# Real-time graphic display of time-sharing system operating characteristics*

*by* JERROLD MARVIN GROCHOW**

*Massachusetts Institute of Technology****
Cambridge, Massachusetts

## INTRODUCTION

The Graphic Display Monitoring System (GDM) is an experimental monitoring facility for Multics, a general purpose time-sharing system implemented at Project MAC cooperatively with General Electric and the Bell Telephone Laboratories.[2,7] GDM allows design, systems programming, and operating staff to graphically view the dynamically changing properties of the time-sharing system. It was designed and implemented by the author to provide a medium for experimentation with the real-time observation of time-sharing system behavior. GDM has proven to be very useful both as a measuring instrument and a debugging tool and as such finds very general use.

Monitoring the activity of a traditional computer system (one with only a single active process) is a fairly simple task. Hardware and software devices can easily be devised to keep track of almost any parameter. Asking the question "What are you doing right now?" to a computer system controlling multiple processes or servicing multiple interactive users, however, proves particularly difficult to answer meaningfully. It becomes necessary to "snapshot" the system (record in some manner its state at a specific time) and interpret this information for the inquirer. Since a basic property of a time-sharing system is that, in fact, it is "doing something else" a few milliseconds from now, what the inquirer really wants to ask is "What are you doing now, and now, and now . . .?" Implicitly, he is also asking to be shown what is happening in an easily interpretable format. The GDM solution to his problem is to provide the user with a real-time, graphical output "eavesdropper."

Statistical studies of time-sharing systems have been performed[1,5,11] in an attempt to provide "after-the-fact" monitoring (in effect answering the question "On the average, what is happening?") and there have been simulations in an effort to provide "predictive monitoring."[5,11] One company has even produced a hardware device to receive system status information over a special wired in channel and record the results on magnetic tape.[12] Other than the "SNUPER Computer"[6] which, however, still requires engineer-installed hardware probes, there has been little work directed towards providing a generalized, real-time, time-sharing system monitoring device. It is felt that while the hardware used for this implementation of GDM is perhaps unusual, the design principles involved and the monitoring methods explored are sufficiently general to provide a framework and a guide for other designers.

The basic goal in designing the GDM System was to produce a time-sharing system monitoring device for use by the staff of the Multics project. Initial requirements implied that it would be on-line, that is, active while Multics was in operation—not just collecting data for future analysis, and would provide

* Work reported herein was supported (in part) by Project MAC, an M. I. T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Nonr-4102(01)

** This paper is based on a thesis submitted in partial fulfillment for the degree of Master of Science at the Massachusetts Initutest of Technology, Department of Electrical Engineering.

*** Project MAC

dynamically changing graphic output (as well as hard copy if desired). It was to be designed such that the act of monitoring did not cause significant interference to the time-sharing system or perturbations in its behavior and such that it would not be necessary to make more than a few minor additions to supervisory procedures in order to incorporate the GDM System (as opposed to monitoring done by inserting entire procedures in critical points in the supervisor in order to collect data; see Scherr[11] for an example). Since the GDM System was to be an experimental tool, it was also considered especially important that it be easily expandable and adaptable to new or different monitoring requests. Coupled with these requirements was the need to involve the expected user community as early as was possible in the project in order to insure its continued use after initial implementation. In this regard, acceptance by the systems programming staff was very encouraging and many currently make use of the GDM facility.

The original GDM System embodies these goals while making use of existing hardware at Project MAC. The Digital Equipment Corporation 338 (see Figure 2) was already on site for use in other experimental work. A more extensive (and less expensive) monitoring system could perhaps be designed if it were possible to choose both the display processor and the method of interface to the time-shared computer. This was not, however, viewed as a major handicap in developing a useful system.

Succeeding sections will discuss the various components of the GDM System and will describe in detail initial experiments and current usage at Project MAC. Compromises in design and special problems due to the particular constraints of the display hardware or software and the Multics system to which they interfaced are also discussed.

