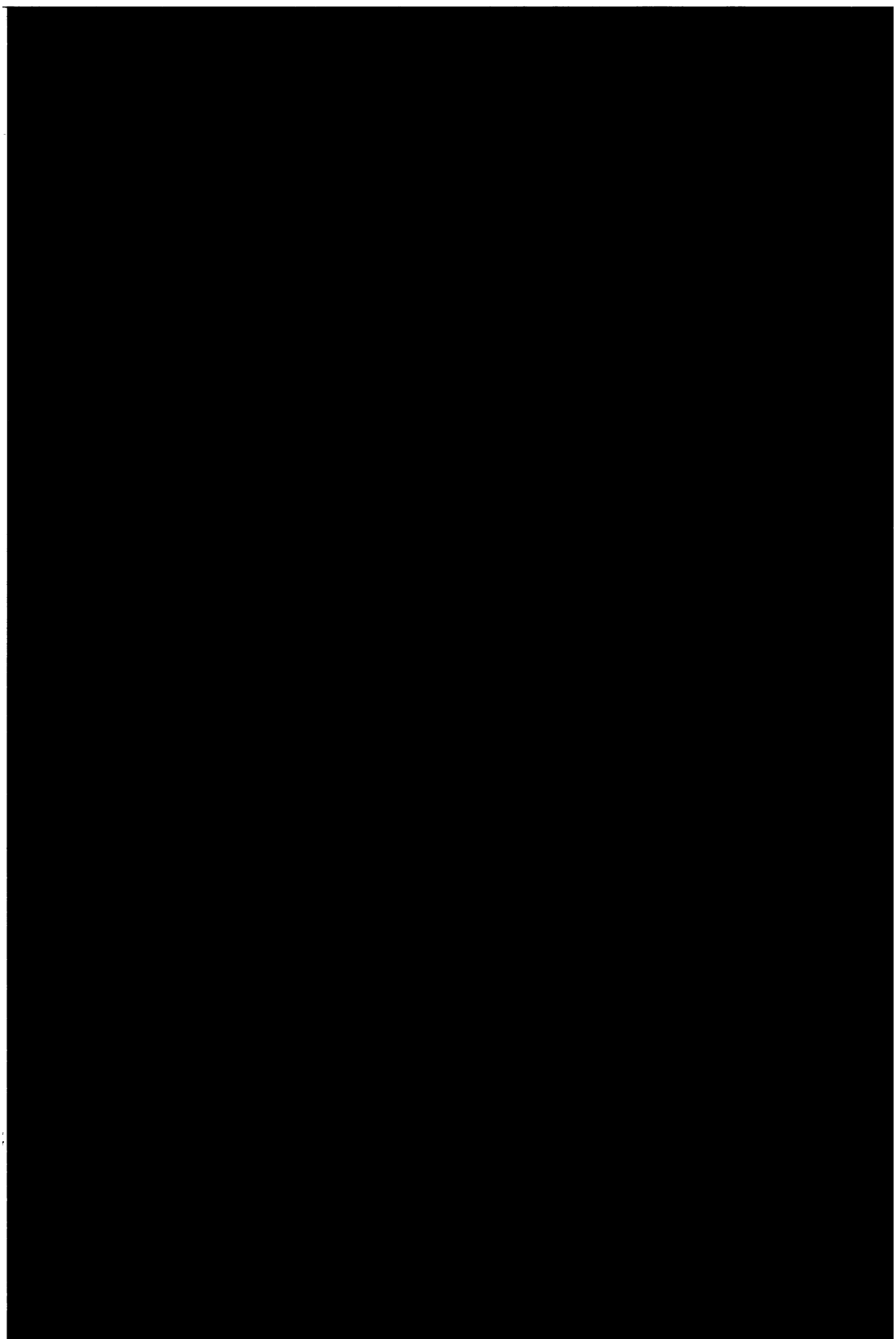


This blank page was inserted to preserve pagination.

THE GRAPHIC DISPLAY AS AN AID IN THE MONITORING OF A TIME-SHARED COMPUTER SYSTEM

by

JERROLD MARVIN GROCHOW

S.B., Massachusetts Institute of Technology
(1968)

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September, 1968

Signature of Author _____
Department of Electrical Engineering, August 30, 1968

Certified by _____
Thesis Advisor

Accepted by _____
Chairman, Departmental Committee on Graduate Students

This empty page was substituted for a blank page in the original document.

THE GRAPHIC DISPLAY AS AN AID
IN THE MONITORING OF
A TIME-SHARED COMPUTER SYSTEM

by

JERROLD MARVIN GROCHOW

Submitted to the Department of Electrical Engineering on
August 30, 1968 in partial fulfillment of the requirements
for the Degree of Master of Science

ABSTRACT

The problem of dynamic observation of the state of a time-shared computer system is investigated. The Graphical Display Monitoring System was developed as a medium for this experimental work. It is an integrated system for creating graphic displays, dynamically retrieving data from Multics Time-Sharing System supervisor data bases, and on-line viewing of this data via the graphic displays.

On-line and simulated experiments were performed with various members of the Multics staff at Project MAC in an effort to determine what data is most relevant for dynamic monitoring, what display formats are most meaningful, and what sampling rates are most desirable. The particular relevance of using a graphic display as an output medium for the monitoring system is noted.

As a guide to other designers, a generalized description of the principles involved in the design of this on-line, dynamic monitoring device includes special mention of those areas of particular hardware or software system dependence.

Several as yet unsolved problems relating to time-sharing system monitoring, including those of security and data base protection, are discussed.

THESIS SUPERVISOR: Fernando J. Corbató
TITLE: Professor of Electrical Engineering

ACKNOWLEDGEMENTS

The author would like to express his appreciation to Professor Fernando J. Corbató for his guidance throughout his association with the author and particularly for his knowledgeable suggestions and observations on GDM monitoring experiments.

The author is also indebted to Prof. E.L. Glaser, now at Case-Western Reserve University, for providing invaluable advice and inspiration on many projects and especially for his introduction to the subject of computer displays.

Thanks are also due to the many members of the Multics development group at Project MAC who participated in the early experiments and without whose help this thesis could not have been completed. The author would particularly like to thank Thomas Skinner and Noel Morris, co-workers on PDP-8 development projects, who contributed several programs to the GDM Monitor System.

TABLE OF CONTENTS

Chapter			Page
ABSTRACT			iii
ACKNOWLEDGEMENTS			iv
LIST OF ILLUSTRATIONS			vi
FRONTISPIECE			vii
I	INTRODUCTION		1
II	WHAT IS THE GDM SYSTEM?		3
	2.1	The Multics/GDM Interface	6
		2.1.1 The GIOC Data Retrieval Program	8
		2.1.2 The PDP-8 Dataphone Routine	8
		2.1.3 The Address Formation Routine	8
		2.1.4 The User Interface	10
	2.2	The GDM Monitor System on the PDP-8/338	10
		2.2.1 Monitor Commands	11
		2.2.2 Monitor Display Control	13
		2.2.3 Other Monitor Facilities	19
	2.3	The GDM Subroutine Package	19
		2.3.1 Data Manipulation Subroutines	19
		2.3.2 The Display Routines	20
		2.3.3 The Pushbutton Routine	22
		2.3.4 The Interval Timer Routine	22
	2.4	The Display Description Language	22
		2.4.1 Format Control	23
		2.4.2 Data Description	24
		2.4.3 Display Component Description	25
		2.4.4 Display Template Address Table	27
		2.4.5 A Sample Display and Description	29
III	A MULTICS MONITORING EXPERIMENT		31
	3.1	A System Overview Display	33
	3.2	A Process Overview Display	34
IV	OBSERVATIONS AND CONCLUSIONS		39
	4.1	Unsolved Problems	39
	4.2	Plans for the Future	41

APPENDICES

A	The PDP-8/338: Structure and Programming	43
B	Segment Addressing in Multics	47
C	Using GDM under the OS/8 Operating System	49
D	GDM Command and Error Summary	55
E	The GDM Subroutine Package	57
F	Macro Definitions for Use with GDM	63
G	DDL Statement Summary	67
H	A Sample Display Template: XRAY	69

BIBLIOGRAPHY 73

LIST OF ILLUSTRATIONS

Figure		Page
1 thru 8	Frontispiece	vii
2-1	GDM Subsystem Interactions	4
2-2	Hardware Configuration	5
2-3	Multics/GDM Interface	7
2-4	GE-645/PDP-8 Message Format	9
2-5	GDM Monitor System Control	12
2-6	GDM Monitor Display Control	14
2-7	Display Template Data Routine Organization	15
2-8	Display Control Data Dispatch Routine	16
2-9	GDM Monitor Clock Routines	18
2-10	Standard Display Types	21a
2-11	GE-645 Display Template Address Table Format	28
2-12	Sample Display Format	30
3-1	Multics Overview Display	32
3-2	Process Overview Display	36
A-1	PDP-8 Instruction Formats	44
B-1	Multics Two-Dimensional Addressing	48
C-1	Display Template Format	52
H-1	The XRAY Display	70

