A MANAGEMENT ORIENTED DESIGN METHODOLOGY FOR DISTRIBUTED BUSINESS SYSTEMS UTILIZING A RING NETWORK OF MINICOMPUTERS

D378
B644m
1977

By

JAMES LEE BOOCKHOLDT

A DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
the Graduate School of Business
in the Graduate School of
The University of Alabama

UNIVERSITY, ALABAMA

1977

PLEASE NOTE:

Print throughout is broken and indistinct. Appendix contains computer print-out with much indistinct print. Filmed as received.

UNIVERSITY MICROFILMS.

ACKNOWLEDGEMENTS

Because a work of this nature is a distillation of a wide range of education and experience, it is not possible to acknowledge all those who have participated in its completion. However, the contributions of the committee supervising this dissertation should be recognized.

For the direction provided by the Chairman, Dr. J. Donald Phillips, I will always be grateful. Without his support and encouragement the research would never have been completed. I am also indebted to Dr. Joseph M. Mellichamp and Dr. Jerry E. Moore for their criticism and suggestions relating to the methodology employed in the study. The contributions of Dr. Donald Crouch and Dr. John O. Mason in aiding the literature search likewise merit special attention.

I would like to express my gratitude to another faculty member who participated in my graduate education, Professor Charles R. Scott. The guidance which he provided to me as I began a teaching career, and the interest he has shown in my development, will always be remembered and appreciated.

TABLE OF CONTENTS

Page

ACKNOWLEDGEMENTS iii

LIST OF TABLES vii

LIST OF FIGURES ix

Chapter

I. INTRODUCTION 1

Role of the Computer in a Business 1
Technological Trends 5
Minicomputer Networks 7
Research Objectives 13
Research Procedure 15

II. ORGANIZATIONAL PERSPECTIVES 19

Organizational Considerations 20
Technical Considerations 26
Minicomputer Limitations 26
Control of EDP Operations 28
Complexity 31
Financial Considerations 36
Summary 42

III. A TYPICAL BUSINESS APPLICATION 43

The Business Environment 44
Informational Requirements 46
A System Design 51
General Ledger Jobstream 53
Purchase Order Jobstream 54
Material Issues Jobstream 55
Material Receipts Jobstream 56
Payables Jobstream 57
Disbursements Jobstream 58
Billing Jobstream 59
Cash Receipts Jobstream 60
Work In Progress Costing Jobstream 61
Finished Goods Costing Jobstream 62
Order Entry Jobstream 63
Sales Analysis Jobstream 64

Employee Data Jobstream 65
Payroll Jobstream 66
Scheduling Jobstream 67
System Files 68
Summary 68

IV. COMPUTER SYSTEM SIMULATION 70

Difficulties 71
Complexity 72
Level of Aggregation 72
Data Sources 74
Misinterpretation of Results 79
Objectives of Simulation 80
Computer Network Models 86
Summary 89

V. A GENERALIZED RING NETWORK SIMULATOR 90

General Description 91
The Operating System 91
The Peripheral Devices 93
The Processor 94
Data Input 97
Computer System Description 98
Jobstream Description 100
Model Construction 104
Macro Construction 107
Control Segment Construction 107
Model Output 116
Performance Measures 116
Standard GPSS Output 118
Summary 120

VI. DESIGN PROCEDURE 122

Design Alternatives 123
Network Size 123
A Network Size Heuristic 126
File Locations 128
A File Location Heuristic 130
Equipment Choices 132
An Application of the Procedure 134
System Description 134
System Tests 138
Summary 140

VII. RESULTS AND CONCLUSIONS 145

Design Procedure Results 146
Network Size 146
File Location 152
Peripheral Equipment Choices 159

Sensitivity to Workload Changes 163
Validation 164
Interpretation of Results 167
Conclusions 170

BIBLIOGRAPHY 172

APPENDIX 1: NETWORK SIMULATOR LISTING 188

APPENDIX 2: SAMPLE MODEL OUTPUT 223

APPENDIX 3: JOBTAPE WRITE PROGRAM LISTING 238

LIST OF TABLES

Table		Page
1	Core Usage, Dual IBM 370/165 Processors	35
2	Agee Company, Transaction Volume	47
3	Agee Company, System Files	69
4	Simulation Characteristics at Different Levels of Aggregation	75
5	Management Information System Life Cycle	82
6	Computer System Description, Input Data	99
7	Job Stream Description, GPSS Transaction Parameters	102
8	Job Stream Description, Parameter Specification Codes	103
9	Model Construction, System Component Designations	106
10	Initial Network Topology, Ten Processors	135
11	Agee Company Hardware Description	137
12	Example Jobstream Characterization, Purchase Order Jobstream	139
13	Agee Company Serial Printer Evaluation Hardware Description	141
14	Agee Company Line Printer Evaluation Hardware Description	142
15	Agee Company Floppy Disk Evaluation Hardware Description	143
16	Network Performance Measures, Eight Daily Jobs	149
17	Network Performance Measures, Fourteen Weekly Jobstreams	153