*What is the GDM system?*

### Subsystems

The GDM System consists of four major components:

A. An input-output procedure running under Multics to transmit data as requested to the display computer.
B. A monitor system operating on the display computer to facilitate the creation, storage, and retrieval of display templates (see below) and to perform various other housekeeping functions.
C. A series of display computer subroutines for manipulating data and generating command sequences for the display.
D. A language for describing desired data manipulation and display formats (Display Description Language), a (planned) compiler for translating such descriptions into display computer assembly language programs, and a set of macro-definitions for simplifying display computer programming and for calling the subroutines mentioned under C.

Figure 1 gives a functional representation of the various GDM subsystems showing the interaction among them, the two computers, and the user. Figure 2 shows the complete hardware configuration. Reference 8 goes into considerable detail about the GDM monitor system software including system flow charts.

### Modes of operation

Use of the GDM System generally falls into one of three classes of operation:

1. Demonstration mode: any of a number of library displays may be viewed to get a general picture of Multics operation at the moment. Data used in these displays is updated periodically according to preprogrammed instructions.
2. XRAY mode (so named because of its similarity to the X-ray System[4]): the user may type the

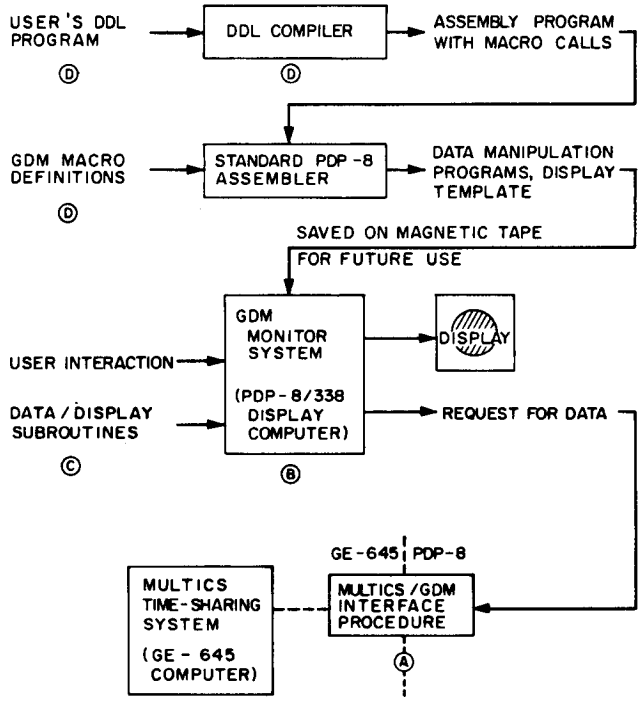


Figure 1—GDM subsystem interactions

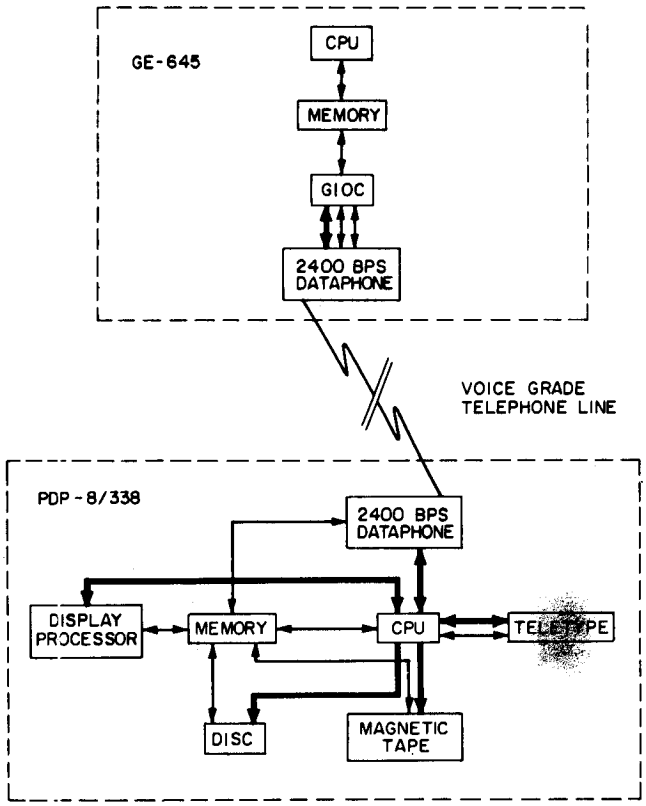


Figure 2—Hardware configuration

segment number and offset of a datum (see Reference 3 for a description of the addressing scheme used in Multics) on the teletype of the display computer and see displayed the octal and ASCII character representation of its contents, updated every second (Figure 3, XRAY display).

3. Display creation mode: the user will go through the process of creating his own display (as outlined in Figure 1) in order to gain desired flexibility in data displayed, format of display, or data sampling rate. Displays are then saved in a special format, the "display template," for use in later experiments or as part of the library.