FRONTISPIECE

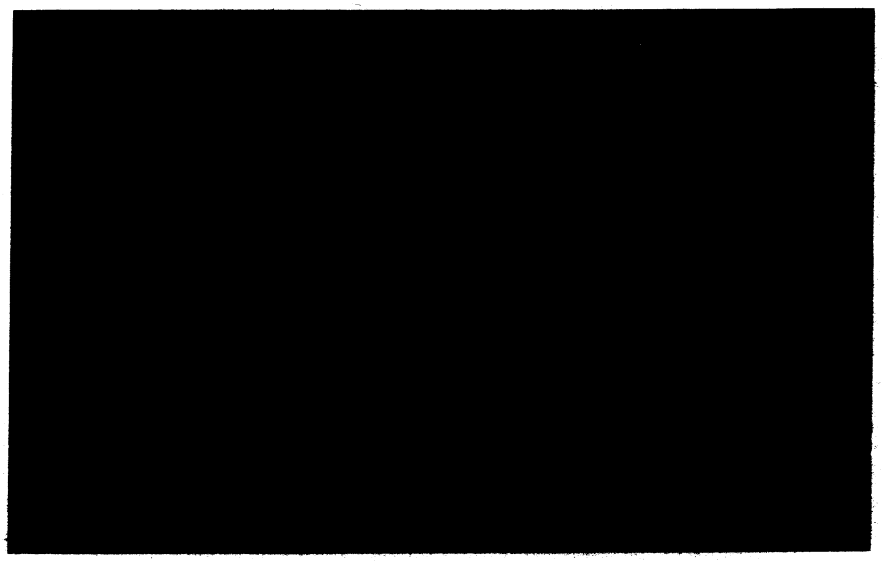


Figure 1. The XRAY Display

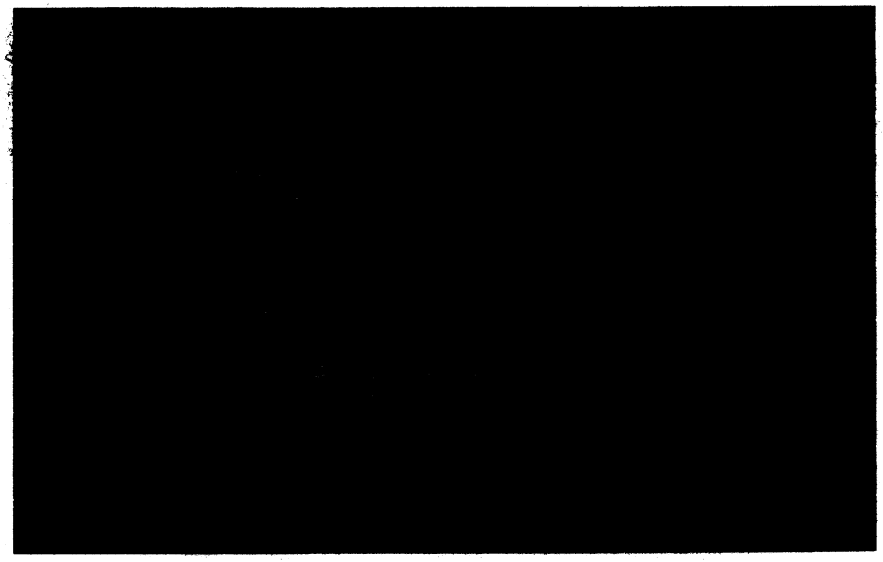


Figure 2. The Core Map Display

FRONTISPIECE

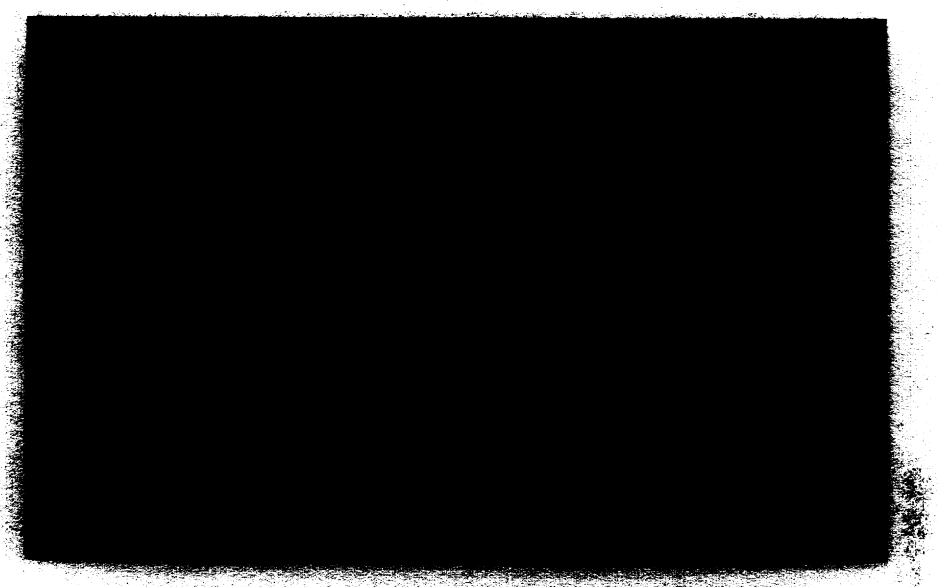


Figure 3. The Active Process Table - Few Users; Light Loading

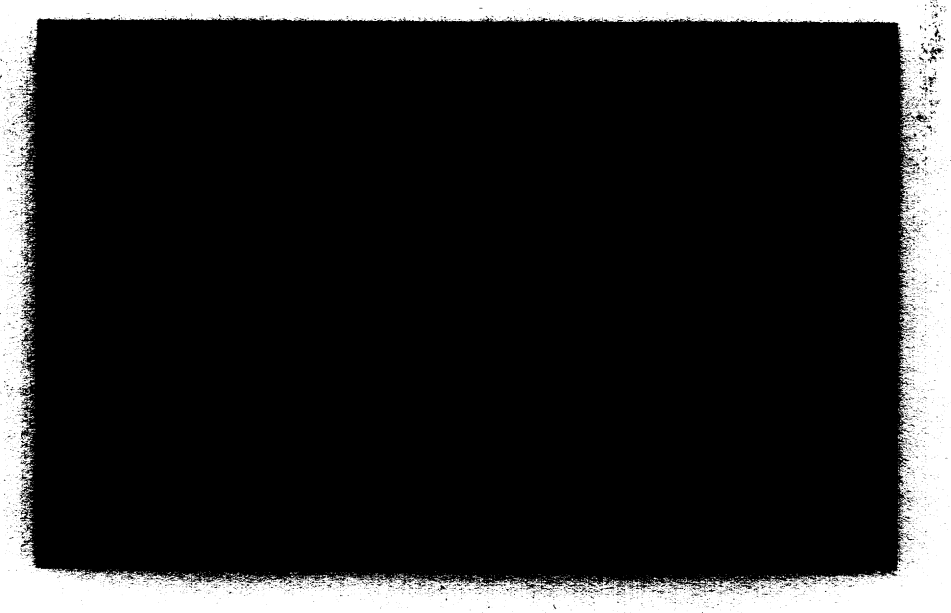


Figure 4. The Active Process Table - Moderate Loading

FRONTISPIECE

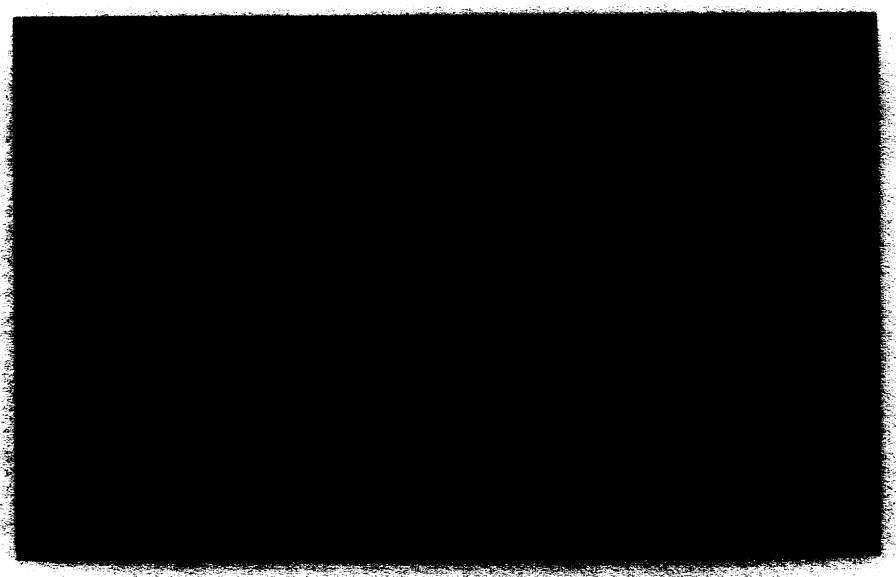


Figure 5. The Active Process Table - Moderate Loading; Few Users Eligible

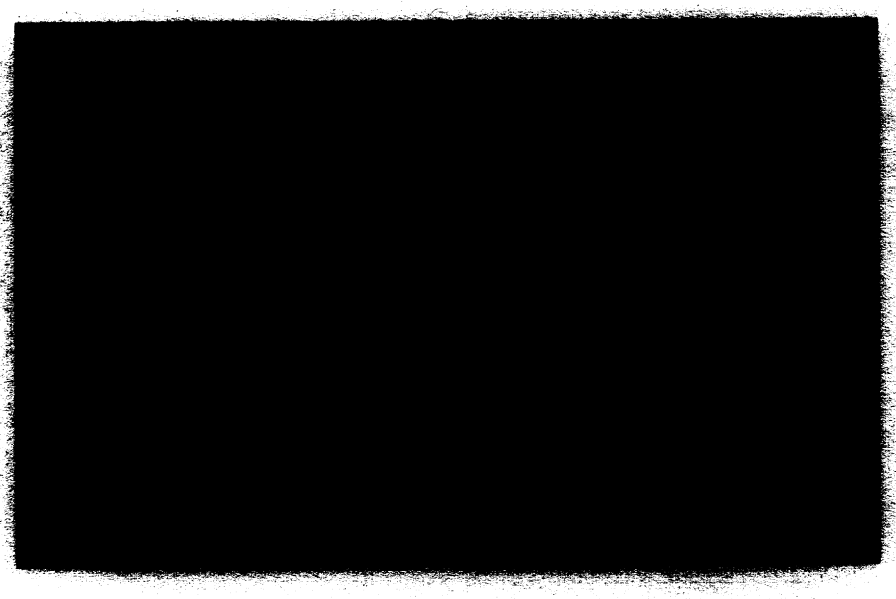


Figure 6. The Active Process Table - Heavy Loading

FRONTISPIECE

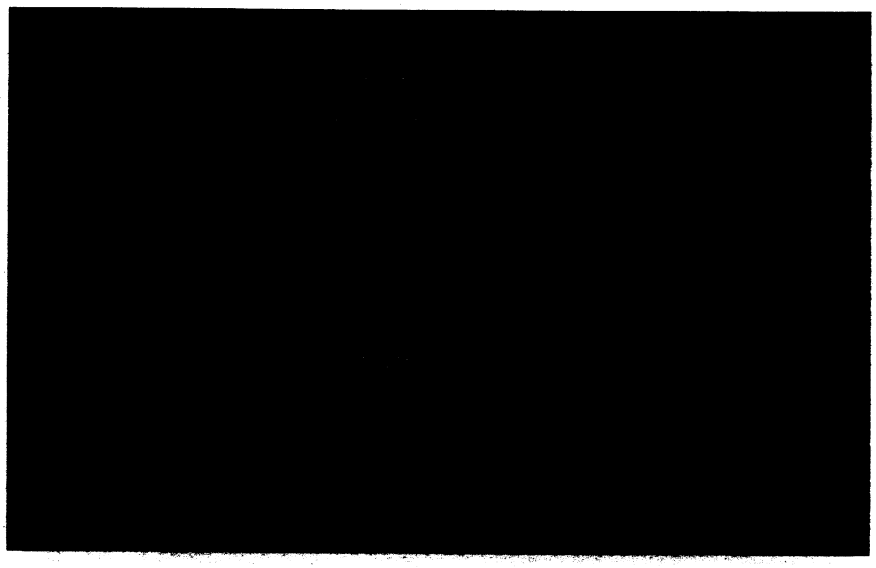


Figure 7. The Segment Meter Table - Page Fault Times

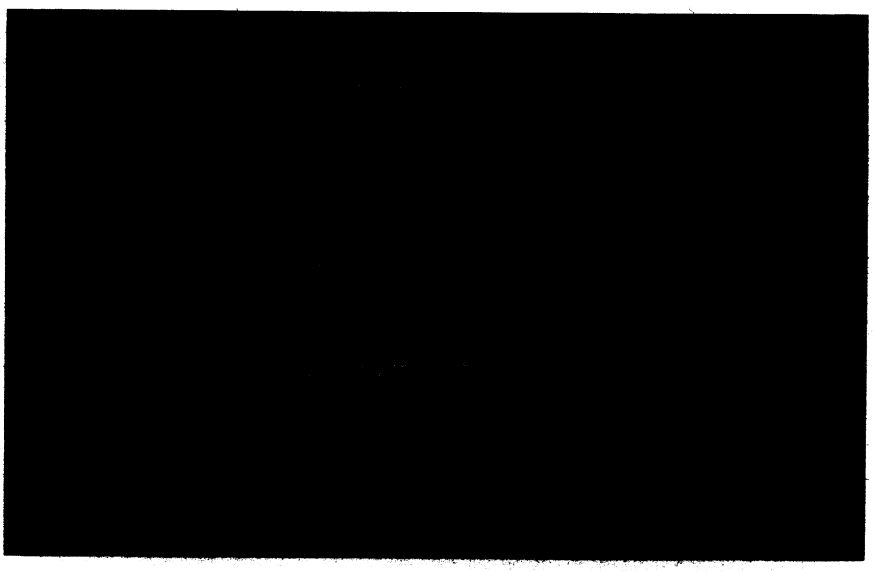


Figure 8. The Segment Meter Table - Linkage Fault Times

FRONTISPIECE

LEGEND

Figure 1. This display is similar to that described in Appendix H. The user supplies the DBR, segment number, and offset, and sees both octal and ASCII contents of the location specified.

Figure 2. Information on core page usage is displayed to get a dynamic picture of the properties of a demand paging scheme (See Chapter 3.1).

Figure 3, 4, 5, and 6. This series of pictures takes information from the Traffic Controller Data Base (See Chapter 3.2) to give an overview of the activity of each user. The four views show various system loadings and user activity at a particular time. Under normal instances, when several user activity levels approach 100%, the system becomes quite sluggish. In this series, "MP" stands for "Multiprogramming State" which indicates whether a user is eligible to run by showing " ◇ " next to his number in this column. In addition, "ST" stands for "Activity State" which shows a " ◇ " next to a number "1" if he is running, a "2" if he is ready to run, a "3" if he is waiting, and a "4" if he is blocked. Entries of "0" in this column are empty. The bar graph shows the percentage of time that a user is ready or running, computed by weighted average over the previous two minutes.

Figure 7 and 8. These two pictures display time distributions from the Segment Metering Table (See Chapter 3.2). They represent the amount of time for page and linkage faults during a particular instance of system initialization. Notice the automatic scaling.

CHAPTER I

INTRODUCTION

The Graphical Display Monitoring System (GDM) is an experimental monitoring facility for the Multics Time-Sharing System (see reference [2])* implemented at Project MAC. It allows design, systems programming, and operating staff to graphically view the dynamically changing properties of the system. It was designed and implemented by the author in an attempt to provide a medium for experimentation with the real-time observation of time-sharing system behavior.

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 output this information to 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 (for example, see [1], [9], and [16]) in an attempt to provide "after-the-fact" monitoring (in effect answering the question "What happened before, and before that...") and there have been simulations (see [9] and [16]) in an effort to provide predictive monitoring. 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 [17]. There has been little work, however, directed towards providing a general, real-time, time-sharing system monitoring device. The GDM System is an attempt to lead work in this direction. It is felt that while this

---

* References, indicated in square brackets, are collected in the Bibliography section which begins on page 73.

implementation is particularly machine dependent, the design principles involved and the monitoring methods explored are sufficiently general so that the GDM System provides a framework and a guide for other designers.

The basic goal of the GDM System was to provide a time-sharing system monitoring device for use by the staff of the Multics project. This implied that it would be on-line, that is, active while the system was in operation -- not just collecting data for future analysis, and would provide dynamically changing graphic output (as well as hard copy if desired). It was to be designed in such a way that the act of monitoring did not cause observable interference to the time-sharing system and that it would not be necessary to make more than a few minor changes to supervisory procedures in order to incorporate the GDM system (as opposed to the type of monitoring done by Scherr on the Compatible Time-Sharing System, [16]). 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, many useful ideas were received as more and more of the systems programming staff was made aware of GDM.

The current GDM System attempts to embody these goals while making use of the existing hardware at Project MAC. A more extensive (and less expensive) system could perhaps have been 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 chapters will discuss the various components of the GDM System in detail and will describe initial experiments as they were performed at Project MAC. Compromises in design and other special problems due to the particular constraints of the display hardware or software and the Multics system to which they interfaced are discussed in a separate chapter.

CHAPTER II

WHAT IS THE GDM SYSTEM?

The GDM System consists of four major components:

A. A 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 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 2-1 gives a functional representation of the various GDM subsystems showing the interaction among them, the two computers, and the user. Figure 2-2 gives a more detailed view of the hardware configuration.

The user of the GDM System will have his choice of several modes of operation:

1. Demonstration mode: any of several "canned" displays may be viewed to get a general picture of system 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 XRAY System, see [7]): the user may type the segment number and offset of a datum (see Appendix B for a description of the addressing scheme used in Multics) on the teletype and see displayed the octal and ASCII character representation of its contents, updated every second.
3. Display template mode: the user will go through the process of creating his own display (as outlined in Figure 2-1) in order to gain desired flexibility in data displayed, format of display, or data sampling rate.

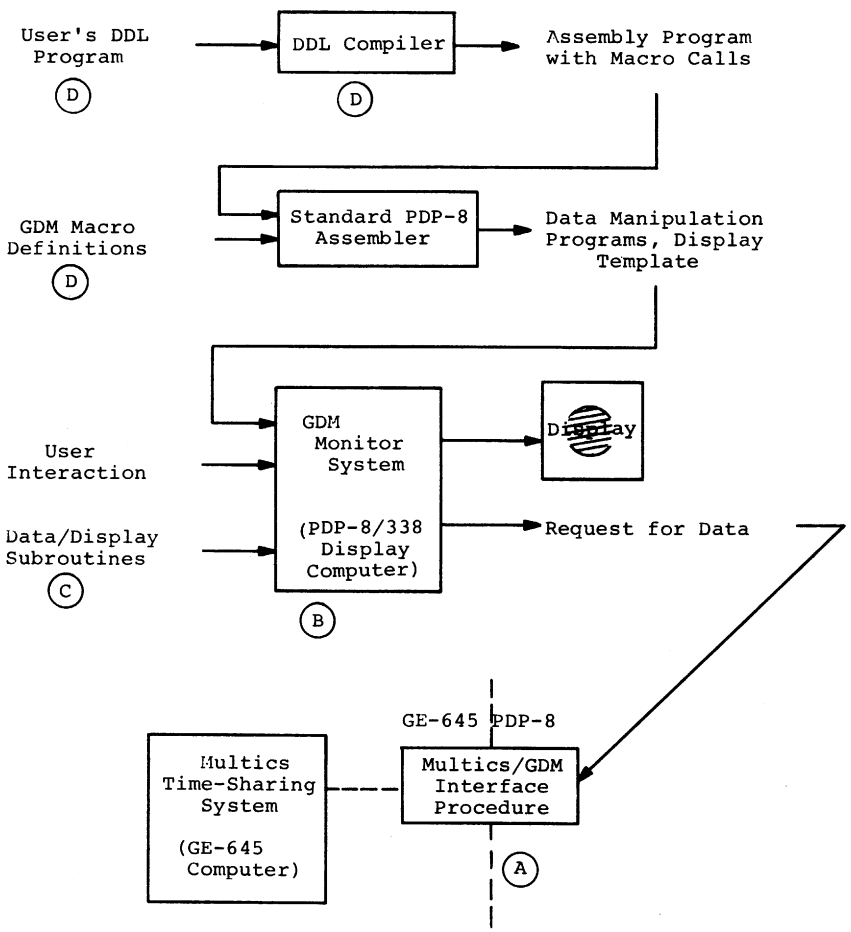
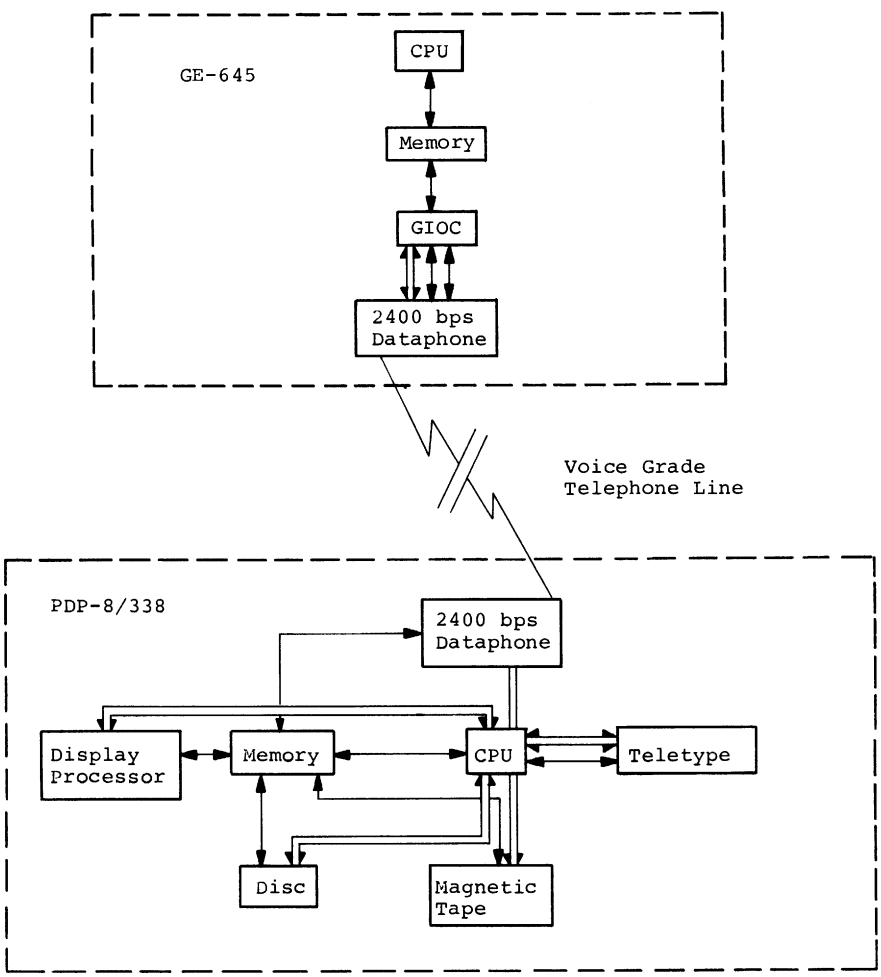


Figure 2-1. GDM Subsystem Interactions



(Single lines represent data transmission; double lines represent transmission of status information and interrupts)

Figure 2-2. Hardware Configuration

Display template mode, the most general use of the GDM System, will require 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 display template, either directly in PDP-8 Assembly Language or using DDL, for input to the GDM Monitor. It is in this mode of operation that all the facilities of the GDM System come into play and in which the most fruitful experimental work can be performed.

The following sections of this chapter will describe the GDM subsystems in sufficient detail so that the user will be able to derive full benefit from the display template mode of operation.

2.1 THE MULTICS/GDM INTERFACE

The GDM System is designed for use in a symbiotic relationship with a time-shared computer. The environment must be capable of supporting an attached display processor functioning basically independently of the time-sharing system but occassionally 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 input-output controller (called the GIOC on the General Electric 645 Computer, see [15]) without interrupting the central processing unit (see Figure 2-2 for a diagram of the hardware configuration). It is necessary, however, to dedicate two GIOC channel pairs (one for transmitting and one for receiving) to the display processor. This is not viewed as a major drawback as there are 4096 channels on each of two GIOC's on the Project MAC GE-645 computer. Those problems that are introduced by this relationship are discussed in a later chapter ("Unsolved Problems").

The Multics/GDM interface must then be capable of providing the following services:

1. Accept address requests by segment number and offset of data to be displayed.
2. Convert this address to an absolute memory location for interpretation by the GIOC.
3. Transmit the datum from the GE-645 to the PDP-8.

Figure 2-3 shows how these services are distributed between procedures in both computers.

The currently implemented interface can be divided into four parts:

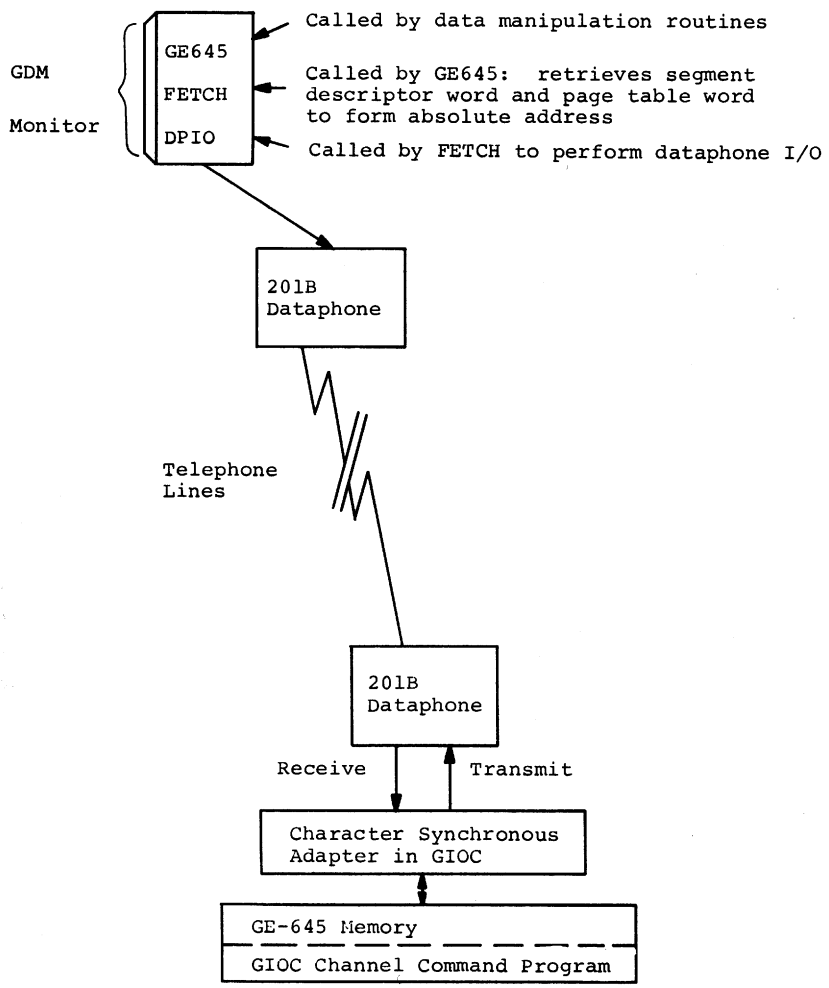


Figure 2-3. Multics/GDM Interface

1. A GE-645 GIOC procedure.
2. A PDP-8 routine for using the dataphone.
3. A PDP-8 routine to simulate the GE-645 addressing mechanism (see Appendix B).
4. A GDM subroutine to interface with the user.

