIT SCRIPT

```
$$RUN *TIME
$$RUN *ED;SCARDS=EAS:EDITS
$$RUN *TIME
$CONTINUE WITH EAS:EDITOR
        FILE EAS:EDITS CONTAINS
XEC $EDT
EDIT EAS:LOOP
TOP: SCAN@A *F 26 '50'
L *F
+3
L *L
L *F
P 10 20
STOP
$EDT
```

FORTRAN SCRIPT

```
$$RUN *TIME
$$RUN *FORTRAN;SCARDS=EAS:FORTRAN PAR=NOSOURCE,NOMAP,NOLOAD
$$RUN *TIME
$$CONTINUE WITH EAS:FORCOMP
```

FORTEX SCRIPT

```
$$RUN *TIME
$$RUN EAS:FORTEX
$$RUN *TIME
$$CONTINUE WITH EAS:FORTX
```

PAGE SCRIPT

```
$$RUN *TIME
$$RUN EAS:PAGE
$$RUN *TIME
$$CONTINUE WITH EAS:PAGER
```

PLISM SCRIPT

```
$$RUN *TIME
$$RUN *PL1;SCARDS=EAS:PLISM PAR=NS,NLD,NS2,NOL
$$RUN *TIME
$$CONTINUE WITH EAS:PLILIT
```

PLILG SCRIPT

```
$$RUN *TIME
$$RUN *PL1;SCARDS=EAS:PLILG PAR=NS,NLD,NS2,NOL
$$RUN *TIME
$$CONTINUE WITH EAS:PLIBG
```

SOURCE CARDS FOR BENCHMARK SCRIPTS

PROGRAM USED BY FORTRAN AND EDIT

```
C     PROGRAM TO COMPILE THE ROSTER OF SUBMARINE OFFICERS ATTACHED TO   PAU00010
C     THE NAVAL POST GRADUATE SCHOOL                                    PAU00020
      DIMENSION XNA(19),RANK(4),WIFE(9),ADDR(5),DUTY(5),DATE(5)         PAU00030
C     READ THE NUMBER OF DATA CARDS NOT INCLUDING THE YELLOW ONE, THE   PAU00040
C     NUMBER OF COPIES OF THE DIRECTORY DESIRED, AND THE DATE           PAU00050
      READ(5,1)NCAR,NCOP,DATE                                           PAU00060
    1 FORMAT(2I4,5A4)                                                   PAU00070
C     THIS LOOP READS ALL THE DATA CARDS AND PUTS THE DATA ON DISK      PAU00080
      DO 90 I=1,NCAR                                                    PAU00090
      READ(5,8)XNA,RANK,NYR,NCRS,NTEL,WIFE,ADDR,CITY,NDPN,DUTY,SMC      PAU00100
    8 FORMAT (19A1,4A1,I2,I3,I7,9A1,5A4,A1,I1,5A2,I4)                   PAU00110
   90 WRITE (2,8)XNA,RANK,NYR,NCRS,NTEL,WIFE,ADDR,CITY,NDPN,DUTY,SMC     PAU00120
      END FILE 2                                                        PAU00130
C     THIS LOOP PRINTS OUT THE ROSTER, REWINDS THE DISK AND CONTINUES   PAU00140
C     DOING THIS UNTIL NCOP COPIES ARE PRINTED.                         PAU00150
      DO 91 I=1,NCOP                                                    PAU00160
      REWIND 2                                                          PAU00170
      WRITE (6,10)                                                      PAU00180
   10 FORMAT ('1',24X,'DIRECTORY OF SUBMARINE OFFICERS ATTACHED TO THE NPAU00190
     1AVAL POSTGRADUATE SCHOOL',///,' 1. THIS DIRECTORY IS UNOFFICIAL ANPAU00200
     2D IS INTENDED PRIMARILY FOR SOCIAL REFERENCE.',//)                PAU00210
      WRITE (6,11)                                                      PAU00220
   11 FORMAT (' 2. THE SUBMARINE LIASON OFFICER IS LCDR PHIL OCONNELL.  PAU00230
     1HIS OFFICE IS LOCATED IN ROOT',/,'   HALL, ROOM 214, ENGINEERING  PAU00240
     2SCIENCE CURRICULUM OFFICE. PHONE NUMBER 646 2426.'//)             PAU00250
      WRITE (6,12)                                                      PAU00260
   12 FORMAT (' 3.  THE SUBMARINE OFFICERS SOCIAL CHAIRMAN IS LCDR  RAY PAU00270
     1 ANDERSON, PHONE 373 5150.',//)                                   PAU00280
      WRITE (6,13)                                                      PAU00290
   13 FORMAT (' 4. THE CHAIRMAN OF THE SUBMARINE WIVES GROUP IS  JANE   PAU00300
     1 WHITE, PHONE 372 7053.',//)                                      PAU00310
      WRITE(6,14)                                                       PAU00320