All modes of operation employ the same type of display template and are listed only to differentiate between the application of the GDM System. System programmers have been trained in five minutes to utilize the many displays already in the library (operation under "Demonstration Mode"). Some use the XRAY display when there are one or two locations of interest at a particular moment, as in the current

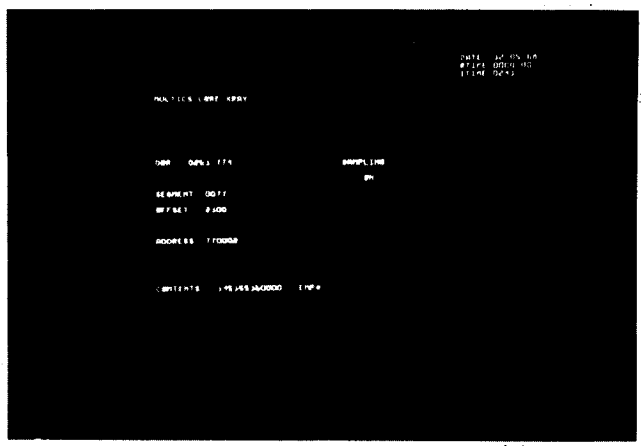

Figure 3—XRAY display

number of available disk pages or the value of a particular time-dependent variable. Display creation mode, the most general use of the GDM System, requires the most work on the part of the user. He must decide what data items to display, how to display them, and how often to sample them. He must then create the data manipulation routines and the display list comprising his particular "display template." Until the DDL compiler is constructed, this work must be done in an extended version of the PDP-8 Assembly Language as seen in Table I (the 338 computer uses the same systems software as its sister PDP-8). It is in this mode of use that all the facilities of the GDM System come into play and in which the most fruitful experimental work can be performed.

*Examples*

Figures 4 and 5 show typical examples of GDM displays. Figure 4, Core Memory Summary Display, displays real-time information on the usage of Multics core memory pages; Figure 5, Active Process State Display, displays user activity information (see below). The display templates for both figures were constructed in about two hours apiece by an experienced user and have provided many hours of system observation for experienced and inexperienced alike.

The display in Figure 5 causes information about each process in Multics to be extracted from the traffic controller data base. The column labelled "MP" is the "multiprogramming state," an indication of a process' eligibility to receive CPU time. Stars to the right of this column indicate the processes that are eligible (state 4). The column "ST" is the "activity state"—running, ready to run (waiting to be serviced), or not ready to run. The star is next to the process currently