2.1.1 The GIOC Data Retrieval Program

The GE-645 procedure is a modified version of the program used by the XRAY System [7]. It is a part of the Multics library and is called to initialize the two GIOC channel pairs dedicated to the PDP-8 whenever Multics is in operation. It is not necessary that the PDP-8 be active or even turned on at this time. Both channels are, however, always in operation: the receive channel waiting for synchronization characters followed by an 18-bit address (this only allows addressing of a 256K memory but can be expanded 6-bits at a time as necessary); the transmit channel transmitting the contents of whatever memory location is specified in its buffers as fast as the dataphone will allow (see Figure 2-3).

2.1.2 The PDP-8 Dataphone Routine

The PDP-8 dataphone routine simply sends a message whenever it wants data and waits for that data to be transmitted from the GE-645. A "key" is transmitted after the data address so that proper synchronization of sending and receiving can be accomplished. The key indicates completion of the address and is sent back to the PDP-8 along with the datum (Figure 2-4 shows the message format). If the sent key and the received key do not match, then the data transmission started before the GIOC had completely received the new address and the data is not wanted. If this is the case, the dataphone routine will re-synchronize and try again. Because the dataphone is only capable of 2400 bit per second full duplex operation, this method of transmission limits the actual number of data requests to a maximum of twenty per second.

2.1.3 The Address Formation Routine

The process of address formation from segment number and offset, called "appending," involves two data requests (in addition to retrieval of the data item itself): one for a descriptor segment word and one for a page table word. The GDM appending procedure has been designed to save the previously fetched DSW and PTW in an effort to cut down on the total number of data fetches required for a display. The incorporation of this small "associative memory" has

PDP-8:	Transmits	Receives
GE-645:	Receives	Transmits

26	KEY
26	C(1-6)
A(1-6)	C(7-12)
A(7-12)	C(13-18)
A(13-18)	C(19-24)
KEY	C(25-30)
177	C(31-36)
	377

A(x-y)= bits x through y of the address
C(x-y)= bits x through y of the contents

All messages are eight bits long. Where less than eight bits are specified, additional zeroes are sent at the beginning of the message.

Figure 2-4. GE-645/PDP-8 Message Format

proven very useful as there are often requests for several data items from the same segment (k|i, k|j, k|1, etc.) and sometimes for consecutive data items (k|i, k|i+1, k|i+2, etc.). (The GE-645 hardware actually contains a 16 location associative memory for this same purpose.)

2.1.4 The User Interface

The structure of many of the data bases in Multics is such that entries are often linked together in a list, or such that entries are really pointers to other data items. For this reason, the user interface to the various data-fetch routines allows several types of requests:

1. Fetch n words starting at location k|1.
2. Fetch the word at k|1. Take the left 18 bits as a new offset, j, and fetch n words starting at k|j.
3. As in 2 but take the right 18 bits as j.
4. Fetch two words, k|1 and k|1+1. Take the left half of k|1 as a new segment number, i; take the left half of k|1+1 as a new offset, j. Fetch n words starting at i|j. (This is the format of the so-called "ITS-pair" used for indirect addressing in the GE-645.)

These four options have been found to cover all but a few obscure cases of data requests. The user can, if he wishes, nest indirection by asking for a data retrieval and using the result as input to another retrieval request.

A general description of the use of this GDM subroutine is included under "The GDM Subroutine Package" later in this chapter. A detailed description of the calling sequence is found in Appendix E.

2.2 THE GDM MONITOR SYSTEM ON THE PDP-8/338

The GDM Monitor provides a convenient interface between the user of the GDM System and OS/8 File System (see Appendix C). By typing simple commands on the PDP-8 teletype, the user can cause display template tape files to be created or loaded, tape "snapshots" of displays to be taken or displayed, display data dispatches to be initiated, and various other miscellaneous functions to be performed. A summary listing of the commands is found in Appendix D.

2.2.1 Monitor Commands

The GDM command format is as follows ("(NL)" indicates the "new line" character):

```
tape_unit command_name arg_1 arg_2 arg_3 ... (NL)
```

where tape_unit is an optional argument indicating which unit number (1-8) is to be regarded as the current tape unit (default setting is tape unit 1). The command_name is one or two letters and the number and type of arguments depends on the command.

To load a display template file from a tape and initiate the display, two commands are required:

```
LN file_name display_name (NL)
display_name (NL)
```

The LN command (mnemonic for "load and name") will look for file 'file_name M' on the current tape unit and will load it onto the disc. Display_name (one or two letters) will be added to the command table so that the display can be started by simply typing its name (as in the second command line above). The LN command accepts multiple argument pairs on a single line and can store up to eight templates on the disc at a time. The "C" command will clear the disc and the display command name table if it is desired to start anew.

Once a display has been initiated by typing its name, the teletype is no longer active. Depressing the "manual interrupt" button on the display pushbutton box will temporarily suspend "execution" (data retrieval is halted) and allow commands to be typed on the teletype. An "R" command will resume the data dispatch.

If it is desired to stop the display at some particular point in its execution in order to study the display or photograph it, it is only necessary to depress the manual interrupt button. The "SN" command will save the display in a special tape file so that this "snapshot" may be viewed later. Typing the "SH" command will accomplish this.

Auxiliary commands are also provided to set the date display and reset the interval timer provided by the system (DD), print the current command and display names (P), and return to the OS/8 Operating System (Q). Commands to create display template files are discussed in Appendix C, "Using GDM under the OS/8 Operating System." Figure 2-5 is a general flow chart of the Monitor System Control.

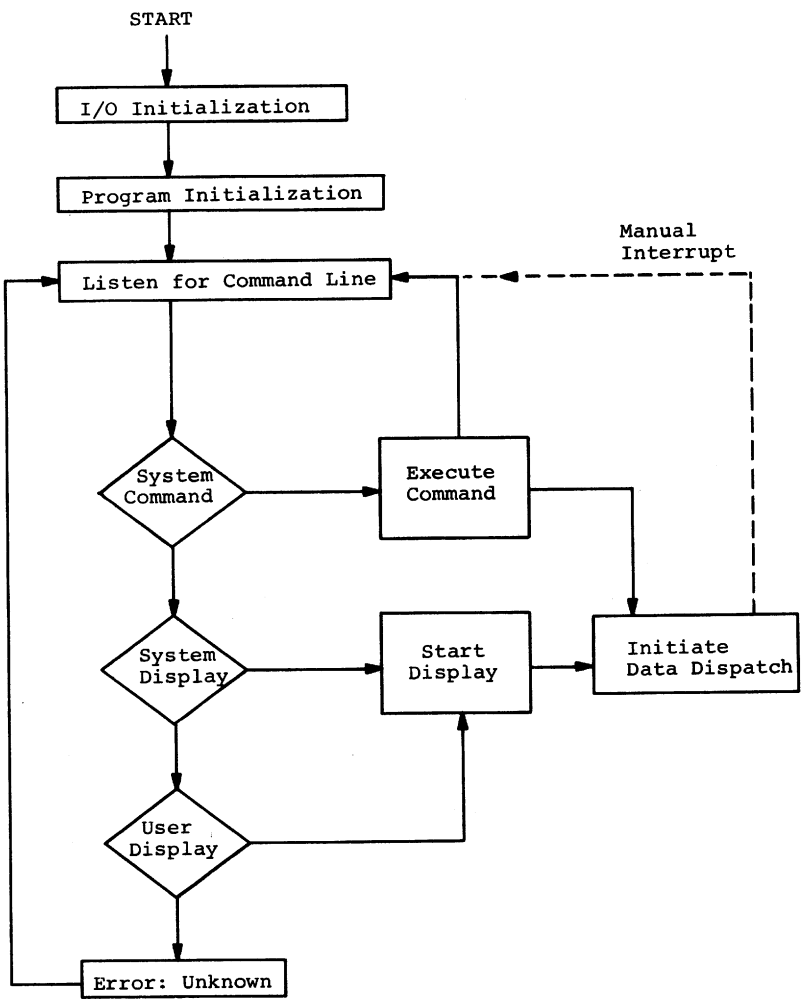


Figure 2-5. GDM Monitor System Control

2.2.2 Monitor Display Control

A major portion of the GDM Monitor is devoted to display control: that is, the "execution" of the particular display in accordance with the various data requests. This is the process that is initiated by typing a display name on the teletype. Figure 2-6 is an overview of the activities required to execute a display after it is brought into core memory.

Before continuing with a discussion of the display control system, it will be advantageous to describe a typical set of instructions for retrieving and manipulating the data to be displayed (see Figure 2-7).

The display template instructions are divided into "data routines." A data routine usually contains the instructions for retrieving, manipulating, and displaying a single data item or a group of related data items. Data routines may also be provided to perform initialization (these might not reschedule themselves) or set the interval timer counting rate.

A typical data routine might perform the following tasks:

1. Reschedule itself according to data sampling rate required.
2. Check for a pushbutton signal (for triggering purposes).
3. Request appropriate data fetches from the GE-645.
4. Mask or scale the datum to extract the relevant information.
5. Cause the item to be displayed in whatever format is desired.

Any routine mentioned in the "data dispatch list" (see Figure 2-7b and below) will be called initially but the routine must call the scheduler if it is to be called again.

Subroutines are provided to perform most functions related to data retrieval and display. These are discussed under "The GDM Subroutine Package."

The data dispatch routine initiates the display data dispatch by calling all routines mentioned in the data dispatch list (see Figures 2-8 and 2-7b). Once all these routines have been called and executed (each is run to completion), the dispatch routine checks the ready list for further work. If the ready list is empty, the dispatch routine will wait for work by constantly checking the list and executing whatever routine appears on it.

There is a possibility of a ready list overflow under this system only if (a) a routine schedules itself more than once; or (b) a routine schedules another routine in addition to itself. The dispatch routine will signal an error on the overflow condition and return to command level. Redesigning the display data routines with

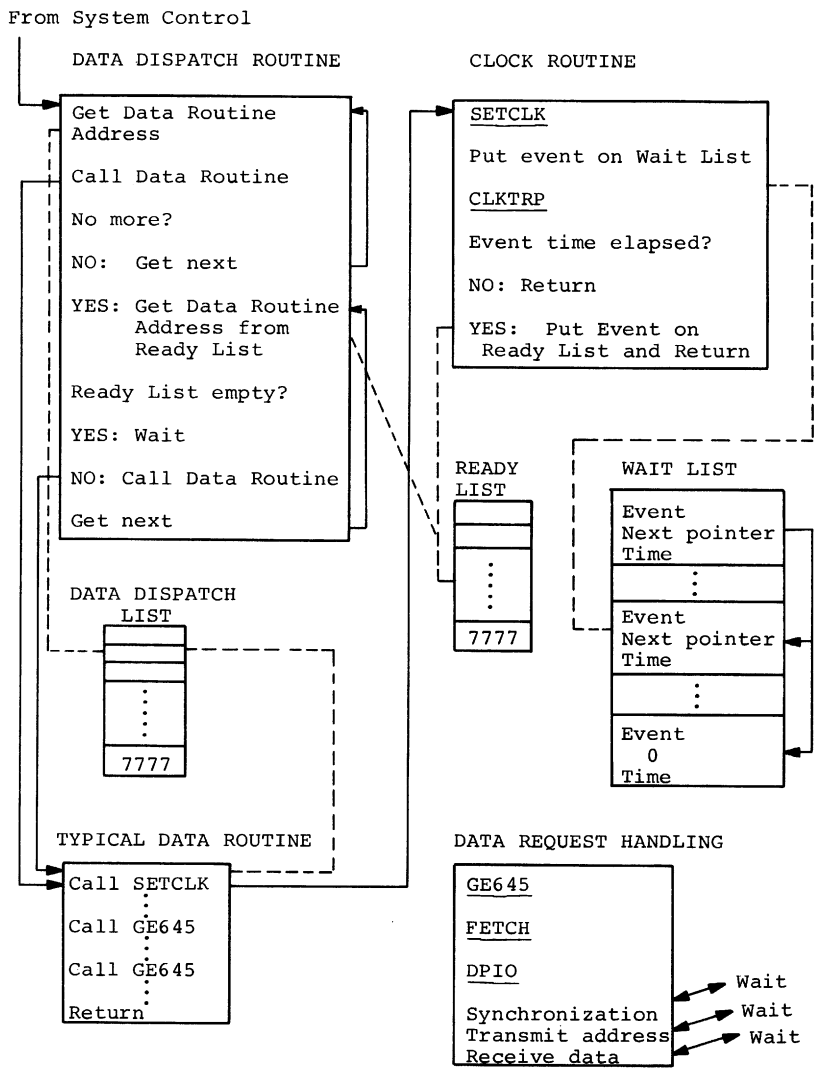
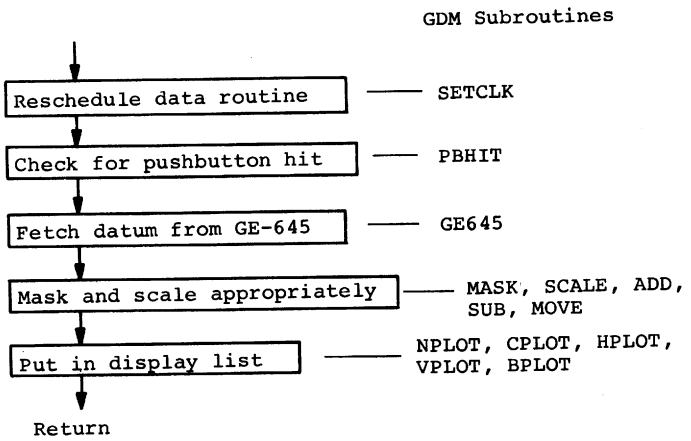
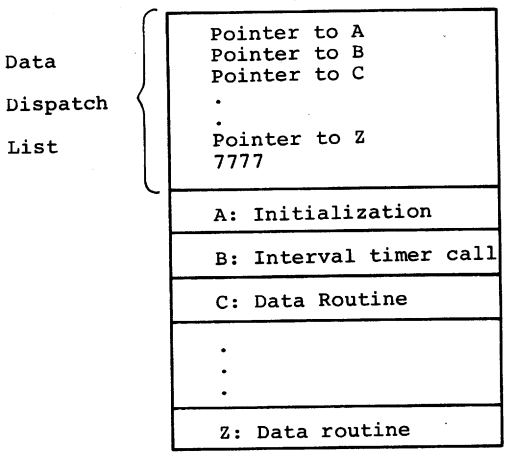


Figure 2-6. GDM Monitor Display Control



(a) Typical Data Routine



(b) Data Dispatch List

Figure 2-7. Display Template Data Routine Organization

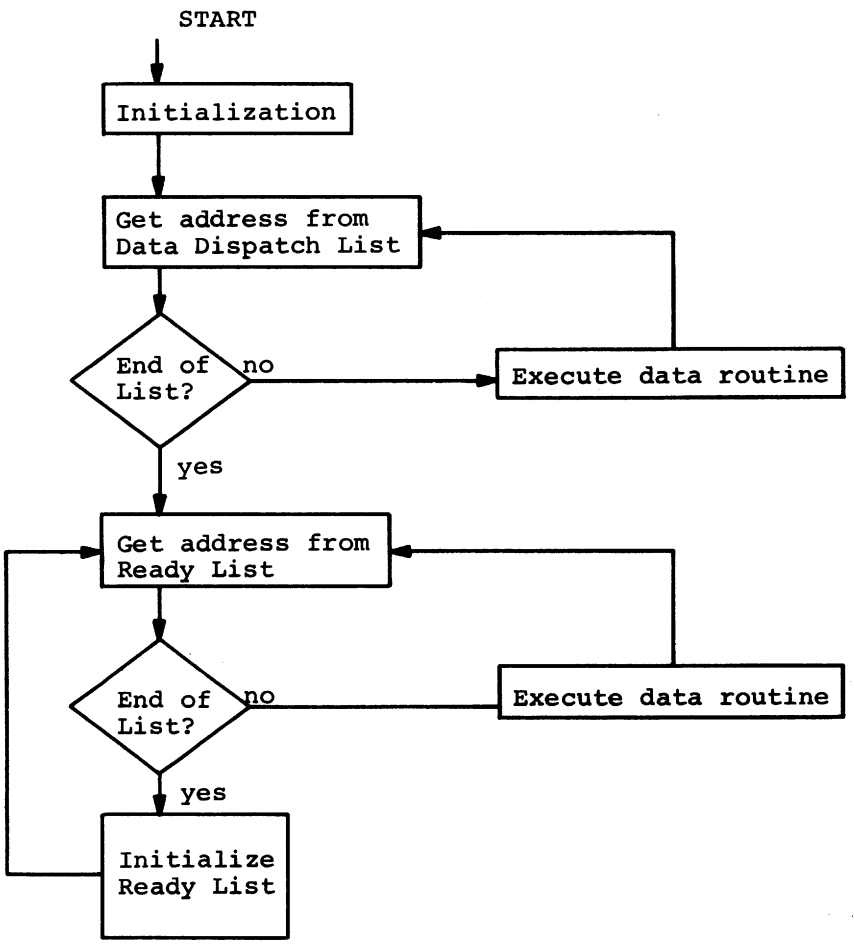


Figure 2-8. Display Control Data Dispatch Routine

fewer event reschedulings will eliminate this problem.