   14 FORMAT (' 5. DOLPHIN PLAYING CARDS, NAPKINS, CALENDARS, COOKBOOKS,PAU00330
     1ETC, CAN BE OBTAINED OR ORDERED THROUGH',/,'  MRS JOAN EGAN, PHONEPAU00340
     2 375 1710.',//)                                                   PAU00350
      WRITE (6,15)                                                      PAU00360
   15 FORMAT (' 6. CHANGES OR ADDITIONS TO THIS DIRECTORY SHOULD BE  BROPAU00370
     1UGHT TO THE ATTENTION OF THE SUBMARINE LIASON OFFICER',/,'  OR THEPAU00380
     2 SOCIAL CHAIRMAN.')                                               PAU00390
      WRITE(6,5)                                                        PAU00400
    5 FORMAT (//////////,1X,'CITY LEGEND:',//,' C CARMEL',11X,' H CARMELPAU00410
     1 HEIGHTS     V CARMEL VALLEY     D DEL REY OAKS     M MARINA',/,' PAU00420
```

```
     2P MONTEREY PEEN CC      T MONTEREY      B PEBBLE BEACH      G PACIFI PAU00430
     3C GROVE      S SEASIDE',/,' A SALINAS      Q BOQ',///)               PAU00440
      WRITE (6,6)                                                          PAU00450
 6    FORMAT (' CURRICULUM CODE:',/,' 360 OPERATIONS ANALYSIS',/,' 367 MPAU00460
     1ANAGEMENT (COMPUTER SYSTEMS)',/,' 368 COMPUTER SCIENCE',/,' 372 MEPAU00470
     2TEOROLOGY',/,' 380 ADVANCED SCIENCE',/,' 440 OCEANOGRAPHY',/,' 460PAU00480
     3 ENGINEERING SCIENCE',/,' 461 BACHELOR OF ARTS/SCIENCE',/,' 521 NUPAU00490
     4CLEAR ENGINEERING',/,' 530 WEAPONS ENGINEERING',/,' 535 UNDER WATEPAU00500
     5R PHYSICS',/,' 570 NAVAL ENGINEERING',/,' 590 ENGINEERING ELECTRONPAU00510
     6ICS',/,' 600 COMMUNICATIONS ENGINEERING',/,' 610 AERONAUTICAL ENGIPAU00520
     7NEERING',/,' 620 COMMUNICATIONS MANAGEMENT',/,' 817 NAVAL MANAGEMEPAU00530
     8NT',/,' 999 P.G. SCHOOL STAFF')                                      PAU00540
      WRITE (6,2)                                                          PAU00550
 2    FORMAT ('1',26X,'YEAR CURR PHONE',40X,'NO.OF LAST DUTY')             PAU00560
      WRITE (6,3)                                                          PAU00570
 3    FORMAT (' NAME',17X,'RANK GRP   NO. NUMBER  WIFE       ADDRESS',14XPAU00580
     1,'CITY  CHILD  STATION',17X,'SMC',/)                                 PAU00590
      WRITE (6,4)                                                          PAU00600
 4    FORMAT ('+','----',17X,'---- ---    --- ------  ----      -------',PAU00610
     114X,'----  -----  -------',17X,'----',//)                            PAU00620
 30   READ (2,8,END=50)XNA,RANK,NYR,NCRS,NTEL,WIFE,ADDR,CITY,NDPN,DUTY     PAU00630
     2,SMC                                                                 PAU00640
      WRITE (6,9)XNA,RANK,NYR,NCRS,NTEL,WIFE,ADDR,CITY,NDPN,DUTY,          PAU00650
     1SMC                                                                  PAU00660
   9  FORMAT (1X,19A1,2X,4A1,2X,I2,3X,I3,1X,I7,1X,9A1,1X,5A4,2X,A1,6X,I    PAU00670
     11,4X,5A2,14X,I4)                                                     PAU00680
      GO TO 30                                                             PAU00690
 50   WRITE (6,7) DATE                                                     PAU00700
  7   FORMAT (///,40X,5A4)                                                 PAU00710
 91   CONTINUE                                                             PAU00720
      STOP                                                                 PAU00730
      END                                                                  PAU00740
```