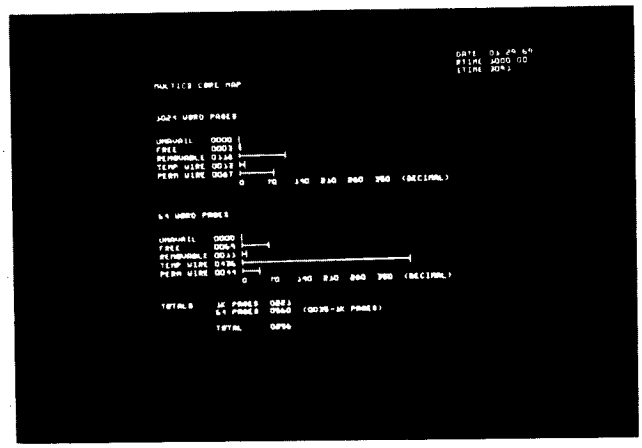

Figure 4—Multics core memory summary display

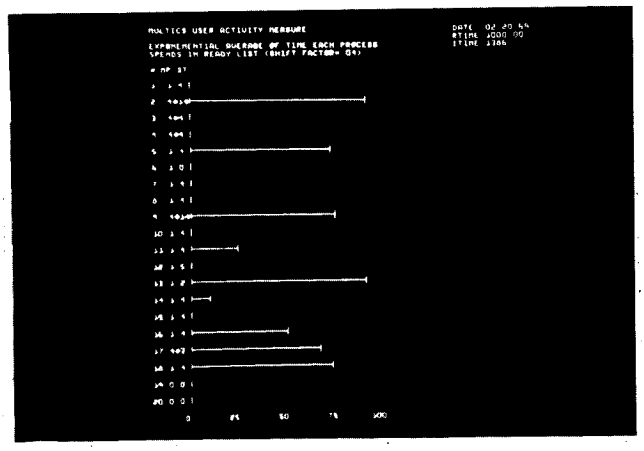

Figure 5—Active process state display

running, state 1. In a multi-processor configuration, there would be more than one such process.

The associated bar graphs also provide a descriptive measure of overall system activity. By "eyeball integration" of the length of the bars, one can get a fairly accurate idea of system loading. Several means of calculating graph lengths have been used (in different display templates all using the same basic form):

1. Whenever a process is ready or running, the length of the bar is increased. When the process is not ready, the associated bar decreases. Each bar changes length as an exponentially weighted sum of ready-running and not-ready time. (This is seen in Figure 5.)
2. Whenever a process is ready, its bar grows in equal time increments. When the process is finally serviced (receives processor time), its bar is reset to zero length.

The display of type 1 gives a general picture of system loading but also shows something of the behavior of the individual process. The scale is calibrated in percentage to indicate that the bar shows the percentage of time a process is requesting or receiving CPU time—a measure of the process' activity. The type 2 display is more useful in getting an uncluttered picture of just how long a "ready" user must wait to run, i.e., how long each process is spending in the queues waiting for service.

The display templates for these two displays differ in about ten instructions (the computation of bar length). The two hours of editing and assembling to get a "first draft" of the display is even less if averaged over the two displays. Herein lies a basic flexibility of the GDM System: once the data to be displayed have been decided upon, it need be only a matter of minutes before it is viewed. Display formats can be easily experimented with and a finished display template can be added to the GDM library for future monitoring without any costly "dedicated system" monitoring runs.

The examples discussed above show simply two ideas. Others have included collecting (and displaying) data on the mean lifetime of a page in the Multics memory (how long does it take before the page is swapped out to secondary storage), the distribution over time of the number of active time-sharing users (very nicely displayed as a graph similar to Figure 6D), and the average number of users referencing particular supervisor segments (built up during the length of the monitoring session). There is a great deal of work yet to be done before we run out of ideas or into the limitations of the GDM system.

*More on the display template*

A display template (DT) consists of three sections:

1. A list of the time-sharing system data items to be sampled (segment number and data base format are sufficient since absolute core locations are determined by GDM at monitoring time).
2. Instructions on display type (numerical, ASCII, bar graph, other graphics, etc.), sampling rate, and data manipulations (averaging, scaling, etc.) for each data item or group of items.
3. A display list: machine instructions for the 338 Display giving text, formatting information, and storage for items to be displayed.