Another situation, equally as bad, is caused by rescheduling an event with a "sleep" time shorter than its execution time. As this event will always be put on the ready list, it will monoplize the processor after a short period. This can only be detected by noticing that certain parts of the display do not change (for instance, the interval timer!). Redesigning the offending routine with a longer sleep time or with the call to reschedule performed before exiting rather than after entering (essentially adding the execution time to the sleep time -- as in Figure 2-7a with dotted line) will cure this situation.

The clock routines provide the facility for scheduling events and signalling that an event is ready to be dispatched. They do this by keeping two lists: the "wait list" of events waiting for a certain period of time to elapse; and the "ready list" of events waiting to be dispatched (see Figure 2-6).

The "SETCLK" routine (see Figure 2-9) maintains the wait list as a thread of events in order of increasing time intervals. Times are stored such that the sum of the times in all blocks up to and including the one in question is the amount of time to elapse before the event is put on the ready list. A maximum of fifteen events are allowed on the wait list at one time and an error will be signalled if an attempt is made to add to a full list. If this occurs, several events should be consolidated into one so that less than the maximum of fifteen will be scheduled at one time.

The PDP-8 clock is set to interrupt the computer every hundredth of a second. The "CLKTRP" routine is called at each interrupt to determine if any event timers have run out. All events for which this is the case are put on the ready list to be executed in turn. If no events are ready, the current timer is decremented and control is returned to the interrupted process.

The limitation of 12-bit memory words in the PDP-8 and the desire to cause clock interrupts no less fequently than every hundredth of a second, puts bounds on event times of .001 second and 40.96 seconds. It should be noted that while events may be scheduled every hundredth of a second, it takes approximately a twentieth of a second to perform a data transfer from the GE-645 and rescheduling faster than this will cause one routine to monopolize the processor (as mentioned above).

An example of the use of these routines is in the interval timer control. Aside from the octal to decimal conversion, the routine does little more than count and reschedule itself to be called in one second.

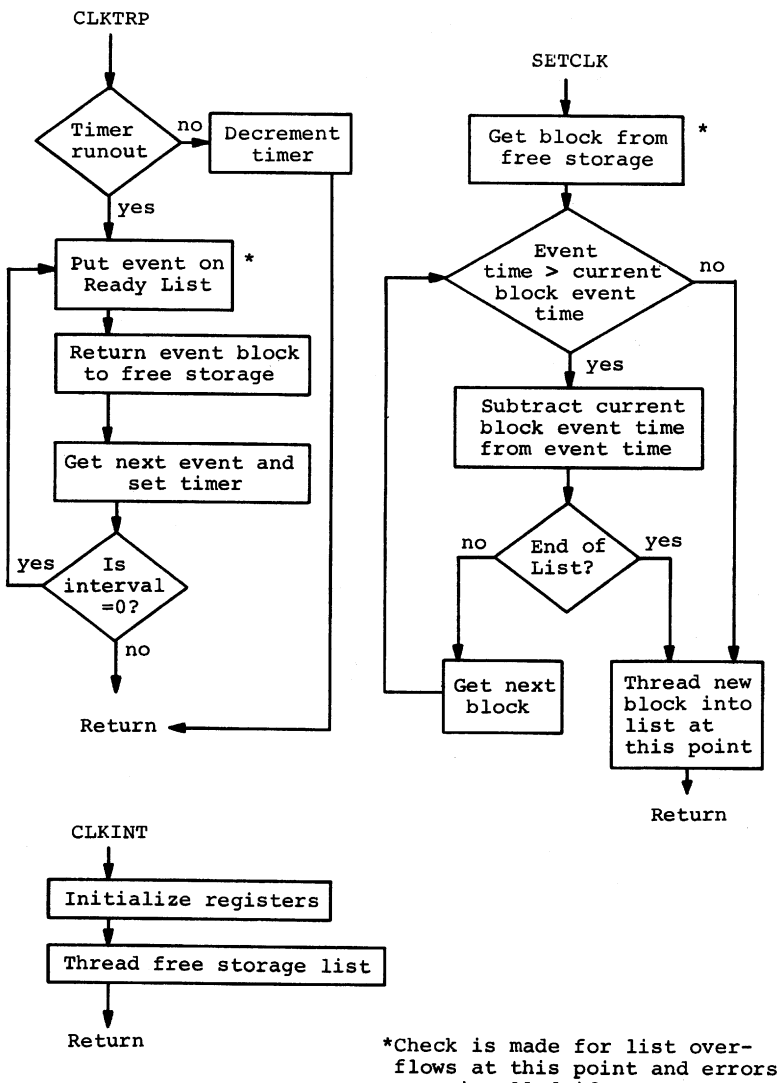


Figure 2-9. GDM Monitor Clock Routines

2.2.3 Other Monitor Facilities

The GDM Monitor also contains routines to process all computer interrupts, interface with the OS/8 File System in writing tapes, and manage display templates assigned to the disc. These routines are not called directly by the user, but rather indirectly in the course of processing a GDM command.

2.3 THE GDM SUBROUTINE PACKAGE

It is reasonable to assume that an intended user of the GDM System will learn the basics of operation and display template preparation in order to conduct his own monitoring experiments. It is not reasonable to assume that he will be desirous of learning all of the peculiarities of PDP-8 and 338 Display programming. Subroutines to perform most of the basic data manipulation and display functions are, therefore, provided by the GDM System. (Appendix E gives specific information on calling procedures.)

2.3.1 Data Manipulation Subroutines

Six subroutines are provided to handle data retrieval and manipulation. "GE645," the routine which controls data transfers, was discussed under "The Multics/GDM Interface" above. In general, it is only necessary to specify the type of request (direct fetch or any of the indirect types) and the display template address table locations of the segment number and offset of the data item. GE645 will always return the data item to three consecutive words (high order bits in the first word, etc.) starting at a prespecified location (referred to as "DATA" in the GDM macro definitions).

Display subroutines all assume that the information in DATA is in a format suitable for the display specified. In order to accomplish this, the user can call upon routines "SCALE" and "MASK" to shift the data item and retain only the relevant bits. The display subroutine descriptions in Appendix E indicate the data format expected.

Certain data items are more meaningfully displayed after comparison with other items. Three subroutines facilitate such manipulations. "ADD" and "SUB" perform the addition or subtraction of two data items (or constants) and return the result to locations "DATA" for plotting. Because GE645 always uses the same area for data transfers, the "MOVE" subroutine is provided for storing data in temporary locations.

If, for instance, it was desired to plot the difference of the left half-words of two data items, A and B (this might be the relative length of a storage area computed from pointers to the beginning and end of the area), the following actions would be necessary:

```
Call to GE645 to retrieve data item A.
        MOVE to store A temporarily in AA.
        GE645 to retrieve B.
        MOVE to store B temporarily in BB.
        SUB to subtract AA from BB (result in DATA).
        SCALE to shift result to right half word.
        MASK to extract number of bits needed for plotting.
```

2.3.2 The Display Subroutines

Several different types of displays can be plotted using the display subroutines. These conform to the types allowed by the Display Description Language (see below) and seem to cover most display requests.

The simplest display shows all or part of the octal or ASCII character representation of the data item. Calls to "NPLOT" and "CPLOT" will accomplish this. NPLOT displays the number of octal digits specified, taking them from the rightmost part of the word (here is where scaling and masking can be used to cause display of only certain bits). CPLOT interprets the appropriate number of 9-bit groups (starting from the high order or leftmost part of the word) as ASCII characters. If a control character or a non-existant character is detected, a special symbol is displayed.

Another useful display type is the bar graph. The 338 hardware makes it especially easy to plot such graphs by providing a "vector" plotting mode. Routine "BPLOT" will handle the mechanics of placing the data item into the display list appropriately.

The plotting of "versus-time" graphs and "multiple-state" graphs, as in Figure 2-10, is accomplished through a special use of the 338 Character Generator (see Appendix A). Character codes for drawing lines of different lengths and skew have been appended to the standard character codes for use in these graphs. "HPLOT" (horizontal plotting) and "VPLOT" (vertical plotting) are used to simplify the procedures. By keeping track of the magnitude of the previous item, they will compute the code of the appropriate character needed to display a continuous graph. HPLOT also has a facility for directing non-intensified horizontal movement of the CRT beam needed for certain types of graphs.

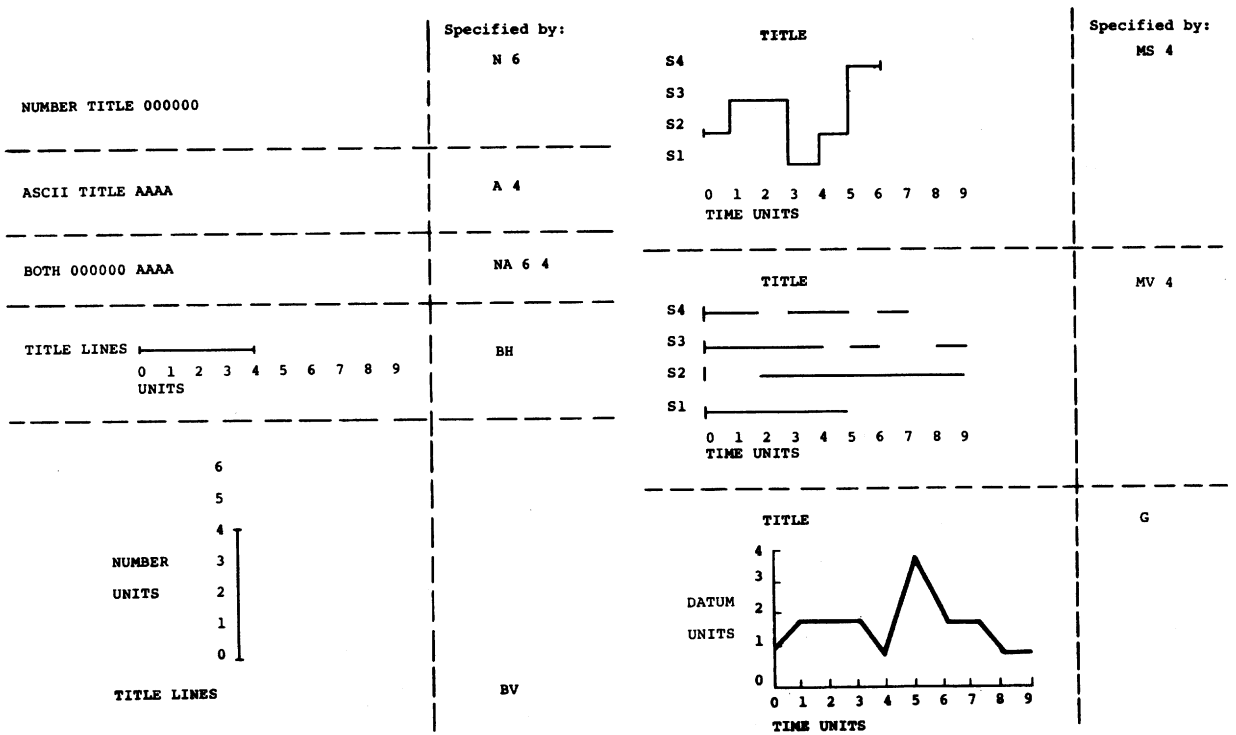


Figure 2-10. Standard Display Types

It should be noted that while the 338 hardware and the GDM software are capable of displaying any conceivable form, those provided by the display subroutines appear to be a reasonable selection. If further experience indicates the advisability of new display types, it is a very simple matter to incorporate them into the system.

2.3.3 The Pushbutton Routine

Routine "PBHIT" controls triggering of display data requests if this is desired (see below, "The Display Description Language"). If pushbutton 0 is on (lighted), all displays that call PBHIT will get the "no hit" return (see Appendix F) unless the specified pushbutton is also lighted. Turning off pushbutton 0 will cause PBHIT to return to the "hit" return no matter what the state of the specified pushbutton.

Figure 2-7 indicates that calls to PBHIT be made after calls to reschedule. This has two implications in triggering mode (pushbutton 0 lighted): (1) the routine will be recalled even if the specified pushbutton is not lighted; and (2) the specified pushbutton will only be interrogated every n seconds, where n is the rescheduling time. These facts should be considered when planning displays with critical timing.

2.3.4 The Interval Timer Routine

The GDM System provides both a date display and an interval timer display in the upper right corner of the screen. As has been mentioned above, the date is set and the timer reset from GDM command level by the "DD" command. The timer routine, however, is initiated by a simple data routine in the display template (an example of this can be found in Appendix H). Once initiated, the timer will count every second, thus providing a record of relative time in a particular monitoring session. Both date and timer will be saved whenever the "SN" command is used to save the current display.

2.4 THE DISPLAY DESCRIPTION LANGUAGE

The Display Description Language (DDL) of the GDM System provides a comprehensive yet well defined way of communicating the user's desires to the rest of the system in the form of a display template.

Statements in DDL are of three types: (1) format control; (2) data description; and (3) display component description. Display components may also be of the following types:

1. numeric information
2. alphabetic information
3. horizontal bar graphs
4. vertical bar graphs
5. "multiple-state time bars"
6. "... versus time" graphs

Each display component statement, along with its referenced data description statement, will cause assembly code to be generated for both a data routine and a portion of a display list (command program for the display). Display initialization sequences and the data routine dispatch table are also generated by the compiler.

The syntax for all three types of DDL statements is explained in detail in the following sections. A DDL "program" consists of any number of type 1 statements and up to 100 each of type 2 and type 3 (several type 3 statements may refer to the same type 2 statement) followed by: END.

2.4.1 Format Control

The object of a DDL program is to provide a complete description of a display. Format controls serve to position the various graphs, numbers, and text strings in a manner pleasing to the eye. Format controls in DDL are separate statements (terminated by a semicolon) and may appear anywhere throughout the program.

The display screen on the 338 is 9-3/8 inches square. This is equivalent to approximately fifty lines of seventy characters each (space between lines is equal to 3/7 of a character). All display commands can be thought of as positioning the cathode ray gun. When an instruction to "intensify" is given, electrons are fired from the gun and an image of the movement is seen. The CRT beam is initially positioned to plot the first character on the first line (upper left corner). Format controls cause non-intensified beam movement (positioning) or text plotting (the beam traces the characters and is then positioned after last character plotted).

The DDL statements indicated below cause beam movements on n characters. If no "n" is present, one character is plotted:

```
NL n;          (new line)
SP n;          (space)
HT n;          (horizontal tab -- 10 spaces)
```

A text plotting statement takes the form:

```
T 'text';
```

Because of limitations in the character generation routines, lower case letters are mapped into the corresponding upper case characters for plotting. If the leading "T" is replaced by "SP," "HT," or "NL," a space, horizontal tab, or new line, respectively, will be plotted before the text string. This option is also available in display description statements wherever a "T" control character is indicated.

2.4.2 Data Description

A data description statement consists of a label, a data unit, and a terminator:

```
<label>:<data unit>;
```

A <label> is any two-digit (decimal) number, thus allowing one hundred data unit statements per display description.

Display component descriptions refer to data units as the information to be displayed. Data units are described by DDL statements according to the following syntax:

```
<data unit>::=<scaled datum>|<scaled datum><op><data unit>
<scaled datum>::=<datum> [<datum>,<sign><scale factor>,<mask>
<op>::=<sign>|,
<datum>::=<octal constant>|<GE-645 word>
<sign>::=+|-
<scale factor>::=0|1|2|3... |42 |43 (octal)
<mask>::= scale factor>
<octal constant>::=0|1|2|3...777777777776|777777777777 (octal)
<GE-645 word>::=G1|G2|G3...G63|G64
```

A <GE-645 word> is a 36-bit binary number retrieved from a specified location in the GE-645 memory. The memory locations corresponding to the <GE-645 word>'s in the display description are specified separately by the user (see the section on "Display Template Address Table").

Certain display component specifications require certain types of data unit specifications (i.e. request for more than one GE-645 word: <scaled datum>,<scaled datum>). The display component description will indicate if an unusual data unit is required.

Data scaling, when specified, is accomplished as follows: the 36-bit datum is shifted right (+) or left (-) by the specified number of bits and padded with zeros as appropriate. The mask is then applied to extract the indicated number of bits from the right (low order) end of the 36-bit number. (These functions are performed by calls to GDM Subroutines MASK and SCALE.) Displays which require a certain number of bits of precision (six bits for sixty-four possible variations, three for eight, etc.) will take them from the rightmost (low order) end of the 36-bit word. Scaling and masking, therefore, should be specified as required by the particular display component referencing the data unit statement.

2.4.3 Display Component Description

Display component descriptions should specify sufficient information to the DDL compiler for it to (1) form the portion of the display list relating to the particular component and (2) set up reference locations for data to be inserted in the display by the referenced data routine. (Because several display components may reference the same data description, the compiler cannot complete the data routines until all references have been found and the appropriate instructions added to the routine.)

Each display component statement consists of several basic arguments and special arguments as necessary:

```
type title data rate trigger optional_args;
```

The display type is one of the following (see Figure 2-10):

N n	numerical display of n digits
A n	ASCII display of n characters
NA n1 n2	n1 digit numerical display followed by an n2 digit ASCII character display
BH	horizontal bar
BV	vertical bar
MS n	type "a" multiple state time bar of n states
MV n	type "b" multiple state time bar of n states
G	"...versus time" graph

Interpretation of these codes by the compiler will cause the appropriate display list to be generated and appropriate links to be established to the data unit referenced later in the statement. As the compiler makes use of the GDM Subroutine Package, scaling and masking should be specified in the data unit statement according to

the display types referencing it.

Numbers to be plotted are taken from the rightmost octal digits of the word (as mentioned above). A request for 6 digits, therefore, would cause the right half of the word to be plotted. Character displays, on the other hand, take bits starting from the left end of the word. A request for 2 characters would cause the 18 bits in the left half of the word to be interpreted as characters. Because the subroutines only look at the indicated bits, no masking or scaling is normally necessary. If, however, it is desired to display a single digit from the high order portion of the data item, the word must be scaled first (by appropriate indication in the data unit specification).