PROGRAM USED BY FORTEX

```
1000 WRITE(6,2000)
     I=0
  10 I=I+1
     A=2. + 2.
     IF(I.EQ.10000) GOTO 1000
     GOTO 10
2000 FORMAT('************')
     END
```

PROGRAM USED BY PAGE

```
      DIMENSION A(1030,30)
5     WRITE (6,100)
      DO 10 I=1, 1030
      DO 10 J=1, 30
10    A(I,J)=100.+50.
100   FORMAT(' START OVER')
      STOP
      END
```

PROGRAM USED BY PLILG

```
 PROJECT:PROC OPTIONS(MAIN);

       /* THIS PROGRAM WAS DONE BY THE COOPERATIVE EFFORTS OF
          J. BAIRD, B. HAINES, R. SPENCER, AND R. WOOLS   */

       /* THIS PROGRAM CONSTRUCTS A RING STRUCTURE USING THE
          26 LETTERS OF THE ALPHABET AND USES THESE LETTERS
          , WHICH CORRESPOND TO THE FIRST LETTER OF A PERSON'S
          LAST NAME, AS ENTRY POINTS TO BUILD A SUBRING
          STRUCTURE OF RECORDS CONTAINING DATA ITEMS: LAST
          NAME, FIRST NAME, OCCUPATION, CITY, AGE.
          THE PROGRAM ALLOWS ONE TO INSERT NEW RECORDS,
          DELETE RECORDS, CHANGE RECORDS, LOCATE RECORDS
          ACCORDING TO LAST NAME AND FIRST NAME, AND PRINT
          A LISTING OF ALL RECORDS IN THE SYSPRINT FILE  */

        DCL ALPHABET CHAR(26) INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ');
        DCL ANSWER  CHAR (10)VAR ;
        DCL BUFFER CHAR(30);
        DCL SYSIN FILE STREAM ENVIRONMENT(F(80));
             OPEN FILE(SYSIN);
        DCL(FIRST_LTR,FRONT,BACK) PTR;
        DCL(NLAST,NFIRST,NOCC,NCITY,OLD_LAST,OLD_FIRST) CHAR(10);
        DCL (POSITION,LISTING) FIXED BIN(1) INIT(0);
        DCL NAGE CHAR(3);
        DCL BUILD ENTRY (CHAR(10));
        DCL BUILD_REC ENTRY (PTR,PTR) ;
        DCL LOOK_UP ENTRY (CHAR(10),CHAR(10)) RETURNS (PTR);
        DCL BUILD_ALPHA ENTRY (PTR) RETURNS (PTR);
        DCL FIND ENTRY (CHAR(10), CHAR(10)) ;
```

```
DCL SYSPRINT FILE STREAM PRINT ENV (F(120));
DCL ALPHA_ARRAY (26) PTR;
DCL LABEL LABEL;
DCL ERROR ENTRY(LABEL);


DCL 1 ALPHA BASED (ALPHA_PTR),
      2 LTR CHAR(1),
      2 NEXT_LTR PTR,
      2 LIST_PTR PTR;


DCL 1 RECORD BASED(NEW_RECORD),
      2 NAME,
           3 LAST CHAR(10),
           3 FIRST CHAR(10),
      2 OCCUPATION CHAR(10),
      2 CITY CHAR(10),
      2 AGE CHAR(3),
      2 NEXT_RECORD PTR;


/* THIS BLOCK STRUCTURE HANDLES THE TERMINAL I/O FOR THE
   PROGRAM  */


 ON ENDFILE(SYSIN) BEGIN;
      DISPLAY(' ');
      DISPLAY( ' ');
      DISPLAY('DO YOU WANT TO PROCESS ANY RECORDS?');
      DISPLAY(' ');
      DIS1:DISPLAY('TYPE IN YES OR NO IN QUOTES');
      DISPLAY(' ');
      DISPLAY(' ') REPLY(BUFFER);
      ON ERROR CALL ERROR(DIS1);
      GET STRING(BUFFER) LIST(ANSWER);
      IF ANSWER='NO' THEN EXIT;
      IF ANSWER¬='YES' THEN DO;
           DISPLAY('YOU DIDNT SAY YES OR NO, WHAT THE HELL DO'||
           ' YOU WANT????');
           GO TO DIS1;
      END;
      DISPLAY('INDICATE WHAT YOU WANT BY TYPING IN ONE OF THE'||
        ' FOLLOWING:');


/* THE FOLLOWING GROUPS ARE USED TO INSERT NEW RECORDS,
   DELETE RECORDS, AND CHANGE RECORDS WITHIN A SUBRING  */
```

```
DO WHILE(BUFFER¬='');
DISPLAY(' ');
DIS2:DISPLAY('INSERT,DELETE,CHANGE,LOCATE,LISTING OR QUIT'
     ||' IN QUOTES');
DISPLAY(' ');
DISPLAY(' ') REPLY(BUFFER);
ON ERROR CALL ERROR(DIS2);
GET STRING(BUFFER) LIST(ANSWER);
IF ANSWER ='INSERT'|ANSWER ='DELETE'|ANSWER ='CHANGE'
     |ANSWER='LOCATE'|ANSWER ='LISTING'|ANSWER ='QUIT'
THEN DO;
IF ANSWER='INSERT' THEN DO;
     DIS3:DISPLAY('INDICATE WHAT YOU WANT CHANGED BY'||

     'TYPING IN ONE OF THE FOLLOWING:');
     DISPLAY('LAST,FIRST,OCCUPATION,CITY, AGE IN QUOTES');
     DISPLAY(' ');
     DISPLAY(' ') REPLY(BUFFER);
     ON ERROR CALL ERROR(DIS3);
     GET STRING (BUFFER) LIST(NLAST,NFIRST,NOCC,NCITY,
          NAGE);