For example, to display a single process' activity as

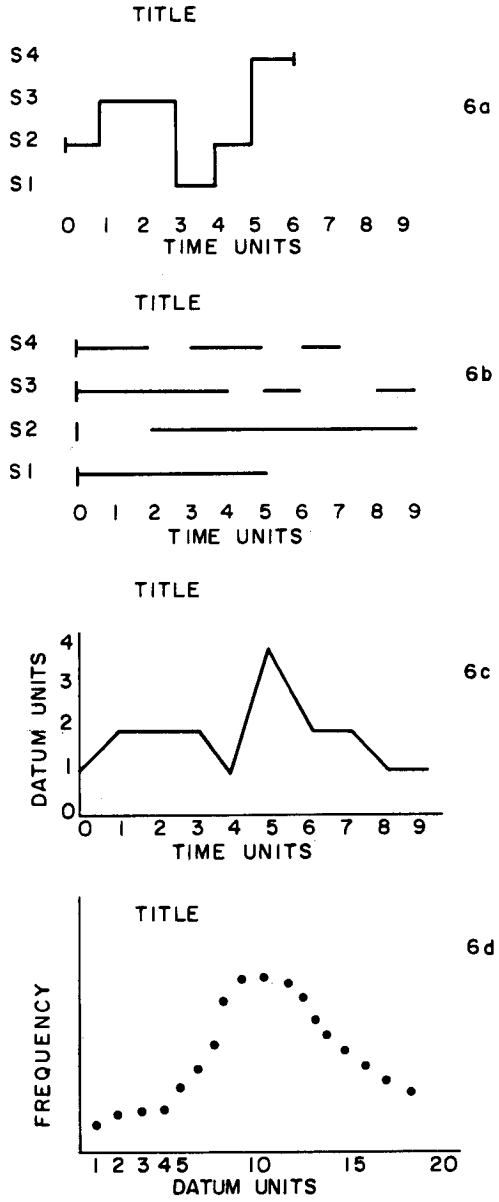


Figure 6—Other standard display types

in Figure 5, a DT would contain about twenty instructions (Table I).

The various non-PDP-8 instructions (call, do, dlstart, etc.) are macro calls to a set of definitions designed as part of the GDM System. Various subroutines (nplot, ge645, sked, etc.) are also provided as interfaces to the GDM monitor and to simplify programming. These features allow the programmer without PDP-8 experience to design a display template with a minimal apprenticeship. (Implementation of a DDL compiler should simplify this even further.) Of course, since all the facilities of the computer are available, data manipulations can be quite complex (although subroutines are provided for such common operations as scaling and masking) and displays quite unusual (Figure 6 shows standard types for which GDM provides some macro facility). The only limit is the designer's imagination and the size of the PDP-8 core memory.

*The Multics/GDM interface*

The GDM System is designed for use in a symbiotic relationship with a time-shared computer. The computer must be capable of supporting a display processor functioning basically independently of the time-sharing system but occasionally interjecting requests for data transmission.

The Multics environment is particularly friendly to this type of system as it is possible to make data requests through the generalized input-output controller (GIOC) of the GE–645[9], without interrupting the central processing unit (Figure 2). It is necessary, however, to dedicate two of the 2048 GIOC channel pairs (one for transmitting and one for receiving) to the display processor. Those problems introduced by this relationship are discussed further below.

The Multics/GDM interface procedures are capable of providing the following services:

1. Accept address request by segment number and offset of data to be displayed (GDM).
2. Convert this address to an absolute memory location for interpretation by the GIOC (GDM to Multics).
3. Transmit the datum from the GE–645 memory to the 338 (Multics).

In general, a GDM-type monitor requires only the simplest method possible of getting data from the time-shared computer to the display processor. On the Project MAC system, this means sending requests to a short I/O program running on the GE–645 GIOC. The 2400 bit per second Dataphone (201B modems) used for this transmission limits the request rate to approximately twenty per second (a negligible disturbance on a one-and-one-half microsecond per instruction processor). Higher data rate transmission can be used with corresponding increases in interference (if we increase the rate to 40,000 bps, the perturbance is still less than .1 percent) and special telephone lines.