Bar graphs have 512 possible positions (taken as the low order 9 bits of the data item). A change of eight in the data item will cause a 1/4 inch change in the bar length. Data should be shifted (and masked!) to take advantage of this.

As was indicated, multiple state time bars are of two types: (a) single datum (MS); and (b) multiple datum (MV). The single datum must be scaled so that the number of bits remaining will give the desired number of states: e.g. three bits will allow eight states; two bits, four states; etc. In multiple datum specifications, there must be as many data unit components (separated by commas in the data description statement) as states. An intensified line will be plotted next to any state whose data item is non-zero at the sampling time. If it is zero, the CRT beam will be moved but no image will be displayed. It is possible to monitor any bit of the data item in this manner if appropriate scaling and masking are used.

The title is specified as a standard text string:

```
T 'text'
```

while the data unit number is specified as:

```
D n
```

where "n" is the label of the data description statement (1 to 100). As mentioned above, each data unit reference causes information to be added to the specified data unit routine. This information will be combined with generated labels in the display list to ensure proper placement of data items for plotting.

Sampling rate is specified in hundredths of seconds, 1 to 4096:

S n

This specification causes information to be added to the referenced data routine so that calls to "SETCLK" can be made with the specified rescheduling time.

The triggering argument is either:

X n

or R

If an "R" is specified, the display is "repeating" and no further action is taken. An "X" specifies triggering by pushbutton "n." A call to "PBHIT" is added to the referenced data routine as shown in Figure 2-7.

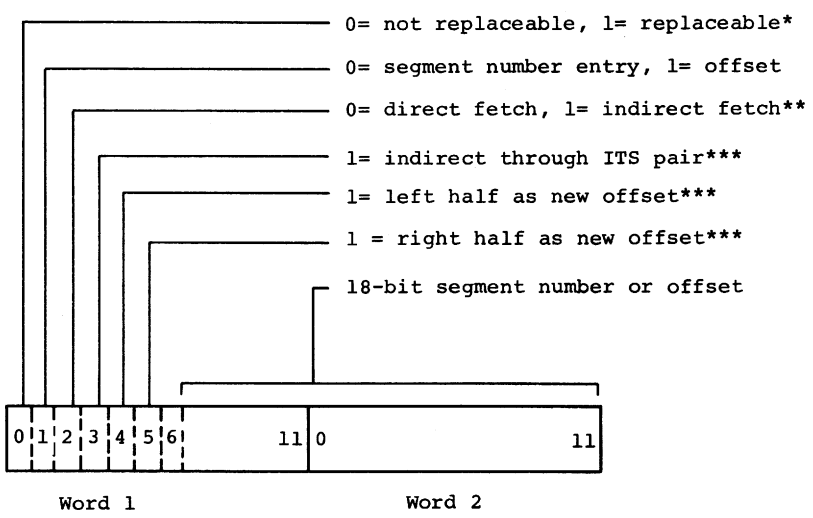
Optional arguments are always to specify additional labels or spaces between graph markings. The DDL summary in Appendix G gives the complete format for each type of statement.

2.4.4 Display Template Address Table

The address table of GE-645 words (see under "Data Description") must be specified independently of the DDL display description. It is expected that this design deficiency will be corrected in the near future. A general description of its format should, however, clarify its relationship to the rest of the display template.

Each segment number or offset is specified in two PDP-8 words as shown in Figure 2-11. The 18-bit number is placed with the high order bits in the right half of the first word and the low order bits in the second word. The left six bits specify control information in accordance with the different data request types mentioned under "The Multics/GDM Interface." Bit 0 indicates that the GDM Monitor can replace these two words with an absolute address if necessary. (This means that the DSW and PTW need only be retrieved once for this item. See Appendix B.) Bit 1 indicates whether this is an offset or a segment number and the remaining bits specify indirection as indicated in the diagram.

The address table has a particular position in the display template (see Figure C-1) and will, as noted above, eventually be put there according to DDL language statements. An assembly language file, created according to instructions contained in Appendix C, should be used until then.



* If an entry is replaceable, it will be replaced with an absolute address after the first reference. Bits 0-5 will be set to 111111, an otherwise illegal bit pattern.

** Information on indirection is contained only in the "offset" entry.

*** Only one bit in the group 3, 4, and 5 should be on at one time.

Figure 2-11. GE-645 Display Template Address Table Format

2.4.5 A Sample Display and Description

A number will vary continuously between 0 and 77 (octal). We wish to display both a horizontal bar and an eight-state time bar as well as to display the actual number. We wish the numeric and bar display to be repeating and the time bar to be triggered by pushbutton 6. (See Figure 2-12.) See Appendix G for further explanation of the display component statement format. (Line numbers in parentheses are for reference and are not part of the program.)

```
(1)   NL;
(2)   HT 2; T 'RANDOM NUMBER' ;
(3)   NL 2;
(4)   N 2 T'NUMBER= ' D1 S100 R;
(5)   NL;
(6)   BH T '64 POSITIONS' D1 S100 R M8 'TEN CHANGES';
(7)   NL;
(8)   MS 8 HT'8 POSITIONS' NL'SEVENTY' NL'SIXTY' NL'FIFTY'
      NL'FORTY' NL'THIRTY' NL'TWENTY' NL'TEN' NL'ZERO'
      D2 S100 X6 M1 'TIME IN UNITS';
(9)   1: [G1, 0, 6];
(10)  2: [G1, 3, 3];
(11)  END.
```

Lines 1, 2, 3, 5, and 7 are format control statements. Line 4 indicates that the datum requested (specified in line 9) should be displayed as a two-digit number while line 6 indicates that the same datum should also be displayed as a bar graph. Line 6 is the specification for the multiple state graph with labeling as shown in the diagram. It should be noted that while data item "G1" is still displayed (line 10 reference), it is scaled and masked differently than for the numeric or bar graph display. The fact that "G1" is referenced in both lines 9 and 10 will assure that the same item is displayed (only one fetch is made).

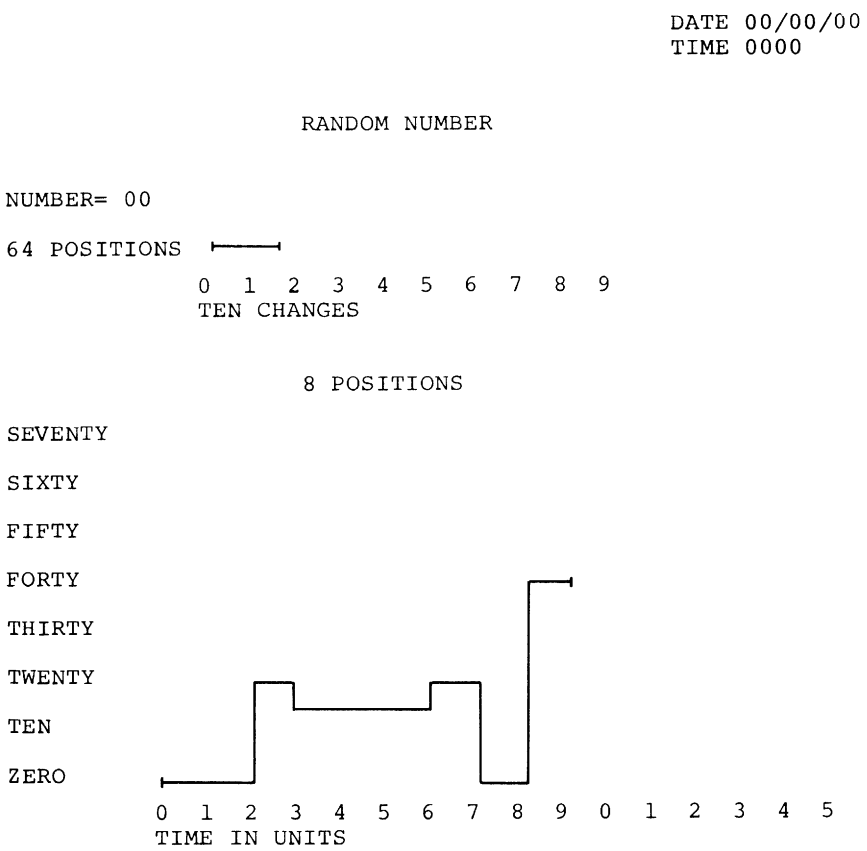


Figure 2-12. Sample Display Format

CHAPTER III

A MULTICS MONITORING EXPERIMENT

One of the more difficult problems in designing a computer monitoring system is in determining just what to monitor. If a hardware monitoring device is used then this is an especially important decision as it is relatively final: redesign will be costly (see [17] for one solution). Where little special hardware is required, as in GDM, care must still be taken to ensure that massive reprogramming is not required when monitoring requirements change.

With this in mind, very few restrictions were put on the GDM System. In particular, almost every system wide data base in Multics can be monitored and many "per-process" data bases are also accessible to GDM observation. As a matter of fact, the contents of any location in core memory may be observed if the GDM user knows what he is looking for! (This points out a problem of security which is discussed further in Chapter IV. It also shows how an experiment might be performed to determine how often a particular core memory page is changed.)

With this flexibility available to any user, the design of useful displays for the GDM demonstration mode is complicated: the data they display must have particular relevance to the observation of overall system performance.

Demonstration displays are, of course, created in exactly the same way as any user display (see Figure 2-1). There are instructions within the system, however, to automatically load these displays when GDM is initialized. XRAY mode of operation is just a special case of demonstration mode in that the XRAY display is also loaded at initialization but is a special interactive display (see Appendix H for a description of the XRAY display).

Several criteria were used in determining which displays would serve as demonstrators. They would, first of all, display as many features of Multics operation as possible. They would attempt to give overviews but would supply details where these would give a better idea of system performance. Finally, demonstration displays were to be designed to show off as many features of GDM displays as possible without cluttering up the picture.

Two basic displays were decided upon for the initial version: (1) a system overview and (2) a process overview.

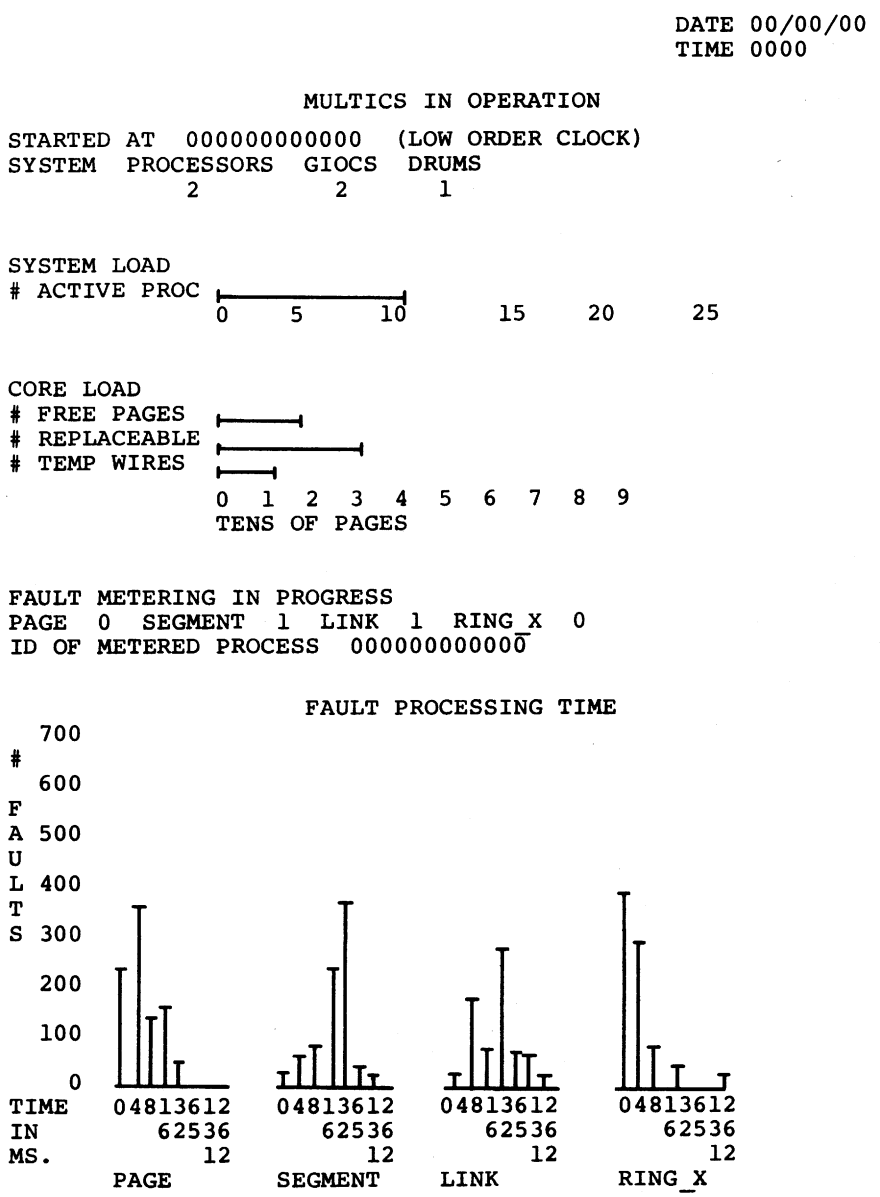


Figure 3-1. Multics Overview Display

3.1 A SYSTEM OVERVIEW DISPLAY

A system overview must provide not only a picture of what is happening in the system, but what the system is! The first Multics overview display (Figure 3-1) attempts to show these things by selectively displaying data from the following Multics data bases:

1. Major Module Configuration Table (MMCT) -- information about the hardware modules (memory, CPU, GIOC, drum, etc.) currently attached to the system.
2. Core Map (CORMAP) -- tables showing the current usage of each page of core memory.
3. Active Process Table (APT) -- contains an entry for each process in the system.
4. Segment Meter Table (SMT) -- a variety of statistics collected for independent metering experiments but contained in very neat core buffers.

Items taken from the MMCT do not change and so are retrieved from the GE-645 only once. The core map, however, is very volatile and is sampled as often as is practical (about every second in the present version). Similarly, the number of active processes is determined every few seconds. SMT entries are sampled according to their volatility (the system start-up time, for instance, need only be checked once!).

The core map is a source of particularly interesting information. A good indication of system loading is provided by observing the number of core pages with wired-down, replacable (liable to be paged out), or free status. The CORMAP segment contains header information (in addition to lists of pages) which gives the total number of pages in each of these and several other categories. These are displayed as bar graphs scaled to commensurate levels for easy viewing. Pushbutton display control is also available.

Any instruction on the GE-645 may cause one of the following faults:

1. Page fault -- the page containing the word requested is not in core.
2. Segment fault -- no part of the segment containing the word requested is in core.
3. Linkage fault -- the segment number of the segment (symbolically) requested must be determined before the instruction can continue.

4. Protection fault -- the segment requested is in a protection ring other than the currently operating procedure (see[11]for a complete description of the Multics protection mechanism).

The segment metering process simply records in the SMT the number of faults of each type as they occur and the time for processing each. The demonstration display will allow viewing the results of this metering via a pushbutton controlled vertical bar graph. By depressing the appropriate button, the graph will record the SMT table entries, as they are made, for one or more of the fault types. It will also show (by indicators on the display) what type of metering is in progress, the identification number of the process being metered, and other relevant parameters.

The active process table contains status information about each system and user process (see below). Determining the total number of entries, however, requires (1) fetching the table size, (2) fetching the empty list size, and (3) dividing by the entry size [16]. The number of active processes is an important indicator of system load and is displayed numerically and as a bar graph.

Initial experiments with this display have been useful in showing the advantages of real-time graphic output of the metering experiments mentioned earlier and have given the additional information about core utilization that has been so long sought after. These experiments also served as an introduction to the GDM System for many of the Project MAC staff and as such elicited many helpful suggestions for dynamic monitoring experiments.

3.2 A PROCESS OVERVIEW DISPLAY

Following an active process through its operation will provide information on how much time the system spends in the supervisor, in waiting, and in computing for the user.

Unfortunately, a demonstration display showing a particular process' history would be hampered by the relatively low data sampling rate allowable under GDM (the fastest rate, for one word transfers, is twenty cycles per second). State changes of a process running in Multics occur too rapidly for GDM to detect them and, even if this were possible, the display would change too rapidly for the eye to follow! (A further discussion of this problem is given in the next chapter.) Instead, an attempt was made to monitor only those state changes that take a process through "command level": that is, the period during which a command line is typed and executed and the time during which the process is waiting for the

next command to be typed. This essentially means that for the purpose of display and monitoring several process states are merged into one, as shown in Figure 3-2.

Previous studies have shown that the average "think time" between commands is on the order of thirty seconds [16]. A display of approximately two minutes of console operation at a time should, therefore, adequately portray the percentages of time spend in the various states. The assumptions made by the display are that (1) a process which is "blocked" (in the state named "blocked") is awaiting console input; (2) one that is not blocked (except "stopped") is executing a command; and (3) one that is "stopped" is not worth looking at for the moment (a new process is found for display). Because of the structure of the Multics process control mechanism, these assumptions are generally valid. The "loading state" of the process, displayed in parallel with console activity, shows whether the process has any dedicated core memory pages (the "loaded" state). This will usually be the case only during the command execution period.

Associated with each active process is an entry in the Active Process Table where this information is available. The various items in this entry tell the identification number, state, class, ready list level number, loading status, and descriptor base register value of the process. Another segment associated with the process, the Process Data Segment, contains (among other items) the current protection ring number and the "block lock count," the number of data bases this process has currently locked. While the particular demonstration display does not show all these, other special purpose displays may in the future.