     CALL BUILD(NLAST);
END;
IF ANSWER='DELETE' THEN DO;
     DIS4:DISPLAY('INDICATE WHICH RECORD IS TO BE'||
     'DELETED BY TYPING IN THE LASTNAME AND THE FIRST'
     ||'NAME IN QUOTES')REPLY(BUFFER);
     ON ERROR CALL ERROR(DIS4);
     GET STRING(BUFFER) LIST(NLAST,NFIRST);
     CALL FIND(NLAST,NFIRST);
     IF POSITION=0 THEN
          IF FRONT->NEXT_RECORD=NULL & BACK= ALPHA_PTR
          THEN DO;
               LIST_PTR=NULL;
               FREE RECORD;
          END;
          ELSE DO;
               BACK->NEXT_RECORD=FRONT->NEXT_RECORD;
               FREE RECORD;
          END;
END;
IF ANSWER='LISTING' THEN DO;
     LISTING=1;
     CALL PRINT;
     LISTING=0;
END;
```

```
IF ANSWER='LOCATE' THEN DO;
    DIS10:DISPLAY('INDICATE WHOSE RECORD YOU'||
    'WANT LOCATED BY TYPING IN THE LAST AND '||
    'FIRST NAME IN QUOTES')REPLY(BUFFER);
    ON ERROR CALL ERROR(DIS10);
    GET STRING(BUFFER) LIST(NLAST,NFIRST);
    CALL FIND(NLAST,NFIRST);
    IF POSITION=0 THEN DISPLAY(LAST||FIRST||OCCUPATION||
         CITY||AGE);
END;
IF ANSWER='CHANGE' THEN DO;
    DIS5:DISPLAY('CHANGE WHAT?');
    DISPLAY('INDICATE EITHER NAME,CITY,'||
    'OCCUPATION,OR AGE IN QUOTES')REPLY(BUFFER);
ON ERROR CALL ERROR(DIS5);
    GET STRING (BUFFER) LIST(ANSWER);
    IF ANSWER='NAME' THEN DO;
    DIS6:DISPLAY('TYPE IN THE FOLLOWING ITEMS IN '||
         'QUOTES :OLD_LAST,OLD_FIRST,NEW_LAST,NEW_FI'||
              'RST')REPLY (BUFFER);
    ON ERROR CALL ERROR(DIS6);
         GET STRING(BUFFER) LIST(OLD_LAST,OLD_FIRST,
              NLAST,NFIRST);
         CALL FIND(OLD_LAST,OLD_FIRST);
    IF POSITION=0 THEN DO;
                   BACK->NEXT_RECORD=FRONT->NEXT_RECORD;
                   NOCC=OCCUPATION;
                   NCITY=CITY;
                   NAGE=AGE;
                   FREE RECORD;
                   CALL BUILD(NLAST);
              END;
    END;
    IF ANSWER ='CITY' THEN DO;
    DIS7:DISPLAY('TYPE IN LAST, FIRST AND THE NEW CITY'||
    'IN QUOTES')REPLY(BUFFER);
              ON ERROR CALL ERROR(DIS7);
         GET STRING(BUFFER) LIST(OLD_LAST,OLD_FIRST,NCITY
              );
         CALL FIND(OLD_LAST,OLD_FIRST);
         IF POSITION=0 THEN CITY=NCITY;
    END;
    IF ANSWER='OCCUPATION' THEN DO;
         DIS8:DISPLAY('TYPE IN LAST,FIRST AND NEW'||
         'OCCUPATION IN QUOTES')REPLY (BUFFER);
         ON ERROR CALL ERROR(DIS8);
         GET STRING(BUFFER) LIST(OLD_LAST,OLD_FIRST,
             NOCC);
```

```
                 CALL FIND (OLD_LAST,OLD_FIRST);
                 IF POSITION=0 THEN OCCUPATION=NOCC;
                 END;
            IF ANSWER='AGE' THEN DO;
                 DIS9:DISPLAY('TYPE IN LAST,FIRST AND NEW AGE'||
                 'IN QUOTES')REPLY(BUFFER);
                 ON ERROR CALL ERROR(DIS9);
                 GET STRING(BUFFER) LIST(OLD_LAST,OLD_FIRST,NAGE)
                      ;
                 CALL FIND(OLD_LAST,OLD_FIRST);
                 IF POSITION=0 THEN AGE=NAGE;
            END;
            IF ANSWER='NAME'|ANSWER='CITY'|ANSWER='AGE'|
                 ANSWER='OCCUPATION' THEN DO;
                 DISPLAY('THE INDICATED RECORD HAS BEEN CHANGED '
                      ||'TO READ:');
                 DISPLAY(LAST||FIRST||OCCUPATION||CITY||AGE);
            END;
            ELSE DO;
                 DISPLAY('I CANT FIGURE OUT WHAT YOU WANT '||
                 'CHANGED; I CAN ONLY ASSUME THAT:');
                 CALL ERROR(DIS5);
            END;
       END;
       IF ANSWER='QUIT' THEN DO;
            DISPLAY('YOU HAVE INDICATED THAT NO MORE WORK IS TO '
                 ||'BE DONE; THEREFORE I WILL PUT A CURRENT LIST'
                 ||'ING IN SYSPRINT BEFORE EXITING:');
            CALL PRINT;
