```
*******      **        **     *******
********     ***      ***    *********
**    **     ****    ****    **
**    **     ** ***** **     **
********     **  ***  **     ********
*******      **   *   **      ********
**           **       **            **
**           **       **            **
**           **       **     *********
**           **       **      *******
```

NASA CR-175271

PERFORMANCE MANAGEMENT SYSTEM

ENHANCEMENT AND MAINTENANCE

Final Report for the period January, 1984 - November, 1984

Thomas G. Cleaver, Ramin Ahour, and Billy R. Johnson

Department of Electrical Engineering

The University of Louisville

Louisville, KY 40292

Prepared For:

NASA
Goddard Space Flight Center
Greenbelt, MD 20771

Under Contract NAS5-26504

November, 1984

(NASA-CR-175271) PERFORMANCE MANAGEMENT SYSTEM ENHANCEMENT AND MAINTENANCE Final Report, Jan. - Nov. 1984 (Louisville Univ.) 86 p HC A05/MF A01 CSCL 09B

N85-1756?

Unclas

G3/60 14016

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle
Performance Management System
Enhancement and Maintenance

5. Report Date
November 1984

6. Performing Organization Code

7. Author(s)
T. G. Cleaver, R. Ahour, and B. R. Johnson

8. Performing Organization Report No.

9. Performing Organization Name and Address
Electrical Engineering Department
University of Louisville
Louisville, KY 40292

10. Work Unit No.

11. Contract or Grant No.
NAS5-26504

12. Sponsoring Agency Name and Address
Mr. Straton Laios, Code 502
Goddard Space Flight Center
National Aeronautics and Space Administration
Greenbelt, MD 20771

13. Type of Report and Period Covered
Final Report
Jan.-Nov. 1984

14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The research described in this report concludes a two-year effort to develop a Performace Management System (PMS) for the NCC computers. PMS provides semi-automated monthly reports to NASA and contractor management on the status and performance of the NCC computers in the TDRSS program. Throughout 1984, PMS was tested, debugged, extended, and enhanced. Regular PMS monthly reports were produced and distributed. PMS continues to operate at the NCC under control of Bendix Corp. personnel.

17. Key Words (Selected by Author(s))
Computer Performance Evaluation
TDRSS
Univac Performance
Performance Monitoring
Performance Analysis

18. Distribution Statement

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified		

*For sale by the Clearinghouse for Federal Scientific and Technical Information, Springfield, Virginia 22151.

EXECUTIVE SUMMARY

This document contains a full description of the research conducted at the University of Louisville over the past year. The purpose of this research was to enhance and debug the current Performance Management System (PMS) for the UNIVAC 1100/82 at the NCC, and to extend PMS to operate on other computers at the NCC.

PMS provides a semi-automated system for generating monthly performace reports to NASA and contractor management. These reports summarize the activity of the operations partition of the UNIVAC 1100/82 at the NCC. This is the computer which supports TDRSS operations. The reports present tabular and graphical data showing such parameters as CPU utilization, memory utilization, and TIP thruput. These graphs and tables are accompanied by textual explanations.

Technical and administrative managers use the PMS reports to track the system's workload (for example, observing the change in workload during a shuttle launch), to anticipate system problems (for example, excessive utilizations which might threaten the system's reliability), and plan for system upgrades (for example, by observing which system resources are heavily or lightly used).

A PMS operator (currently from Bendix) must devote one to two days a month to perform the tape manipulation operations and report generation functions required by PMS. More time may be required if system anomalies require research to explain the behavior in the PMS report, or if system malfunctions require special data recovery procedures.

During this research, PMS was debugged, and there are now no known instabilities in PMS. Furthermore, PMS was enhanced making it simpler, more reliable, and faster for the PMS operator. In particular, PMS now uses fewer tapes and is menu-driven. The current version of PMS is called PMS1R16. It is available in the NCC tape library on tape number 0754.

In order for PMS to be operated by personnel other than teh developers at the University of Louisville, it was necessary to fully document its features. Therefore, the following three manuals hve been generated under this contract:

Installation, Operation, and Maintenance Manual

Program Reference Manual

File Reference Manual

Copies of these manuals have been delivered to the Technical Officer for this contract and the PMS operator at Bendix. Other copies are archived at the University of Louisville.

PMS is currently in operation at the NCC, and it should continue to provide useful information to management for years to come.

PMS2 was developed under this contract to extend PMS to operation on the backup partition of the 1100/82 and the development computer at the NCC, a UNIVAC 1100/80A. This new

PMS version is more general than the one in current operation. It monitors more system parameters and provides a more "generic" report. By reconfiguring system data files (not programs) it can be made to operate on any of the three UNIVAC 1100 configurations at the NCC. PMS2 has been tested, and has been shown to produce usable reports, but it has not been approved for implementation on the NCC computers. The University of Louisville PMS researchers recommend that steps be taken to approve and activate PMS2 for implementation on all the NCC UNIVAC 1100 series computers. It should provide the superior performance reporting capability for all these computers that is now provided only for the operations partition of the UNIVAC 1100/82.

There were no inventions developed under this contract. There were no subcontracts which contained Patent Rights or New Technology clauses.

TABLE OF CONTENTS

NOMENCLATURE

ARM	-	Arcnive Module
DTV	-	Digital Television
ECL	-	Executive Control Language
FEP	-	Front End Processor
GSFC	-	Goddard Space Flight Center
GSTDN	-	Ground Spaceflight Tracking and Data Network
MOMP01	-	Manual Operation Module
NASA	-	National Aeronautics and Space Administration
NCC	-	Network Control Center
PDL	-	Program Design Language
PMS	-	Performance Management System
SIP	-	Software Instrumentation Package
STDN	-	Space Tracking and Data Network

1. INTRODUCTION

1.1 Statement of Problem

Under the extension of the National Aeronautics and Space Administration (NASA) Contract NAS5-26504 with the University of Louisville, the maintenance and further development of a semi-automated Performance Management System (PMS) was undertaken for the Network Control Center (NCC) of the Tracking and Data Relay Satellite System (TDRSS). PMS analyzes the major resources and system workloads of the NCC computer, a Univac 1100/82. The PMS system prepares and prints monthly reports documenting the performance of NCC's UNIVAC 1100/82.[1] It was desired by NASA that PMS's manual operations (the procedures required to produce the monthly reports) should be enhanced for smoother operation of PMS. The contract also required that PMS be extended so that it could evaluate all the NCC UNIVAC 1100 series computers at the NCC. This report describes in detail the enhancements and modifications performed on PMS to meet these requirements.

1.2 Background

Performance evaluation, the determination of how well a system is able to complete its specified tasks, is essential to the successful application of virtually every technology. System designers, installation directors, data processing and corporate members, system analysts, program managers, and computer users at all levels have to cope with problems that could be solved substantially more easily and more satisfactorily with some knowledge of performance evaluation methodologies, techniques, and tools.[2]

All computer systems have to address problems that require considerable involvement in performance evaluation activities. For example, configuration design, system tuning, upgrading, scheduling, operation's management, and short and long term planning are important aspects involved in a computer system. As is the case with the NCC, the increasing complexity of modern computer systems, the increasing significance of tasks being delegated to computers and future upgrading to their system, have necessitated that computer system performance become an important consideration for their developers and those responsible for system operation.[3]

To analyze the performance of any computer system, the productivity, responsiveness, and utilization of the system must be quantified into a set of indices to be compared with performance standards. Methods used to determine these indices fall within three categories: direct measurement, simulation, and analytical techniques. Direct measurement uses monitors that sum events as they occur in the system and convert counter values to indices. Hardware or software monitors may be used. Hardware monitors use probes connected to the computer while software monitors are usually incorporated in the computer's operating system and run concurrently with application programs. Simulation; the second method of evaluation, uses system specifications, information collected by monitors, or both to model the system's hardware and its workload. A simulation program which uses these modeled parameters imitates the system by reproducing a sequence of events which corresponds to actual events taking place in the system. A third method of performance evaluation using analytical techniques represents a system by a set of mathematical equations. These equations are developed using Markovian queuing network theory or operational analysis.[4]

All three performance methods have advantages over each other. Direct measurement is the most accurate, but requires an operational system. The system's hardware and software must be in the final stages of development. Simulation requires development of complex programs to simulate the target system. These programs take time to develop and debug, but they do not require the target system to be developed. Because the system's operation is imitated, simulation can produce results which are almost as accurate as direct measurement. Analytical techniques may produce results quickly, and such techniques are easily changed in response to system changes. They are usually not as accurate as simulation or direct measurement. Performance indices for a system and modification assessments that can be produced quickly are useful for efficient system management.[5]

All three performance evaluation methods were used to evaluate the NCC computers in previous research at the University of Louisville under contract NAS5-26504. Chief responsibility for development of measurement methods, simulation, and analytical system modeling was undertaken by J. G. Darnley, G. G. Crush, and A. M. Long tively. The measurement methods gathered all the data from the NCC's computers necessary to perform the simulation and modeling experiments. The monitor used to gather this in- formation is called Software Instrumentation Package (SIP). SIP accumulates and records most of the activities of the UNIVAC's operating system and hardware devices. Because data collected by SIP require very large files, data reduction programs were developed to reduce the SIP data into useable formats and also to produce performance summary reports from the UNIVAC 1100/22 then installed in the NCC. At this point, simulation and modeling programs analyzed the data-base created by SIP/PAR to evaluate the TDRSS network.[6] As a result of the application of the above methods, it was deter- mined that the UNIVAC 1100/22 used by NCC for TDRSS activity was inadequate, but an upgrade to an 1100/82 model would suffice.[7]

An outgrowth of the above research was a proposal to use the measurement methods to monitor the operations partition of the new UNIVAC 1100/82 and report on its performance. A system needed to be developed to obtain detailed analysis of the data-base that could be obtained from SIP-collected data and PAR reduction of it. This system would be run periodically and would produce reports on the current status of the NCC's computers could be included in this new system. It was desired that the reports produced could be used by management, as well as technical specialists. The members of the team that designed the performance management system for the NCC were R. D. Shelton, T. G. Cleaver, A. M. Long, M. Shive, L. B. Drake, and A. B. Shah.[8]

The NCC's computers are a part of an extension to the existing Ground Spaceflight Tracking and Data Network (GSTDN) which is responsible for communications with low orbiting NASA spacecraft. To increase the satellite communications and coverage significantly, the TDRSS was designed because user satellites were frequently out of communication with any ground station. Another major advantage of TDRSS is that more modern technology is used than with STDN, so that a wider range of communication rates is provided. The Network Control Center controls and monitors are a part of the TDRSS system. The NCC has front end processors (FEP) which are UNIVAC Varian 77 600 minicomputers, and a custom digital TV (DTV) system which drives operator workstations. The NCC is built around a UNIVAC 1100/82 mainframe computer with dual processors that normally runs as a partitioned system, one processor for operation and the other for testing. There is also a UNIVAC 1100/80A mainframe computer used for software development.

The PMS system was developed for the NCC which is located at the Goddard Spaceflight center (GSFC).[9]

Throughout 1983, PMS was developed by the University of Louisville design team, and in October 1983, it was delivered to NASA/GSFC as an operational system. Below is a brief discussion of PMS as configured on that delivery date. More detail is available in the Final Report for 1983 (see bibliography).

The PMS system was broken down into seven modules. The first module which needed to be accessed when generating a performance report was the Manual Operations Module (MOM). This module prompted the PMS operator for information, such as the operator's name, report date, report period, and threshold values. Also, MOM set up the major units of PMS for execution.[10]

Data elements were stored intwo classed of files: Internal Data Base and the External Data Base. The Internal Data Base contained all data files required during a run of PMS. The External Data Base contained the PMS system files that were stored on magnetic tapes, including the trend files. These trend files contained daily averages for parameters such as the NCC workload, CPU utilization, memory utilization, and utilization of the busiest disk drive. The Archive Module (ARM) automatically interfaced PMS's Internal Data Base and External Base.

The Data Reduction Module (DRM) reduced SIP performance data collected during the month. Raw performance data was collected continuously by SIP and was written to data files as blocks of raw performance data every thirty minutes. A PAR program reduced the data collected from SIP, then created the Consolidated Data Base (CDB) files. The CDB files contained performance data for an entire report period. These data were chronological, and if data were missing for any time period, flags were provided.[11]

The Report Generation module produced the monthly performance report. (See the Installation, Operation, and Maintenance Manual, Appendix I for an example of a typical report.) Two tables were contained in this report: Busiest Day Summary and Daily Average Report. The Daily Average Report gave a summary of daily performance averages during an entire month. The Busiest Day Summary table reported on the busiest day during the report period. This day could be specified by the PMS operator.

There were scatter plots in the monthly report for CPU, memory, and disk utilization. These scatter plots showed how the major resources relate to each other. Trend graphs were contained that reported on the NCC's CPU, memory, and highest disk utilization during the report period. Conditional reports were automatically plotted if threshold values were exceeded. These values were manually set by the PMS operator in the Manual Operations Module. It was possible to plot up to 25 conditional reports.

A Performance Monitor Module (PMM) along with MOMP01 allowed the PMS operator to start and stop the PMS software at various stages when generating the monthly report. This was helpful when testing and debugging the PMS system. When data integrity errors occured when executing PMS, the PMM would stop execution and provide error statements for unrecoverable errors.[12]

The Maintenance Module (MAM) made it easier for the PMS operator to maintain the PMS system. This included maintenance of command files which compile and link programs. The PMS monthly performance report and the system programs could be printed on the UNIVAC's line printer by executing the appropriate command files. Also contained were PAR programs that enabled the operator to determine if a SIP cycle file could be reduced by PMS.

PMS was not a general system and would only evaluate NCC's UNIVAC 1100/82 operations partition. PMS could not reduce data from the backup partition of the UNIVAC 1100/82 or a UNIVAC 1100/80A computer. As of December, 1983, PMS would reduce Software Instrumentation Package (SIP) cycles properly and produce a satisfactory monthly performance report as described. PMS required at least four magnetic tapes: a system tape, two trend tapes (one used for the purpose of positioning the other one) and a SIP tape or tapes. Data copied on the SIP tape was collected using the Software Instrument Package, located on the UNIVAC 1100/82. The data collected required reduction of large amounts of performance data which would eventually be used in the PMS monthly performance report. The trend tapes contained reduced performance data from the previous three months. The system tape stored all programs and maintenance routines contained in the PMS system. PMS required a minimum of human input to generate a monthly report. PMS was designed modularly to help with debugging and future modifications to the system.

The modular design of PMS helped limit the size and functions of all its software routines. The modules and units were designed with minimum interface between each unit. This allowed for changes to be made to the system by changing small programs instead of large ones. Ease of future develpments to PMS was increased because of the flexibility designed into the system.

1.3 Preview of Report

The following text will describe the major modifications made to the PMS system. in the first section, the design methodologies used by developers of PMS are discussed. Then bugs found in PMS and enhancements made to PMS are discussed. Finally, the development of the new general version of PMS, which runs on all NCC 1100 series computers, is discussed.

2. MAINTENANCE OF PMS

2.1 Preliminary Activities

In accordance with the statement of work for this contract, the project team produced monthly PMS reports for the first six months of the contract. This activity had the following beneficial effects:

1. New team members gained experience in learning and using PMS.
2. High-quality monthly performance reports were prepared and distributed.
3. It provided a training period for the Bendix Corp. personnel who were to take over the operation of PMS.
4. System instabilities were detected and eliminated.

As a consequence of producing these monthly PMS reports, an important and interesting anomaly was noted in the behavior of the UNIVAC 1100/82. The CPU percent utilization would vary between two levels, 10% and 70%. This "bimodal" behavior was interesting in that it obviously represented some sort of "thrashing" or "deadly embrace" phenomenon, and it was important in that excessively high CPU utilization threatens the reliability of the computer system.

Subsequent studies indicated that the problem may have been caused by competition for resources between SIP and OSAM. In accordance with this presumption, OSAM use was cut back severly, and the PMS reports reflected a reduction in the bimodal activity. Still, the University of Louisville staff regard the source of the problem to be unproved, and we continue to recommend that definitive tests be performed.

2.2 Debugging the PMS System

PMS is a large and complex software performance system. When dealing with programs on this scale, bugs in a system are inevitable. PMS was written modularly with the application of "top down" design techniques. This eased the difficulty in finding errors in the system.

The first major bug that was corrected was found when generating the 1983 December report. PMS had difficulty in creating files needed to produce the December monthly report. Apparently, PMS did not handle new year transitions properly. The error was found in Data Reduction Module DRMPR4. A logical condition was tested improperly. When the first month of the year was calculated, it was defined as thirteen. Consequently, all subsequent months calculated were offset by one (the second month of the new year is calculated as the first month). The system was corrected and tested again, but the monthly report still could not be produced. Later, it was discovered that a trend file had a date that was out of sequence. The bad datum was corrected and PMS was executed, this time producing a correct monthly report.

Program module ARMEI3 did not close a file called "MISCELLAN" properly, generating an error when other modules read the file. MISCELLAN contains five records. After ARMEI3 finished writing the fourth record into MICELLAN file, the file was closed; thereby, altering the number of records in the file from five to four. Any program that tried to read a fifth record in the MISCELLAN file would generate an error. To correct this error, ARMEI3 was changed to write a dummy fifth record in the MISCELLAN file.

A scheme was prepared for avoiding an annoying message. When PMS had difficulty handling a SIP cycle, a program module occasionally generated a message "***DRMFB1#3 - NONFATAL ERROR IN READING SPHISTORY" to the terminal. PMS may even be in an infinite loop when this message appears on the screen. After this message is displayed several times on the monitor, the PMS system operator wonders if a SIP cycle cannot be reduced. If the SIP cycle cannot be reduced, it should be marked for skipping and PMS should be run again. To help the operator make this decision, the PMS software was corrected so that the error message is written to the monitor once, then it is written to a file called "TRACER" along with a counter, so that the number of times the error message occurs is readily apparent. If the PMS system is in an infinite loop trying to reduce a SIP cycle, the DRMFB1 error message can only be written into the TRACER file several thousand times. Once there is an overflow in the TRACER file, the PMS system will crash. The PMS operator will then know that a particular SIP cycle cannot be reduced, and will mark this cycle for skipping before running the system again.

Once modifications were made to the PMS system, they were documented. Change Request forms were filled out to reflect changes made to the system. The program's PDL was also changed to show the new logical flow of its modified program. Overall, the PMS system did not have any major errors. Changes made to PMS were subtle modifications, but necessary if PMS was to operate reliably.

2.3 Enhancing the PMS System

Previously, PMS was not especially user friendly. This made it difficult for beginning PMS operators to run the system properly. An indepth knowledge of PMS and UNIVAC ECL was required to produce satisfactory PMS reports consistently. This is especially apparent when trying to run the Report Generation module to produce the tables and graphs contained in the reports.

In order to increase PMS ease of operation, it was decided that PMS should be accessed through a menu driven program. The menu program executes all pertinent procedures of PMS to ensure PMS proper operation. By having a menu driven program of the major operations of PMS, the operator will be more aware of the types of operations that PMS can perform since these operations are listed in the menu program. An indepth knowledge of how PMS functions operate is still required of the PMS operator when problems develop, but the menu program makes it easier for the operator to be competent in routine operation of PMS. He is more aware of the different functions PMS is allowed to perform.

The menu program is a FORTRAN program (See the Program Reference Manual for a complete listing) that is automatically executed by the procedure that loads the PMS system from magnetic tape to disk files. Through the menu program, PMS may be set up for operation and executed. The menu program has 15 options. Option A deletes all the files needed by MOMP01. Normally, the files used in the execution of MOMP01 are handled properly, but if there is a problem when executing MOMP01, this option is selected. It will allow the user to start from scratch and enter a new set of parameters in MOMP01. Option B allows a user to set up the PMS system for a monthly performance report run. This involves entering the operator name, beginning and ending dates of the monthly report, SIP cycles to be used by PMS, threshold values for devices configured on NCC's UNIVAC 1100/82, and modules of PMS selected for execution. Option C executes the PMS system. The modules of PMS that will be run are determined in module MOMP01. When changes need to be made to any program in PMS, Option D will compile the edited program. Once modified programs are compiled, Option E may be selected to map (link) the entire PMS system. If it is desired to save PMS on any tape number, especially after modifications are made to PMS, Option F may be selected. To obtain a complete listing of all programs and ECL command files contained in the PMS system, Option G is selected. To print the monthly performance report generated by PMS, Option H is selected. The monthly performance report is sent to the UNIVAC's main line printer. Option I gives instructions on how to execute the PARQA. PARQA dedetermines whether a particular SIP cycle may be used by PMS. Whenever an operator desires a listing of files currently on a magnetic tape, Option J may be used. The operator only needs to input the number of the tape he wants to look at. As PMS is used, the two work files (PMS and UTILITIES) accumulate an excessive number of versions of a file element. In order to minimize the disk space useage of PMS, Option K is selected to delete all file elements, except the most recent version. Option L concentrates two files, 66 columns each, into a file of 132 columns. The interlaced files are used as documentation in the PMS monthly performance report. Option M performs a catalog that lists disk files created under a user's account number. A catalog is performed periodically so that the PMS operator can delete unwanted files to conserve disk space. Option N lists the current disk file usage of a user while logged on a UNIVAC 1100 series computer. This option is helpful to the operator in case PMS does not execute successfully. The operator can determine if all files needed by PMS were assigned. Option O exits the menu program and returns the user to the UNIVAC's operating system.

At the NCC, there was a limit for how much disk space each user is allocated. Because of the size of the PMS system, the disk space allocated for PMS was nearly full. PMS was analyzed to determine where in the system disk usage could be minimized. Previously, PMS copied SYSBALLOG files from tape to disk even if they are designated to be skipped by module MOMP01. A scheme was prepared in the ARM units to prevent loading in the SIP files to disk that will be unused in a monthly report run. This greatly reduced the disk usage of PMS.

Originally, the PMS system operated using three tapes: two trend tapes (which stored trend files) and a system tape (which stored the system files such as PMS and UTILITIES). When the system tape and trend tapes were changed periodically to save a new version of PMS files and trend files, this created difficulty for the PMS operator. It was easy for the operator to specify incorrect tape numbers that were to replace the former ones. This arises from the fact that modules which reference the system and trend files had the tape numbers hardcoded into their modules. If new trend tapes were to be used, the operator had to change the tape numbers in the programs. A new system tape required changing a tape number in an ECL command file that stores the PMS system from disk files to magnetic tape.

A further problem was that the use of three tapes was inconvenient and error-prone. The mounting and dismounting of three tapes by the UNIVAC operator took a long time, and the tape drives used were not always reliable. Changes were undertaken so that the files contained on the system and trend tapes would be stored on a single tape.

Changes were also made so that the user is now prompted for the system tape number the PMS system is to be stored on. Once all modules of PMS are executed, the system is saved on the specified tape number. The entire system is then deleted. This procedure forces the PMS operator to reload the PMS system every time he wants to generate a monthly report and reduces the chance tnat files needed by the system are rolled out. Once the PMS system is loaded from tape to disk the menu program is executed so that the the operator can immediately select which functions of PMS he wants to perform.