18 Initial Network Topology
Six Processors 157

19 File Relocation Tests 160

20 Optimal Network Topology
Six Processors 161

21 Peripheral Device Selection Tests 162

LIST OF FIGURES

Figure		Page
1	Information Flow - Manual Systems	3
2	Information Flow - Centralized Computer Systems	4
3	Basic Network Configurations	9
4	Input/Output Relationships in a System Model	16
5	Knight's Economies of Scale	38
6	Cost Comparison, Large Scale Versus Mini Processors	41
7	General Ledger	53
8	Purchase Order	54
9	Material Issues	55
10	Material Receipts	56
11	Payables	57
12	Disbursements	58
13	Billing	59
14	Cash Receipts	60
15	Work In Process Costing	61
16	Finished Goods Costing	62
17	Order Entry	63
18	Sales Analysis	64
19	Employee Data	65
20	Payroll	66

21 Scheduling 67

22 Single Site Configuration 95

23 Macro Construction, Print Macro 108

24 Macro Construction, Input Macro 109

25 Macro Construction, Read Disk Macro 110

26 Macro Construction, Write Disk Macro 111

27 Model Construction, Control Segment Interpretation and Processor Functions . . 113

28 Model Construction, Control Segment Communications Function 115

29 Network Size Algorithm 127

30 File Location Algorithm 131

31 Agee Company, Data Processing Workloads 148

32 Network Performance, Turnaround Time versus Size, Eight Daily Jobstreams 150

33 Network Performance, Throughput versus Size, Eight Daily Jobstreams . . . 151

34 Network Performance, Turnaround Time versus Size, Fourteen Weekly Jobstreams 154

35 Network Performance, Throughput versus Size, Fourteen Weekly Jobstreams . 155

CHAPTER I

INTRODUCTION

Many organizations, particularly those with a decentralized structure, have found that a centralized data processing department is an expensive and inefficient means for providing the types of information required for effective management [155]. The need has arisen for locally based information processing in such organizations.

Role of the Computer in a Business

The potential of the computer for providing information useful to management began to be realized in the 1950's and was first utilized in accounting applications. The success in this area, along with the apparent power of the computer to process large amounts of data routinely and efficiently, led to applications in other functional areas of business. Applications were developed in marketing, production control, materials management, and product engineering. Large scale systems were developed which utilized the computer to assume many of the clerical tasks associated with operating a business. Later these systems were expanded to accumulate types of information which were not feasible with manual systems.

However, this computational power was expensive to establish and to maintain; such systems provided new forms of overhead which had to be borne by the revenue-generating segments of an organization. The technological capabilities required by the new information processing medium dictated the establishment of large staffs of systems specialists, programmers, and key punch operators. Thus, managers became concerned with methods of achieving a high level of data gathering and information processing capability in a cost-effective manner. The most efficient way of supporting the costs of specialized equipment and personnel was by establishing a centralized data processing branch in the organization. This branch, usually a separate department, accepted inputs of raw data from the functional areas of the business and provided them in return with information in a form useful for making decisions at the functional level. Such a centralized data processing department interrupted the previously existing data accumulation and summarization procedure within the functional area. The process of converting raw data to information was taking place external to the functional area which generated the data and utilized the information. (See Figure 1 and Figure 2).

The demands placed on such a centralized data processing branch by the several functional departments in a business results in its frequent failure to perform satisfactorily for any of them. The needs of managers in different functional areas may conflict, making it