            EXIT;
       END;
       END;
       ELSE DO;
            DISPLAY('I CANT FIGURE OUT WHAT IT IS YOU WANT DONE'
             ||'; ICAN ONLY ASSUME THAT:');
            CALL ERROR(DIS2);
       END;
  END;
  END;


/* THIS GROUP OF 8 STATEMENTS IS THE MAIN PROGRAM WHICH
   OBTAINS THE DATA FROM THE SYSIN FILE AND CALLS THE
   BUILD PROCEDURE TO BUILD A SUBRING STRUCTURE (DATA
   RECORDS)  */


  DO I=1 TO 26;
```

```
                ALPHA_ARRAY(I)=NULL;
            END;
            FIRST_LTR=NULL;
            DO WHILE('1'B);
                GET LIST(NLAST,NFIRST,NOCC,NCITY,NAGE);
                CALL BUILD (NLAST);
            END;


     /* THE FOLLOWING PROCEDURE ALLOWS FOR CORRECTION OF TERMINAL
            INPUT ERRORS */


    ERROR:PROC (LABEL);
            DCL LABEL LABEL;
            DISPLAY(' ');
            DISPLAY('THERE IS AN ERROR IN THE INPUT');
            DISPLAY('THE LAST INPUT FROM THE TERMINAL WAS:');
            DISPLAY(' ');
            DISPLAY(BUFFER);
            DISPLAY(' ');
            DISPLAY('CHECK IT THEN RE-INPUT THE CORRECT DATA AS FOLLOWS:');
            GO TO LABEL;
    END ERROR;


          /* THIS FUNCTION CALLS PROCEDURES WHICH CONSTRUCT (1) A
             RING WHICH HAS THE FIRST LETTER OF A PERSON'S LAST NAME
             AND (2) A SUBRING WHICH HAS DATA ITEMS: LAST NAME, FIRST
             NAME, OCCUPATION,CITY,AGE */


     BUILD:  PROC(NLAST) ;
                DCL NLAST CHAR(10);
            IF FIRST_LTR=NULL THEN DO;
                ALPHA_PTR=BUILD_ALPHA(FIRST_LTR);
            CALL BUILD_REC (ALPHA_PTR,FRONT);
            RETURN;
            END;
            ELSE DO;
                ALPHA_PTR=FIRST_LTR;
                DO WHILE('1'B);
                    IF SUBSTR(NLAST,1,1)¬=ALPHA_PTR->LTR THEN DO;
                        ALPHA_PTR=ALPHA_PTR->NEXT_LTR;
                        IF ALPHA_PTR=FIRST_LTR THEN DO;
                            ALPHA_PTR=BUILD_ALPHA(FIRST_LTR);
                            CALL BUILD_REC(ALPHA_PTR,FRONT);
                            RETURN;
```

```
                  END;
              END;
              ELSE DO;
                  FRONT=LOOK_UP (NLAST,NFIRST);
                  IF POSITION¬=0 THEN
                      CALL BUILD_REC (ALPHA_PTR,FRONT);
                  RETURN;
              END;
          END;
      RETURN;
END BUILD;


    /* THE FOLLOWING PROCEDURE BUILDS A SUBRING, UNDER THE FIRST
       LETTER OF A PERSON'S LAST NAME, WHICH CONTAINS DATA ITEMS:
       LAST NAME, FIRST NAME, OCCUPATION, CITY, AGE */


BUILD_REC:  PROC (ALPHA_PTR,FRONT);
      DCL (ALPHA_PTR,FRONT) PTR;
      ALLOCATE RECORD;
      LAST=NLAST;
      FIRST=NFIRST;
      OCCUPATION=NOCC;
      CITY=NCITY;
      AGE=NAGE;
      IF ALPHA_PTR=FRONT THEN
          IF LIST_PTR=NULL THEN DO;
              NEXT_RECORD=NULL;
              LIST_PTR=NEW_RECORD;
              RETURN;
          END;
          ELSE DO;
              NEXT_RECORD=LIST_PTR;
              LIST_PTR=NEW_RECORD;
              RETURN;
          END;
      ELSE DO;
          NEXT_RECORD=FRONT->NEXT_RECORD;
          FRONT->NEXT_RECORD=NEW_RECORD;
          RETURN;
      END;
END BUILD_REC;

    /* THE FOLLOWING PROCEDURE LOOKS TO SEE WHERE DATA ITEMS, LAST
       LAST AND FIRST NAME OF A RECORD, ARE TO BE INSERTED IN THE
       SUBRING   */
```

```
LOOK_UP:  PROC (NLAST,NFIRST) PTR;
          DCL (NLAST,NFIRST) CHAR(10);
          FRONT=ALPHA_PTR->LIST_PTR;
          BACK=ALPHA_PTR;
          POSITION =1;
          DO WHILE (FRONT¬=NULL);
               IF NLAST||NFIRST>FRONT->LAST||FRONT->FIRST THEN DO;
                    BACK=FRONT;
                    FRONT=FRONT->NEXT_RECORD;