The enhancements discussed make PMS more reliable and faster running. Also, the system is contained on one tape (including the trend files) and is more automatic since the operator no longer concerns himself with the correct trend tape numbers needed by the ARM units.

2.4 Documentation of PMS

Once all enhancement developments were performed on PMS, three documens were produced:

PMS Installation, Operation, and Maintenance Manual

PMS Program Reference Manual

PMS File Reference Manual

Copies of these three documents have been provided to NASA and Bendix. Other copies are on file at the University of Louisville.

The Installation, Operation, and Maintenance Manual describes PMS operator procedures. Guidelines for installing, running and maintaining PMS are included. The operator is shown how to access SIP cycles on the UNIVAC 1100/82, then store them to magnetic tape(s). Once the SIP tape(s) are created, procedures describe how to determine the valid SIP cycles to be used in a PMS monthly performance report run. There is also an error recovery section that explains operator recovery options in case difficulty should arise while running the PMS system. Next, there is a brief description of the internal function of PMS. There is also an explanation of the chronological execution of the PMS system, while PMS generates a monthly performance report. The last section describes maintenance of PMS. Instructions are given that show how to compile programs, map (link) programs, minimize PMS disk usage, and save the PMS system on magnetic tape.

The Program Reference Manual gives listings of the PDL and code for all PMS programs, including supporting utility programs. This will serve as the primary reference for system modifications and enhancements.

The File Reference Manual provides descriptions of all data files used by PMS. It includes file formats, variable names, and sample listings.

3. BENCHMARK TESTING WITH SHORT SIP INTERVALS

Bendix and CSC recognized that PMS reports provide performance data in a convenient and readable form, and that PMS reports could be produced much faster (one day) than other performance reports. They were therefore interested in using PMS to aid in analyzing benchmark tests and V and V tests on the 1100/82. Both Bendix and CSC requested that the University of Louisville personnel determine if PMS could be applied to such tasks.

PMS is configured to process SIP data taken at 30-minute SIP intervals over a period of several days. Benchmark and V and V tests use one-minute SIP intervals over a few hours. It was questionable whether PMS could easily be adapted to these short SIP intervals.

A test run of PMS with one-minute intervals was attempted. PMS processed the one-minute SIP intervals without difficulty. Normal PMS procedures were used to produce a PMS monthly report. Only three to four hours of data could be processed because the file generated by SIP would become too large for PMS to handle. The report produced using the one-minute SIP interval cycle contained all the tables and graphs normally generated by PMS. The trend graphs and Daily Average Report table produced trivial results because these graphs and tables are designed to summarize data for up to one month. Since only three to four hours of SIP data were used, the cycle was much too small to obtain meaningful results. In the Busiest Day Summary table and other standard report graphs, useable data was recorded for memory, CPU, and highest disk utilization. Also, the conditional graphs that plot threshold parameters vs. time of day were successful. The resolution of the graphs was coarse because PMS summed the one-minute SIP interval raw performance data into 30-minute time slots.

It was determined that, with suitable modifications, PMS could be reconfigured to provide useful reports on benchmark tests. This remains an area of possible future research.

4. TRAINING OF A PMS OPERATOR

At the onset of the this year's research, NASA specified Bendix as the contractor who would eventually take control of the operation of PMS and the distribution of monthly PMS reports. In accordance with the contract, University of Louisville personnel undertook The training of the Bendix operator, E. Z. Block.

Throughout the first half of 1984, the Bendix operator observed and assisted in the generation of monthly PMS reports. He became familiar with PMS operation, the gathering of SIP data, and the tape handling routines.

In June of 1984, the operation of PMS was turned over to Bendix. In July of 1984, Bendix released its first PMS report, the report for the month of June.

As changes and upgrades to PMS were developed, the new releases were issued to Bendix. PMS reports continue to be generated by their PMS operator.

5. PMS2: DESIGN OF A GENERAL PMS SYSTEM

The PMS system described in the previous section was designed specifically to analyze data from the operations partition of the UNIVAC 1100/82. The contract required that PMS be extended so that it would operate on the backup partition of the 1100/82 and also on the 1100/80A, a software development machine.

5.1 Design Criteria

After extensive interviews with NASA and contractor personnel to determine the form and content of the new PMS, it was decided that PMS should, as nearly as possible, be the same for all three configurations. Differences among the three machine configurations would be handled by selecting different options and data files, but the PMS programs would remain the same for all three. This new PMS was dubbed "PMS2".

In order for PMS to run as a general performance system that could evaluate the NCC UNIVAC 1100 series computers, some of PMS's parameter specifications needed to be changed to match the environment of all three. All of the new parameters needed to be common for all the NCC UNIVAC installations. The parameters that were determined to be most useful were CPU utilization, memory utilization, average disk utilization, TIP throughout, TIP response time, demand terminals in use, demand response time, and batch jobs open. The PMS report given in Appendix C, however, includes FEP and DTV information since these parameters are valuable for the NCC's 1100/82, for which the mentioned report was produced. The parameters which are general enough to be included in the new PMS tabular output were listed earlier and the three new parameters, demand terminals in use, demand response time, and batch jobs open, can easily overwrite the DTV and FEP data tabulated in the Report Generation module of PMS in a step towards the universilization of PMS.

5.2 Implementation

PMS1, the version of PMS which runs exclusively on the UNIVAC 1100/82, was used as the foundation for PMS2. The first task was to redesign the PAR reduction unit, DRMPR4, to acquire the additional parameters required of PMS2.

Although acquisition of TIP response time and demand response time were challenging problems, they were eventually acquired and DRMPR4 successfully passed its integration test. A listing of DRMPR4 is given in Appendix A.

DRMFRU, the Fortran Reduction Unit, had to be completely rewritten for PMS2. This complex and lengthy program creates the SPHISTORY file which is a structured SIP Data History. Fields of SPHISTORY represent the different parameters collected by PMS. The created SPHISTORY file is used by the FORTRAN Data Base Unit (DRMFB1, DRMFB2), to create the Consolidated Data Base files used by the rest of

PMS. The program is extensively commented and has detailed PDL. The listing of the program is given in Appendix B.

In order to make the new desired parameters available to PMS by the PAR reduction routines, a number of files had to be entirely reformatted. These files, namely PARAMADEF, PARAFLNMS, and THRESHOLD were each modified to include, respectively, a definition of the new parameter (needed to identify it from other parameters in PARDATAFL), the name given to the file which will be used to accumulate data at various times for the parameter, and a threshold setting, which, when exceeded, will trigger the generation of a conditional report for the parameter.

An entirely new file, called SYSCONFIG, was created. This file was developed in an effort to probe some crucial characteristics of the system which could then be displayed on the cover page of the PMS report. The existence of such a file became particularly desirable due to the need to identify a system report for the 1100/80A. The four records of this file are:

TOTEM: The main memory size of the system.
NUMCPU: Number of CPUs configured for a day.
EXLEVL: Executive level for a day.
DSKCNT: Number of disks up.

Upon completion, PMS2 was tested with a subset of the data for the month of August, 1984. The resulting PMS report is given in Appenix C.

The sample report produced by PMS2 proves that it can be successfully applied to performance management for NCC computers. Consideration should be given to applying PMS2 opertionally on all the NCC UNIVAC computers.

5.3 Toward a Universal PMS System

In May of 1984, University of Louisville personnel presented the current research at a meeting of the UNIVAC users' group, USE.[13] PMS2 was described, and its extension to PMS3 was also described. PMS3 was to be a version of PMS which would be able to run on any UNIVAC 1100 series computer, not just those at the NCC.

We received twenty-six requests from systems managers of UNIVAC installations to be beta test sites for PMS3. It is clear from this that there is great demand for a software tool of this kind.

As the present contract is expiring, we have no immediate plans to develop or release PMS3, but this remains a possible avenue of future research.

6. CONCLUSIONS

PMS has been an operational system since November of 1983. Although PMS could produce monthly performance reports at that time, a beginning PMS operator without much experience using the system would have some difficulty generating a PMS report. The enhancements made to PMS simplify the user's interface. An operator can now generate PMS reports without detailed knowledge of the internal operations of PMS. Documentation on PMS is now supplied to help the operator in case difficulties arise when executing PMS. The operator now has standard procedures to follow when installing, maintaining, and operating PMS. PMS is a more automatic system because the entire system is now contained on a single tape. By having the operator prompted for the tape number of the system's tape, errors are reduced when running PMS. All known bugs in PMS have been eliminated, thereby significantly increasing the reliability of PMS.

PMS is a viable and valuable tool for performance analysis of the UNIVAC 1100/82. It should continue to provide PMS monthly reports to NASA and contractor management into the foreseeable future.

PMS2, the general PMS system, is ready to provide performance analysis for all UNIVAC 1100 series computers at the NCC.