All displays currently in use sample the GE-645 at rates at or near the available maximum. Displays with a number of data items occasionally resort to special

TABLE I—A display template to monitor a T-S user's activity

```
*address_table
tc_data              /segment name
540; 541             /locations within the segment

*data_routines
a                    /name table of routines to be
                     /called by the GDM monitor
7777                 /end of table
a, 0                 /PDP-8 subroutine format
   call ge645, 1, 2  /get first data item
   call nplot, mp, 1 /plot "MP" state number
   call ge645, 1, 3  /get next data item
   call nplot, st ,1 /plot "ST" state number
   jms calc          /call to machine language
                     /subroutine to calculate bar
                     /graph length
   do hplot, bar     /plot horizontal bar graph
   call sked, 144, a /reschedule "a" to be called
                     /by monitor in one second
   jmp i a           /PDP-8 subroutine format

*display-list
    dlstart          /macro instruction to start dis-
                     /play
    nl; nl           /"new line" for formatting
mp, 0                /storage for "MP"
    sp2              /spaces for display formatting
st, 0                /storage for "ST"
    sp2              /formatting
    hbar bar         /macro to create bar graph display
    escape           /display instruction macro
    top              /display instruction macro to
                     /cause refreshing of display
```

sampling methods in order to update important items at least once a second: about the rate at which the human eye can follow a dynamic display with that much information.

*Advantages and disadvantages of GDM*

Advantages, disadvantages, capabilities, and limitations of GDM can be grouped into two categories: those relating to its monitoring ability; and those relating to its ability to report the information monitored.

### Monitoring ability

Several factors determine the usefulness of any type of monitor. These include the number and type of events it can monitor, the rate at which it can monitor them, and the interference that this observation will cause to the system being monitored.

One of the capabilities of GDM is a facility to change the point of observation easily: this is accomplished through the use of the display template. A new display template can be designed and operational in a short time and, once constructed, can be added to a library for future recall. No hardware changes need be made, no plug boards rewired, no probes changed to monitor a new or different event. Another display template with a few basic instructions is all that is needed to change the "probe" of GDM.

GDM, as constructed, is a sampling monitor. Current dataphone connections limit requests for data items to about twenty per second as mentioned above. Faster dataphone, direct connections or other means can be used to influence sampling rate. The current rate is such that "microsecond" events cannot be monitored. Transient data items will be missed if their core location changes many times in a second. Current displays, therefore, limit themselves to observing only "wired" data, this is, data whose core location need be determined only once during a particular monitoring period although the data itself may change many times. As approximately 80 percent of the Multics supervisors, data bases fall into this category at the current time, this is not particularly restrictive.

Monitoring which requires the collection of a large number of statistics over a very short time period similarly is hindered by the current configuration although "long-time" statistics are collected and displayed by a number of display templates.

Under Multics, short-time event monitoring is performed by special software embedded in the Multics supervisor.[10] A GDM display is used to observe, in real time, the data base of this monitor in order to see the time build up of the statistics and to note any abnormalities that might be missed by observing averages after an hour or more of operation. In this way, the advantages of a real-time display are combined with monitoring embedded in the time-sharing system (which causes significant interference when turned on) to provide a very useful tool.

The area of system interference has already been discussed but one item should be emphasized. In the Multics configuration, GDM need take only GIOC time—not CPU time. In computer systems where this is not possible, interference will still be negligible if the GDM monitor "steals" only enough information to make a useful display. Five hundred cycles per second is still only .1 percent on a two-microsecond

cycle time computer and this is more than sufficient for even the most complex display.

### Reporting ability

Output of information is another area in which flexibility is crucial. Displays in Figures 3, 4, and 5 show only numbers, characters, and bar graphs. Displays have also been constructed with the types of graphs shown in Figure 6 and many others have been suggested for particular applications. It has been found that displaying the same information in different ways often presents an entirely different picture of what is going on. The only price to be paid for this flexibility is programmer time and even then it is no more difficult to display a bar graph (or any other type) than it is to simply show a number. Several display templates showing the same data in different formats can be made almost as easily as a single one and the best added to the GDM library.