               END;
               ELSE IF NLAST||NFIRST=FRONT->LAST||FRONT->FIRST
                         THEN DO;
                              POSITION=0;
                              RETURN(FRONT);
                         END;
                         ELSE RETURN(BACK);
          END;
          RETURN(BACK);
END LOOK_UP;

      /* THE FOLLOWING PROCEDURE BUILDS THE RING WHICH CONTAINS THE
         LETTERS OF THE ALPHABET. THE LETTERS OF THIS RING ARE THEN
         USED AS ENTRY POINTS FOR THE INSERTION OF RECORDS CONTAINING
         THE FIRST LETTER OF A PERSON'S LAST NAME CORRESPONDING
         TO THE LETTER OF THE ALPHABET IN THE RING */


BUILD_ALPHA:   PROC (FIRST_LTR) PTR;
          DCL FIRST_LTR PTR;
          DCL TEMP2 FIXED BIN(15);
          ALLOCATE ALPHA;
          TEMP2=INDEX(ALPHABET,SUBSTR(NLAST,1,1));
          ALPHA_ARRAY(TEMP2)=ALPHA_PTR;
          LTR=SUBSTR(NLAST,1,1);
          LIST_PTR=NULL;
          FRONT=ALPHA_PTR;
          IF FIRST_LTR=NULL THEN DO;
               FIRST_LTR=ALPHA_PTR;
               NEXT_LTR=FIRST_LTR;
          END;
          ELSE DO;
               NEXT_LTR=FIRST_LTR->NEXT_LTR;
               FIRST_LTR->NEXT_LTR=ALPHA_PTR;
          END;
          RETURN (ALPHA_PTR);
```

```
END BUILD_ALPHA;


      /* THE FOLLOWING PROCEDURE PRINTS OUT A LISTING OF THE ENTIRE
         STRUCTURE OR A LISTING OF THE SYSPRINT FILE  */


PRINT:  PROC;
        DCL (TEMP) PTR;
       IF LISTING=0 THEN OPEN FILE (SYSPRINT);
        DO I=1 TO 26;
             IF ALPHA_ARRAY(I)¬=NULL THEN DO;
                  TEMP=ALPHA_ARRAY(I);
                  NEW_RECORD=TEMP->LIST_PTR;
                  DO WHILE(NEW_RECORD¬=NULL);
                      IF LISTING=0 THEN PUT EDIT(LAST,FIRST,OCCUPATION,
                      CITY,AGE)(SKIP,A(10),A(10),A(10),A(10),A(3));
                  ELSE DISPLAY(LAST||FIRST||OCCUPATION||CITY||AGE);
                       NEW_RECORD=NEW_RECORD->NEXT_RECORD;
                  END;
             END;
        END;
        RETURN;
 END PRINT;


      /* THE FOLLOWING PROCEDURE LOCATES THE LETTER OF THE ALPHABET
         CORRESPONDING TO THE FIRST LETTER OF A PERSON'S LAST
         NAME AND SEARCHES THROUGH THE SUBRING FOR THE LAST
         NAME AND FIRST NAME OF THE RECORD DESIRED  */


FIND:PROC(NLAST,NFIRST);
         DCL(NLAST,NFIRST) CHAR(10);
         ALPHA_PTR=FIRST_LTR;
         IF FIRST_LTR=NEXT_LTR THEN IF SUBSTR(NLAST,1,1)¬=LTR
              THEN DISPLAY('RECORD NOT IN FILE, YOU BLEW IT    ');
              ELSE DO;
                   NEW_RECORD=LOOK_UP(NLAST,NFIRST);
                   IF POSITION¬=0 THEN DISPLAY('RECORD NOT IN FILE,'||
                   'YOU BLEW IT');
                   RETURN;
              END;
         ELSE IF SUBSTR(NLAST,1,1)=LTR THEN DO;
                   NEW_RECORD=LOOK_UP(NLAST,NFIRST);
                   IF POSITION¬=0 THEN DISPLAY('RECORD NOT IN FILE,'||
                   'YOU BLEW IT');
                   RETURN;
```

```
          END;
               ELSE DO;
                    ALPHA_PTR=ALPHA_PTR->NEXT_LTR;
                    DO WHILE(ALPHA_PTR¬=FIRST_LTR);
                         IF SUBSTR(NLAST,1,1)=LTR THEN DO;
                              NEW_RECORD=LOOK_UP(NLAST,NFIRST);
                              IF POSITION¬=0 THEN DISPLAY('RECORD NOT'||
                              'IN FILE, YOU BLEW IT');
                              RETURN;
                         END;
                         ELSE ALPHA_PTR=ALPHA_PTR->NEXT_LTR;
                    END;
                    IF ALPHA_PTR=FIRST_LTR THEN
                         DISPLAY('RECORD NOT IN FILE, YOU BLEW IT    ');
               END;
 END FIND;
  END PROJECT;
```