APPENDIX A

PROGRAM LISTING

PMS2*PMS.DRMPR4

ORIGINAL PAGE IS
OF POOR QUALITY

```
[illegible]
@ UNIT INVOCATION NAME:  DRMPR4 (RAW SIP TO PAR HISTORY FILE)
@                      (PAR LANGUAGE PROGRAM)
@
@ PURPOSE:  TO CONVERT SIP PRODUCED PERFORMANCE DATA INTO FIXED FORMAT
@          HISTORY FILE
@
@ INVOCATION METHOD:  @RAM,PAR,C? SIPFIL.,PMS*PMS.DRMPR4,TPFS.
@                          DRMPR4 "NAMES"
@
@ EXTERNAL UNITS/SUBDIVISIONS OF THIS MODULE:
@ NAME     FUNCTION                          CALLED BY      CALLS      LANGUAGE
@ GETNIV   POSITION TO NEXT TIME             RSTFHIST         -          PAR
@          INTERVAL OF SIP DATA
@ SCALE    SCALES DOUBLE WORD INTO           BASEIV,ANALIV    -          PAR
@          A SINGLE WORD
@ INITIV   ALLOCATES ARRAYS AND              RSTFHIST         -          PAR
@          INITIALIZES DATA ITEMS
@ BASEIV   BASES DATA FROM A TIME            RSTFHIST       SCALE        PAR
@          INTERVAL BLOCK FOR
@          DIFFERENCING
@ ANALIV   ANALYZES (SUBTRACTS) THE          RSTFHIST     SCALE,WRITIV PAR
@          BASE DATA FROM NEXT BLOCK
@ WRITIV   OUTPUTS A TIME INTERVAL TO        ANALIV           -          PAR
@          THE HISTORY FILE
@
@ INTERMEDIATE VARIABLES:
@ NAME        DESCRIPTION
@ FELT        HISTORY ELEMENT IN OUTPUT FILE
@ INTLEN      LENGTH OF COLLECTION PERIOD IN MINUTES
@
@ FILE/ELEMENT REFERENCES:
@ FILE/ELEMENT NAME                USE               CONTENTS
@ SIPFIL.                           I        RAW SIP PERFORMANCE DATA
@ TPFS.NAMES                        C        ELEMENT OF REDUCED SIP DATA
@                                            FROM FILE SIPFIL.
@
@ DEVELOPMENT HISTORY:
@ AUTHORS
@ R.E. STURGEON/J.G. DARNLEY   REQT:  APPENDIX 3    INITIAL DRAFT 3/17/83
@                              HIPC:  DRM           SECOND DRAFT 3/31/83
@                                                   THIRD DRAFT 5/5/83
@                                CHANGED VIA CHANGE REQUEST E4-1 3/15/84
@                                CHANGED 7/06/84 FOR GENERAL PMS SYSTEM
@ UNIT FLOW
@-------------------------------------------------------------------------------
@ OPEN OUTPUT HISTORY ELEMENT TPFS.NAMES
@ ALLOCATE INTERNAL ARRAYS FOR STORING INTERMEDIATE RESULTS
@ INITIALIZE VARIABLES TO ZERO (SEE MODULE INITIV)
@ GET AN INPUT DATA BLOCK FROM SIPFIL.
@ DO UNTIL SUMMARY BLOCK OR READ ERROR OR EOF
@      BASE THIS BLOCK (SEE MODULE BASEIV)
@      GET NEXT BLOCK (SEE MODULE GETNIV)
@      IF NEXT BLOCK VALID (NOT SUMMARY OR ERROR OR EOF)
@      THEN
@          ANALYZE NEXT BLOCK AGAINST BASE DATE (SEE MODULE ANALIV)
@          OUTPUT RESULTS OF ANALYSIS TO TPFS.NAMES (SEE MODULE WRITIV)
```

```
&       END IF
& END DO
& CLOSE OUTPUT TPF...
& END
&---------------------------------------------------------------------
FUNCTSIZE:=8.0.                             & SET FUNCTION SIZE UP TO 8K
LINSIZ:=132                                 & OUTPUT RECORD SIZE IS 132 CHAR
INTLEN:=30                                  & SIZE OF OUTPUT INTERVAL IN MINS
FUNCT DRMPF4 PELT                           & MAIN FUNCTION FOR HISTORY COLLECT
     INTRTC:=INTLEN*360.0.                  & SIZE OF INTERVAL IN RTC UNITS
     I:=TIMSTP[]///432000.00                & COMPUTE SIF INTERVALS
     READ
     INTRND:=((TIMSTP[]///432.000CF)-1)/2 & INTERVAL ROUND UP FACTOR
     REWIND
     READ                                   & POSITION TO START OF FILE
     CTIME:=TIMSTP[]///432000.00            & START TIME REL TODAY
     INTVNO:=(CTIME+INTRND)/INTRTC          & CURRENT INTERVAL NUMBER
     ITINTV:=INTVNO                         & SAVE THE INITIAL INTERVAL
     SYR:=SETDAY[,,S3]+64                   & SIF COLLECTION YEAR
     SMO:=SETDAY[,,S1]                      & SIP COLLECTION MONTH
     SDAY:=SETDAY[,,S2]                     & SIP COLLECTION DAY
     SDATE:=100*((100*SYR)+SMO)+SDAY        & SIF COLLECTION YYMMDD
     INITIV
     POPEN PELT                             & OPEN OUTPUT FILE
     IMEM:=MEMSIZE*64                       & SYSTEM MEMORY SIZE IN WORDS
     INTVNO:=INTVNO-1                       & SET TO START INTERVAL NUMBER
     WHILE ETYPE<>63                        & LOOP UNTIL A SUMMARY BLOCK IS READ
          BASEIV                            & BASE THIS INTERVAL
          GETNIV                            & READ NEXT INTERVAL
          IF INTVIM>0                       & WAS SIP ON DURING LAST INTERVAL
               ANALIV                       & YES THEN ANALIZE THE INTERVAL
          ENDIF
     ENDWH
     PCLOSE
END
FUNCT GETNIV                                & READ NEXT INTERVAL DATA
     NT:=(INTVNO+2)*INTRTC                  & CALCULATE TIME OF NEXT INTERVAL
     WHILE 1                                & READ UNTIL NEXT INTERVAL FOUND
          READ
          IF ISTAT<>0                       & ANY READ ERROR?
               I:=BLOCK                     & SAVE LAST GOOD BLOCK NUMBER
               REWIND
               READ I                       & READ TO LAST GOOD BLOCK
               ETYPE:=63                    & FORCE SUMMARY BLOCK TYPE
          ENDIF
          NCTIME:=TIMSTP[]///432000.00      & THE TIME STAMP FOR THIS BLOCK
          IF NCTIME < CTIME                 & PASS MIDNIGHT
               NT:=NT///432000.000
          ENDIF
          IF ETYPE=63
               I:=NT+INTRND
          ELSE
               I:=NCTIME+INTRND
          ENDIF
          LEAVE ETYPE=63                    & LEAVE IF EITHER AT END OR PASS TIME
          LEAVE I>NT
     ENDWH
```

```
      INTVTM:=NCTIME-CTIME
      CTIME:=NCTIME                        & SET NEW CURRENT TIME
      INTVNO:=I/INTRTC -1                  & CALCULATE NEXT INTERVAL
      IF INTVTM<0                          & PASS MIDNIGHT ?
      INTVTM:=INTVTM+432000000
           SDAY:=SDAY+1                    & YES ADVANCE DAY
           IF SDAY > DAYMO(SMO)            & PASS END OF THE MONTH?
                SDAY:=SDAY-DAYMO(SMO)      & YES
                SMO:=SMO+1                 & ADVANCE MONTH
                IF SMO > 12                & PASS DEC
                     SMO:=SMO-12
                     SYR:=SYR+1            & ADVANCE YEAR
                ENDIF
           ENDIF
      ENDIF
      SDATE:=100*((100*SYR)+SMO)+SDAY      & SIF COLLECTION YYMMDD
END
FUNCT SCALE X,Y                            & SCALE A DOUBLE WORD INTO SW
      UW:=X
      LW:=Y
      LWS:=LW/01000                        & SCALE DOWN BY 512
      RND:=(Y**0777)/0400                  & ROUND UP
      IF LWS<0
           LWS:=LWS+077777777              &THE HICKLIN FIX
      ENDIF
      UWS:=UW*0100000000
      SW:=UWS+LWS+RND                      &STORE COMBINED RESULT
END
FUNCT INITIV                               & ALLOCATE ALL ARRAYS NEEDED
     ALLOCATE BP01,NCAU,BP02,NCAU,BP03,NCAU,BP04,NCAU,BP05,NCAU,;
              BP06,NCAU,BP07,NCAU,BP08,NCAU,BP09,NCAU,BP10,NCAU,;
              BP11,NCAU,BP12,NCAU,BP13,NCAU,BP14,NCAU,BP15,NCAU,;
              BP16,NCAU,BP17,NCAU,BP18,NCAU,BP19,NCAU,BP20,NCAU,;
              BP21,NCAU,BP22,NCAU,BP23,NCAU,BP24,NCAU,BP25,NCAU,;
              BP26,NCAU,BP27,NCAU,BP28,NCAU,BP29,NCAU,BP30,NCAU,;
              BP31,NCAU,BP32,NCAU,BP33,NCAU,BP34,NCAU
     ALLOCATE PC01,NCAU,PC02,NCAU,PC03,NCAU,PC04,NCAU,PC05,NCAU,;
              PC06,NCAU,PC07,NCAU,PC08,NCAU,PC09,NCAU,PC10,NCAU,;
              PC11,NCAU,PC12,NCAU,PC13,NCAU,PC14,NCAU,PC15,NCAU,;
              PC16,NCAU,PC17,NCAU,PC18,NCAU,PC19,NCAU,PC20,NCAU,;
              PC21,NCAU,PC22,NCAU,PC23,NCAU,PC24,NCAU,PC25,NCAU,;
              PC26,NCAU,PC27,NCAU,PC28,NCAU,PC29,NCAU,PC30,NCAU,;
              PC31,NCAU,PC32,NCAU,PC33,NCAU,PC34,NCAU
     ALLOCATE BIU01,NIOU,BIU02,NIOU,BIU03,NIOU,BIU04,NIOU,BIU05,NIOU,;
              BIU06,NIOU,BIU07,NIOU,BIU08,NIOU,BIU09,NIOU,BIU10,NIOU,;
              BIU11,NIOU,BIU12,NIOU,BIU13,NIOU,BIU14,NIOU,BIU15,NIOU,;
              BIU16,NIOU
     ALLOCATE IU001,NIOU,IU002,NIOU,IU003,NIOU,IU004,NIOU,IU005,NIOU,;
              IU006,NIOU,IU007,NIOU,IU008,NIOU,IU009,NIOU,IU010,NIOU,;
              IU011,NIOU,IU012,NIOU,IU013,NIOU,IU014,NIOU,IU015,NIOU,;
              IU016,NIOU
     ALLOCATE BLC01,NLC,BLC02,NLC,LC001,NLC,LC002,NLC
     ALLOCATE BUT01,NUNITS,BUT02,NUNITS,BUT03,NUNITS,BUT04,NUNITS,;
              BUT05,NUNITS,BUT06,NUNITS,BUT07,NUNITS,BUT08,NUNITS,;
              BUT09,NUNITS
     ALLOCATE UT001,NUNITS,UT002,NUNITS,UT003,NUNITS,UT004,NUNITS,;
              UT005,NUNITS,UT006,NUNITS,UT007,NUNITS,UT008,NUNITS,;
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
             UT",G,A,WRIT.
      ALLOCATE DAYMO,12,OPPTCC,2,RESULT,16,BGCS,32,FRAVPG,2,NOPROG,E
      FLAG:=0                                     . INITIALIZE FIRST PASS FLAG
      DAYMO(1):=31                                . TABLE OF DAYS IN MONTH
      IF (SYR//4)=0                               . LEAP YEAR?
           DAYMO(2):=29
      ELSE
           DAYMO(2):=28
      ENDIF
      DAYMO(3):=31
      DAYMO(4):=30
      DAYMO(5):=31
      DAYMO(6):=30
      DAYMO(7):=31
      DAYMO(8):=31
      DAYMO(9):=30
      DAYMO(10):=31
      DAYMO(11):=30
      DAYMO(12):=31
      FOR I,0,31
           BGCS(I):=0
      ENDFOR
      VC01:=0                                     @ SIP RTC TIME
      VC09:=0                                     @ TRANSACTION SCHEDULED
      VC16:=0                                     @ SYSTEM IDLE
      VC23:=0                                     @ GCS INPUT CHARACTERS
      VC24:=0                                     @ GCS OUTPUT CHARACTERS
      VC25:=0                                     @ GCS ACTIVE TIME
      VC30:=0                                     @ DOWNED MEMORY
      VC31:=0                                     @ RESIDENT EXEC MEMORY
      VC32:=0                                     @ EXEC SEGMENTS MEMORY
      VC33:=0                                     @ COMMON BANK MEMORY
      VC34:=0                                     @ TIP MEMORY
      VC35:=0                                     @ R/T MEMORY
      VC36:=0                                     @ DEMAND MEMORY
      VC37:=0                                     @ BATCH MEMORY
      VC43:=0                                      @ DEMAND RESPONSE TIME
      VC44:=0                                     @ SWAP COUNT
      FOR I,0,NCAU-1                              @ LOOP FOR ALL PROCESSORS
           PC01(I):=0                             @ EXEC 0 ACTIVE TIME
           PC02(I):=0                             @ EXEC 1 ACTIVE TIME
           PC16(I):=0                             @ REAL TIME ACTIVE
           PC17(I):=0                             @ TIP ACTIVE
           PC18(I):=0                             @ DEADLINE BATCH TIME
           PC19(I):=0                             @ BATCH TIME
           PC20(I):=0                             @ DEMAND TIME
           PC22(I):=0                             @ PROCESSOR IDLE TIME
           PC23(I):=0                             @ EXEC 3 ACTIVE TIME
           IF NCAU>1                              @ MULTI PROCESSOR SYSTEM?
                PC32(I):=0                        @ III
                PC33(I):=0                        @ EXEC T/S
                PC34(I):=0                        @ USER T/S
           ENDIF
      ENDFOR
      FOR I,0,NIOU-1                              @ LOOP FOR ALL IOU S
           IOC01(I):=0                            @ REQUEST THRU CHN 0
           IOC02(I):=0                            @ # 100 WD BLKS THRU CHN 0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
            IOC02(I):=0                      & REQUEST THRU CHN 1
            IOC03(I):=0                      & # 100 WD BLKS THRU CHN 1
            IOC04(I):=0                      & REQUEST THRU CHN 2
            IOC05(I):=0                      & # 100 WD BLKS THRU CHN 2
            IOC07(I):=0                      & REQUEST THRU CHN 3
            IOC08(I):=0                      & # 100 WD BLKS THRU CHN 3
            IOC09(I):=0                      & REQUEST THRU CHN 4
            IOC10(I):=0                      & # 100 WD BLKS THRU CHN 4
            IOC11(I):=0                      & REQUEST THRU CHN 5
            IOC12(I):=0                      & # 100 WD BLKS THRU CHN 5
            IOC13(I):=0                      & REQUEST THRU CHN 6
            IOC14(I):=0                      & # 100 WD BLKS THRU CHN 6
            IOC17(I):=0                      & REQUEST THRU CHN 7
            IOC15(I):=0                      & # 100 WD BLKS THRU CHN 7
       ENDFOR
       FOR I,0,NLC-1                         & FOR EACH LOGICAL CHANNEL
            LCOC1(I):=0                      & REQUEST QUEUED TO LC
            LCOC2(I):=0                      & CUMULATIVE QUEUED UNITS TO LC
       ENDFOR
       FOR I,0,NUNITS-1                      & FOR ALL UNITS
            UTDC1(I):=0                      & INPUT REQUEST FOR UNIT
            UTDC2(I):=0                      & OUTPUT REQUEST FOR UNIT
            UTDC3(I):=0                      & #WDS INPUT FROM UNIT
            UTDC4(I):=0                      & # 100 WD BLK OUTPUT TO UNIT
            UTDC5(I):=0                      & DATA TRANSFER TIME
            UTDC6(I):=0                      & EXISTENCE TIME
            UTDC7(I):=0                      & REQUEST QUEUED
            UTDC8(I):=0                      & CUMULATIVE QUEUE SIZE
       ENDFOR
  END
  FUNCT LASEIV
       LASTIM:=TIMSTP[]/5000               & SET BEGINNING TIME INTERVAL
       BOPNCT:=UPDTCT[]                      & # UPDATES TO OPEN COUNTERS
       BINCCT:=SBINCT[]                      & # UPDATES TO IN CORE COUNTERS
       BC09:=SBTPX[]                         & TRANSACTIONS SCHEDULED
       BC16:=0                               & ALL CPUS IDLE TIME
       FOR I,0,NCAU-1                        & FOR ALL CPUS SUM
            BC16:=BC16+LSLTM[LSLSDL,I]       & TIME IN SYSTEM IDLE LOOP
       ENDFOR
       BC23:=0                               & ZERO GCS CHARACTERS IN
       BC24:=0                               & ZERO GCS CHARACTERS OUT
       FOR I,0,31                            & SUM CHARACTER CTS FOR ALL LINES
            J:=I*2                           & NOTE ONLY SINGLE GCS PROCESSED
            BC24:=BC24+SDCTC[J]              & OUTPUT CHARACTER COUNTS
            J:=J+1                           & ADVANCE TO INPUT SLOT
            BC23:=BC23+SDCTC[J]              & INPUT CHARACTER COUNTS
       ENDFOR
       BSFIUT:=0                             & ZERO SUM OF MEMORY TIME
       FOR K,0,31                            & SUM UP ACCUMULATED TIME FOR ALL
            BSFIUT:=BSFIUT+SFIUT[K]          & SIZE INTERVALS
       ENDFOR
       FOR K,0,7                             & FOR EACH MAIN STORAGE CATEGORY
            I:=2*K                           & SAVE THE TIME WEIGHTED TOTAL
            J:=I+1                           & ALLOCATED MAIN STORAGE
            SCALE SFMTCG[I],SFMTCG[J]        & GET DOUBLE WORD ENTRY AND
            BFMTCG(K):=SW                    & AND SAVE IT
            SCALE SFAVFG[I],SFAVFG[J]
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
          DFAVPO(I):=SW
     ENDFOR
     BCNT:=UPKCPAC]
     BCW2:=0
     FOR M,0,NSTHF
        B.44:=B.44+CMASHT[M]
     ENDFOR
     B.44:=0                                & COMPUTE NUMBER SWAPS
     FOR I,0,5
          B.44:=B.44+SDSWPT[I]
     ENDFOR
     FOR I,0,NCAU-1                         & LOOP FOR ALL PROCESSORS
          DF11(I):=LSLTM[LEXEC,I]           & EXEC 2 ACTIVE TIME
          DF12(I):=LSLTM[LEXONE,I]          & EXEC 1 ACTIVE TIME
          DF16(I):=LSLTM[LRT,I]             & REAL TIME ACTIVE
          DF17(I):=LSLTM[LTIP,I]            & TIP ACTIVE
          DF18(I):=LSLTM[LDLN,I]            & DEADLINE BATCH TIME
          DF19(I):=LSLTM[LBATCH,I]          & BATCH TIME
          DF20(I):=LSLTM[LDEM,I]            & DEMAND TIME
          DF22(I):=LSLTM[LSLPOL,I]          & PROCESSOR IDLE TIME
          DF23(I):=LSLTM[LX3,I]             & EXEC 3 ACTIVE TIME
     ENDFOR
     FOR I,0,NIOU-1                         & LOOP FOR ALL IOU S
          BIU01(I):=SBCMRQ[0,I]             & REQUEST THRU CHN 0
          SCALE SBCMWD[0,I],SBCMWD[1,I]
          BIU02(I):=SW                      & # 100 WD BLKS THRU CHN 0
          BIU03(I):=SBCMRQ[1,I]             & REQUEST THRU CHN 1
          SCALE SBCMWD[2,I],SBCMWD[3,I]
          BIU04(I):=SW                      & # 100 WD BLKS THRU CHN 1
          BIU05(I):=SBCMRQ[2,I]             & REQUEST THRU CHN 2
          SCALE SBCMWD[4,I],SBCMWD[5,I]
          BIU06(I):=SW                      & # 100 WD BLKS THRU CHN 2
          BIU07(I):=SBCMRQ[3,I]             & REQUEST THRU CHN 3
          SCALE SBCMWD[6,I],SBCMWD[7,I]
          BIU08(I):=SW                      & # 100 WD BLKS THRU CHN 3
          BIU09(I):=SBCMRQ[4,I]             & REQUEST THRU CHN 4
          SCALE SBCMWD[8,I],SBCMWD[9,I]
          BIU10(I):=SW                      & # 100 WD BLKS THRU CHN 4
          BIU11(I):=SBCMRQ[5,I]             & REQUEST THRU CHN 5
          SCALE SBCMWD[10,I],SBCMWD[11,I]
          BIU12(I):=SW                      & # 100 WD BLKS THRU CHN 5
          BIU13(I):=SBCMRQ[6,I]             & REQUEST THRU CHN 6
          SCALE SBCMWD[12,I],SBCMWD[13,I]
          BIU14(I):=SW                      & # 100 WD BLKS THRU CHN 6
          BIU15(I):=SBCMRQ[7,I]             & REQUEST THRU CHN 7
          SCALE SBCMWD[14,I],SBCMWD[15,I]
          BIU16(I):=SW                      & # 100 WD BLKS THRU CHN 7
     ENDFOR
     FOR I,0,NUNITS-1                       & FOR ALL UNITS
          BUT01(I):=SBUREQ[0,I]             & INPUT REQUEST FOR UNIT
          BUT02(I):=SBUREQ[1,I]             & OUTPUT REQUEST FOR UNIT
          BUT03(I):=SBUTRN[0,I]             & WRDS INPUT FROM UNIT
          BUT04(I):=SBUTRN[1,I]             & # WRDS OUTPUT TO UNIT
          BUT05(I):=SBUTIM[0,I]             & DATA TRANSFER TIME
          BUT06(I):=SBUEXT[0,I]             & EXISTENCE TIME
          BUT07(I):=SBUQUE[0,I]             & REQUEST QUEUED
          BUT08(I):=SBUQSZ[0,I]             & CUMULATIVE QUEUE SIZE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      ENDFOR
   END
   FUNCT ANALYV
      CLRTIM:=TIMSTP[3/1000                  @ SET END TIME OF THIS INTERVAL
      V10:=INTVTM*2                          @ SET THE TIME IN THIS INTERVAL
      ACPNCT:=UPDTCT[3-BUPACT                @ # OF TIMES OPEN COUNT UPDATED
      V12:=SRTPX[3-BCUS                      @ # TRANS SCHEDULED
      V016:=0
      FOR I,0,NCAU-1                         @ FOR ALL CPUS SUM
            V016:=V016+LS.TM[LSLSDL,I]       @ TIME IN SYSTEM IDLE LOOP
      ENDFOR
      V016:=V016-B016                        @ THIS INTERVAL TIME IN SYSTEM IDLE LOOP
      V016:=V016*2
      V023:=0
      V024:=0
      V025:=0
      FOR I,0,21                             @ SUM FOR ALL GCS LINES
            J:=I*2                           @ NOTE ONLY SINGLE GCS PROCESSED
            IF SDCTST[J]=0                   @ WAS LINE ACTIVE THIS INTERVAL
                  VC25:=VC25+SLCTTT[J]-BGCS(I) @ LINE ACTIVE TIME
                  BGCS(I):=SLCTTT[J]         @ RESET TOTAL ACTIVE TIME
            ELSE
                  VC25:=VC25+INTVTM          @ LINE ACTIVE TIME/INTERVAL
                  BGCS(I):=BGCS(I)+INTVTM    @ TOTAL TIME LINE ASSIGNED
            ENDIF
            VC24:=VC24+SBCTCC[J]             @ OUTPUT CHARACTER COUNTS
            J:=J+1                           @ ADVANCE TO INPUT SLOT
            VC23:=VC23+SDCTCC[J]             @ INPUT CHARACTER COUNTS
      ENDFOR
      VC23:=VC23-BC23                        @ GCS CHARACTERS IN
      VC24:=VC24-BC24                        @ GCS CHARACTERS OUT
      VC25:=VC25*2                           @ GCS LINE ACTIVE TIME
      MUTICS:=0                              @ ZERO SUM OF MEMORY TIME
      FOR M,0,30                             @ SUM UP ACCUMULATED TIME FOR ALL
            MUTICS:=MUTICS+SFJUT[M]          @ SIZE INTERVALS
      ENDFOR
      MUTICS:=(MUTICS-BSFJLT)/01000
      FOR M,0,7                              @ FOR EACH MAIN STORAGE CATEGORY
            I:=2*M                           @ SAVE THE TIME WEIGHTED TOTAL
            J:=I+1                           @ ALLOCATED MAIN STORAGE
            SCALE SFMTCG[I],SFMTCG[J]        @ GET DOUBLE WORD ENTRY AND
            RESULT(M):=(SW-BFMTCG(M))/MUTICS @ MEMORY FOR CATEGORY
            SCALE SFAVPG[I],SFAVPG[J]
            NOPROG(M):=(100*RESULT(M))/((SW-BFAVPG(M))/MUTICS)
      ENDFOR
      V030:=RESULT(0)*64                     @ OWNED MEMORY
      V031:=RESULT(1)*64                     @ RES EXEC MEMORY
      V032:=RESULT(2)*64                     @ EXEC SEGS MEMORY
      V033:=RESULT(3)*64                     @ COMMON BANKS MEMORY
      V034:=RESULT(4)*64                     @ TIP MEMORY
      V035:=RESULT(5)*64                     @ R/T MEMORY
      V036:=RESULT(6)*64                     @ DEMAND MEMORY
      V037:=RESULT(7)*64                     @ BATCH MEMORY
      V038:=NOPROG(4)
      V039:=NOPROG(5)
      V040:=NOPROG(6)
      V041:=NOPROG(7)
```

```
B44A:=
FOR N,1,NOTHI
   B,44A:=B44A+LMRSPT[N]
ENDFOR
V443:=(BL43A-B44A)/(1*(BMRSPA[J-B43]))
V444:=0                              @ COMPUTE NUMBER SWAPS
FOR I,1,7
     V444:=V444+SDSWPT[I]
ENDFOR
V444:=V444-B444                      @ NUMBER OF SWAPS
FOR I,0,NCAU-1                       @ LOOP FOR ALL PROCESSORS
     P001(I):=(LSLTM[LXMO,I]-BP01(I))*2 @ EXEC 0 ACTIVE TIME
     P002(I):=(LSLTM[LEXONE,I]-BP02(I))*2 @ EXEC 1 ACTIVE TIME
     P016(I):=(LSLTM[LRT,I]-BP16(I))*2 @ REAL TIME ACTIVE
     P017(I):=(LSLTM[LTIF,I]-BP17(I))*2 @ TIF ACTIVE
     P018(I):=(LSLTM[LDLN,I]-BP18(I))*2 @ DEADLINE BATCH TIME
     P019(I):=(LSLTM[LBATCH,I]-BP19(I))*2 @ BATCH TIME
     P020(I):=(LSLTM[LDEM,I]-BP20(I))*2 @ DEMAND TIME
     P022(I):=(LSLTM[LSLPDL,I]-BP22(I))*2 @ PROCESSOR IDLE TIME
     P023(I):=(LSLTM[LX3,I]-BP23(I))*2    @ EXEC 3 ACTIVE TIME
ENDFOR
FOR I,0,NIOU-1                       @ LOOP FOR ALL IOU S
     IU001(I):=SBCMRQ[0,I]-BIU01(I) @ REQUEST THRU CHN 0
     SCALE SBCMWD[0,I],SBCMWD[1,I]
     IU002(I):=512*(SW-BIU02(I))/100 @ # 100 WD BLKS THRU CHN 0
     IU003(I):=SBCMRQ[1,I]-BIU03(I) @ REQUEST THRU CHN 1
     SCALE SBCMWD[2,I],SBCMWD[3,I]
     IU004(I):=512*(SW-BIU04(I))/100 @ # 512 WD BLKS THRU CHN 1
     IU005(I):=SBCMRQ[2,I]-BIU05(I) @ REQUEST THRU CHN 2
     SCALE SBCMWD[4,I],SBCMWD[5,I]
     IU006(I):=512*(SW-BIU06(I))/100 @ # 100 WD BLKS THRU CHN 2
     IU007(I):=SBCMRQ[3,I]-BIU07(I) @ REQUEST THRU CHN 3
     SCALE SBCMWD[6,I],SBCMWD[7,I]
     IU008(I):=512*(SW-BIU08(I))/100 @ # 512 WD BLKS THRU CHN 3
     IU009(I):=SBCMRQ[4,I]-BIU09(I) @ REQUEST THRU CHN 4
     SCALE SBCMWD[8,I],SBCMWD[9,I]
     IU010(I):=512*(SW-BIU10(I))/100 @ # 512 WD BLKS THRU CHN 4
     IU011(I):=SBCMRQ[5,I]-BIU11(I) @ REQUEST THRU CHN 5
     SCALE SBCMWD[10,I],SBCMWD[11,I]
     IU012(I):=512*(SW-BIU12(I))/100 @ # 100 WD BLKS THRU CHN 5
     IU013(I):=SBCMRQ[6,I]-BIU13(I) @ REQUEST THRU CHN 6
     SCALE SBCMWD[12,I],SBCMWD[13,I]
     IU014(I):=512*(SW-BIU14(I))/100 @ # 100 WD BLKS THRU CHN 6
     IU015(I):=SBCMRQ[7,I]-BIU15(I) @ REQUEST THRU CHN 7
     SCALE SBCMWD[14,I],SBCMWD[15,I]
     IU016(I):=512*(SW-BIU16(I))/100 @ # 100 WD BLKS THRU CHN 7
ENDFOR
FOR I,0,NUNITS-1                     @ FOR ALL UNITS
     UT001(I):=SBUREQ[0,I]-BUT01(I) @ INPUT REQUEST FOR UNIT
     UT002(I):=SBUREQ[1,I]-BUT02(I) @ OUTPUT REQUEST FOR UNIT
     UT003(I):=SBUTRN[0,I]-BUT03(I) @ #WDS INPUT FROM UNIT
     UT004(I):=SBUTRN[1,I]-BUT04(I) @ # 512 WD BLK OUTPUT TO UNIT
     UT005(I):=(SBUTIM[0,I]-BUT05(I))*2 @ DATA TRANSFER TIME
     UT006(I):=(SBUEXT[0,I]-BUT06(I))*2 @ EXISTENCE TIME
     UT007(I):=SBUQUE[0,I]-BUT07(I) @ REQUEST QUEUED
     UT008(I):=SBUQSZ[0,I]-BUT08(I) @ CUMULATIVE QUEUE SIZE
ENDFOR
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
        WRTIV                                         & WRITE OUT THIS INTERVAL'S DATA
    END
    FUNCT WRTIV
        IF FLAG=0                                     & CHECK IF FIRST PASS
            ACTCAU:=NCAU                              & THIS SECTION COMPUTES THE ACTUAL
            ACTIOU:=NIOU                              &   NUMBER OF CAUS AND IOUS IN THE
            FOR I,0,NCAU-1                            &   SYSTEM.
                IF PC01(I)<=0
                    ACTCAU:=ACTCAU-1
                ENDIF
            ENDFOR
            FOR I,0,NIOU-1
                J:=IU001(I)+IU003(I)+IU005(I)+IU007(I)+IU009(I)+;
                    IU011(I)+IU013(I)+IU015(I)
                IF J<=0                               & IF THIS UNIT IS INACTIVE
                    ACTIOU:=ACTIOU-1                  &   SUBTRACT ONE FROM THE COUNT
                ENDIF
            ENDFOR
            P LOCAL,SDATE[!D6],INTVNO[!D4],"01000000",WEEKDAY[!F6],INTLEN[!D4],;
                 IMEF[!D9],ACTCAU[!D3],ACTIOU[!D3],NCU[!D3],NOU[!D3],;
                 NUNITS[!D3],EXECVERS[:2!F]
            FLAG:=1
        ENDIF
        OID:=2000001
        P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],LASTIM[!D11],CURTIM[!D11]
        OID:=OID+1
        P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],V001[!D11],V009[!D11],V016[!D11],;
          V022[!D11],V024[!D11],V025[!D11],V030[!D11],V031[!D11],V032[!D11]
        OID:=OID+1
        P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],V033[!D11],V034[!D11],;
          V035[!D11],V036[!D11],V037[!D11],V044[!D11]
        OID:=OID+1
        P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],V038[!D11],V039[!D11],;
        V040[!D11],V041[!D11],V042[!D11]
        FOR I,0,NCAU-1
          IF PC01(I)>0                                & SKIP INACTIVE CAU
             OID:=3000001+(I*1000)
             P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],PC01[I!D11],PC02[I!D11],;
                  PC16[I!D11],PC17[I!D11],PC18[I!D11],PC19[I!D11],PC20[I!D11],;
                   PC22[I!D11],PC23[I!D11]
          ENDIF
        ENDFOR
        FOR I,0,NIOU-1                                & LOOP FOR EACH IOU
          J:=IU001(I)+IU003(I)+IU005(I)+IU007(I)+IU009(I)+;
              IU011(I)+IU013(I)+IU015(I)
          IF J>0                                      & SKIP INACTIVE IOU
             OID:=4001001+(I*1000)
             P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],IU001[I!D11],IU002[I!D11],;
                  IU003[I!D11],IU004[I!D11],IU005[I!D11],IU006[I!D11],;
                  IU007[I!D11],IU008[I!D11],IU009[I!D11]
             OID:=OID+1
             P LOCAL,SDATE[!D6],INTVNO[!D4],OID[!D8],IU010[I!D11],IU011[I!D11],;
                  IU012[I!D11],IU013[I!D11],IU014[I!D11],IU015[I!D11],;
                  IU016[I!D11]
          ENDIF
        ENDFOR
        FOR I,0,NUNITS-1                              & LOOP FOR EACH DEVICE
```

```
                UTL:=5.TDDL*(I+1)
                % LOCAL,SD/TE[I[0:3],INTVAG[I[0:3],CI[L[0:3],UTLC1[I[0:11],UTCC2[I[0:11],;
                    UT..C3[I[0:11],UTCC4[I[0:11],UTCC5[I[0:11],UTCC6[I[0:11],;
                    UTCC7[I[0:11],UTCC8[I[0:11]
            ENDFOR
        END
        FUNCTSIZE:=0                                    % SET FUNCTION SIZE TO ZERO

SPRTIS FIS.DYMFRG
```

APPENDIX B

PROGRAM LISTING

PMS2*PMS.DRMFRU

```
PMS*RPS*ELS(1).DRMFRU

      C-----------------------------------------------------------------
      C UNIT INVOCATION NAME:   DRMFRU (DATA REDUCTION MODULE - FORTRAN
      C  REDUCTION UNIT)
      C
      C PURPOSE:   TO CONVERT PAR REDUCED DATA TO FILES OF PERFORMANCE DATA
      C  FOR USE BY THE REMAINDER OF PMS.
      C
      C INVOCATION METHOD:   @XQT PMS.DRMFRU
      C
      C   FILE/RECORD REFERENCES:
      C    FILE NAME        USE                 CONTENTS
      C    PMS*PARAMADEF.    I          DEFINITIONS OF PARAMETERS AND DEVICES
      C                                 USED
      C    PMS*OPERINPUT.    I          PMS OPERATOR INPUTS
      C    PMS*IOCONFIGU.    I          I/O CONFIGURATIONS
      C    PMS*PARDATAFL.    I          PERFORMANCE DATA REDUCED BY PAR
      C    PMS*SPHISTORY.    O          PERFORMANCE DATA OUTPUT FILE
      C    PMS*TRACER.       O          PROGRAM EXECUTION MESSAGES
      C    PMS*SYSCONFIG.    O          SYSTEM CONFIGURATION DATA
      C
      C INTERMEDIATE VARIABLES:
      C   NAME          TYPE                DESCRIPTION
      C  CNT           INT            ROW COUNTER FOR ARRAYS
      C  NUMCDE        INT            NUMBER OF CDE FILES IN PARAFLNMS.
      C  PERFUJ        REAL/COMMON    PER MULTIPLIER FACTOR
      C  FILCDB        INT            NUMBER OF CDB FILES IN PARAMADEF.
      C  DEVMNE        CHAR           DEVICE MNEMONIC FROM PARAMADEF.
      C  UNIT          CHAR           DEVICE UNIT ID FROM PARAMADEF.
      C  MODEL         CHAR           MODEL ID FROM PARAMADEF.
      C  DATTYP        CHAR           DATA TYPE FROM PARAMADEF.
      C  PARMS         INT/COMMON     ONE DIMENSIONAL ARRAY OF 15 ELEMENTS
      C                               CONTAINING CDE FILE NUMBERS FOR PARAMETERS
      C  DEVDEF        CHAR/COMMON    TWO DIMENSIONAL ARRAY CONTAINING DEVICE
      C                               MNEMONIC, MODEL NO., UNIT NO. AND DATA TYPE
      C                               FOR ALL DEVICES IN PARAMADEF.
      C  DEVDAT        REAL/COMMON    TWO DIMENSIONAL DEVICE DATA ARRAY
      C  NUMDEV        INT/COMMON     NUMBER OF DEVICED SPECIFIED IN PARAMADEF.
      C  DAYCNT        INT            NUMBER OF CYCLES TO BE REDUCED IN ONE
      C                               INVOCATION OF DRMFRU, RECORD 5 OF MISCELLAN.
      C  REPDAT        CHAR           REPORT DATE FROM OPERINPUT.
      C  INDATE        REAL           DATES READ IN FROM OPERINPUT.
      C  STDATE        INT            START DATE OF REPORT
      C  EDDATE        INT            END DATE OF REPORT
      C  DAYLOP        INT            MAIN LOOP COUNTER FOR DAYS OF SIF TO BE
      C                               REDUCED
      C  IOCTYP        INT/COMMON     IOCONFIGU. RECORD TYPE
      C  SAVDAT        INT/COMMON     SAVED DATE
      C  PARTYP        INT/COMMON     RECORD TYPE FROM PARDATAFL.
      C  BLKDAT        INT/COMMON     SIF DATA BLOCK DATE
      C  INTVAL        INT/COMMON     INTERVAL NUMBER FROM PARD/TAFL.
      C  SUBTYP        INT/COMMON     RECORD SUBTYPE FROM PARDATAFL.
      C  NEXTDY        CHAR/COMMON    NEXT DAY FOUND INDICATOR
      C  ELPTIM        REAL/COMMON    ELAPSED TIME READ FROM PARDATAFL.
      C  PROCNM        INT/COMMON     ONE DIMENSIONAL ARRAY WITH 4 ELEMENTS
      C                               SPECIFYING PROCESSOR NUMBER IN USE
```