The demonstration display shows several active processes (actually, up to about twenty can be displayed at once if the extra spaces are "squeezed out" of the picture shown in Figure 3-2) at once in order to provide the observer with another type of system overview. The GDM user is shown a record of console activities for many of the user processes active in the system so that he can, at once, see the span of activity rates and get a general, "mental" average of system load. A display which simply showed some sort of statistical measure of all process' activity would not adequately portray the available information. Designers of a developing system, such as Multics, would more than likely want to view the individual data items than see a statistic which will tend to obscure the presence of minor, but important, deviations from the norm. The data sampling rate of the GDM Monitor, while limited and generally a disadvantage, proves to be sufficient in this case to

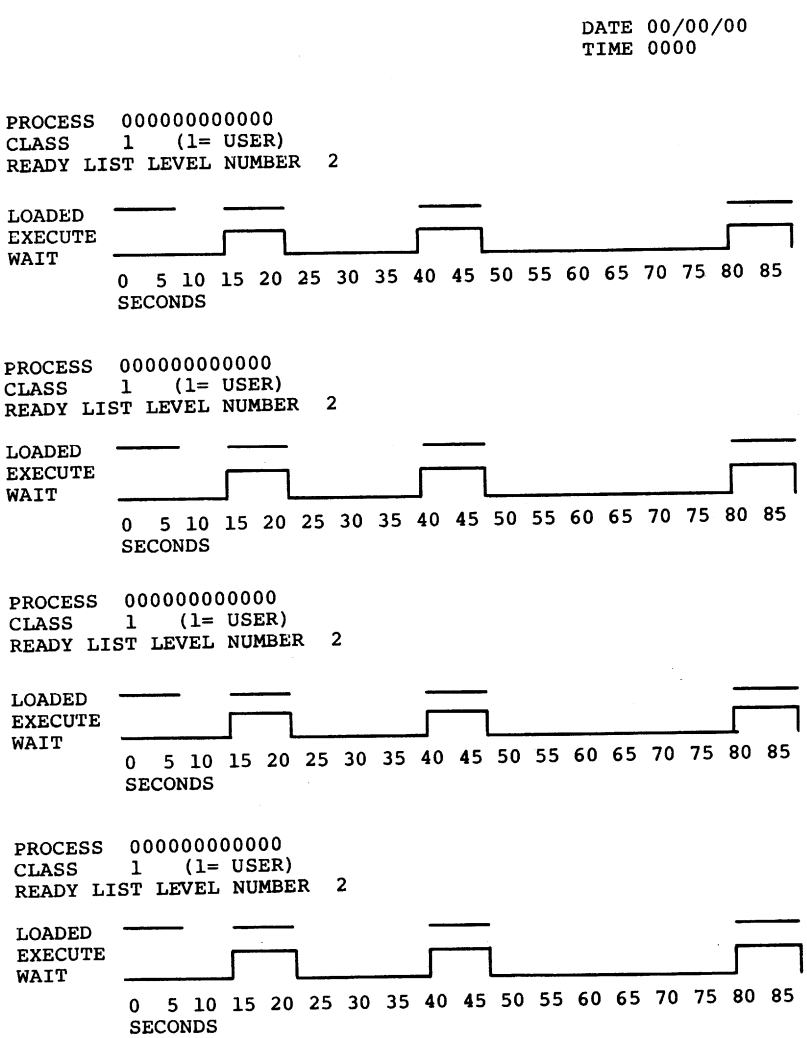


Figure 3-2. Process Overview Display

monitor a large number of users at once.

Initial experiments with this display have shown it to be useful in gauging the "intensity" of console activity. The number of commands a user issues per period of console time will be a major factor in determining every user's average response time. Intensity of use, therefore, is an important parameter to measure when deciding upon the maximum allowable number of simultaneous users. Further experimentation with this type of display is definitely indicated.

This empty page was substituted for a blank page in the original document.

CHAPTER IV

OBSERVATIONS AND CONCLUSIONS

The GDM System was developed as part of an effort to prove the feasiblity of on-line, real-time monitoring of time-shared computer systems. As such, it was necessary to demonstrate its usefulness via pre-planned displays. The System is sufficiently "open-ended," however, so that the real advantages of such a system will be seen when the programming systems staff uses GDM to measure the effects of installing new supervisor modules, or increasing the amount of core memory dedicated to an individual process, or removing a memory bank, or changing the scheduling algorithm, or any of a host of other tasks.

The number of experiments that can be performed with GDM is almost endless. Many of these will have to wait, in this case, for the Multics user community to be established so that measurements can be made on an operational (rather than prototype) system. At other installations, it will only be necessary to modify GDM (or a GDM-like system) to the peculiarities of the time-sharing system it is to work with. A discussion of some of the problems encountered at the present installation will, perhaps, aid this effort.

4.1 UNSOLVED PROBLEMS

The GDM Monitor was generally designed to accommodate all requests for display management: from creation to execution to storage. These are fairly complete mechanisms and should suffice for any system. The Monitor requires, however, that data requests be compatible with the system to be monitored: the GDM subroutine "GE645" (discussed in Chapter II) and its associated routines, are specifically designed to allow efficient retrieval from a computer with two dimensional addressing, such as the GE-645 used by Multics. This is a limitation which will require the GDM implementor to analyze the method of data retrieval for his own system- possibly a sizable task. A more generalized data retrieval package would perhaps have furthered machine independence of GDM but this was viewed as a very secondary goal.

The general problem of the interface between the time-shared computer and the display processor deserves further investigation. The very pragmatic approach of using available equipment was taken in designing the current system and has worked fairly well. The

biggest problem to date has been the "low speed" of the dataphone interconnection. With appropriate control and synchronization messages, a maximum of twenty 36-bit words can be retrieved in one second (see Chapter II for a more complete discussion of the retrieval mechanism). This, of course, keeps interference to the time-sharing system at a negligible level but also severely limits the type of monitoring that can be performed.

As mentioned in Chapter II, an important monitoring experiment would be one in which a process is traced through all of its state transitions (perhaps this information could be recorded by GDM and "played back" slower than real-time). Not only would the percentage of fault time (page, segment, ring crossing, or any other types peculiar to the system) be measured, but the temporal relationship of fault and computation time periods would also be determined. This would, of course, be useful in attempting to reduce fault time and interference, a goal of all time-sharing system designers. As most GDM subsystems are input/output bound (most of their time is spent in waiting for I/O completion), an obvious solution to the problem is to increase the dataphone speed. This and another suggestion are discussed further below.

Another major problem, not only in GDM, but in any monitoring system, is the determination of what access the monitor will have to system and user data bases. Many data bases which provide useful monitoring information may also provide information about a user or about the system which should not be seen by everyone. This issue of system security is an important one if time-sharing users with proprietary information are to be encouraged. In GDM, the security problem has been temporarily circumvented: first of all, the users of GDM are currently the Multics design staff who, presumably, know what information is available anyway; second, user information is, at best, transient in core and the GDM user would have a difficult time keeping track of some item (as noted in Chapter II); and third, the GIOC channel program cannot "monitor" requests according to preassigned access privileges-this can only be accomplished by a process running under Multics and, therefore, taking CPU time (the Multics I/O system, in fact, must postpone access checks until the data has already been brought into its private buffers). The issue is certainly not resolved but it is expected that the GDM System will have to be made more secure if non-systems programming staff are to be allowed to conduct monitoring experiments.

A partial solution to the data base protection problem was proposed during the design of the GE-645 (long before GDM was dreamed of!). This was to include "memory bounds registers" in the

GIOC Character Synchronous Adapter (the adapter used by the 2400 bps dataphone), thus limiting the areas of memory which could be interrogated. This was, however, never implemented.

In short, GDM has proven useful but could be much more so. The next section will outline several plans that have been made with this in mind.

4.2 PLANS FOR THE FUTURE

The GDM System was designed and implemented as an experimental system. It is expected that modifications and improvements will be made as the System is used by systems programming staff and others. Such changes cannot be predicted as only further experience will tell what courses of action are the most fruitful. Ideas for upgrading general system characteristics, however, are currently being investigated.

As has been mentioned in several places above, the biggest problem in monitoring any rapidly changing environment is in determining how much and how often to monitor. GDM presently does not have sufficient capabilities in either area. Experiments with simulated data have shown that optimum sampling time, from the point of view of producing readable displays, is on the order of one second. For data that changes drastically from reading to reading, somewhat slower rates should be used. This implies that a useful device would be one that samples events at a rate commensurate with their occurrence, but displays them at a rate suitable for human consumption.

One suggested method to accomplish this is to build up a buffer of monitoring information in some communications region accessible to the display processor. In the Multics environment, this can be accomplished by causing the event in question to record its own occurrence in a small reserved core area in the GE-645. This data can then be read at whatever rate the display processor interface will allow.

The disadvantage of such a system is the necessity of inserting extra statements in crucial supervisor procedures. The monitoring may indeed cause a degradation of service sufficiently great one proposed took ten per cent of the central processor time [16]. Another tack, reported in Chapter I, is to provide direct signal paths from certain computer registers to a hardware device that records them [17]. This eliminates the problem of system interference but introduces one of the inflexibility of the monitoring arrangement. It is felt that this approach can be

fruitfully pursued in special purpose monitoring devices but is not the path of a general device, as GDM is intended to be.

The computer interface is another area which can be improved. Such channels range from dataphones to direct connections. The 2400 bit per second dataphone currently in use exercises the full bandwidth of a single voice-grade telephone line. Higher speed dataphones are available, upwards of 40,000 bits per second, for use with several lines in parallel. This is considerably more versatile than the direct connection, although slower, as it can be easily transfered to another computer system. Here again, it is felt that a higher speed dataphone would aid the GDM effort. The complications introduced by the direct connection would probably prove more advantageous in other time-sharing system graphical experiments.

These final comments are intended to serve as suggestions for continuing work in the area of on-line monitoring of time-sharing systems via graphic displays. The work to date has been productive, both as an academic problem and as a useful groundwork for future projects. It is hoped that these will be forthcoming.

APPENDIX A

THE PDP-8/338: STRUCTURE AND PROGRAMMING

(For more complete information on the PDP-8, see [6]; for further information on the 338 Display, see [5].)

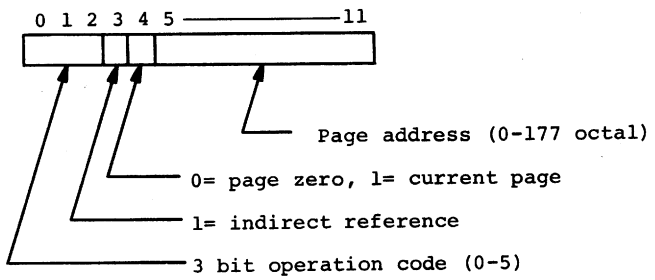
The PDP-8 is a general purpose high-speed digital computer manufactured by the Digital Equipment Corporation. The Project MAC installation (Figure 2-2) has 8192 words of 12-bit core memory, a 32,768 word fixed head disc, two DECtape magnetic tape drives for DEC fixed block length tapes, hardware multiply and divide instruction extension, and a 12-bit programmable clock, as well as interfaces to a 200 bit per second dataphone (103A) and a 2400 bit per second dataphone (201B). All devices are capable of interrupting the computer and the disc, tape, and high-speed dataphone transfer data directly to core memory (by a system of cycle stealing known as "three cycle data break"). The basic PDP-8 instruction repetoire contains eight operation codes:

1. Logical AND
2. Add to accumulator
3. Index memory location and skip if zero
4. Store accumulator
5. Jump to subroutine and deposit return address
6. Jump
7. Input/output instruction (IOT)
8. "Operate" group instructions

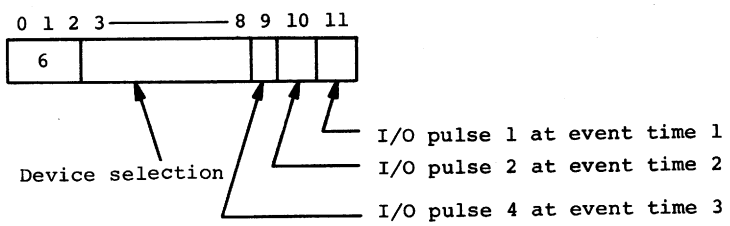
There are approximately 75 IOT's on the Project MAC PDP-8 and more can be added as needed. The "operate" group contains multiply, divide, shift, and skip on accumulator instructions. There are about forty of these.

The core memory is divided into two 4096 word core "fields." Communication between these is provided by "change data field" and "change instruction field" instructions (IOT's). Each 4096 is further divided into 128 word "pages" so that each instruction can address any location on its page or page zero (see Figure A-1 for instruction formats). Out of page references (other than to page zero) are made through indirect words. Only single level indirection is possible.

Memory Reference Instructions



IOT Instruction



"Operate" Micro-coded Instruction

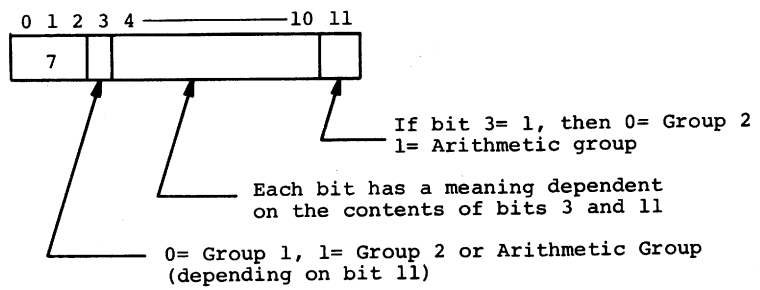


Figure A-1. PDP-8 Instruction Formats

The 338 Display is a versatile stored program display processor sharing the PDP-8 memory. Display instruction lists are stored in memory and accessed during a memory "break" cycle occurring once every three normal instruction cycles. The 338 has instructions to interrupt the PDP-8 processor and similarly, the PDP-8 has IOT instructions to start and stop the display, and load or interrogate its various status registers.

The display itself has a 75 inch square programmable area of which a 9-3/8 inch square sector is viewable at one time (1024 points in each direction). The 338 is also equipped with a light pen, a bank of twelve pushbuttons, and a "manual interrupt" button, all capable of causing PDP-8 interrupts or raising flags interpretable by display instructions.

Display instructions are executed in control state or data state with the same code taking on a different meaning depending on the state. Control instructions perform such functions as changing the scale, intensity, or sector, interrogating the pushbuttons or light pen, performing jumps or "push" jumps or "pops" and, of course, changing to data state. Data state instructions cause actual CRT beam movements in one of seven modes: point, increment, vector, vector continue, short vector, character, and graph plot. Character mode uses the VC38 Character Generator to execute an efficient dispatch to "character routines" written in increment or short vector mode.

The GDM Monitor occupies approximately 4000 locations of core memory, the PDP-8 File System occupies an additional 2500, the display character generator routines, 500, and the remaining 1500 locations are reserved for the current display template and data manipulation routines (although they seldom get that large). One half of the disc, about 16,000 words, is available for display template storage while the other half is reserved for the disc resident PDP-8 Operating System. An average display template occupies 25 tape blocks for the source file, 12 for the binary file, and 5 for the "DT" form of a standard 1465 block DECtape. Most of the display template is plotted in character mode with portions in vector and short vector modes.

This empty page was substituted for a blank page in the original document.

APPENDIX B

SEGMENT ADDRESSING IN MULTICS

(See [3] for further background and a more complete description.)

Each user process running in Multics has at its disposal a virtual memory (or address space) which is mapped into core memory by several hardware and software devices. Each address in virtual memory has two components: a "segment number" (up to 2**14) and an "offset" within that segment (up to 2**18). Segments are further divided into "pages" for the purposes of efficient core memory management but this mechanism is invisible to the user.

The "generalized address" is mapped into an actual core memory address with the help of a "page table" for each segment and a single "descriptor segment" (that is, one for each user). (See Figure B-1) A hardware register, the descriptor base register, is set to point to the user's descriptor segment. The hardware addressing mechanism then interprets the generalized address as follows: The segment number, k (see the diagram), is taken as an offset into the descriptor segment. The contents of the location pointed to is a pointer to the page table for segment k. Pages are 1024 words long in Multics, thus the high order bits of the offset, l, are interpreted as the page number, i (the offset into the segment's page table).

The core location specified by generalized address k|l is, therefore, "computed" by the formula:

$$\text{location} = C(C(C(DBR)+k)+i)+j$$

where C(x) = contents of location x

i = offset into page table = [l/1024]

j = offset within page i = l modulo 1024

In order to simulate the hardware mechanism of the GE-645, the GDM system must know the base address of the descriptor segment in question. It can then retrieve the appropriate descriptor segment word and page table word for the generalized address being converted to a core address. If the user restricts himself to requesting data from "wired-down" data bases (that is, data bases whose core locations never change after Multics initialization), then this computation need only be performed the first time a location is referenced.

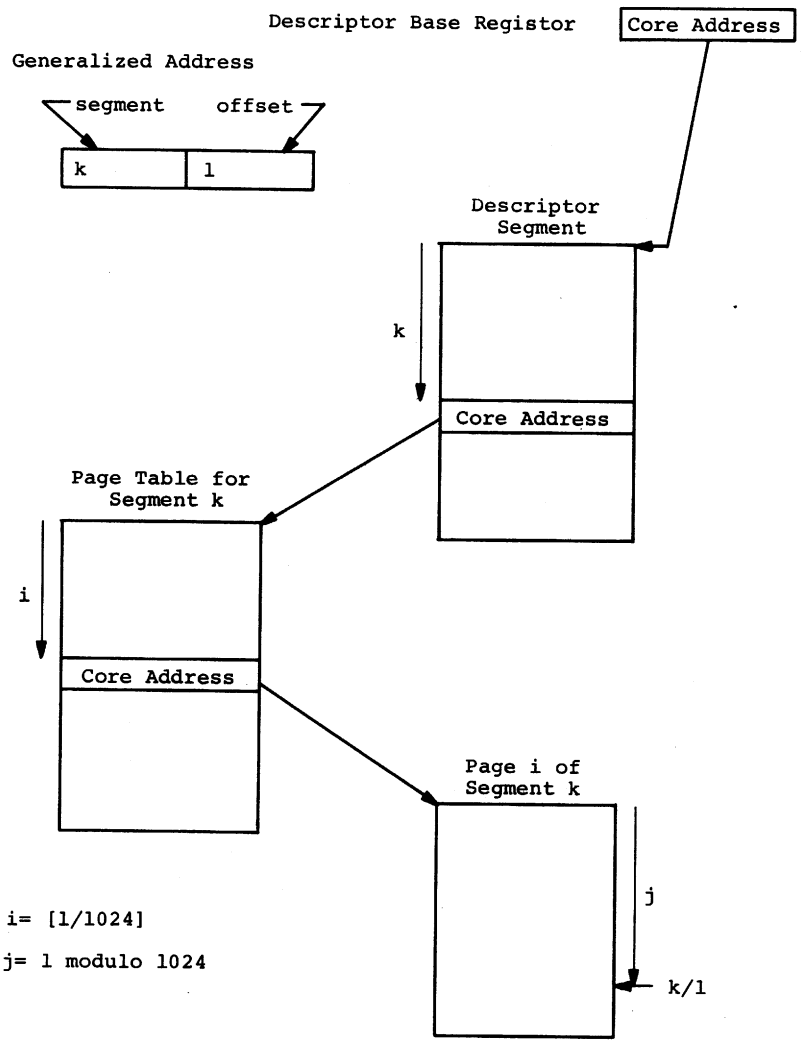


Figure B-1. Multics Two-Dimensional Addressing

APPENDIX C

USING GDM UNDER THE OS/8 OPERATING SYSTEM

The OS/8 Operating System is a tape oriented monitor system for the PDP-8 (see [12] and [18].. The GDM Monitor System and the GDM Subroutine Package are stored as a single tape file with name 'GDM S' ("S" indicates a "saved" file).