PROGRAM USED BY PLISM

```
 NEWRAP: PROC OPTIONS (MAIN);
 TRY:    BEGIN;
         DCL I FIXED DEC (3) INITIAL (1);
         DCL (ROOT,X,X1,XJ,XI) FLOAT DEC (16) INITIAL (0);
         DCL E FIXED DEC (3) INITIAL (0.0001);
         DCL BUFFER CHAR (80);
         DISPLAY ('INPUT') REPLY (BUFFER);
         DISPLAY (BUFFER);
         GET STRING (BUFFER) LIST (XJ);
         DCL (D,DJ,DI) FLOAT DEC (8) INITIAL (30);
 AGAIN:  XI=XJ;
         IF DF((XI))=0 THEN GOTO TRY;
         XJ=XI-F((XI))/DF((XI));
         DI=DJ;
         DJ=ABS(XJ-XI);
         IF DJ>E THEN
 STUP:        IF DJ>DI THEN
                   GOTO TRY;
              ELSE DO;
                   IF I>20 THEN
                        GOTO TRY;
                   ELSE DO;
                        DISPLAY ('AT ITERATION '||I||' X'||I||'='||XJ);
                        I=I+1;
                        GOTO AGAIN;
                   END;
```

```
             END;
        ELSE
             IF ABS (F((XJ)))>E THEN
                  GOTO STUP;
             ELSE DO;
                  DISPLAY ('ROOT = '||XJ);
                  STOP;END;
        DCL F ENTRY (FLOAT DEC(16)) RETURNS (FLOAT DEC (16));
F:      PROC (X) FLOAT DEC (16);
        DCL X FLOAT DEC (16);
        X=X**2-EXP(X)-3;
        RETURN (X);
        END F;
        DCL DF FNTRY (FLOAT DEC (16)) RETURNS (FLOAT DEC (16));
DF:     PROC (X) FLOAT DEC (16);
        DCL (X,Y) FLOAT DEC (16);
        Y=X-1;
        X=2*X-X*EXP(Y);
        RETURN (X);
        END DF;
        END NEWRAP;
```