ORIGINAL PAGE IS OF POOR QUALITY

```
C   TOTMEM        INT/COMMON      MAIN MEMORY SIZE
C   NUMCPU        INT/COMMON      NUMBER OF CPUS CONFIGURED FOR A DAY
C   EXLEVL       CHAR/COMMON     EXECUTIVE LEVEL FOR A DAY
C   DSKCNT        INT/COMMON      NUMBER OF DISKS UP
C   TIMEON        INT/COMMON      SIF DATA BLOCK TURN-ON TIME
C   TIMEOF        INT/COMMON      SIF DATA BLOCK TURN-OFF TIME
C   TIFCNT        INT/COMMON      TIF TRANSACTION COUNT
C   SYSIDL        REAL/COMMON     SYSTEM IDLE TIME
C   REXMEM        INT/COMMON      RESIDENT EXECUTIVE MEMORY SIZE
C   EXCMEM        INT/COMMON      EXECUTIVE SEGMENT MEMORY SIZE
C   COMMEM        INT/COMMON      COMMON BANK MEMORY SIZE
C   TIFMEM        INT/COMMON      TIF MEMORY SIZE
C   RELMEM        INT/COMMON      REAL TIME MEMORY SIZE
C   DEMMEM        INT/COMMON      DEMAND MEMORY SIZE
C   BATMEM        INT/COMMON      BATCH MEMORY SIZE
C   SWPCNT        INT/COMMON      SWAP COUNT
C   TIFPRG        REAL/COMMON     TIF PROGRAMS IN MEMORY
C   RTMPRG        REAL/COMMON     REAL-TIME PROGRAMS IN MEMORY
C   DMDPRG        REAL/COMMON     DEMAND PROGRAMS IN MEMORY
C   BATPRG        REAL/COMMON     BATCH PROGRAMS IN MEMORY
C   BLDATE        INT/COMMON      SIF DATA BLOCK DATE
C   IDLE          REAL/COMMON     TOTAL PROCESSOR IDLE TIME
C   PRCIDL        REAL/COMMON     PROCESSOR IDLE TIME
C   HSTARY        REAL/COMMON     ONE DIMENSIONAL ARRAY OF 102
C                                 ELEMENTS HOLDING DATA FOR SPHISTORY.
C                                 CREATION
C   ENDFIL        CHAR/COMMON     END OF PARDATAFL. INDICATOR
C   CPUCNT        INT/COMMON      COUNT OF ACTIVE CPUS
C   CDBMAX        INT/COMMON      HIGHEST PARAMADEF. CDB FILE NO.
C   DMDRST        CHAR/COMMON     DEMAND RESPONSE TIME
C
C DEVELOPMENT HISTORY
C    AUTHOR                 REFERENCES               DESCRIPTION
C  R. AHOUR                                     INITIAL DRAFT   5/17/84
C
C UNIT FLOW
C-----------------------------------------------------------------------
C ***PROCESS 1*** READ PARAMADEF FILE AND STORE DATA IN PARMS UNTIL
C  EOF IS REACHED
C
C RESET ROW COUNTER FOR ARRAYS, CNT, TO ZERO
C RESET CDBMAX TO ZERO
C INITIALIZE ALL ELEMENTS OF PARMS ARRAY TO ZEROS
C INITIALIZE ALL ELEMENTS OF DEVDEF ARRAY TO BLANKS, " "
C INITIALIZE ALL ELEMENTS OF DEVDAT ARRAY TO ZEROS
C OPEN PARAMADEF.
C READ TWO FIELDS OF FIRST RECORD INTO NUMCDB, FLPFUJ
C  ((NOTE: ALL OF THE DATA TYPES BELOW MAY NOT BE USED IN ANY
C   PARTICULAR CONFIGURATION OF PARAMADEF.))
C DO UNTIL EOF OF PARAMADEF
C    READ FIVE FIELDS OF PRESENT RECORD OF PARAMADEF INTO FILCDB,
C     DEVNME, UNIT, MODEL, DATTYP
C     ((NOW TEST DATA TYPE FOR PARAMETER WITH FILCDB FILE NO.))
C    IF DATTYP IS "TIPWL" ((TIP WORK LOAD, AVG JOBS IN MEMORY))
C    THEN PARMS(1) = FILCDB
C    ENDIF
C    IF DATTYP IS "DMDWL" ((DEMAND WORK LOAD, AVG JOBS IN MEMORY))
```

```
C      THEN PARMS(2) = FILCDE
C      ENDIF
C      IF DATTYP IS "BATWL" ((BATCH WORK LOAD, AVG JOBS IN MEMORY))
C      THEN PARMS(3) = FILCDE
C      ENDIF
C      IF DATTYP IS "REALWL" ((REAL TIME WORK LOAD, AVG JOBS IN MEMORY))
C      THEN PARMS(4) = FILCDE
C      ENDIF
C      IF DATTYP IS "TIPRT" ((TIP RESPONSE TIME))
C      THEN PARMS(5) = FILCDE
C      ENDIF
C      IF DATTYP IS "DMDRT" ((DEMAND RESPONSE TIME))
C      THEN PARMS(6) = FILCDE
C      ENDIF
C      IF DATTYP IS "CPUUL" ((CPU UTILIZATION))
C      THEN PARMS(7) = FILCDE
C      ENDIF
C      IF DATTYP IS "MEMUL" ((MEMORY UTILIZATION))
C      THEN PARMS(8) = FILCDE
C      ENDIF
C      IF DATTYP IS "SWPRT" ((SWAP RATE))
C      THEN PARMS(9) = FILCDE
C      ENDIF
C      IF DATTYP IS "CIOUL" ((I/O CHANNEL UTILIZATION))
C      THEN PARMS(10) = FILCDE
C      ENDIF
C       ((CHECK TO SEE IF DATA TYPE REPRESENTED A DEVICE.  IF SO,
C        THEN PLACE READ RECORDS INTO THE DEVICE ARRAY))
C      IF DATTYP WAS DISK UTILIZATION, "DSKUL", OR FEP WORK LOAD,
C       "FEPWL", OR DTV WORK LOAD, "DTVWL", OR OTHER DEVICE WORK
C       LOADS, "DEVWL"
C      THEN
C         INCREMENT CNT BY ONE
C         SET DEVDEF(CNT,1) TO DEVNME
C         SET DEVDEF(CNT,2) TO MODEL
C         SET DEVDEF(CNT,3) TO UNIT
C         SET DEVDEF(CNT,4) TO DATTYP
C         SET DEVDAT(CNT,5) TO FILCDE
C      ENDIF
C      IF CDEFILE NO. IS GT MAXIMUM FILE NUMBER THUS FAR
C      THEN SET CDEFILE NO. AS THE LARGEST FILE NUMBER READ FROM PARAMADEF
C      ENDIF
C ENDDO
C SET NUMBER OF DEVICES, NUMDEV, TO CNT
C CLOSE PARAMADEF.
C ***END PROCESS 1***
C
C
C
C ***PROCESS 2A***  READ RECORDS OF MISCELLAN. FILE
C OPEN MISCELLAN.
C  ((RECORD 3 OF THIS FILE CONTAINS THE NUMBER OF DAYS OF DATA))
C READ RECORD 3 OF MISCELLAN. INTO DAYCNT
C CLOSE MISCELLAN.
C IF DAYCNT EQ 0
C THEN
C    WRITE TO TERMINAL "DRPFRU FINISHED - NO DATA REDUCED THIS LOOP"
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      READ (IN1,10) NUMCDE,FEPFUJ
      READ (IN1,20,ERR=920,END=395) FILCDE,DEVMNE,UNIT,MODEL,DATTYP
      TEMP=TEMP+1
      IF (DATTYP.EQ.'TIPWL') PARMS(1) = FILCDE
      IF (DATTYP.EQ.'DYCWL') PARMS(2) = FILCDE
      IF (DATTYP.EQ.'WATWL') PARMS(3) = FILCDE
      IF (DATTYP.EQ.'RELWL') PARMS(4) = FILCDE
      IF (DATTYP.EQ.'TIFRT') PARMS(5) = FILCDE
      IF (DATTYP.EQ.'DYCRT') PARMS(6) = FILCDE
      IF (DATTYP.EQ.'CPLLL') PARMS(7) = FILCDE
      IF (DATTYP.EQ.'HEMLL') PARMS(8) = FILCDE
      IF (DATTYP.EQ.'SWFRT') PARMS(9) = FILCDE
      IF (DATTYP.EQ.'DICLL') PARMS(10) = FILCDE
C
      IF ((DATTYP.EQ.'DSKWL').OR.(DATTYP.EQ.'FEPWL').OR.(
     &  DATTYP.EQ.'CTVWL').OR.(DATTYP.EQ.'DEVWL')) THEN
         CNT = CNT + 1
         DEVDEF(CNT,1) = DEVMNE
         DEVDEF(CNT,2) = MODEL
         DEVDEF(CNT,3) = UNIT
         DEVDEF(CNT,4) = DATTYP
         DEVDAT(CNT,5) = FILCDE
      ENDIF
      IF (FILCDE.EQ.29) THEN
         WRITE(10,*)'PDP2 HAS SPECIFICATION::'
         WRITE(10,*)DEVDAT(CNT,1),DEVDAT(CNT,2),DEVDAT(CNT,3),
     &    DEVDAT(CNT,4),'.'
      ENDIF
      IF (FILCDE.GT.CDEMAX) CDEMAX = FILCDE
      IF (TEMP.NE.NUMCDE) GO TO 30
 40   NUMDEV = CNT
      WRITE(10,*)'TOTAL OF ',NUMDEV,' DEVICES READ FROM PARAMADEF.'
      CLOSE (IN1,ERR=925)
C   ***END PROCESS 1***
C
C
C   ***PROCESS 2A***  READ RECORDS OF MISCELLAN. FILE
      OPEN (IN1,ERR=930,FILE='MISCELLAN.')
      READ (IN1,50) DAYCNT
      CLOSE (IN1,ERR=935)
      WRITE(10,*)'NUMBER OF CYCLES TO BE REDUCED THIS LOOP: ',DAYCNT
      IF (DAYCNT.EQ.0) GO TO 205
C   ***END PROCESS 2A***
C
C
C   ***PROCESS 2B***  READ DATES FROM OPERINPUT. FILE
      OPEN (IN1,ERR=940,FILE='OPERINPUT.')
      READ (IN1,60) REPDAT
      READ (IN1,70) INDATE
      STDATE = IFIX(INDATE*100 + 0.5)
      READ (IN1,70) INDATE
      EDDATE = IFIX(INDATE*100 + 0.5)
      CLOSE (IN1,ERR=945)
      WRITE(10,*)'REPDATE: ',REPDAT,', START DATE: ',STDATE
      WRITE(10,*)'END DATE: ',EDDATE
C   ***END PROCESS 2B***
C
```

```
C
C   ***PROCESS 3***  REDUCE DATA FROM IOCONFIGU. AND PARDATAFL.
      OPEN (IN1,ERR=990,FILE='PARDATAFL.')
      OPEN (IN2,ERR=990,FILE='IOCONFIGU.')
      OPEN (IN3,ERR=990,FILE='CPHISTORY.')
      WRITE(10,*)'DRMFRU: OPENED FILES, READY TO REDUCE ',DAYCNT
      WRITE(10,*)'CYCLES OF DATA.'
 90   READ (IN2,50,ERR=985,END=987) IOCTYP
      IF (IOCTYP.NE.10) GO TO 90
      DO 140 DAYLOP=1,DAYCNT
 100     CALL DRMFR1
         WRITE(10,*)'IOCONFIGU. FILE PROCESSED SUCCESSFULLY IN DRMFR1'
         WRITE(10,*)'FOR CLOCK ',DAYLOP,'.'
         SAVDAT = 0
         IF (DAYLOP.EQ.1) THEN
 110        READ (IN1,120,ERR=970,END=977) BLKDAT,INTVAL,PARTYP,SUBTYP
            IF (PARTYP.NE.01) GO TO 110
         ENDIF
         SAVDAT = BLKDAT
         NEXTDY ='N'
         ENDFIL ='N'
 130     CALL DRMFR2
         IF ((ELPTIM.EQ.0).OR.(BLKDAT.LT.STDATE).OR.(BLKDAT.GT.EDDATE)
     &    .OR.(BLKDAT.GT.(SAVDAT+1))) THEN
            WRITE (6,*) 'DRMFRU - PAR FILE NOT IN PROPER SEQUENCE OR
     &       ELAPSED TIME OUT OF BOUNDS'
            GO TO 99999
         ELSE
            CALL DRMFR3
            CALL DRMFR4
         ENDIF
         IF ((NEXTDY.EQ.'N').AND.(ENDFIL.EQ.'N')) GO TO 130
         IF ((ENDFIL.EQ.'Y').AND.(DAYLOP.NE.DAYCNT)) GO TO 993
 140  CONTINUE
      OPEN (14,ERR=995,FILE='SYSCONFIG.')
      WRITE(14,125) TOTMEM
      WRITE(14,126) NUMCPU
      WRITE(14,127) EXLEVL
      WRITE(14,128) DSKCNT
      WRITE(10,*)'DRMFRU WRITE TO SYSCONFIG:'
      WRITE(10,*)'TOTMEM: ',TOTMEM,', NUMCPU: ',NUMCPU
      WRITE(10,*)'EXLEVEL: ',EXLEVL,', DSKCNT: ',DSKCNT,'.'
      CLOSE (14,ERR=997)
      CLOSE (IN1,ERR=980)
      CLOSE (IN2,ERR=980)
      CLOSE (IN3,ERR=990)
      WRITE(10,*)'DRMFRU: CLOSED OLD FILES'
C   ***END PROCESS 3***
C
      GO TO 999
 900  FORMAT ('***DRMFRU ERROR A', A3)
C
 995  WRITE(6,900) '995'
      WRITE(6,*)'UNEXPECTED TERMINATION OF PARAMADEF.'
      GO TO 99999
 910  WRITE (6,900) '910'
      GO TO 99999
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
  915    WRITE (6,900) '915'
         GO TO 99999
  920    WRITE (6,900) '920'
         WRITE (6,*) 'FAILURE TO READ PARAMADEF. FILE'
         GO TO 99999
  925    WRITE (6,900) '925'
         GO TO 99999
  930    WRITE (6,900) '930'
         GO TO 99999
  935    WRITE (6,900) '935'
         GO TO 99999
  940    WRITE (6,900) '940'
         GO TO 99999
  945    WRITE (6,900) '945'
         GO TO 99999
  950    WRITE (6,900) '950'
         GO TO 99999
  955    WRITE (6,900) '955'
         GO TO 99999
  960    WRITE (6,900) '960'
         GO TO 99999
  965    WRITE (6,900) '965'
         WRITE (6,*) 'FAILURE TO READ IOCONFIGL. FILE'
         GO TO 99999
  967    WRITE (6,900) '967'
         WRITE (6,*) 'UNEXPECTED TERMINATION OF IOCONFIGL. FILE'
         GO TO 99999
  970    WRITE (6,900) '970'
         WRITE (6,*) 'FAILURE TO READ PARDATAFL. FILE'
         GO TO 99999
  973    WRITE (6,900) '973'
         WRITE (6,*) 'PARDATAFL. DOES NOT EXIST.'
         GO TO 99999
  975    WRITE (6,900) '975'
         GO TO 99999
  980    WRITE (6,900) '980'
         GO TO 99999
  985    WRITE (6,900) '985'
         GO TO 99999
  990    WRITE (6,900) '990'
         GO TO 99999
  993    WRITE(6,900) '993'
         WRITE(6,*)'DRMFRU: UNEXPECTED TERMINATION OF PARDATAFL.'
         GO TO 99999
  995    WRITE (6,900) '995'
         GO TO 99999
  997    WRITE (6,900) '997'
         GO TO 99999
  998    PRINT *,'DRMFRU FINISHED - NO DATA REDUCED THIS LOOP'
  999    PRINT *,'**********SUCCESSFUL EXECUTION OF DRMFRU**************'
         WRITE(*,*)'SUCCESSFUL EXECUTION OF DRMFRU'
         CLOSE (IN,ERR=973)
         GO TO 9999
99999    CALL FSETC (448)
         WRITE(*,*)'DRMFRU EXECUTION HALTED'
 9999    STOP
         END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C
C
C
C-----------------------------------------------------------------------------
C  SUBROUTINE DRFFR1  (DATA REDUCTION MODULE - FORTRAN REDUCTION 1)
C
C  PURPOSE:    TO PROCESS IOCONFIGU.
C              THIS MODULE READS IOCONFIGU. FOR ONE DAY TO BUILD PART OF
C              THE DEVDAT ARRAY.
C
C  INVOCATION METHOD:  CALL PMS.DRFFR1
C
C  FILE/RECORD REFERENCES:
C   FILE NAME            USE                 DESCRIPTION
C   PMS*IOCONFIGU.        I             I/O CONFIGURATION FILE
C
C  LOCAL VARIABLES:
C    NAME                TYPE                DESCRIPTION
C    IOID                INT            UNIT NUMBER OF IOCONFIGU.
C    IOMNEM              CHAR           DEVICE MNEMONIC FROM IOCONFIGU.
C    IOUNIT              CHAR           I/O UNIT NUMBER
C    IOMODL             CHAR           MODEL CODE FOR THE DEVICE
C    IOSTAT              INT            UP OR DOWN STATUS OF THE DEVICE
C    DEVFND              CHAR           DEVICE FOUND INDICATOR FLAG
C
C  UNIT FLOW
C-----------------------------------------------------------------------------
C
C  INITIALIZE IOCTYP TO 0
C   ((FIND NEXT HEADER RECORD OF IOCONFIGU., HAVING IOCTYP OF 10))
C  DO UNTIL IOCTYP=10 OR EOF ((NORMAL EXITS)), OR ERROR CNDITION
C     READ IOCTYP FROM IOCONFIGU. ((FORMAT (12X,I2) ))
C     IF IOCTYP = 05
C     THEN
C          ((RE-READ RECORD FROM BUFFER.  THIS CAN BE DONE BY READING FROM
C           UNIT NUMBER ZERO))
C        READ FIELDS OF CURRENT RECORD OF IOCONFIGU. INTO IOID, IOMNEM,
C         IOUNIT, IOMODL, AND IOSTAT  ((FORMAT (17X, I3, 19X, A3, 2X, A1,
C          A6, 5X, I1) ))
C        SET DEVICE FOUND INDICATOR, DEVFND, TO "N"
C        DO (FOR CNT=1 TO NUMBER OF DEVICES, NUMDEV) OR (UNTIL DEVFND="Y")
C            ((MATCH THE DEVICE IN IOCONFIGU. WITH THE DEVICE IN THE
C             DEVDEF ARRAY.  WHEN A MATCH IS FOUND, SAVE THE UP/DOWN STATUS
C             AND SET THE NEXT UNUSED DEVDAT ELEMENT TO THE UNIT NUMBER
C             READ FROM IOCONFIGU.))
C           IF (DEVDEF(CNT,1) = IOMNEM) AND (DEVDEF(CNT,2) = IOMODL)
C            AND (DEVDEF(CNT,3) = IOUNIT)
C           THEN
C              SET UP/DOWN CODE IN DEVDAT(CNT,8) TO IOSTAT
C              SET DEVFND TO "Y"
C               ((SET THE SEVEN ALTERNATIVE DEVICE NUMBERS OF DEVDAT))
C              IF DEVICE NUMBER IN DEVDAT(CNT,1) HAS NOT YET BEEN SET,
C               STILL 0,
C              THEN SET DEVDAT(CNT,1) TO IOID
C              ELSE
C                 IF DEVDAT(CNT,2) = 0
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
C                         THEN SET DEVDAT(CNT,2) TO IOID
C                         ELSE
C                            IF DEVDAT(CNT,3) =
C                            THEN SET DEVDAT(CNT,3) TO IOID
C                            ELSE
C                               IF DEVDAT(CNT,4) = 0
C                               THEN SET DEVDAT(CNT,4) TO IOID
C                               ELSE
C                                  IF DEVDAT(CNT,13) = 0
C                                  THEN SET DEVDAT(CNT,13) TO IOID
C                                  ELSE
C                                     IF DEVDAT(CNT,14) = 0
C                                     THEN SET DEVDAT(CNT,14) TO IOID
C                                     ELSE
C                                        IF DEVDAT(CNT,15) = 0
C                                        THEN SET DEVDAT(CNT,15) TO IOID
C                                        ELSE SET DEVDAT(CNT,16) TO IOID
C                                        ENDIF ((FOR ALTERNATE #6))
C                                     ENDIF ((FOR ALTERNATE # 5))
C                                  ENDIF ((FOR ALTERNATE #4))
C                               ENDIF ((FOR ALTERNATE #3))
C                            ENDIF ((FOR ALTERNATE #2))
C                         ENDIF((FOR ALTERNATE #1))
C                      ENDIF ((FOR DEVICE NUMBER))
C                   ENDIF
C                ENDDO
C             ENDIF
C  ENDDO
C  RETURN
C-----------------------------------------------------------------------
C  CODE
C-----------------------------------------------------------------------
      SUBROUTINE DRMFR1
C
C     **LOCAL DECLARATIONS**
        INTEGER    IOID,IOSTAT,CNT,IOCTYP
        CHARACTER IOMNEM*3,IOUNIT*1,IOMODL*6,DEVFND*1
C     **GLOBAL DECLARATIONS**
        INTEGER    SAVDAT,PARTYP,ELADAT,INTVAL,CPUCNT,PROCNM(4),TOTMEM
        INTEGER    TIMEON,TIMEOF,TIPCNT,RESMEM,EXCMEM,COMMEM,TIPMEM,RELMEM
        INTEGER    NUMDEV,DEMMEM,BATMEM,SWPCNT,OLDATE,PARMS(15),CDBMAX
        INTEGER    NUMCPU,DSKCNT,IN,IN1,IN2,IN3
        REAL       FEPFUJ,ELPTIM,SYSIDL,TIPPRG,DEVDAT(99,16)
        REAL       RTMPRG,DMDPRC,BATPRG,PROIDL,HSTARY(102)
        CHARACTER  DEVDEF(99,4)*6,NEXTDY*1,EXLEVL*12,ENDFIL*1
        COMMON     FEPFUJ,PARMS,DEVDEF,DEVDAT,NUMDEV,SAVDAT,CPUCNT,ENDFIL
        COMMON     PARTYP,ELADAT,INTVAL,NEXTDY,ELPTIM,PROCNM,TOTMEM,DSKCNT
        COMMON     NUMCPU,EXLEVL,TIMEON,TIMEOF,TIPCNT,SYSIDL,RESMEM,EXCMEM
        COMMON     COMMEM,TIPMEM,RELMEM,DEMMEM,BATMEM,SWPCNT,TIPPRG,RTMPRG
        COMMON     DMDPRC,BATPRG,OLDATE,PROIDL,HSTARY,IN,IN1,IN2,IN3,CDBMAX
        COMMON     DMDRST
C
  210   FORMAT(12X,I3)
  220   FORMAT(17X,I3,19X,A3,2X,A1,A6,5X,I1)
C
        IOCTYP = 0
        DO 25. CNT=1,NUMDEV
```

```
              DEVDAT(CNT,1)=0.0
              DEVDAT(CNT,2)=0.0
              DEVDAT(CNT,3)=0.0
              DEVDAT(CNT,4)=0.0
              DEVDAT(CNT,13)=0.0
              DEVDAT(CNT,14)=0.0
              DEVDAT(CNT,15)=0.0
              DEVDAT(CNT,16)=0.0
250     CONTINUE
260     READ (IN3,210,ERR=666,END=240) IOCTYP
        IF (IOCTYP.EQ.25) THEN
           READ (3,220) IOID,IOMMEM,IOUNIT,IOMODL,IOSTAT
           DEVFND = 'N'
           DO 300 CNT=1,NUMDEV
              IF ((DEVDEF(CNT,1).EQ.IOMMEM).AND.(DEVDEF(CNT,2).EQ.
     &            IOMODL).AND.(DEVDEF(CNT,3).EQ.IOUNIT)) THEN
                 DEVDAT(CNT,6) = IOSTAT
                 DEVFND = 'Y'
                 IF (DEVDAT(CNT,1).EQ.0.0) THEN
                    DEVDAT(CNT,1) = IOID
                 ELSE
                    IF (DEVDAT(CNT,2).EQ.0.0) THEN
                       DEVDAT(CNT,2) = IOID
                    ELSE
                       IF (DEVDAT(CNT,3).EQ.0.0) THEN
                          DEVDAT(CNT,3) = IOID
                       ELSE
                          IF (DEVDAT(CNT,4).EQ.0.0) THEN
                             DEVDAT(CNT,4) = IOID
                          ELSE
                             IF (DEVDAT(CNT,13).EQ.0.0) THEN
                                DEVDAT(CNT,13) = IOID
                             ELSE
                                IF (DEVDAT(CNT,14).EQ.0.0) THEN
                                   DEVDAT(CNT,14) = IOID
                                ELSE
                                   IF (DEVDAT(CNT,15).EQ.0.0) THEN
                                      DEVDAT(CNT,15) = IOID
                                   ELSE
                                      DEVDAT(CNT,16) = IOID
                                   ENDIF
                                ENDIF
                             ENDIF
                          ENDIF
                       ENDIF
                    ENDIF
                 ENDIF
              ENDIF
              IF (DEVFND.EQ.'Y') GO TO 325
300        CONTINUE
325     ENDIF
        IF (IOCTYP.NE.'0') GO TO 260
340     RETURN
440     CALL FSETC (440)
        STOP
        END
C
```

```
C
C
C
C--------------------------------------------------------------------
C SUBROUTINE DRMFFD (DATA REDUCTION MODULE - FORTRAN REDUCTION 3)
C
C PURPOSE:      TO PROCESS PARDATAFL. THIS MODULE READS ALL THE INFORMATION
C               FOR ONE BLOCK OF DATA (15 MINUTE PERIOD) INTO THE DEVDAT
C               ARRAY AND SETS OTHER VARIABLES IN PREPARATION FOR THE
C               COMPUTATIONS WHICH WILL FOLLOW.
C
C INVOCATION METHOD:  CALL FMS.DRMFFD
C
C FILE/RECORD REFERENCES:
C  FILE NAME            USE                DESCRIPTION
C  FMS*PARDATAFL.        I              DATA REDUCED FROM PAR
C
C LOCAL VARIABLES:
C  NAME                 TYPE               DESCRIPTION
C   OLDINT              INT            OLD INTERVAL NUMBER FROM PARDATAFL.
C   SDEVUN              INT            DEVICE UNIT NUMBER
C   SINREQ              INT            INPUT REQUESTS
C   SOTREQ              INT            OUTPUT REQUESTS
C   SDATTM              REAL           DATA TRANSFER TIME
C   SEXTIM              REAL           EXISTENCE TIME
C   SREQUD              INT            NUMBER OF REQUESTS QUEUED
C   SCUMSZ              INT            CUMULATIVE QUEUE SIZE
C
C UNIT FLOW
C--------------------------------------------------------------------
C
C INITIALIZE PROIDL TO 0
C SET OLDINT = INTVAL
C  ((CLEAR OUT DEVDAT ARRAY COLUMNS CORRESPONDING TO TOTAL REQUESTS IN,
C   TOTAL REQUESTS OUT, EXISTENCE TIME, AND DATA TRANSFER TIME: COLUMNS
C   7-12))
C DO FOR CNT=1 TO NUMBER OF DEVICES, NUMDEV
C    DO FOR COL=7 TO 12
C       INITIALIZE DEVDAT(CNT,COL) TO 0
C    ENDDO
C ENDDO
C CLEAR ALL FOUR ELEMENTS OF PROCAM ARRAY
C INITIALIZE CPUCNT, ELEMENT COUNTER OF PROCAM, TO 0
C  ((WHEN INTVAL EXCEEDS OLDINT, A BLOCK HAS BEEN COMPLETED))
C DO WHILE INTVAL IS EQ TO OLDINT
C     ((CHECK TO SEE IF PARDATAFL. HEADER IS BEING LOOKED AT))
C     IF RECORD TYPE, PARTYP, EQ 01
C      ((RE-READ FROM BUFFER PARAMETERS FROM PARDATAFL. HEADER))
C     THEN READ PARDATAFL. AGAIN, INTO: TOTMEM, NUMCPU, EXLEVL
C      ((FORMAT (12X, I4, 1X, I2, 47X, A12) ))
C     ENDIF
C     IF (PARTYP = 2) AND (SUBTYP = 1)
C      ((RE-READ FROM BUFFER WITH AN APPROPRIATE FORMAT WHICH TAKES US TO
C        THE DESIRED PARAMETER OF RECORD. THE FOLLOWING FIVE IF STATEMENTS
C        ARE ALSO STRUCTURED UPON THIS BASIS))
C     THEN READ PARDATAFL. AGAIN, INTO: TIMEON, TIMEOF  ((FORMAT (12X, I11,
C      1X, I11) ))
```

```
C      ENDIF
C      IF (PARTYP = 2) AND (SUBTYP = 2)
C      THEN READ PARDATAFL, AGAIN, INTO: ELPTIM, TIPCNT, SYSIDL, REXMEM,
C       EXCMEM  ((FORMAT (20X, F11.4, 1X, I11, 1X, F11.4, 49X, I11,
C        1X, I11) ))
C      ENDIF
C      IF (PARTYP = 2) AND (SUBTYP = 3)
C      THEN READ PARDATAFL, AGAIN, INTO: COMMEM, TIPMEM, RELMEM, DEMMEM,
C       DATMEM, SWPCNT  ((FORMAT (21X, I11, 1X, I11, 1X, I11, 1X, I11) ))
C      ENDIF
C      IF (PARTYP = 2) AND (SUBTYP = 4)
C      THEN READ PARDATAFL, AGAIN, INTO: TIPPRC, RTMPRG, DMDPRG, BATPRG,
C        PRDRST
C       ((FORMAT(21X, F11.4, 1X, F11.2, 1X, F11.2, 1X, F11.2, 1X, F11.2)))
C      ENDIF
C      IF PARTYP = 3
C      THEN
C         INCREMENT CPUCNT BY 1
C          ((FOLLOWING READ DONE WITH FORMAT (I6, 11X, I3, 85X, F11.4) ))
C         READ PARDATAFL, AGAIN, INTO: BLDATE, PROCNM(CPUCNT), IDLE
C          ((UPDATE PROCESSOR IDLE TIME))
C         PROIDL = PROIDL + IDLE
C      ENDIF
C      IF PARTYP = 5   ((A DEVICE))
C      THEN
C          ((FOLLOWING READ DONE WITH FORMAT (17X, I3, 1X, I11, 1X, I11,
C           25X, F11.4, 1X, F11.4, 1X, I11, 1X, I11) ))
C         READ PARDATAFL, AGAIN, INTO: SDVEUN, SINREQ, SOTREQ, SDATTM,
C          SEXTIM, SREQUD, SCUMSZ
C          ((CHECK TO FIND DEVICE UNIT NUMBER, SDVEUN  IN DEVDAT DEVICE
C           NUMBER OF THE 7 ALTERNATIVE DEVICE NUMBERS))
C         DO FOR CNT=1 TO NUMBER OF DEVICES, NUMDEV
C            IF (DEVDAT(CNT,1) = SDEVUN)
C              OR (DEVDAT(CNT,2) = SDEVUN)
C              OR (DEVDAT(CNT,3) = SDEVUN)
C              OR (DEVDAT(CNT,4) = SDEVUN)
C              OR (DEVDAT(CNT,13) = SDEVUN)
C              OR (DEVDAT(CNT,14) = SDEVUN)
C              OR (DEVDAT(CNT,15) = SDEVUN)
C              OR (DEVDAT(CNT,16) = SDEVUN)
C            THEN
C               UPDATE TOTAL REQUESTS IN, DEVDAT(CNT,7) BY ADDING
C                SINREQ TO IT
C               UPDATE TOTAL REQUESTS OUT, DEVDAT(CNT,8) BY ADDING
C                SOTREQ TO IT
C               UPDATE CUMULATIVE QUEUE SIZE, DEVDAT(CNT,9), BY ADDING
C                SCUMSZ TO IT
C               UPDATE NO. OF REQUESTS QUEUED, DEVDAT(CNT,10), BY ADDING
C                SREQUD TO IT
C               UPDATE EXISTENCE TIME, DEVDAT(CNT,11), BY ADDING
C                SEXTIM TO IT
C               UPDATE DATA TRANSFER TIME, DEVDAT(CNT,12) BY ADDING
C                SDATTM TO IT
C            ENDIF
C          ENDDO
C       ENDIF
C       READ NEXT PARDATAFL, RECORD INTO: BLKDAT, INTVAL, PARTYP, SUBTYP
```

```
C          ((FORMAT (I6, 1X, I4, 1X, I2, 2X, I2) ))
C          ((IF WE GET TO ANOTHER HEADER IN PARDATAFL., ONE DAY OF DATA HAS
C           BEEN PROCESSED))
C         IF END OF PARDATAFL., SET ENDFIL FLAG TO "Y" AND EXIT (RETURN)
C         IF PARTYP = 1
C         THEN SET NEXTDY TO "Y"
C  ENDDO
C  RETURN
C
C
C  ---------------------------------------------------------------------
C  CODE
C ----------------------------------------------------------------------
           SUBROUTINE GRNFR2   ,
C
C       **LOCAL DECLARATIONS**
           INTEGER OLDINT,SOLVUN,SINREG,SOTREG,SKEWUD,SCUMSZ,CNT,COL,SUBTYP
           REAL     SCATTN,SEXTIM,IDLE
C       **GLOBAL DECLARATIONS**
           INTEGER    SAVDAT,PARTYP,BLKDAT,INTVAL,CPUCNT,PROCNM(4),TOTMEM
           INTEGER    TIMEON,TIMEOF,TIPCNT,REXMEM,EXCMEM,COMMEM,TIPMEM,RELMEM
           INTEGER    NUMDEV,DEMMEM,BATMEM,SWPCNT,BLDATE,PARMS(15),CDBMAX
           INTEGER    NUMCPU,DSKCNT,IN,IN1,IN2,IN3
           REAL       FEPFUJ,ELPTIM,SYSIDL,TIPPRG,DEVDAT(99,16)
           REAL       RTMPRG,DMDPRG,BATPRG,PROIDL,HSTARY(102)
           CHARACTER  DEVDEF(99,4)*6,NEXTDY*1,EXLEVL*12,ENDFIL*1
           COMMON     FEPFUJ,PARMS,DEVDEF,DEVDAT,NUMDEV,SAVDAT,CPUCNT,ENDFIL
           COMMON     PARTYP,BLKDAT,INTVAL,NEXTDY,ELPTIM,PROCNM,TOTMEM,DSKCNT
           COMMON     NUMCPU,EXLEVL,TIMEON,TIMEOF,TIPCNT,SYSIDL,REXMEM,EXCMEM
           COMMON     COMMEM,TIPMEM,RELMEM,DEMMEM,BATMEM,SWPCNT,TIPPRG,RTMPRG
           COMMON    DMDPRG,BATPRG,BLDATE,PROIDL,HSTARY,IN,IN1,IN2,IN3,CDBMAX
           COMMON    DMDRST
C
   340     FORMAT(32X,I9,1X,I3,17X,A12)
   350     FORMAT(21X,I11,1X,I11)
   360     FORMAT(21X,F11.4,1X,I11,1X,F11.4,49X,I11,1X,I11)
   370     FORMAT(21X,I11,1X,I11,1X,I11,1X,I11,1X,I11,1X,I11)
   380     FORMAT(21X,F11.2,1X,F11.2,1X,F11.2,1X,F11.2,1X,F11.2)
   390     FORMAT(I6,11X,I3,85X,F11.4)
   400     FORMAT(17X,I3,1X,I11,1X,I11,25X,F11.4,1X,F11.4,1X,I11,1X,I11)
   430     FORMAT(I6,1X,I4,1X,I2,3X,I3)
C
           PROCNM(1) = 0
           PROCNM(2) = 0
           PROCNM(3) = 0
           PROCNM(4) = 0
           CPUCNT    = 0
           PROIDL    = 0.0
           OLDINT = INTVAL
           DO 320 CNT=1,NUMDEV
              DO 310 COL=7,16
                 DEVDAT(CNT, COL) = 0.0
   310        CONTINUE
   320     CONTINUE
   330     IF (PARTYP.EQ.01) THEN
              READ (5,340) TOTMEM,NUMCPU,EXLEVL
           ENDIF
```

```
           IF ((PARTYP.EQ.2).AND.(SUBTYP.EQ.1)) THEN
              READ (1,350) TIMEON,TIMEOF
           ENDIF
           IF ((PARTYP.EQ.2).AND.(SUBTYP.EQ.2)) THEN
              READ (1,360) ELPTIM,TIPCNT,SYSIDL,REAMEM,EACMEM
           ENDIF
           IF ((PARTYP.EQ.2).AND.(SUBTYP.EQ.3)) THEN
              READ (1,370) COMMEN,TIPMEM,RELMEM,LENMEM,BATMEM,SWPCNT
           ENDIF
           IF ((PARTYP.EQ.2).AND.(SUBTYP.EQ.4)) THEN
              READ (1,380) TIPPRG,RTMPRC,DMDPRC,BATPRG,DMDRST
           ENDIF
           IF (PARTYP.EQ.3) THEN
              CPUCNT = CPUCNT + 1
              READ (1,390) PLDATL,PROCNM(CPUCNT),IDLE
              PRCIDL = PRCIDL + IDLE
           ENDIF
           IF (PARTYP.EQ.5) THEN
              READ (1,400) SDEVUN,SINREQ,SOTREQ,SDATTM,SEXTIM,SREQUD,SCUMSZ
              DO 410 CNT=1,NUMDEV
                 IF ((DEVDAT(CNT,1).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,2).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,3).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,4).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,13).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,14).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,15).EQ.SDEVUN)
     C            .OR. (DEVDAT(CNT,16).EQ.SDEVUN)) THEN
                    DEVDAT(CNT,7) = DEVDAT(CNT,7) + SINREQ
                    DEVDAT(CNT,8) = DEVDAT(CNT,8) + SOTREQ
                    DEVDAT(CNT,9) = DEVDAT(CNT,9) + SCUMSZ
                    DEVDAT(CNT,10) = DEVDAT(CNT,10) + SREQUD
                    DEVDAT(CNT,11) = DEVDAT(CNT,11) + SEXTIM
                    DEVDAT(CNT,12) = DEVDAT(CNT,12) + SDATTM
                 ENDIF
     410      CONTINUE
           ENDIF
           READ(IN1,420,END=430) BLKDAT,INTVAL,PARTYP,SUBTYP
           IF (PARTYP.EQ.01) NEXTDY = 'Y'
           IF (INTVAL.EQ.OLDINT) GO TO 330
           GO TO 440
     430   ENDFIL = 'Y'
     440   RETURN
           END
     C
     C
     C
     C
     C-----------------------------------------------------------------
     C  PURPOSE:       TO COMPLETE THE HSTARY ARRAY WHICH WILL BE OUTPUTED TO
     C                 SPHISTORY. FILE
     C
     C  INVOCATION METHOD:  CALL PRS.DPPRNT
     C
     C  FILE/RECORD REFERENCES:
     C   FILE NAME          USE               DESCRIPTION
     C
```

```
C  LOCAL VARIABLES:
C    NAME               TYPE              DESCRIPTION
C    TMOF               REAL              TIMEOF CONVERTED TO REAL NUMBER
C    TMON               REAL              TIMEON CONVERTED TO REAL NUMBER
C    DSKTRN             REAL              TOTAL DISK TRANSFER TIME
C    TOTREQ             REAL              TOTAL NUMBER OF REQUESTS
C    DEVTIM             REAL              DEVICE TIME
C    TRUQLN             REAL              TRUE QUEUE LENGTH
C    TRUSER             REAL              TRUE SERVICE TIME
C    DISKCNT            INT/COMMON        COUNTER OF DISKS UP
C
C UNIT FLOW
C-------------------------------------------------------------------------
C
C INITIALIZE ALL ELEMENTS OF HSTARY ARRAY TO -1.0, SYMBOL OF MISSING DATA
C CONVERT TIMEON FROM SEC. TO HR. REPRESENTATION, RESULT IN TMON
C CONVERT TIMEOF FROM SEC. TO HR. REPRESENTATION, RESULT IN TMOF
C  ((ADJUST FOR FAILURE TO RESET SIP AFTER 24 HOURS))
C SET HSTARY(1) TO TMON MINUS NUMBER OF HRS. PAST 24 HRS.
C SET HSTARY(2) TO TMOF MINUS NUMBER OF HRS. PAST 24 HRS.
C COMPUTE DATE OF THE DAY BEING PROCESSED IN THE FORM YYMM.DD FROM SAVDAT
C  AND TMON AND SAVE IN HSTARY(3)
C  ((HENCE FORTH ELEMENTS OF HSTARY ARRAY WILL BE REFERENCED AS THREE
C   PLUS THE ARRAY COUNTER TO TAKE INTO ACCOUNT THE THREE ELEMENTS ALREADY
C   CREATED))
C IF PARMS(1) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN COMPUTE TIP THRUPUT AS THE QUOTIENT OF TIP TRANSACTION COUNT,
C  TIPCNT, AND ELAPSED TIME, ELPTIM, RESULT IN HSTARY(PARMS(1)+3)
C ENDIF
C IF PARMS(2) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN SET HSTARY(PARMS(2)+3) TO AVERAGE NUMBER OF DEMAND JOBS IN MEMORY,
C  DMDPRG
C ENDIF
C IF PARMS(3) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN SET HSTARY(PARMS(3)+3) TO AVERAGE NUMBER OF BATCH PROGRAMS IN
C  MEMORY, BATPRG
C ENDIF
C IF PARMS(4) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN SET HSTARY(PARMS(4)+3) TO AVERAGE NUMBER OF REAL TIME PROGRAMS
C  IN MEMORY, RTMPRG
C ENDIF
C IF PARMS(5) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN
C      ((MAKE SURE DIVISION BY ZERO DOES NOT HAPPEN))
C    IF TIPCNT NE ZERO
C    THEN COMPUTE TIP RESPONSE TIME AS (TIPPRG * ELPTIM) / TIPCNT,
C       RESULT IN HSTARY(PARMS(5)+3)
C    ENDIF
C ENDIF
C IF PARMS(6) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN HSTARY(PARMS(6)+3) = DMDRST (DEMAND RESPONSE TIME)
C ENDIF
C IF PARMS(7) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C THEN COMPUTE CPU UTILIZATION AS 100 * (1-((SYSIDL+PROIDL)/CPUCNT) /
C  ELPTIM), RESULT IN HSTARY(PARMS(7)+3)
C ENDIF
C IF PARMS(8) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C  THEN COMPUTE MEMORY UTILIZATION AS 100*((FEXMEM+E)CMEM+COMMEM+TIPMEM+
C   RELMEM+DEMME+DATMEM)/TCTMEM), RESULT IN HSTARY(PARMS(6)+3)
C  ENDIF
C  IF PARMS(9) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C  THEN COMPUTE THE SWAP RATE AS SWPCNT / ELPTIM, RESULT IN
C   HSTARY(PARMS(9)+3)
C  ENDIF
C  IF PARMS(10) HAS BEEN SET TO A CDB FILE NO., NO LONGER EQ TO 0
C  THEN
C      INITIALIZE TOTAL DISK TRANSFER TIME, DSKTRN, TO 0
C      INITIALIZE DISK COUNTER, DSKCNT, TO ZERO
C       ((CALCULATE TOTAL DISK TRANSFER TIME))
C      DO FOR CNT=1 TO TOTAL NUMBER OF DEVICES, NUMDEV,
C         IF (DATA TYPE IN (DEVDEF(CNT,4) IS "DSKUL") AND (IOSTAT,
C          DEVDAT(CNT,6), IS UP, EQ TO 1)
C         THEN
C            ADD TO DSKTRN THE NEW DATA TRANSFER TIME IN DEVDAT(CNT,12)
C            INCREMENT DISK COUNTER, DSKCNT, BY 1
C         ENDIF
C      ENDDO
C      COMPUTE I/O CHANNEL UTILIZATION AS (DSKTRN*0.5*100) / ELPTIM, RESULT
C       IN HSTARY(PARMS(10)+3)
C  ENDIF
C  DO FOR CNT=1 TO NUMDEV
C      COMPUTE TOTAL NUMBER OF REQUESTS, TOTREQ, BY SUMMING INPUT REQUESTS,
C       DEVDAT(CNT,7), AND OUTPUT REQUESTS, DEVDAT(CNT,8)
C       ((COMPUTE DEVICE UTILIZATION/THRUPUT FOR DTV AND OTHER I/O DEVICES
C        UP))
C      IF (DEVDEF(CNT,4) IS "DTVWL") OR (DEVDEF(CNT,4) IS "DEVWL")
C       AND DEVDAT(CNT,6) IS UP, EQ TO 1)
C      THEN COMPUTE DEVICE UTILIZATION/THRUPUT AS TOTREQ DIVIDED BY ELPTIM,
C       RESULT IN HSTARY(DEVDAT(CNT,5)+3)
C      ELSE CLEAR HSTARY(DEVDAT(CNT,5)+3)
C      ENDIF
C       ((COMPUTE DEVICE UTILIZATION/THRUPUT FOR FEPS UP))
C      IF (DEVDEF(CNT,4) IS "FEPWL")
C      THEN
C         IF (DEVDAT(CNT,6) IS UP, EQ TO 1)
C         THEN COMPUTE DEVICE UTIL. THRUPUT :(TOTREQ*FEPFUJ) / ELPTIM,
C          RESULT IN HSTARY(DEVDAT(CNT,5)+3)
C         ELSE CLEAR HSTARY(DEVDAT(CNT,5)+3)
C         ENDIF
C       ENDIF
C       ((COMPUTE DEVICE UTILIZATION/THRUPUT FOR DISKS UP))
C      IF DEVDEF(CNT,4) IS "DSKUL"
C      THEN
C         IF DEVDAT(CNT,6) IS UP, EQ TO 1
C         THEN
C            COMPUTE DEVICE TIME, DEVTIM, AS SUM OF EXISTENCE TIME,
C             DEVDAT(CNT,11), AND DATA TRANSFER TIME, DEVDAT(CNT,12)
C            IF (TOTAL NUMBER OF REQUESTS, TOTREQ, NE 0) AND (NUMBER OF
C             REQUESTS QUEUED, DEVDAT(CNT,10), NE 0)
C            THEN CALCULATE TRUE QUEUE LENGTH, TRUQLN, AS
C             1 + (DEVDAT(CNT,9)/(DEVDAT(CNT,10)-1))*(DEVDAT(CNT,10)/TOTREQ)
C            ELSE SET TRUQLN TO 1 AS QUEUE IS EMPTY
C            ENDIF
C            IF TOTREQ NE 0
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C              THEN COMPUTE THE SERVICE TIME, TRUSER, AS DEVICE TIME, DEVTIM,
C               DIVIDED BY THE QUOTIENT OF TOTREQ AND TRUQLN
C              ELSE CLEAR TRUSER
C              ENDIF
C              COMPUTE DEVICE UTILIZATION/THRUPUT, HSTARY(DEVDAT(CNT,5)+3), AS
C               100 * TOTREQ * TRUSER / ELPTIM
C           ELSE CLEAR HSTARY(DEVDAT(CNT,5)+3)
C           ENDIF
C       ENDIF
C   ENDDO
C   RETURN
C
C
C   ------------------------------------------------------------------------
C   CODE
C   ------------------------------------------------------------------------
          SUBROUTINE DRMFR3
C
C       **LOCAL DECLARATIONS**
          INTEGER CNT,I
          REAL     TMOF,TMON,DSKTRN,TOTREQ,DEVTIM,TRUQLN,TRUSER
C       **GLOBAL DECLARATIONS**
          INTEGER    SAVDAT,PARTYP,BLKDAT,INTVAL,CPUCNT,PROCNM(4),TOTMEM
          INTEGER    TIMEON,TIMEOF,TIPCNT,REXMEM,EXCMEM,COMMEM,TIPMEM,RELMEM
          INTEGER    NUMDEV,DEMMEM,BATMEM,SWPCNT,BLDATE,PARMS(15),CDBMAX
          INTEGER    NUMCPU,DSKCNT,IN,IN1,IN2,IN3
          REAL       FEPFUJ,ELPTIM,SYSIDL,TIPPRG,DEVDAT(99,16)
          REAL       RTMPRG,DMDPRG,BATPRG,PROIDL,HSTARY(102)
          CHARACTER DEVDEF(99,4)*6,NEXTDY*1,EXLEVL*12,ENDFIL*1
          COMMON     FEPFUJ,PARMS,DEVDEF,DEVDAT,NUMDEV,SAVDAT,CPUCNT,ENDFIL
          COMMON     PARTYP,BLKDAT,INTVAL,NEXTDY,ELPTIM,PROCNM,TOTMEM,DSKCNT
          COMMON     NUMCPU,EXLEVL,TIMEON,TIMEOF,TIPCNT,SYSIDL,REXMEM,EXCMEM
          COMMON     COMMEM,TIPMEM,RELMEM,DEMMEM,BATMEM,SWPCNT,TIPPRG,RTMPRG
          COMMON    DMDPRG,BATPRG,BLDATE,PROIDL,HSTARY,IN,IN1,IN2,IN3,CDBMAX
          COMMON    DMDRST
C
          DO 490 I=1,102
             HSTARY(I) = -1.0
   490    CONTINUE
          TMON = (1.0*TIMEON) / 3600
          TMOF = (1.0*TIMEOF) / 3600
          HSTARY(1) = TMON - (IFIX(TMON/24.0) * 24.0)
          HSTARY(2) = TMOF - (IFIX(TMOF/24.0) * 24.0)
          HSTARY(3) = (SAVDAT * 0.01) + (IFIX(TMON / 24.0) * 0.01) + 0.005
          IF (PARMS(1).NE.0) THEN
             HSTARY(PARMS(1)+3) = TIPCNT / ELPTIM
          ENDIF
          IF (PARMS(2).NE.0) THEN
             HSTARY(PARMS(2)+3) = DMDPRG
          ENDIF
          IF (PARMS(3).NE.0) THEN
             HSTARY(PARMS(3)+3) = BATPRG
          ENDIF
          IF (PARMS(4).NE.0) THEN
             HSTARY(PARMS(4)+3) = RTMPRG
          ENDIF
          IF (PARMS(5).NE.0) THEN
```

```
                IF (TIPCNT.NE.0) THEN
                   HSTARY(PARMS(5)+3) = TIPPRC + ELPTIM / TIPCNT
                ENDIF
             ENDIF
             IF (PARMS(6).NE.0) THEN
                HSTARY(PARMS(6)+3) = DMCNET
             ENDIF
             IF (PARMS(7).NE.0) THEN
                HSTARY(PARMS(7)+3) = 100 * (1-((SYSIDL+PROIDL) / CPUCNT)
     C          / ELPTIM)
             ENDIF
             IF (PARMS(8).NE.0) THEN
                HSTARY(PARMS(8)+3) = 100*(REXMEM+EXCMEM+COMMEM+TIPMEM+RELMEM+
     C          DEVMEM+DATMEM)/TOTMEM
             ENDIF
             IF (PARMS(9).NE.0) THEN
                HSTARY(PARMS(9)+3) = SWPCNT / ELPTIM
             ENDIF
             IF (PARMS(10).NE.0) THEN
                DSKTRN = 0.0
                DSKCNT = 0
                DO 500 CNT=1,NUMDEV
                   IF ((DEVDEF(CNT,4).EQ.'DSKUL')
     C              .AND.(DEVDAT(CNT,6).EQ.0)) THEN
                      DSKTRN = DSKTRN + DEVDAT(CNT,12)
                      DSKCNT = DSKCNT + 1
                   ENDIF
       500      CONTINUE
                HSTARY(PARMS(10)+3) = DSKTRN + 0.5 * 100 / ELPTIM
             ENDIF
             DO 510 CNT=1,NUMDEV
                IF ((DEVDEF(CNT,4).EQ.'CTVWL')
     C           .OR.(DEVDEF(CNT,4).EQ.'DEVWL')) THEN
                   IF (DEVDAT(CNT,6).EQ.0) THEN
                      HSTARY(DEVDAT(CNT,5)+3) =
     C                 (DEVDAT(CNT,7)+DEVDAT(CNT,8))/ELPTIM
                   ELSE
                      HSTARY(DEVDAT(CNT,5)+3) = 0.0
                   ENDIF
                ENDIF
                IF (DEVDEF(CNT,4).EQ.'FEPWL') THEN
                   IF (DEVDAT(CNT,6).EQ.0) THEN
                      HSTARY(DEVDAT(CNT,5)+3)=((DEVDAT(CNT,7)+DEVDAT(CNT,8))
     C                * FEPFUD) / ELPTIM
                   ELSE
                      HSTARY(DEVDAT(CNT,5)+3) = 0.0
                   ENDIF
                ENDIF
                IF (DEVDEF(CNT,4).EQ.'DSKUL') THEN
                   IF (DEVDAT(CNT,6).EQ.0) THEN
                      TOTREQ = DEVDAT(CNT,7) + DEVDAT(CNT,8)
                      DEVTIM = DEVDAT(CNT,11) + DEVDAT(CNT,12)
                      IF ((TOTREQ.NE.0.0).AND.(DEVDAT(CNT,10).NE.0)) THEN
                         TRUQLN = 1 + (DEVDAT(CNT,9) / DEVDAT(CNT,10) - 1) *
     C                    DEVDAT(CNT,10) / TOTREQ
                      ELSE
                         TRUQLN = 1.0
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
                   ENDIF
                   IF (TOTREQ.GT.0.0) THEN
                      TRUSER = DEVTIM / (TOTREQ * TRUSER)
                   ELSE
                      TRUSER = 0.0
                   ENDIF
                   HSTARY(DEVDAT(CNT,5)+7) = 100 * (TOTREQ * TRUSER /
     :                ELPTIM)
                ELSE
                   HSTARY(DEVDAT(CNT,5)+3) = 0.0
                ENDIF
             ENDIF
 510      CONTINUE
          RETURN
          END
C
C
C
C
C
C--------------------------------------------------------------------------
C SUBROUTINE DRMFR4  (DATA REDUCTION MODULE - FORTRAN REDUCTION 4)
C
C PURPOSE:   THIS MODULE WILL DO SOME FINAL DATA VERIFICATION AND THEN
C            WRITE THE HSTARY RECORDS INTO SPHISTORY.
C
C INVOCATION METHOD:  CALL PMS.DRMFR4
C
C FILE/RECORD REFERENCES:
C  FILE NAME              USE              DESCRIPTION
C  PMS*SPHISTORY           O          SIP HISTORY FILE HOLDING DEVICE DATA
C
C INTERMEDIATE VARIABLES:
C   NAME                 TYPE                DESCRIPTION
C   CNT                  INT          COUNTER FOR LOOPS
C   IN3                  INT/COMMON   SPHISTORY FILE REFERENCE NUMBER
C   NUMDEV               INT/COMMON   NUMBER OF DEVICES SPECIFIED IN
C                                     PARAMADEF.
C   HSTARY               REAL/COMMON  ONE DIMENSIONAL ARRAY OF 102 ELEMENTS
C                                     HOLDING DATA FOR SPHISTORY. CREATION
C   IDATE                INT          DATE FORMATTING VARIABLE
C   MONTH                INT          MONTH OF BLOCK DATE
C   DAY                  INT          DAY OF BLOCK DATE
C   CALNDR               INT          ONE DIMENSIONAL ARRAY OF 12 ELEMENTS
C                                     HOLDING THE NUMBER OF DAYS IN EACH OF
C                                     THE 12 MONTHS
C   YEAR                 INT          LAST TWO DIGITS OF YEAR
C   LEAP                 REAL         USED FOR LEAP YEAR DETERMINATION
C   REM                  REAL         REMAINDER OF YEAR DIVIDED BY 4
C
C UNIT FLOW
C--------------------------------------------------------------------------
C ((HSTARY ARRAY HAS CDDMAX+3 RECORDS.  FIRST 3 RECORDS ARE THE TMON,
C  TMOFF, AND DATE RECORDS. THE OTHER CDDMAX RECORDS ARE DATA COLLECTED))
C DO FOR CNT=1 TO CDDMAX+3
C      ((HSTARY ELEMENT WITH VALUE 99999.99 REPRESENTS IMPERFECT DATA))
C    IF HSTARY(CNT) GT 9999.00
```