To resume the GDM System, simple type 'GDM (NL)' on the teletype. The System will be loaded from tape and started at location 200 in field 0 (GDM standard starting address). GDM will respond with 'GDM (NL)' and wait for commands to be typed. Issuing a "Q" command will cause a return to OS/8 (a jump to the bootstrap at location 7600 in core field 0 is executed).

Several other types of tape files are used by the GDM System:

x A	ASCII text file. Output of editor (see below).
x B	binary image file. Output of assembler (see below).
x M	display template file created by GDM "DT" command from file 'x B.'
x N	display snapshot file created by GDM command "SN."

The OS/8 text editor, TECO, is a versatile context editor using the display. Typing 'TECO (NL)' causes it to be resumed. A short listing of TECO commands follows. For a more complete list, see [18]. For further information on the TECO program in general, see [8].

A "$" indicates the teletype "ALT MODE" character; typing "$$" causes execution of a command string:

nRx$	open file 'x A' for reading on tape unit n.
nWx$	open file 'x A' for writing on tape unit n.
A	read next page from file opened for reading and append to buffer.
P	write current buffer on tape file opened for writing.
XR	Close current file opened for reading.
XW	close current file opened for writing.
K	clear buffer.
BJ	move pointer to beginning of buffer.
ZJ	move pointer to end of buffer
±nC	move pointer n characters forward or backward.

±nL	move pointer n lines up or down.
S...$	search for character string and move pointer.
I...$	insert character string at pointer.
±nD	delete n characters forward or backward.
±nK	delete n lines up or down.
Q	return to OS/8.

The OS/8 assembler, MACRO, is called by typing 'MACRO (NL)' on the teletype. It will respond with:

R:

and wait for a tape file name to be typed. It is only necessary to type the primary file name as MACRO always looks for text files with second name "A" (on tape unit 1 only).

MACRO is a two-pass assembler which halts at the end of pass 1. Pressing the "continue" switch on the PDP-8 will cause pass 2 to be entered and the following to be typed:

W:

The primary name of the output file should be typed. A file with second name "B" will be created on tape unit 2. After this action, MACRO will ask again for the name of the input file:

R:

This must be the same as in pass 1! At the completion of assembly, a symbol table will be typed.

The complete MACRO Language is described in reference 4, but certain features are of particular relevance to GDM users. The GDM System provides a set of macro definitions (see Appendix F for a listing) to simplify display template programming. These are stored as a separate tape file with name 'GDMMAC A' and should be used when assembling any display template without the aid of the DDL compiler. If 'TEMPL A' is the name of the display template file, the following should be the appearance of the teletype printout at the end of the assembly (user responses are underlined):

R (NL)	(OS/8 ready)
MACRO (NL)	(resume assembler)
R:GDMMAC (NL)	(read macro definitions)

```
PR:TEMPL (NL)          (read display template file)
$                      (end of pass 1)
                       (user presses "continue")
W:TEMPL (NL)           (name of display template output file)
R:GDMMAC (NL)
PR:TEMPL (NL)
$                      (end of pass 2)

(symbol table follows)
```

This "concatenation" of input files is caused by the presence of the pseudo-op "PAUSE" at the end of the macro definitions file rather than the pseudo-op "$" indicating the end of input.

Another feature of MACRO is the ability to set the core field and location of the assembled output. Display templates should have the format indicated in Figure C-1 (comments begin with a "/").

To create file 'TEMPL M,' suitable for display via GDM, assuming that binary file 'TEMPL B' has been created as outlined above, the following console session should ensue:

```
R (NL)                 (OS/8 ready)
GDM (NL)
GDM (NL)               (GDM response when loaded)
Q (NL)                 (return to OS/8)
R (NL)
LOADGO 200 TEMPL (NL)  (load TEMPL, restart GDM--see below)
GDM (NL)
DT TEMPL (NL)          (create file 'TEMPL M')

(further work with GDM)
```

When at OS/8 command level, the following commands may also be of general use (see [18] for a more complete list):

```
LOAD a b c ... (NL)
START (x) y (NL)
LOADGO (x) y a b c ... (NL)
```

The proper combination of commands will cause file 'a B' (for Binary), 'b B', 'c B', etc. to be loaded and execution to start at location y in core field x.

Any file with second name "S" (saved) may be resumed by simply typing its name. The most commonly used (in addition to MACRO and

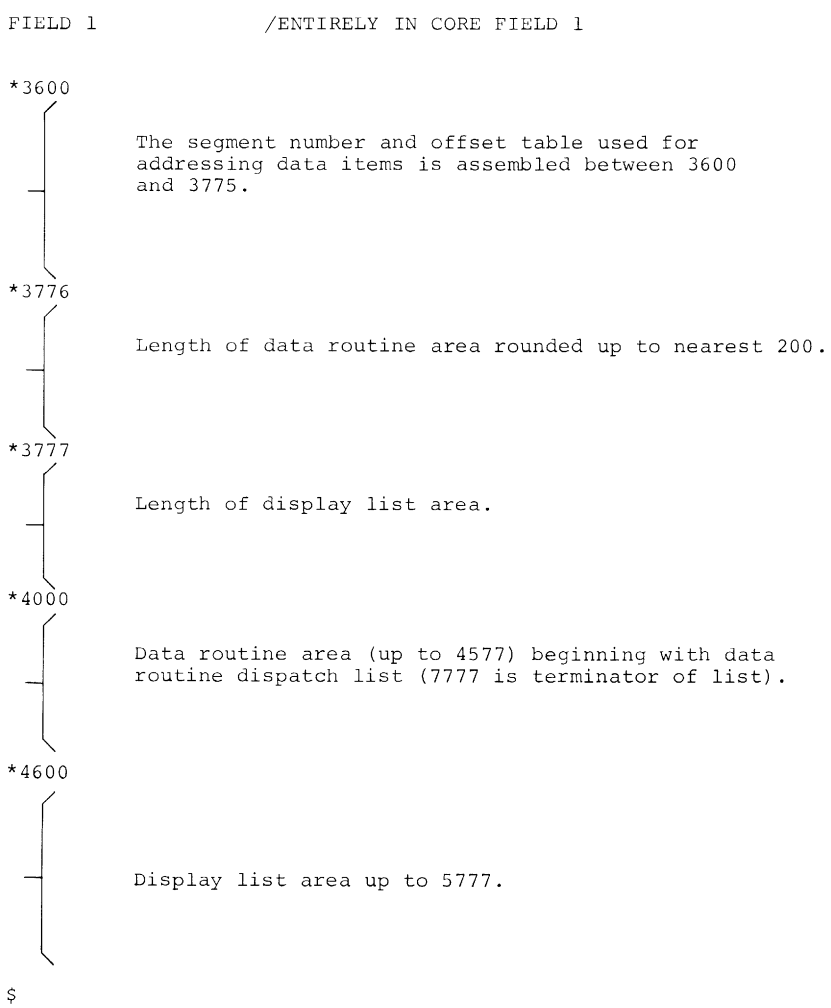


Figure C-1. Display Template Format

TECO) are: COPY (copy file from one tape to another) and L (file directory utility program).

After typing 'COPY (NL)' the program will be loaded from tape and respond with:

```
T
```

The user types:

```
a1 b1 a2 b2 a3 b3 a4 b4 ... (NL)
```

The program copies file 'a1 b1' from tape unit 1 to tape unit 2 and names it 'a2 b2' (not identical to 'a1 b1'); then 'a3 b3', etc.

Typing 'L (NL)' resumes (from tape) the file directory utility command. L accepts the following requests:

```
wL      display file directory of tape on unit w. (Tape unit
        number may precede any command.)
P       print file directory on teletype.
B       display free block list.
D a1 b1 a2 b2 ... delete the files indicated. Second name may be
        "*" indicating all files with the same first name.
R a1 b1 a2 b2 ... rename file 'a1 b1' to 'a2 b2,' etc.
Q       return to OS/8 command level.
```

This empty page was substituted for a blank page in the original document.

APPENDIX D

GDM COMMAND AND ERROR SUMMARY

All GDM commands, system or user defined, are one or two characters. Arguments that are tape file primary names may be up to six characters. The general command format is:

```
tape_unit command_name arg_1 arg_2 arg_3 ... (NL)
```

where the tape_unit (a number from 1 to 8) may optionally precede any command. Arguments are as specified for each command.

To initiate a display, simply type its name on the teletype. To return to GDM command level, depress the "manual interrupt" button on the display pushbutton box. Commands indicated with a "*" destroy the current user display image and should not be used before an "R" command:

P — prints all known command names on the teletype.

DD xx yy zz ... — sets the date display. When no arguments are typed, DD resets the interval timer.

C — removes all user defined display names from the command table and removes all user displays from the disc.

LN al bl ... — loads display file 'al M' and gives it command name 'bl', etc.*

SN a — saves the current display in snapshot file 'a N.' Issues an "R" command at completion (see below).

SH a — displays file 'a N' for photographing, etc.*

DT a — creates display file 'a M.' This command assumes that a display template binary image file has been previously loaded (see Appendix C).

SV a — creates file 'a S' containing the GDM System (see Appendix C).

R — resume the user display after "manual interrupt" caused suspension.

I1 — displays command usage information.

I2 — displays a short description of all system commands.

Q — returns to OS/8 command level.

GDM error messages are always typed on the teletype in the following format:

E:xy

where 'x' is the error code (if any), and 'y' is the number of the command line argument which caused the error (errors caused after processing of the command will output the number of the last argument).

Following is a list of error codes and their probable causes:

U Unknown command or display name.

O Overflow: command line too long, too many display templates, tried to read or write a tape file which was too big.

R Ready list overflow. This is caused when the data dispatch routine cannot keep up with the number of requests for processing. A display template redesign is indicated (sampling rates should be changed).

W Wait list overflow. There are too many requests for rescheduling of events. A display template redesign is indicated (reduce the number of events!).

N No current display. Tried to "snapshot" before initiating a display or tried to issue an "R" (resume display) after executing a command that destroys the display image.

F OS/8 File System error. Usually indicates that the file named does not exist on the tape indicated when trying to read, or that the file already exists when trying to write.

T Tape error. Try again or call field service engineer.

D Disc error. Try again or call field service engineer.

P Dataphone error. Reinitialize display. If this type of error occurs frequently, call the field service engineer.

APPENDIX E

THE GDM SUBROUTINE PACKAGE

The list of GDM Subroutines provides the following information:

1. Symbolic name and number of arguments
2. Purpose of arguments
3. Brief statement of function and usage

All subroutines are addressed through a page zero transfer table. The addition of a new subroutine, therefore, requires only the addition of its actual address to the transfer table and of its name (symbolic) and transfer table address to the GDM Macro Definitions (Appendix F).

Subroutines with only a single argument can be addressed by making the following macro call:

```
DO SUBR,ARG
```

where SUBR is the (symbolic) name of the routine and ARG is its argument. Similarly, for subroutines with two arguments, use the macro call:

```
CALL SUBR,ARG1,ARG2
```

Subroutines TIME and PBHIT (discussed in Chapter II) are called by special macros:

```
TIME
PBHIT number
```

Data Manipulation Routines:

Name: GE645 Number of arguments: 2

1. Display template address table number of segment number of datum.
2. As in 1, but for offset of datum.

This routine will initiate a request for data according to the instructions contained in the two address table entries. The data item is returned to three consecutive locations (36-bits) beginning

at "DATA" (see Appendix F).

Name: MOVE Number of arguments: 2

1. Address of data to move.
2. Address of locations to move data to.

This routine simply transfers the three words at the location specified by argument 1 to the three words starting at the location specified by argument 2. It can be used for moving data before performing additions and subtractions (see below).

Name: ADD Number of arguments: 2

1. Address of addend
2. Address of addend

Performs a 36-bit addition of the items specified. Because of the method of addition, neither argument may be in DATA. The result is left in DATA.

Name: SUB Number of arguments: 2

1. Address of subtrahend.
2. Address of minuend.

Performs a 36-bit subtraction by inverting the subtrahend and calling ADD. As in ADD, neither item may be initially in DATA and the result is left in DATA.

Name: SCALE Number of arguments: 1

1. The number of bits (in octal) to shift the datum left(+) or right(-).

This routine will shift the 36-bit number in DATA left or right the indicated number of bits (up to 43 octal). Vacated bit positions will be filled with zeros. This can be used before applying a "MASK" to get at certain bits of the datum.

Name: MASK Number of arguments: 1

1. The number of bits to be retained in the datum (masked).

The low-order bits (memory location DATA+2) of the data item are logically AND'ed with the specified mask and returned to DATA+2. The argument specifies the number of bits (from the right) to be put in the mask and, therefore, retained in the data word. This routine should be used before calling a plotting routine requiring a particular number of bits of precision.

Display Subroutines:

Name: NPLOT Number of arguments: 2

1. The number of characters to plot (up to 14 octal).
2. Address of an area in the display list to put the display codes.

Will interpret the item in DATA as (up to) 12 octal digits for display. Character codes are generated and put in the locations starting at the address specified in argument 2. If less than 12 digits are to be plotted, they are taken as the rightmost digits.

Name: CPLOT Number of arguments: 2

1. Number of characters to be plotted (up to 4).
2. Address of an area in the display list to put the display codes.

Similar to NPLOT except that the item is interpreted as four 9-bit ASCII characters. If less than 4 characters are to be plotted, they are taken as the leftmost characters. (Special characters and illegal codes appear as a "box" in the display).

Name: BPLOT Number of arguments: 1

1. Address of the display list location for plotting information.

Takes the low-order 9-bits of the item in DATA, appends control information and puts this into the location specified. This routine is for plotting either vertical or horizontal bar graphs and should be used with display macros VBAR and HBAR (see Appendix F).

Name: HPLOT Number of arguments: 2

1. 0 = intensified display; 1 = non-intensified display.
2. Display list location for code.

Plots a horizontal line, one character space in width, intensified or not as indicated in argument 1. This routine (along with VPLOT and ERASE) is used for plotting multiple state time graphs and should be used with display macros MSBEG and MSEND (see Appendix F).

Name: VPLOT Number of arguments: 2

1. Length of vertical line (number of line spaces).
2. Display list location for code.

Plots a vertical line from 1 to 7 line spaces in length, up (+) or down (-) as specified by argument 1. This routine should be used for plotting multiple state time graphs.

Name: ERASE Number of arguments: 2

1. Display code to search for.
2. Display list address to start search.

Will check consecutive display locations for the code specified by argument 1 and will replace all other codes found with zero. This routine can be used to "erase" a portion of a display list, e.g. to replot a multiple state time graph.

Miscellaneous Routines:

Name: SETCLK Number of arguments: 2

1. Rescheduling time in hundredths of a second (octal).
2. Name (label) of subroutine to reschedule.

Will put the subroutine name specified on the wait list to be called when the rescheduling time specified has elapsed. The subroutine will be called with the accumulator equal to zero and with no arguments. Return will be expected at the location immediately following the call.

Name: RD24 Number of arguments: 2

1. Number of octal digits to read (up to 10 octal).
2. Address of locations to put data.

Will read an octal number from the teletype and store it right-justified in the locations specified by the first argument (the locations specified will receive the high order digits, the following location will get the low order). This routine can be used to read data item addresses from the teletype.

Name: PBHIT Number of arguments: 1

1. The pushbutton number (6-11 only) indicated by the appropriate bit (bit 6 = pushbutton 6, etc.).

If pushbutton 0 is on, this routine will check to see if the pushbutton indicated is also on (has been hit). Returns immediately following the call if not; one location after that if there has been a hit. If pushbutton 0 is not on, returns immediately following the call.

Name: TIME Number of arguments: 0

Increments the interval timer and reschedules itself for a one second "sleep."

This empty page was substituted for a blank page in the original document.