BIBLIOGRAPHY

1. Haines, W.R. and Porterfield, J.H., Jr., An Emperical Investigation of the CP/67 and TSS/360 Time-Sharing System, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1971.

2. University of Michigan, MTS Volume 1: MTS and the Computing Center, 3d ed., February 1971.

3. University of Michigan, MTS Volume 2: Public File Descriptions, 3d ed., April 1971.

4. University of Michigan, MTS Volume 3: Subroutine and Macro Descriptions, 3d ed., July 1971.

5. University of Michigan, MTS Volume 11: Plot Description System, 3d ed., April 1971.

6. University of Michigan, MTS Michigan Terminal System, 2d ed., v. 1, December 1967.

7. University of Michigan, MTS Michigan Terminal System, 2d ed., v. 2, December 1967.

8. University of Michigan, Computing Center Memos (CCMEMOS), M1-M149, 1971.

9. University of Michigan, Computing Center News (CCNEWS), C17-C291, 1971.

10. University of Michigan, MTS Distribution 2.0 Documentation, 1970.

11. University of Michigan, MTS HASP Operators Guide, 1970.

12. Alexander, M.T., Time Sharing Supervisor Programs, The University of Michigan Computing Center, May 1970.

13. Naval Postgraduate School, Users Manual, W.R. Church Computer Center, March, 1970.

14. Syms, G.H., Haines, W.R., and Porterfield, J.H., Jr., An Emperical Comparison of CP/67 and TSS/360 Time Sharing Systems, paper to be presented at the Fifth Hawaii International Conference on System Science, January 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Assistant Professor G.H. Syms, Code 53 Zz Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4.	Instructor J. J. Budway, Code 53 Zy Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
5.	LCDR Elbert F. Hinson, USN 9 Legare Street Charleston, South Carolina 29401	1
6.	Michael Alexander The University of Michigan Computing Center Building 1075 Beal Avenue Ann Arbor, Michigan 48105	1
7.	T. A. Marsland Department of Computing Science The University of Alberta Edmonton 7, Alberta Canada	1

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Postgraduate School Monterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified
	2b. GROUP

3. REPORT TITLE
A Comparative Study of the Michigan Terminal System (MTS) with Other Time Sharing Systems for the IBM 360/67 Computer.

4. DESCRIPTIVE NOTES *(Type of report and, inclusive dates)*
Master's Thesis; December 1971

5. AUTHOR(S) *(First name, middle initial, last name)*
Elbert Farrell Hinson

6. REPORT DATE December 1971	7a. TOTAL NO. OF PAGES 83	7b. NO. OF REFS 14

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

The principle of benchmark measurement is applied to yet another time sharing system, the Michigan Terminal System (MTS), and thus complete the comparison of the three time sharing systems for the IBM 360/67 Computer. MTS proves to be the superior time sharing system.

MTS is also evaluated from both a user's view and a performance view against the CP/67 and OS/MVT combination now in use at the Naval Postgraduate School.

The concept of Load Factor and its use is expanded. Use of measurement techniques to determine load factors are discussed with extension of the load factor concept to measure system loads on a continuous basis.

An Operator's Manual for MTS is presented to aid beginners in the operation of MTS. It is believed that this manual, or a generalized version, should be distributed as part of the MTS documentation.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

Security Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Benchmark						
Time Sharing System Evaluation						
Load Factor						
MTS						
CP/CMS						
CP/67						
OS/360						
OS/MVT						
Time Sharing System Performance						
IBM 360/67 Computer						

DD FORM 1 NOV 65 1473 (BACK)
0101-807-6821

Security Classification

A-31409

Thesis 133310
H574 Hinson
c.1 A comparative study of the Michigan Terminal System (MTS) with other time sharing systems for the IBM 360/67 computer.