ORIGINAL PAGE IS
OF POOR QUALITY


```
C       THEN SET HSTARY(CNT) TO -1.00 TO REPRESENT MISSING DATA
C       ENDIF
C  ENDDO
C   ((SPLIT HSTARY(3), THE DATE IN THE FORM YYMM.DD, INTO 3 INTEGER
C     VARIABLES))
C  SET IDATE TO THE INTEGER PART OF HSTARY(3)
C  SET MONTH TO MONTH REPRESENTATION OF IDATE
C  SET DAY TO DAY REPRESENTATION OF HSTARY(3)
C  SET YEAR TO YEAR REPRESENTATION OF HSTARY(3)
C   ((IF YEAR IS DIVISIBLE BY 4, IT IS A LEAP YEAR))
C  SET LEAP TO LEAP / 4.0
C  SET REMAINDER, REM, TO REMAINDER PART OF LEAP
C  IF REMAINDER NE ZERO, LOOKING AT A LEAP YEAR,
C  THEN RESET NUMBER OF DAYS OF FEB., CALNDR(2), TO 29
C  ENDIF
C  IF (MONTH LIES IN THE RANGE OF VALID MONTH NUMBERS 1 THROUGH 12)
C   AND (DAY IS LT OR EQ TO THE NUMBER OF DAYS IN MONTH, CALNDR(MONTH),
C     CORRECTED FOR EXTRA DAY IN FEBRUARY DURING LEAP YEARS)
C  THEN WRITE TO SPHISTORY, RECORDS OF HSTARY(CNT), CNT=1 TO CDBMAX+3
C  ENDIF
C  RETURN
C
C
C-----------------------------------------------------------------------
C  CODE
C-----------------------------------------------------------------------
            SUBROUTINE DPMFR4
C
C        **LOCAL DECLARATIONS**
            REAL        LEAP,REM
            INTEGER     CNT,IDATE,MONTH,DAY,YEAR
            INTEGER     CALNDR(12) /31,28,31,30,31,30,31,31,30,31,30,31/
C        **GLOBAL DECLARATIONS**
            INTEGER     SAVDAT,PARTYP,ELKDAT,INTVAL,CPUCNT,PROCNM(4),TOTMEM
            INTEGER     TIMEON,TIMEOF,TIMCNT,REXMEM,EXCMEM,COMMEM,TIPMEM,RELMEM
            INTEGER     NUMDEV,DEMMEM,BATMEM,SWPCNT,BLDATE,PARMS(15),CDBMAX
            INTEGER     NUMCPU,DSKCNT,IN,IN1,IN2,IN3
            REAL        PERFUJ,ELPTIM,SYSIDL,TIPPRG,DEVDAT(99,16)
            REAL        RTMPRG,CMDPRG,BATPRG,PROIDL,HSTARY(102)
            CHARACTER   DEVDEF(99,4)*6,NEXTDY*1,EXLEVL*12,ENDFIL*1
            COMMON      PERFUJ,PARMS,DEVDEF,DEVDAT,NUMDEV,SAVDAT,CPUCNT,ENDFIL
            COMMON      PARTYP,ELKDAT,INTVAL,NEXTDY,ELPTIM,PROCNM,TOTMEM,DSKCNT
            COMMON      NUMCPU,EXLEVL,TIMEON,TIMEOF,TIMCNT,SYSIDL,REXMEM,EXCMEM
            COMMON      COMMEM,TIPMEM,RELMEM,DEMMEM,BATMEM,SWPCNT,TIPPRG,RTMPRG
            COMMON      CMDPRG,BATPRG,BLDATE,PROIDL,HSTARY,IN,IN1,IN2,IN3,CDBMAX
            COMMON      CMDRST
C
            DO 730 CNT=1,(CDBMAX+3)
                IF (HSTARY(CNT).GT.9000.00) HSTARY(CNT) = -1.00
      730   CONTINUE
            IDATE = IFIX(HSTARY(3))
            MONTH = MOD(IDATE,100)
            DAY   = IFIX((HSTARY(3)+0.005)*100.) - (100*IDATE)
            YEAR  = IFIX(IDATE/100)
            LEAP  = YEAR / 4.
            REM   = LEAP - IFIX(LEAP)
            IF (REM.LT.0.001) CALNDR(2) = 29
            IF ((MONTH.GT.0).AND.(MONTH.LT.13)
```

```
     &   .AND.(DAY).LE.CALNDR(MONTH))) THEN
            IF (((CDBMAX+3).GT.16) THEN
               WRITE(IN3,700) (HSTARY(CNT),CNT=1,16)
            ELSE
               WRITE(IN3,700) (HSTARY(CNT),CNT=1,CDBMAX+3)
            ENDIF
            IF (((CDBMAX+3).GT.32) THEN
               WRITE(IN3,700) (HSTARY(CNT),CNT=17,32)
            ELSE
               IF  (((CDBMAX+3).GT.16) THEN
                  WRITE(IN3,700) (HSTARY(CNT),CNT=17,CDBMAX+3)
               ENDIF
            ENDIF
            IF (((CDBMAX+3).GT.48) THEN
               WRITE(IN3,700) (HSTARY(CNT),CNT=33,48)
            ELSE
               IF  (((CDBMAX+3).GT.32) THEN
                  WRITE(IN3,700) (HSTARY(CNT),CNT=33,CDBMAX+3)
               ENDIF
            ENDIF
            IF (((CDBMAX+3).GT.64) THEN
               WRITE(IN3,700) (HSTARY(CNT),CNT=49,64)
            ELSE
               IF  (((CDBMAX+3).GT.48) THEN
                  WRITE(IN3,700) (HSTARY(CNT),CNT=49,CDBMAX+3)
               ENDIF
            ENDIF
            IF (((CDBMAX+3).GT.80) THEN
               WRITE(IN3,700) (HSTARY(CNT),CNT=65,80)
            ELSE
               IF  (((CDBMAX+3).GT.64) THEN
                  WRITE(IN3,700) (HSTARY(CNT),CNT=65,CDBMAX+3)
               ENDIF
            ENDIF
         ENDIF
  700    FORMAT(16(F7.4,1X))
         RETURN
         END
C
C
C
C
```