For those who desire hard copy, GDM, in its current configuration, offers only photographs of its displays (stopped at any instant of time, saved on tape for future reference or photographing). Plotters of various kinds could perhaps be connected in tandem with a dynamic display and requested to plot a particular instance, even while the CRT display is still changing. Here again, the designer is limited only by the hardware available and his imagination.

## CONCLUSIONS AND OBSERVATIONS

The GDM System at Project MAC has served in two major capacities:

1. As a monitoring "control center".
2. As a debugging tool.

The very nature of a multiple-access computer system makes it very difficult to determine at one location exactly what is happening at all terminals. The GDM display, conveniently located near the main body of Multics programmers, is readily consulted to determine the state of a rampant user program, the availability of secondary storage space, or just the general health of the system (a slave display might possibly be installed near the computer itself or in the office of the system administrator as well). Many system programmers have, at one time or another, brought up the GDM System on their own initiative to find out various, otherwise unobtainable, pieces of information (a "cookbook" instruction sheet has been provided for just this purpose). A visit to the GDM display is always included as part of the standard system tour for visitors.

As a debugging aid, GDM has been invaluable. It is responsible for the detection of many system bugs—often transient or time dependent—that were not easily isolatable by previously available means.

One of the features of GDM that has made it so useful is its ability to simplify the act of dynamic display creation to the point where this is no more difficult than writing a simple assembly language program. This flexibility has paid many times over for the effort of implementation.

Finally, GDM can be readily adapted for use with other time-sharing systems: only two Multics-dependent modules exist in the monitor and display templates can be designed to suit any system.

GDM was designed as an experimental system and as such has been very useful at Project MAC. Its use during a period of intense debugging of the Multics system has proven its development worthwhile.

## ACKNOWLEDGMENTS

The author would like to express his gratitude to Professor F. J. Corbató, advisor for his Master's Thesis, for his continued support and aid during the period of this work. Thanks are also due to many members of the Multics development group at Project MAC without whose help this work could not have been undertaken and in particular, Thomas Skinner Noel Morris, and Professor J. H. Saltzer.

## REFERENCES

1 E G COFFMAN L C VARIAN
*Further experimental data on the behavior of programs in a paging environment*
CACM Vol 11 No 7 1968 471-474

2 F J CORBATÓ V A VYSSOTSKY
*Introduction and overview of the Multics system*
Proc FJCC 1965 185-196

3 R C DALEY J B DENNIS
*Virtual memory, processes, and sharing in Multics*
CACM Vol 11 No 5 1968 306-333

4 D J EDWARDS
*GE-645 core memory X-ray program*
Multics System Programmers' Manual Section BE.13
Cambridge Mass MIT Project MAC internal doc 1966

5 G ESTRIN L KLEINROCK
*Measures, models, and measurements for time-shared computer utilities*
Proc ACM Nat Meeting 1967 85-95

6 G ESTRIN et al
*SNUPER COMPUTER, a computer in instrumentation automation*
Proc SJCC 1967 645-656

7 E L GLASER J F COULEUR G A OLIVER
*System design of a computer for time sharing applications*
Proc FJCC 1965 185-196

8 J M GROCHOW
*The graphic display as an aid in the monitoring of a time-shared computer system*
Project MAC Tech Rpt MAC-TR-54 Thesis Cambridge Mass Sept 1968

9 J F OSSANA L E MIKUS S D DUNTEN
*Communications and input/output switching in a multiplex computing system*
Proc FJCC 1965 231-240

19 J H SALTZER J W GINTELL
*The instrumentation of Multics*
Presented at the Second ACM Symposium on Operating System Principles Princeton N J 1969

11 A L SCHERR
*An analysis of time shared computer systems*
Project MAC Tech Rpt MAC-TR-18 Thesis Cambridge Mass June 1965

12 F D SCHULMAN
*Hardware measurement device for IBM System/360 time sharing evaluation*
Proc ACM Nat Meeting 1967 103-109