APPENDIX F

MACRO DEFINITIONS FOR USE WITH GDM

The following listing of the GDM macro definitions conforms to the current version in all aspects except for the presence of additional comments. Definitions are of two types:

```
X = Y
```

which allows the use of symbolic name "X" when the value "Y" is required; and

```
DEFINE X A1 A2 A3 <body>
```

which causes a macro definition to be entered in the assembler macro definition table. The body of the macro will be inserted in the assembly language program wherever a statement of the form indicated below appears:

```
X A1,A2,A3
```

In the following definitions, comments begin with a slash mark (/), commands are delimited either by a semicolon (;) or a new line, and a combination of command as is indicated by an apostrophe (!). All subroutine calls (the "JMS" instruction) are made indirectly ("I") through a page zero ("Z") communications region.

```
/DEFINITIONS USED IN DATA ROUTINES
DATA=103                    /GE645 PUTS DATA IN 103,104,105
ADATA=50                    /LOCATION 50 CONTAINS A 103

DEFINE TIME                 /WILL CALL THE INTERVAL TIMER
      <0                    /STANDARD PDP-8 SUBROUTINE FORMAT
         JMS I Z 47         /CALL VIA COMMUNICATIONS REGION
         JMP I .-2>         /RETURN

DEFINE PBHIT ARG1           /CHECKS FOR A PUSHBUTTON HIT
      <CLA!CLL!CMA!RAL      /INSTRUCTION TO SET THE AC TO -2
         TAD (ARG1          /ADD ARG1 TO THE AC
```

```
        DCA .+3             /PUT IN SHIFT INSTRUCTION
        LSR                 /SET UP PUSHBUTTON MASK
        0
        JMS I Z 60          /CALL PUSHBUTTON CHECK ROUTINE
        CMA                 /CHECK RESULTS
        SZA!CLA             /AC=0 INDICATES NO HIT
                            /NO HIT RETURN LOCATION
        >                   /HIT RETURN LOCATION

/THE FOLLOWING SUBROUTINES ARE CALLED BY INCLUDING A STATEMENT OF
/THE FOLLOWING TYPE:
                            /CALL SUBR,ARG1,ARG2
GE645=51                    /LOCATIONS ARE INDIRECT WORDS
ADD=52
SUB=53
MOVE=56
SETCLK=57
SKED=57                     /ALIAS
ERASE=61
HPLOT=62
VPLOT=63
NPLOT=64
CPLOT=65
RD24=67

DEFINE CALL NAME ARG2 ARG3
      <TAD (ARG2            /PUT FIRST ARGUMENT IN AC
         JMS I Z NAME       /CALL SUBROUTINE
         ARG3>              /SECOND ARGUMENT FOLLOWS

/THE FOLLOWING SUBROUTINES ARE CALLED BY INSERTING A STATEMENT OF
/THE FOLLOWING TYPE:
                            /DO SUBR,ARG
SCALE=54
MASK=55
BPLOT=66

DEFINE DO NAME1 ARG4
      <TAD (ARG4            /PUT ARGUMENT IN AC
         JMS I Z NAME1>     /CALL SUBROUTINE
/END OF DATA ROUTINE DEFINITIONS
```

```
/DISPLAY LIST DEFINITIONS
/ALL USER PORTIONS OF THE DISPLAY LIST ARE IN DATA STATE AND
/CHARACTER MODE. TO LEAVE DATA STATE, AN "ESCAPE" CHARACTER MUST
/APPEAR IN THE DISPLAY LIST. IT MUST APPEAR BEFORE THE "TOP" MACRO
/(SEE BELOW).

/SPECIAL CHARACTER CODES
BS=10                          /BACKSPACE
HT=11                          /HORIZONTAL TAB
SP=40                          /SPACES
SP1=40
SP2=37
SP3=36
SP4=35
SP5=34
SP6=33
SP7=32
SP8=31
SP9=30
ESCAPE=177                     /LEAVE DATA STATE, ENTER COMMAND STATE

DEFINE NL <15;12>              /NEW LINE = CARRIAGE RETURN, LINE FEED
DEFINE DLSTART                 /STANDARD DISPLAY LIST START
<2011;76;0517;1107;1777;4000;1151>
/CALLS SYSTEM DISPLAY LIST SUBROUTINE, SETS BEAM TO UPPER LEFT CORNER
/OF SCREEN AND ENTERS DATA STATE IN CHARACTER MODE.

DEFINE TOP                     /JUMP TO START OF DISPLAY LIST
<2001;4600>                    /MUST BE LAST ITEM IN DISPLAY LIST

DEFINE HBAR DARG1              /PLOTS HORIZONTAL BAR
<4;1;177;6360;1121;4000
DARG1,    4000
          6340;1151;2;4>
/MAKES USE OF SPECIAL CHARACTERS TO PLOT END LINES. ARGUMENT IS LABEL
/OF LOCATION TO PUT DATA TO BE PLOTTED.

DEFINE YBAR DARG2              /PLOTS VERTICAL BAR
<177;1141;4006;4063;6360;1121
DARG2,    4000
          4000;6340;1141;4003;4066;1151>
/AS IN HBAR
```

```
DEFINE MSBEG DARG3              /PLOTS BEGINNING OF M.S.T. GRAPH
<4;1
DARG3,     0>
/ARGUMENT IS LABEL OF FIRST LOCATION OF M.S.T. GRAPH DATA AREA

DEFINE MSEND DARG4              /PLOTS END OF M.S.T. GRAPH
<DARG4, 0;0
          2;4>
/ARGUMENT IS LABEL OF END OF M.S.T. GRAPH DATA AREA
/END OF DISPLAY DEFINITIONS

PAUSE

/USER DISPLAY TEMPLATE FILE WILL BE ASSEMBLED HERE.
```

APPENDIX G

DDL STATEMENT SUMMARY

The Display Description Language contains, at present, three types of statements:

1. Format control
2. Data description
3. Display component description

The syntax for type 2 statements is given in Chapter II and is not repeated here.

The general format for display component statements is:

```
type title data rate trigger optional_args;
```

where optional_args is of the form:

```
mark_distance title
```

which specifies additional graph formatting as shown in the diagrams (Figure 2-10).

T 'text';	plot text
NL n;	plot n line spaces
SP n;	plot n spaces
HT n;	plot 10 spaces (use tab)
N n T'text' Dd Ss Xx;	plot n octal digits from the datum
A n T'text' Dd Ss Xx;	plot n ASCII characters from the datum
NA nl n2 T'text' Dd Ss Xx;	plot nl digits and n2 characters
BH T'text' Dd Ss Xx Mm 'text';	plot horizontal bar graph
BV T'text' Dd Ss Xx Mm 'text';	plot vertical bar graph
MS n T'text' Dd Ss Xx Mm 'text';	plot type "a" multiple state time graph
MV n T'text' Dd Ss Xx Mm 'text';	plot type "b" multiple state time graph

This empty page was substituted for a blank page in the original document.

APPENDIX H

A SAMPLE DISPLAY TEMPLATE: XRAY

The following is a PDP-8 Assembly Language (MACRO-8) program which, when assembled with the GDM Macro Definitions (Appendix F), is suitable for use as an input to the GDM "DT" command (display template creation). This particular program is the display used for the GDM XRAY mode of operation, as shown in Figure H-1. It illustrates a method of display design which will be automated by use of DDL and the associated compiler. It should serve as a sufficient example, however, to the GDM display template designer. It also demonstrates some of the special features of GDM (such as teletype input) that can be used if necessary to the particular display. (Macro definitions are contained in Appendix F; further information on PDP-8 programming can be found in Appendix A and reference [6].)

```
FIELD 1
*3600
GETBL,    0;0;0;0              /SPACE FOR ADDRESS TO BE FILLED IN
*3776
DTLTH-4000                     /LENGTH OF DATA ROUTINE AREA
DLLTH-4600                     /LENGTH OF DISPLAY LIST
*4000
                  /DATA ROUTINE DISPATCH TABLE
T
INIT
PB
DISPL
7777
                  /DATA ROUTINES
INIT,     0                    /INITIALIZATION
          CALL MOVE,ZERO,GETBL
          JMP I INIT
ZERO,     0;0;0;0
T,        TIME                 /MACRO TO SET INTERVAL TIMER
PB,       0                    /CHECKS FOR PUSHBUTTON HIT
          CALL SKED,62,PB      /HALF SECOND
          PBHIT 6
          JMP I PB             /NO HIT
```

```
                                                    DATE 00/00/00
                                                    TIME 0000


                    XRAY


TYPE      123456|123456        (SEGNO|OFFSET)
          123456#              (ABSOLUTE)


SEGMENT NUMBER        000000
OFFSET       000000

ABSOLUTE ADDRESS      000000



CONTENTS     000000000000  AAAA
```

Figure H-1. The XRAY Display

```
        CALL RD24,6,DATA     /READ TELETYPE
        CALL MOVE,DATA,GETBL/NPLOT NEEDS DATA IN DATA
        JMS BCHECK           /CHECK FOR BREAK CHARACTER
PBDIS,  CALL GE645,1,2       /FETCH DATA
        CALL NPLOT,12,NUM    /NUMERIC PLOT
        CALL CPLOT,4,CHAR    /CHARACTER PLOT
        JMP I PB
BCHECK, 0                    /MACHINE LANGUAGE SUBROUTINE CHECKS
                             /BREAK CHARACTER AND PLOTS
        CALL RD24,1,BREAK    /GET BREAK CHARACTER
        TAD BREAK
        TAD (-336            /SEE IF|
        SZA!CLA
        JMP BABS             /ASSUME
        TAD GETBL
        TAD (4000            /SEGMENT NUMBER ENTRY
        DCA GETBL
        CALL NPLOT,6,SEGNO   /DISPLAY SEGMENT
        CALL RD24,6,DATA     /GET OFFSET
        CALL MOVE,DATA,GETBL+2 /NPLOT NEEDS DATA IN DATA
        TAD GETBL+2
        TAD (6000            /OFFSET ENTRY
        DCA GETBL+2
        CALL NPLOT,6,OFFSET
        JMP I BCHECK
BABS,   TAD GETBL
        TAD (7700            /ABSOLUTE ENTRY
        DCA GETBL
        CALL NPLOT,6,ABS     /PLOT
        JMP I BCHECK
BREAK,  0;0                  /TEMPOARY STORAGE
DISPL,  0                    /SAMPLE AND DISPLAY DATA
        CALL SKED,144,DISPL
        TAD GETBL            /SEE IF LOCATION SPECIFIED YET
        SNA!CLA
        JMP I DISPL          /NOT YET
        TAD DISPL            /SET UP RETURN
        DCA PB
        JMP PBDIS            /DISPLAY
*4200
DTLTH,  0                    /TO MEASURE LENGTH
*4600
```

```
                    /DISPLAY LIST
          DLSTART               /MACRO
/THE FOLLOWING IS MAINLY TEXT FOR THE DISPLAY. ASSEMBLY LANGUAGE
/REQUIRES A " IN FRONT OF A CHARACTER.
          NL;NL;HT;HT;"X;"R;"A;"Y
          NL;NL;NL;HT;"T;"Y;"P;"E
          SP2;"1;"2;"3;"4;"5;"6;"|;"1;"2;"3;"4;"5;"6;SP3
          "(;"S;"E;"G;"N;"O;"|;"O;"F;"F;"S;"E;"T;");NL;HT;SP6
          "1;"2;"3;"4;"5;"6;"#;SP9;"(;"A;"B;"S;"O;"L;"U;"T;"E;")
          NL;NL
          HT;"S;"E;"G;"M;"E;"N;"T;SP;"N;"U;"M;"B;"E;"R;SP2
                    /STORAGE FOR SEGMENT NUMBER DISPLAY
SEGNO,    0;0;0;0;0;0
          NL;HT;"O;"F;"F;"S;"E;"T;SP8
                    /STORAGE FOR OFFSET DISPLAY
OFFSET,   0;0;0;0;0;0;NL;NL
          "A;"B;"S;"U;"L;"U;"T;"E;SP;"A;"D;"D;"R;"E;"S;"S;SP
                    /STORAGE FOR ABSOLUTE ADDRESS
ABS,      0;0;0;0;0;0
          NL;NL;NL
          "C;"O;"N;"T;"E;"N;"T;"S;SP3
                    /STORAGE FOR CONTENTS
NUM,      0;0;0;0;0;0;0;0;0;0;0;0
CHAR,     0;0;0;0
          TOP                   /MACRO
DLLTH,    0                     /TO MEASURE LENGTH
                    /END OF DISPLAY TEMPLATE FILE
$
```

BIBLIOGRAPHY

(ACM = Association for Computing Machinery)

[1] Coffman, E.G., and Varian, L.C., "Further Experimental Data on the Behavior of Programs in a Paging Environment," Communications of the ACM, vol. 11, no. 7, July, 1968, pp. 471-474.

[2] Corbató, F.J., and Vyssotsky, V.A., "Introduction and Overview of the Multics System," Proceedings of the Fall Joint Computer Conference, Las Vegas, Nevada, November 30, 1965, pp. 185-196.

[3] Daley, Robert C., and Dennis, Jack B. "Virtual Memory, Processes, and Sharing in MULTICS," Communications of the ACM, vol. 11, no. 5, May, 1968, pp. 306-333.

[4] Digital Equipment Corporation, PDP-8 Programming Manual, MACRO-8, Maynard, Massachusetts, 1965.

[5] Digital Equipment Corporation, Programmed Buffered Display Type 338 Programming Manual, Maynard, Massachusetts, 1967.

[6] Digital Equipment Corporation, Programmed Data Processor-8 Users' Handbook, Maynard, Massachusetts, 1966.

[7] Edwards, D.J., "GE 645 Core Memory X-Ray Program," Multics System Programmers' Manual, Sec. BE.13, Cambridge, Massachusetts, M.I.T., Project MAC, Internal Document, November 22, 1966.

[8] Edwards, D.J., TECO 6, Project MAC Memorandum MAC-M-191, Cambridge, Massachusetts, M.I.T., Oct. 29, 1964.

[9] Estrin, G., and Kleinrock, L., "Measures, Models, and Measurements for Time-Shared Computer Utilities," Proceedings of the ACM National Meeting, August, 1967, pp. 85-95.

[10] Glaser, E.L., Couleur, J.F., and Oliver, G. A., "System Design of a Computer for Time Sharing Applications," Proceedings of the Fall Joint Computer Conference, Las Vegas, Nevada, November 30, 1965, pp. 185-196.

[11] Graham, Robert M. "Protection in an Information Processing Utility," Communications of the ACM, vol. 11, no. 5, May 1968, pp. 365-369.

[12] Grochow, Jerrold M., and Skinner, Thomas P. "An Integrated Disk-Tape Operating System for the 338 Buffered Display Computer," Proceedings of the Fall Symposium of the Digital Equipment Users' Society, Anaheim, California, November 1967, pp. 129-135.

[13] Jacks, Edwin L. "A Laboratory for the Study of Graphical Man-Machine Communication," Proceedings of the Fall Joint Computer Conference, San Francisco, California, Oct., 1964, pp. 343-350.

[14] Kennedy, James R., "A System for Time-Sharing Graphic Consoles," Proceedings of the Fall Joint Computer Conference, 1966, pp. 211-222.

[15] Ossanna, J.F., Mikus, L.E., and Dunten, S.D. "Communications and Input/Output Switching in a Multiplex Computing System," Proceedings of the Fall Joint Computer Conference, Las Vegas, Nevada, November, 1965, pp. 231-240.

[16] Scherr, Allen L. An Analysis of Time-Shared Computer Systems, Project MAC Technical Report MAC-TR-18 (Thesis), Cambridge, Massachusetts, M.I.T., June, 1965.

[17] Schulman, Franklin D. "Hardware Measurement Device for IBM System/360 Time Sharing Evaluation," Proceedings of the ACM National Meeting, August, 1967, pp. 103-109.

[18] Skinner, T. P. Users' Guide to OS/8, Project MAC Memorandum MAC-M-341, Cambridge, Massachusetts, M.I.T., Dec. 14, 1966.

CS-TR Scanning Project
Document Control Form

Date : 3/27/96

Report # LCS-TR-54

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 86 (93-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:
☐ Single-sided or
☒ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☒ Unknown ☐ Other:_____

Check each if included with document:
☒ DOD Form ☒ Funding Agent Form ☒ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): XII, ALSO PAGES FOLLOWING, TITLEPAGE, 37, 45, 53, 61, 67

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description : Page Number:

(A) IMAGE MAP: (1-86) UNN TITLE PAGE, UNN'D BLANK, III-XII, 1-37, UNN BLANK, 39-45, UNN BLK, 47-53, UNN BLANK, 55-61, UNN BLK, 63-67, UNN BLK, 69-74, (87-93) SCANCONTROL, COVER, FUNDING AGENT, DOD, TRGTS (3)

(B) NOTE: SIZE OF MATERIAL SCANNED WAS 5 3/4" X 9".

Scanning Agent Signoff:

Date Received: 3/27/96 Date Scanned: 4/4/96 Date Returned: 4/11/96

Scanning Agent Signature: Michael N. Cook

UNCLASSIFIED
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)* Massachusetts Institute of Technology Project MAC	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP None

3. REPORT TITLE

The Graphic Display as an Aid in the Monitoring of
A Time-Shared Computer System

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

M.S. Thesis, Department of Electrical Engineering, September 1968

5. AUTHOR(S) *(Last name, first name, initial)*

Grochow, Jerrold M.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1968	80	18

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
Office of Naval Research, Nonr-4102(01) b. PROJECT NO. NR-048-189 c. RR 003-09-01 d.	MAC-TR-54 (THESIS) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES

This document has been approved for public release and sale;
its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Advanced Research Projects Agency 3D-200 Pentagon Washington, D.C. 20301

13. ABSTRACT

The Graphical Display Monitoring System was developed as a medium for dynamic observation of the state of a time-shared computer system. The system is integrated to create graphic displays, dynamically retrieve data from the Multics' Time-Sharing System supervisor data bases, and allow on-line viewing of this data via the graphic displays. On-line and simulated experiments were performed with various members of the Project MAC Multics staff to determine the most relevant data for dynamic monitoring, the most meaningful display formats, and the most desirable sampling rates. The particular relevance of using a graphic display as an output medium for the monitoring system is noted.

As a guide to other designers, a generalized description of the principles involved in the design of this on-line, dynamic monitoring device includes special mention of those areas of particular hardware or software system dependence. Several as yet unsolved problems relating to time-sharing system monitoring, including those of security and data base protection, are discussed.

14. KEY WORDS

Computers	Machine-aided cognition	Real-time computers
Dynamic monitoring	Multiple-access computers	Time-sharing
Graphic data display	On-line computers	Time-shared computers

DD FORM 1 NOV 65 1473 (M.I.T.)

UNCLASSIFIED
Security Classification

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 04/04/1996

M.I.T. Libraries
Document Services