@PRT,S FNS.DAMFE1

APPENDIX C

SAMPLE REPORT

GENERATED BY PMS2

```
 PPPPPPPP      MM        MM      SSSSSSSS
PPPPPPPPPP     MMM      MMM     SSSSSSSSSS
PP      PP     MMMM    MMMM    SS        SS
PP      PP     MM MM  MM MM    SSS       SS
PPPPPPPPPP     MM   MM   MM      SSS
PPPPPPPPP      MM        MM        SSS
PPP            MM        MM          SSS
PPP            MM        MM            SSS
PPP            MM        MM    SS        SS
PPP            MM        MM    SS        SS
PPP            MM        MM     SSSSSSSSSS
PPP            MM        MM      SSSSSSSS
```

TRACKING AND DATA RELAY SATELLITE SYSTEM

PERFORMANCE MANAGEMENT SYSTEM

NETWORK CONTROL CENTER

MONTHLY REPORT

DISTRIBUTION LIST:

MR. Isley/800.1
MR. Barsky/820
MR. Tagler/830
MR. Spintman/850
MR. Brumberg/870
MR. Laios/820.1
MR. Goodson/820.1
MR. Parker/820.1
MR. Packard/823
MR. Butler/823
MR. Grunby/823
MS. Frazier/BEFC/TDRSS
MR. Goorevich/CSC
MR. Dunham/CSC
DR. Holcomb/MITRE
DR. Kelly/Datametrics
DR. Cleaver/UL
DR. Shelton/UL

PMS OPERATOR:

RAMIN AHOUR

PMS DEVELOPER:

ELECTRICAL ENGINEERING DEPT.
UNIVERSITY OF LOUISVILLE
LOUISVILLE, KY 40292
(502)-588-6289

PERIOD: 08/06/84 TO 08/09/84

REPORT DATE: 10/23/84

TABLE OF CONTENTS AND LIST OF FIGURES

SECTION — PAGE

** - Omitted reports.

C.B

TABLE 5.1 - BUSIEST DAY SUMMARY : 08/07/84

TIME ON (11)	TIMEOFF (12)	TIP THRUPUT (2)	TIP RESPTIM (3)	CPU UTIL. (4)	MEM UTIL. (5)	HIGHEST DSKUTIL (6)	AVERAGE DSKUTIL (7)	HIGHEST FEPUTIL (8)	AVERAGE FEPUTIL (9)	DTV THRUPUT (10)
23.63	.02	.59	28.81	6.50	54.00	7.35	1.07	.25	.14	1.20
.03	.53	.50	33.69	37.04	57.00	17.07	2.88	.23	.11	1.04
.13	.63	.22	64.06	34.00	53.00	16.32	2.75	.22	.09	.84
.63	1.13	.22	64.90	5.08	51.00	1.78	.46	.22	.09	.85
1.13	1.63	.21	59.41	4.22	51.00	1.31	.30	.22	.19	.89
1.63	2.13	.23	60.29	4.46	51.00	1.34	.37	.22	.09	.89
2.13	2.63	.22	64.40	3.94	51.00	1.31	.27	.22	.09	.87
2.63	3.13	.29	49.15	4.83	50.00	1.72	.50	.22	.09	.99
3.13	3.63	.24	59.38	4.18	50.00	1.37	.32	.22	.09	.87
3.63	4.13	.26	53.13	4.40	50.00	1.39	.39	.21	.08	.94
4.13	4.63	.40	35.20	5.57	50.00	3.17	.73	.23	.09	1.13
4.63	5.13	.39	35.80	5.77	50.00	2.77	.80	.21	.08	1.14
5.13	5.63	.50	28.08	6.31	50.00	4.71	.90	.23	.10	1.52
5.63	6.13	.94	15.02	9.16	50.00	5.68	1.17	.26	.14	2.13
6.13	6.63	2.26	6.66	18.21	49.00	11.63	2.11	.37	.18	4.54
6.63	7.13	1.72	8.74	14.19	50.00	5.11	1.14	.32	.17	3.84
7.13	7.63	1.37	11.41	12.49	51.00	2.07	.63	.32	.17	2.87
7.63	8.13	1.96	7.98	17.41	52.00	26.03	2.89	.36	.18	4.16
8.13	8.53	2.16	7.33	15.81	52.00	3.56	.67	.35	.18	4.68
8.63	9.13	2.42	6.59	17.71	52.00	7.71	1.40	.35	.18	5.30
9.13	9.63	2.70	5.57	20.39	51.00	14.27	2.22	.36	.18	5.85
9.63	10.13	2.24	6.48	19.28	50.00	25.10	3.09	.35	.18	4.78
10.13	10.63	1.49	10.09	13.88	51.00	4.93	.97	.35	.18	3.28
10.63	11.13	1.99	7.33	17.60	50.00	6.86	1.26	.48	.20	4.05
11.13	11.63	1.79	8.36	16.83	51.00	3.19	.80	.49	.21	3.78
11.63	12.13	1.71	9.38	14.52	51.00	3.03	.79	.44	.20	3.53
12.13	12.63	2.05	8.63	19.11	54.00	3.70	1.16	.49	.22	4.40
12.63	13.13	1.37	13.65	13.08	56.00	3.72	1.04	.41	.18	2.65
13.13	13.63	.55	34.50	5.67	56.00	2.63	.68	.22	.13	1.17
13.63	14.13	.55	34.73	5.61	56.00	3.64	.72	.22	.13	1.11
14.13	14.63	.57	33.28	6.34	56.00	2.82	.87	.21	.12	1.13
14.63	15.13	.63	30.40	6.18	56.00	3.20	.85	.21	.08	1.26
15.13	15.63	.58	32.68	5.45	56.00	2.47	.61	.22	.09	1.14
15.63	16.13	.60	31.70	7.35	57.00	4.07	.94	.22	.09	1.19
16.13	16.63	.62	30.53	21.31	58.00	8.59	1.67	.21	.08	1.25
16.63	17.13	.64	29.81	16.87	58.00	7.69	1.80	.22	.11	1.30
17.13	17.63	.65	29.21	9.32	57.00	7.76	1.58	.26	.14	1.21
17.63	18.13	.52	36.54	11.30	57.00	3.20	.90	.23	.13	1.03
18.13	18.63	.58	32.62	6.83	56.00	3.20	.82	.22	.13	1.16
18.63	19.13	.45	41.98	4.98	57.00	1.93	.42	.22	.13	.98
19.13	19.63	.42	45.67	4.54	57.00	1.89	.30	.21	.13	.85
19.63	20.13	.55	33.68	18.12	58.00	9.45	1.18	.22	.13	.67
20.13	20.63	.48	37.61	5.81	56.00	2.83	.56	.22	.13	.08
20.63	21.13	.46	39.05	5.06	56.00	1.95	.45	.22	.13	.01
21.13	21.63	.51	35.43	5.03	55.00	1.94	.44	.22	.13	.63
21.63	22.13	.42	40.56	4.54	54.00	1.89	.34	.22	.13	.90
22.13	22.63	.44	39.02	4.50	54.00	1.90	.31	.22	.13	.90
22.63	23.13	.51	32.99	5.08	54.00	2.03	.49	.22	.13	1.00
23.13	23.63	.54	31.21	5.54	54.00	2.57	.62	.23	.13	1.07

******* DENOTES DATA NOT AVAILABLE TO PMS 2...12: SEE SECTION 5 FOR EXPLANATION

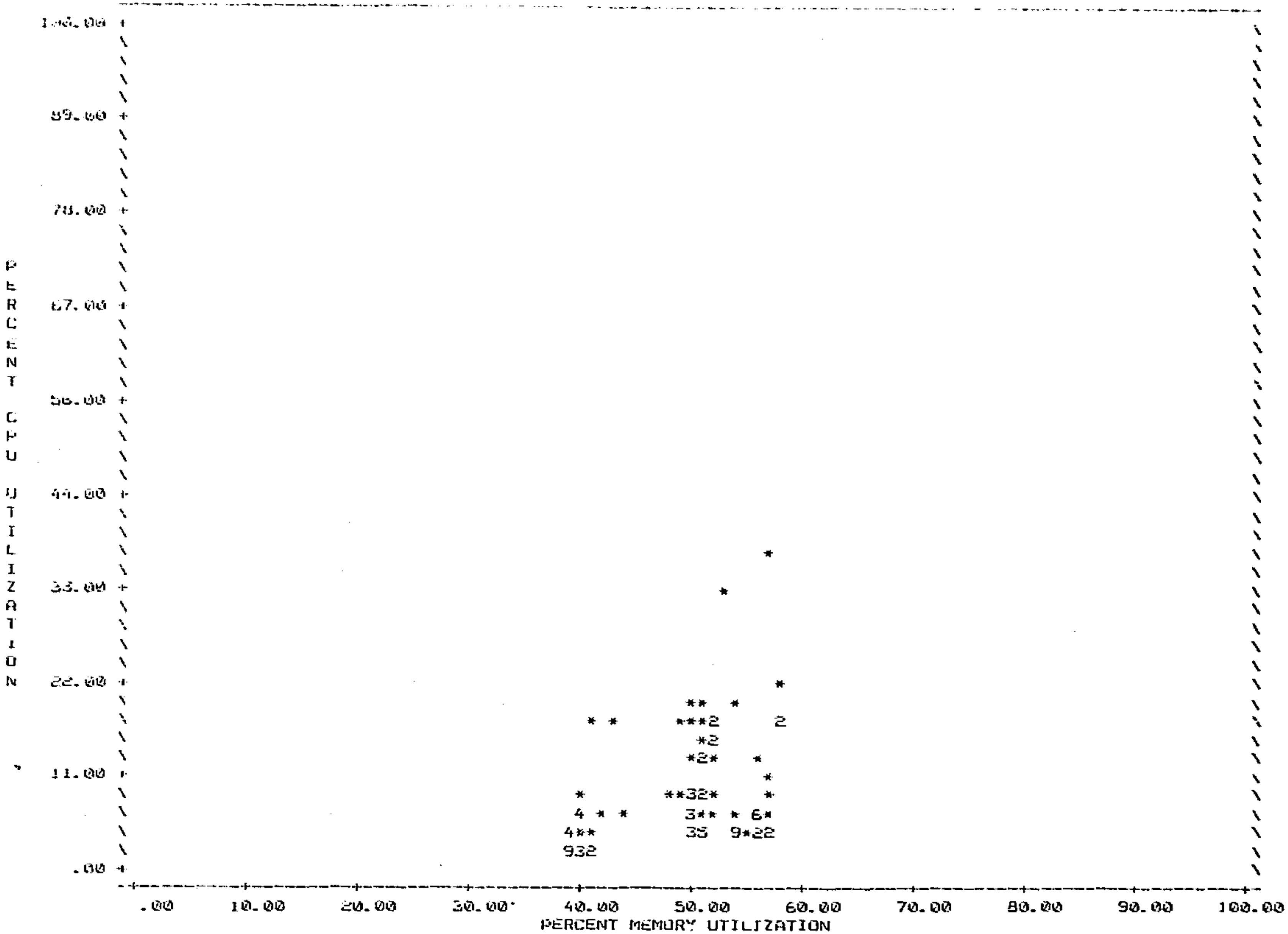
TABLE 3.2

DAILY AVERAGE REPORT

DAYS OF MONTH (1)	TIP THRUPUT (2)	TIP RESPTM. (3)	CPU UTIL. (4)	MEM. UTIL. (5)	HIGHEST DSKUTIL (6)	AVERAGE DSKUTIL (7)	HIGHEST FEPUTIL (8)	AVERAGE FEPUTIL (9)	DTV THRUPUT (10)
8408.06#	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
8408.07	.89	30.67*	10.85*	53.39*	5.22	1.01	.27	.13	1.89*
8408.08	.44	71.84*	5.57*	44.29*	3.47	.61	.25	.11	1.29*
8408.09#	.90	6.55*	17.35*	41.00*	20.72	2.22	.15	.12	.71*

# DENOTES ENTRY WITH LESS THAN 40 SIP BLOCKS OF DATA
*DENOTES THRESHOLD EXCEEDED ENTRIES
-1.0 DENOTES DAYS OR DATA NOT AVAILABLE TO PMS
1...10: SEE SECTION 5 FOR EXPLANATION OF ENTRIES

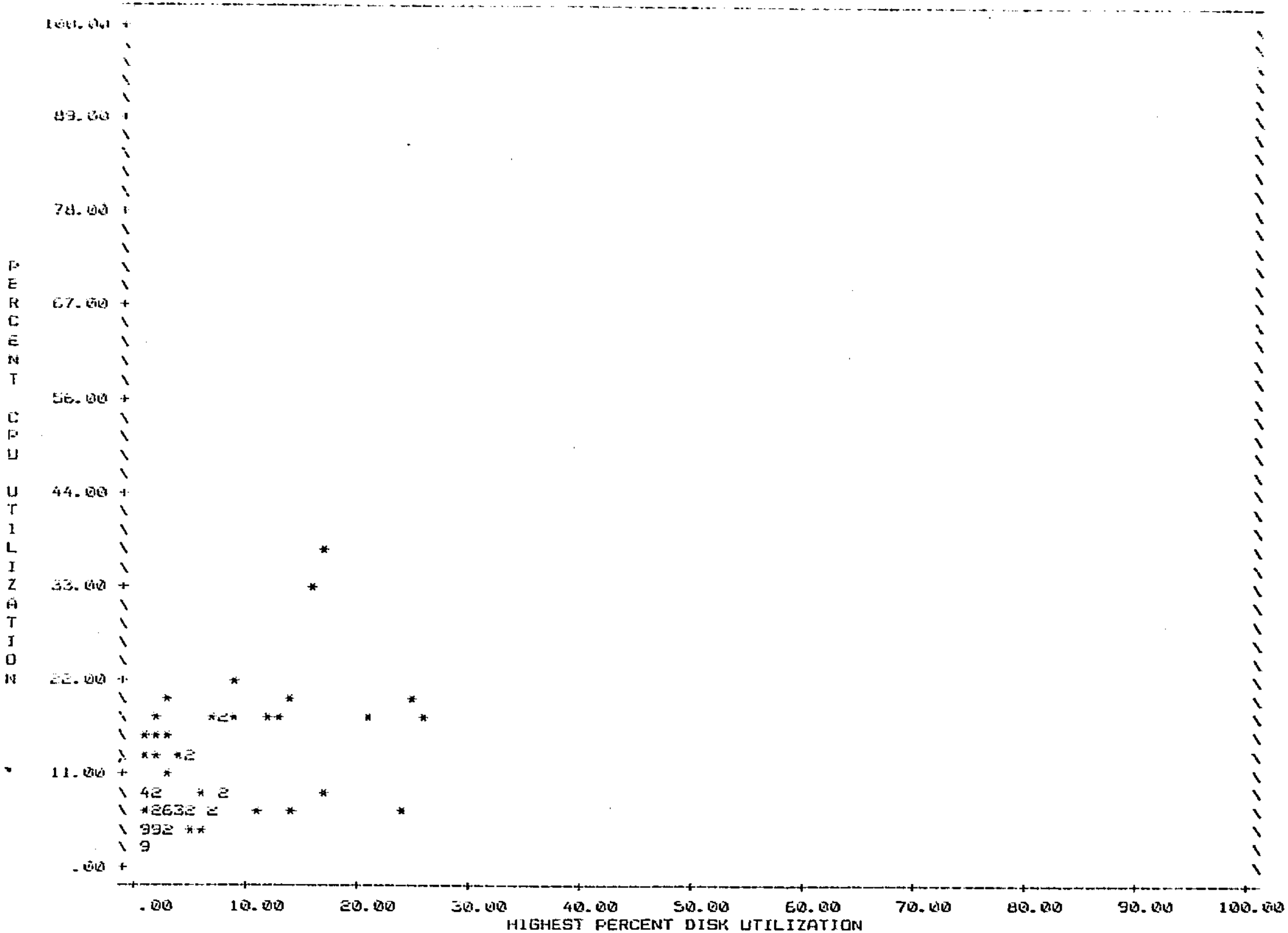
ORIGINAL PAGE IS
OF POOR QUALITY



YMIN= 2.17 YMAX= 37.04 YMEAN= 8.39

XMIN= 39.00 XMAX= 58.00 XMEAN= 48.90

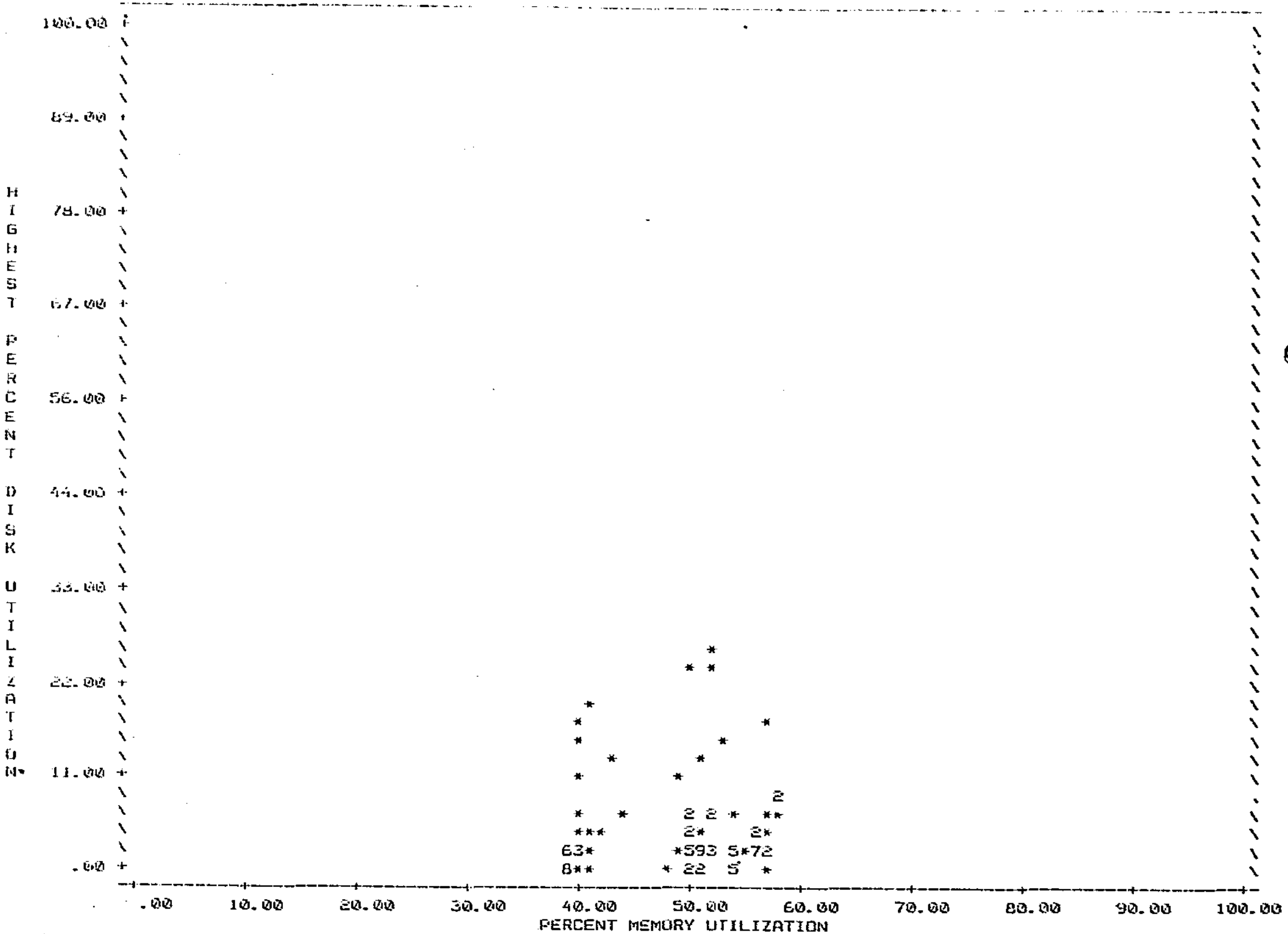
Figure 3.1 - CPU UTILIZATION vs MEMORY UTILIZATION



YMIN= 2.17 YMAX= 37.04 YMEAN= 8.39

XMIN= .84 XMAX= 26.03 XMEAN= 4.54

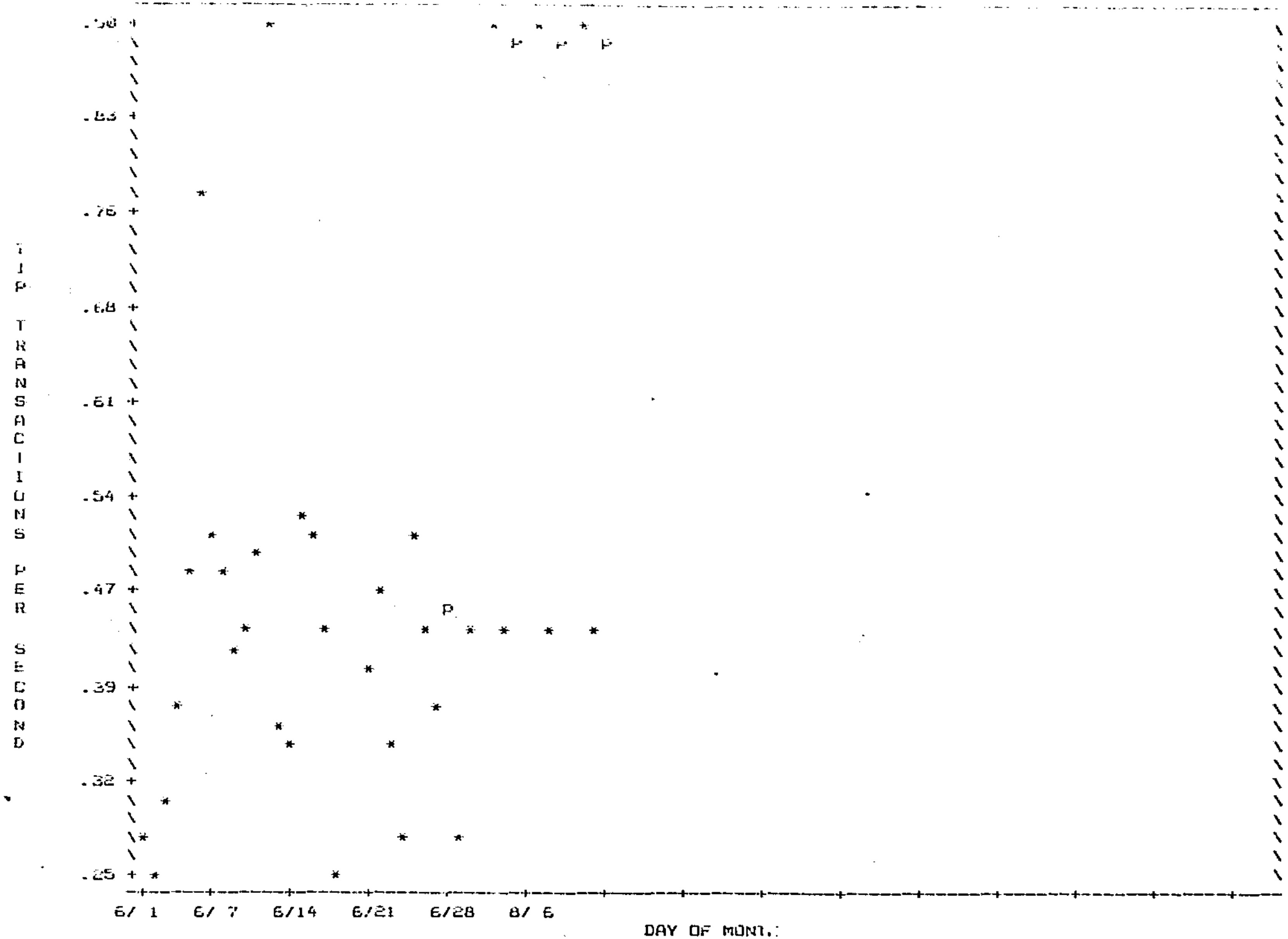
Figure 3.2 – CPU UTILIZATION vs HIGHEST DISK UTILIZATION



YMIN= .84 YMAX= 26.03 YMEAN= 4.54

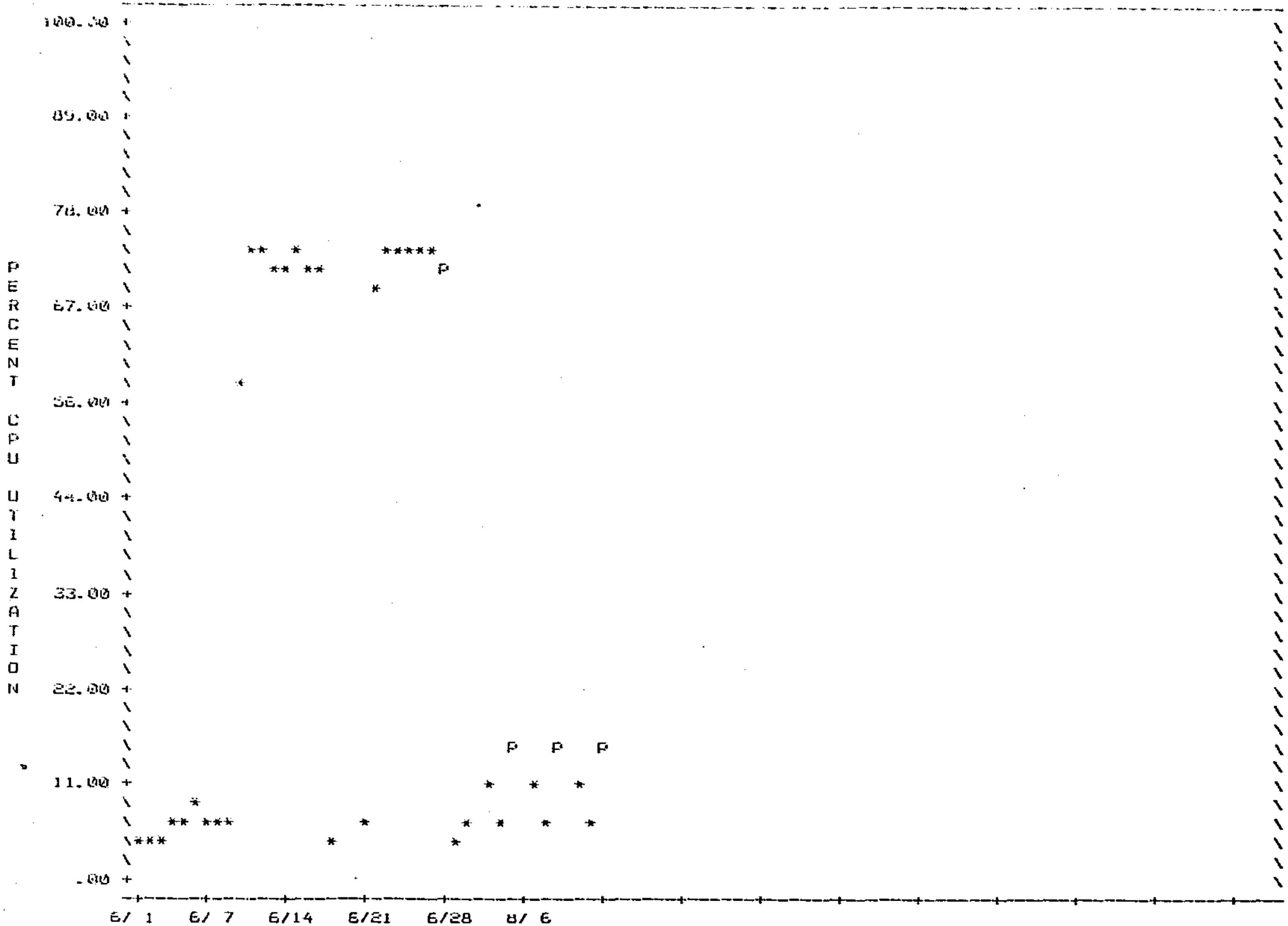
XMIN= 39.00 XMAX= 58.00 XMEAN= 48.90

Figure 3.3 - HIGHEST DISK UTILIZATION vs MEMORY UTILIZATION



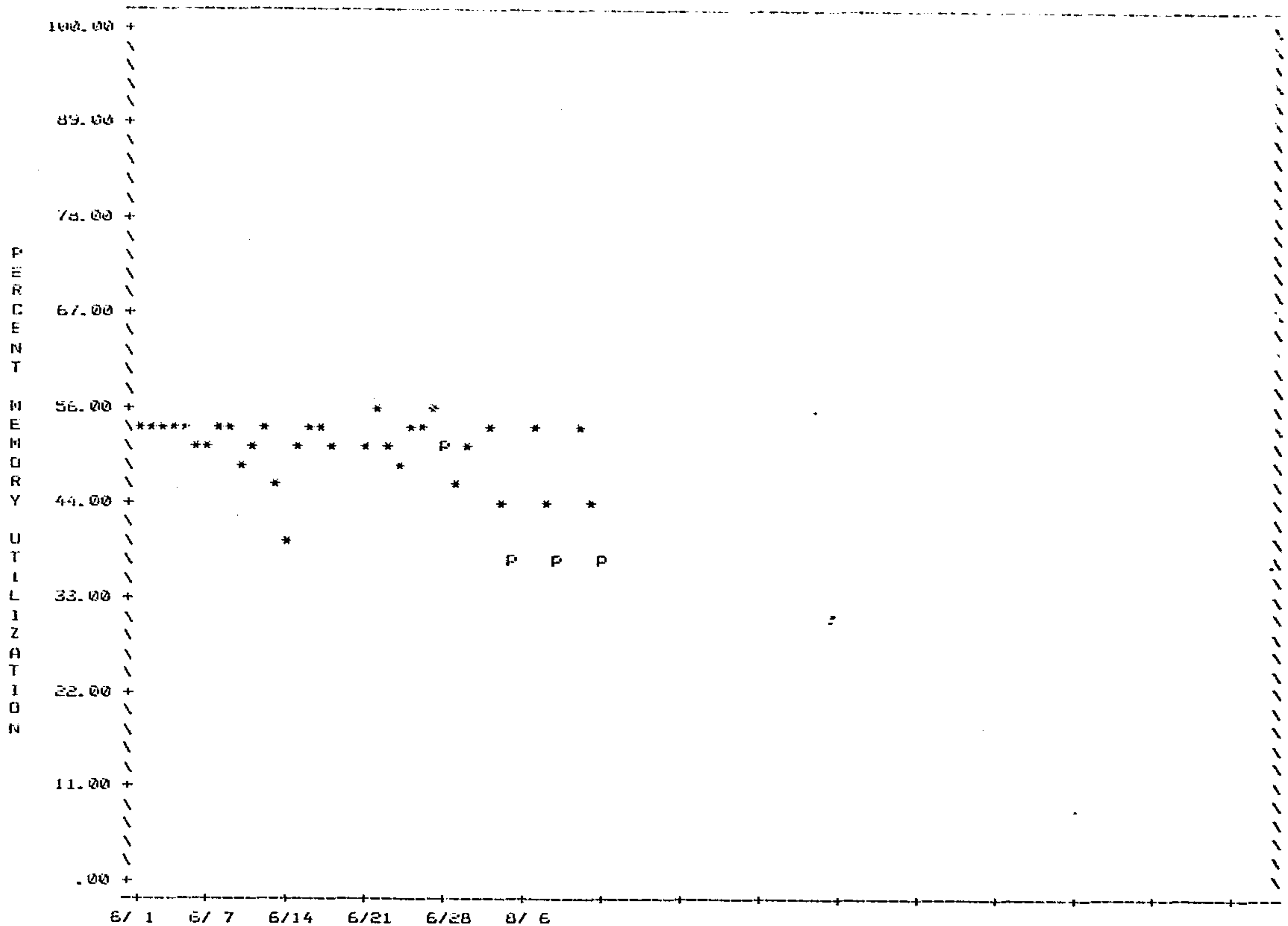
YMIN= .25 YMAX= .90 YMEAN= .45

Figure 3.4 – TIP TRANSACTIONS PER SECOND vs DAY OF MONTH



YMIN= 4.86 YMAX= 73.21 YMEAN= 29.82

Figure 3.5 - CPU UTILIZATION vs DAY OF MONTH



YMIN= 40.90 YMAX= 54.71 YMEAN= 44.33

Figure 3.6 - MEMORY UTILIZATION vs DAY OF MONTH

ORIGINAL PAGE IS
OF POOR QUALITY

10/23/84

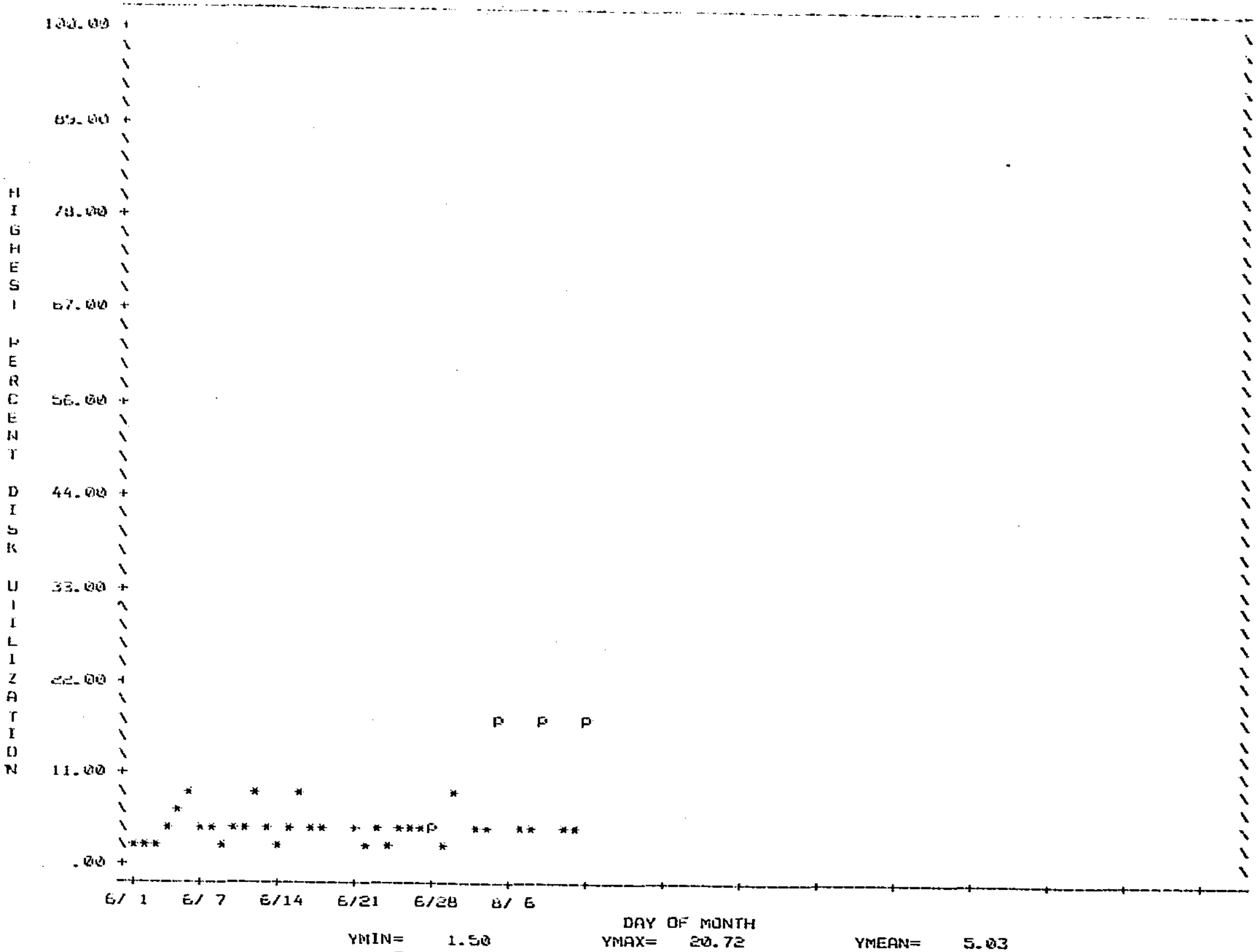


Figure 3.7 - HIGHEST DISK UTILIZATION vs DAY OF MONTH

ORIGINAL PAGE IS
OF POOR QUALITY

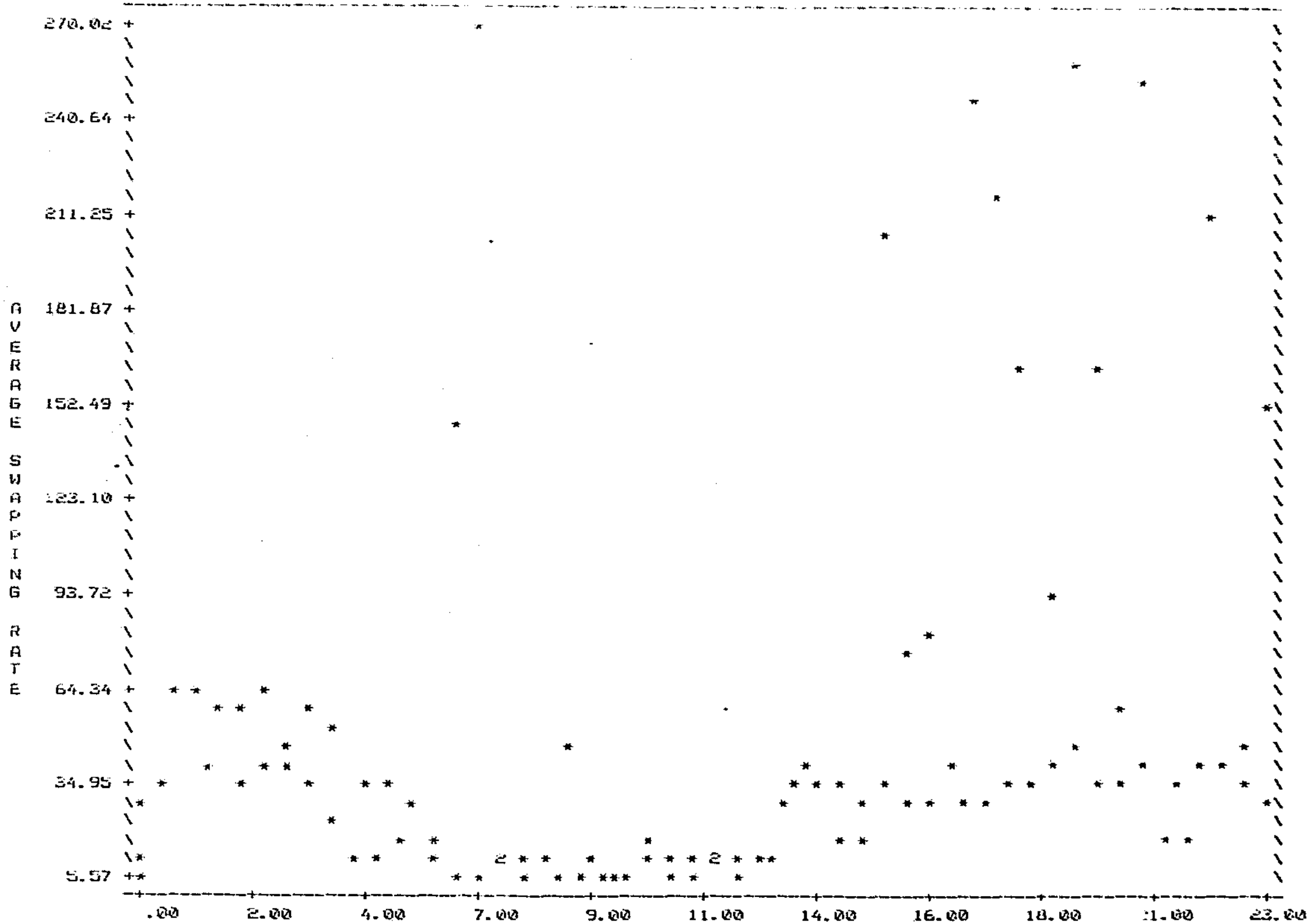
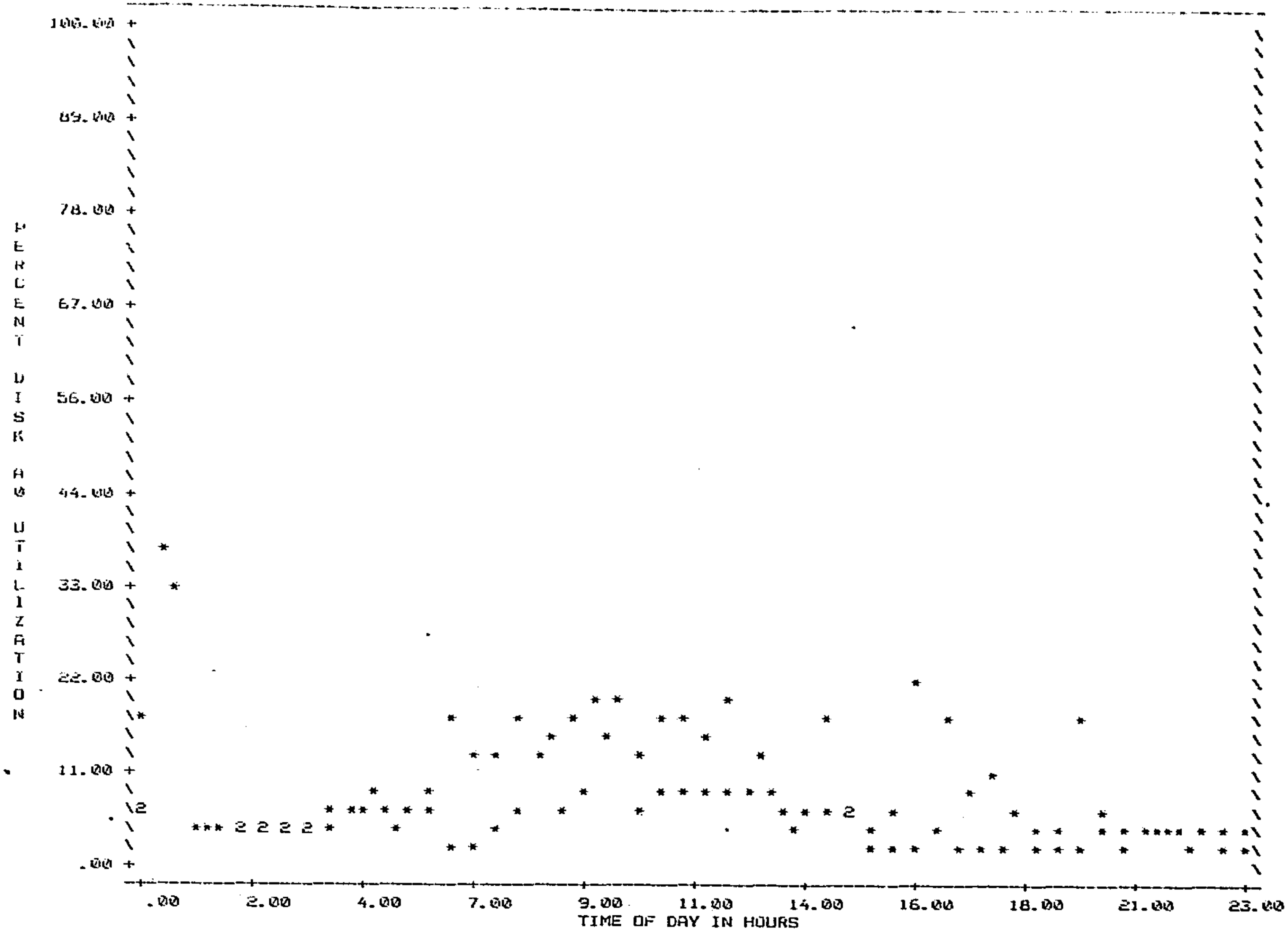


Figure 4.5 - AVERAGE SWAPPING RATE vs TIME OF DAY

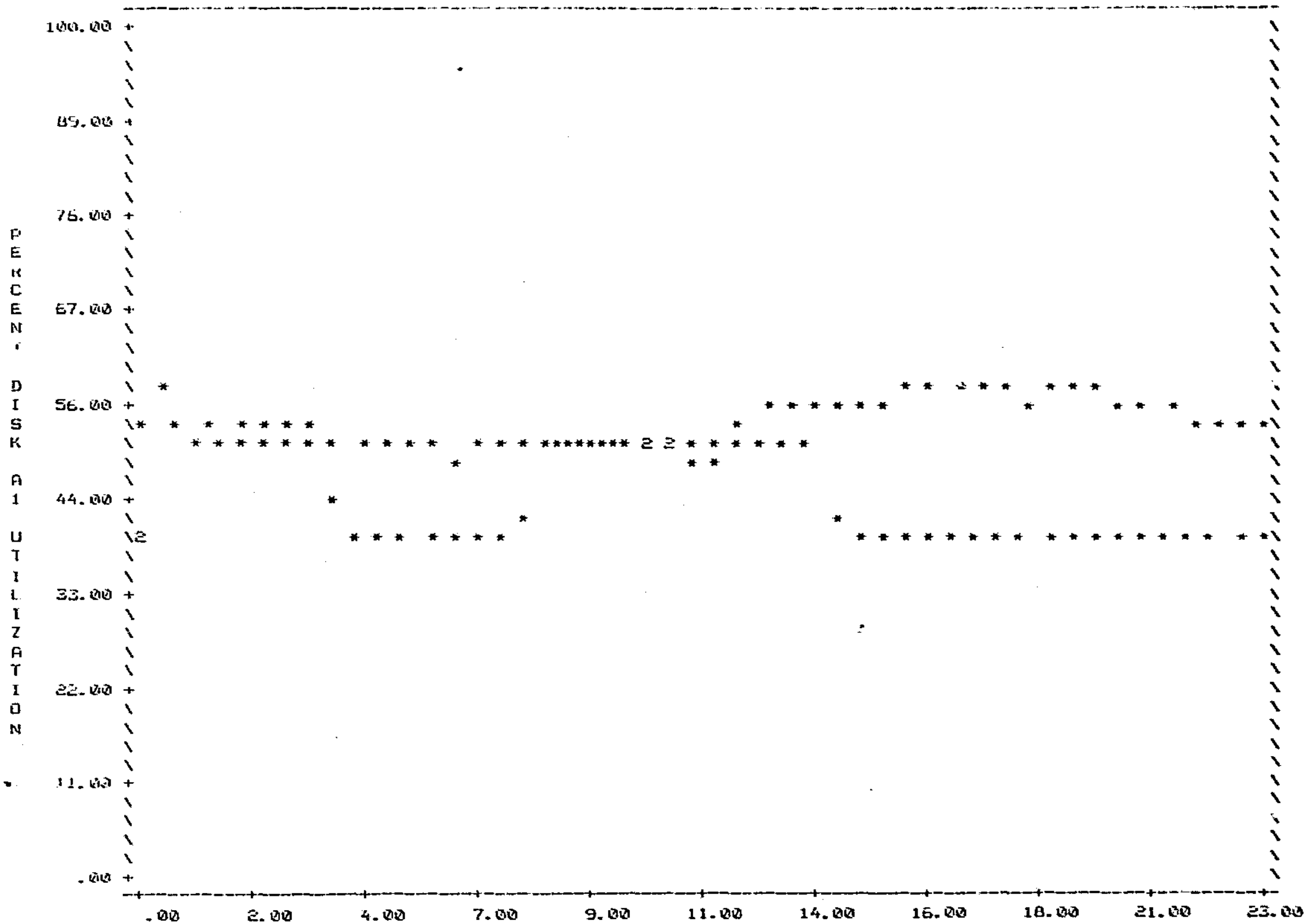
ORIGINAL PAGE IS
OF POOR QUALITY



YMIN= 2.17 YMAX= 37.04 YMEAN= 8.39

Figure 4.7 – DISK A0 UTILIZATION vs TIME OF DAY

ORIGINAL PAGE IS OF POOR QUALITY



YMIN= 39.00 YMAX= 58.00 YMEAN= 48.90

Figure 4.8 - DISK A1 UTILIZATION vs TIME OF DAY

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX D

REFERENCES
AND
BIBLIOGRAPHY

REFERENCES

1. Shive, W. M., Design of a Performance Management System for the TDRTSS NCC. M.Eng. Thesis, Dept. of Applied Math and Computer Science, University of Louisville, May 1983, p. 1.

2. Long, A. M., Performance Forecasting for the TDRSS Network Control Data System. M.Eng. Thesis, Dept. of Electrical Engineering, University of Louisville, August, 1983, pp. 1-2.

3. Ibid., pp. 2-4.

4. Crush, G. G., Statistical Analysis and GPSS Simulations for NASA Network Control Center. M.Eng. Thesis, Dept. of Applied Math and Computer Science, University of Louisville, December, 1982, pp. 34-37.

5. Long, A. M., Performance Forecasting for the TDRSS Network Control Data System, M.Eng. Thesis, Dept. of of Electrical Engineering, p. 4.

6. Darnley, J. G., Computer Performance Measurements on a Sperry UNIVAC 1100 Computer System. M.Eng. Thesis, Dept. of Electrical Engineering, University of Louisville, May, 1983, pp. 11-37.

7. Cleaver, T. G., G. G. Crush, J. G. Darnley, Anna Long, and R. D. Shelton. "A Performance Evaluation Case Study of the TDRSS Network Control Center." Proceedings of the 1983 Computer Measurement Group International Conference, Vol. 14, pp. 105-115.

8. Long, pp. 13-40.

9. Cleaver, T. G., B. R. Johnson, and R. Ahour, Progress Report for the Month of April, 1984, NASA Contract NAS5-26504, Dept. of Electrical Engineering, University of Louisville, May, 1984, Appendix A.

10. Shive, pp. 14-15.

11. Ibid., p. 15.

12. Ibid., p. 17.

13. Shelton, R. D., T. G. Cleaver, L. B. Drake, A. B. Shah, and W. M. Shive, "Performance Measurement with PMS3", Proceedings of the Spring, 1984 USE Conference, April, 1984.

BIBLIOGRAPHY

Cleaver, T. G., B. R. Johnson, and R. Ahour, Progress Report For the Month of April, 1984, NASA Contract NAS5-26504, Dept. of Electrical Engineering, University of Louisville, May, 1984.

Cleaver, T. G., G. G. Crush, J. G. Darnley, A. M. Long, and R. D. Shelton, "A Performance Evaluation Case Study of the TDRSS Network Control Center." Proceedings of the 1983 Computer Measurement Group International Conference, Vol. 14.

Crush, G. G., Statistical Analysis and GPSS Simulations for NASA Network Control Center, M.Eng. Thesis, Dept. of Applied Math and Computer Science, University of Louisville, December, 1982.

Darnley, J. G., Computer Performance Measurements on a Sperry UNIVAC 1100 Computer System. M.Eng. Thesis, Dept. of Electrical Engineering, University of

Shive, W. M., Design of a Performance Management System for the TDRSS NCC. M.Eng. Thesis, Dept. of Applied Math and Computer Science, University of Louisville, May, 1983.

Long, A. M., Performance Forecasting for the TDRSS Network Control Center Data System. M.Eng. Thesis, Dept. of Electrical Engineering, University of Louisville, August, 1983.

Cleaver, T. G., L. B. Drake, A. M. Long, A. B. Shah, R. D. Shelton, and W. M. Shive, Performance Management support for the TDRSS NCCDS. Final Report for Period January-October, 1983. University of Louisville, Oct. 1983.

Shive, W. M., R. D. Shelton, and T. G. Cleaver, "Software Engineering for Small Groups: An Academic Case Study", presented at ACM Computer Science Conference, Philadelphia, Feb. 1984.