FIGURE 1

INFORMATION FLOW - MANUAL SYSTEMS

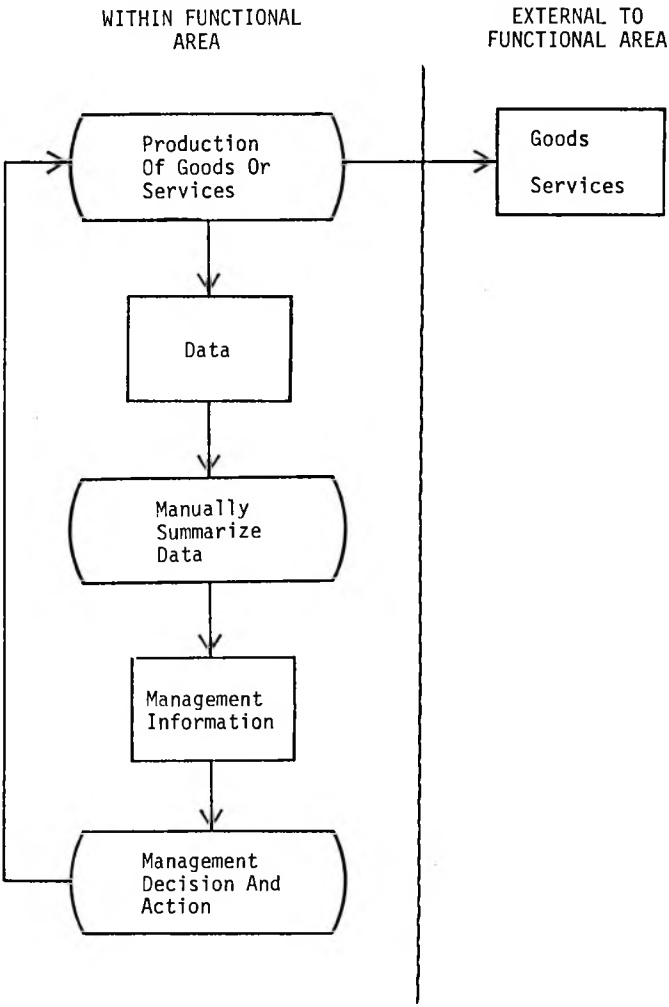
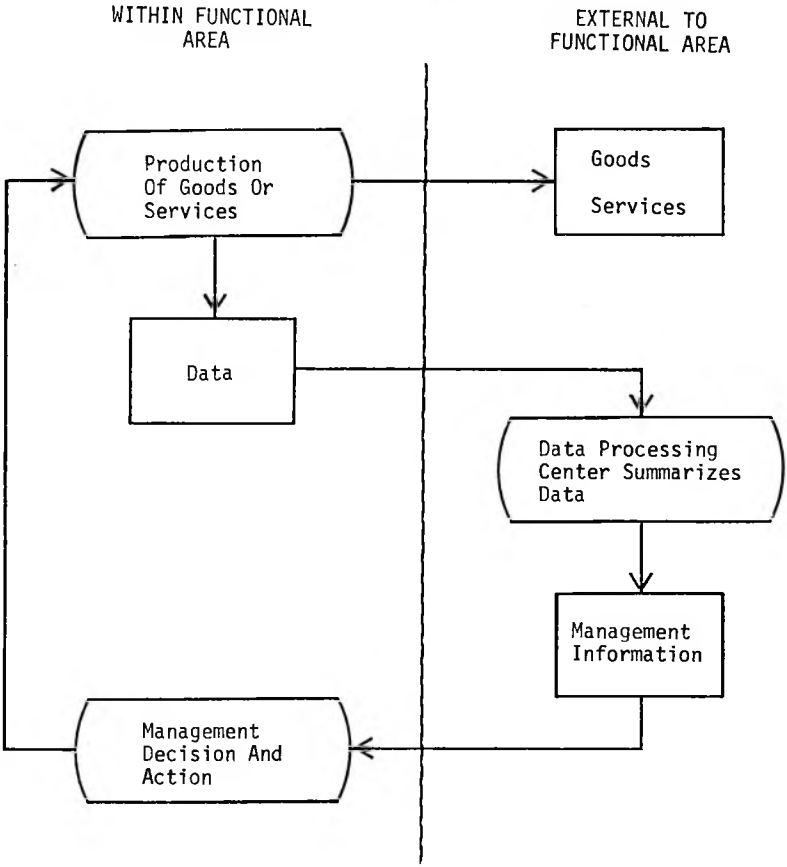


FIGURE 2

INFORMATION FLOW - CENTRALIZED COMPUTER SYSTEMS



impossible to meet all of their needs simultaneously. As systems become more extensive in their application, their complexity increases. The results are increased costs and the further removal of information processing from functional user areas. The economies of scale resulting from a centralized data processing department in an organization were, however, usually considered sufficient to justify the ensuing perturbations to the organization's structure and to the information flow in it. These and other problems resulting from the centralized structure will be discussed more fully in Chapter II.

Managers and data processing professionals alike have long recognized the problems associated with the centralized structure. Until recently, however, technological constraints have made it the most acceptable alternative in terms of costs and benefits.

Technological Trends

Changes in the technology of computing have taken place at a remarkable rate since the introduction of computers to business applications in 1951. Trends in these changes have significant implications for the types of information processing systems which businesses can be expected to employ in the future. Three factors have been noted which reflect these trends [159]:

1. Users are attempting to implement information systems of steadily growing complexity;

2. The cost of personnel has been rising rapidly; and
3. The cost of most hardware, particularly electronics, has been dropping even more rapidly.

As personnel costs and system complexity increase, so will the costs of software development and system malfunctions. And as the costs of processors and components decrease, it will become more feasible for an organization to provide duplication in its data processing facilities. Thus, the economies of scale which justified the creation of large data centers servicing many functional areas of an organization can be expected to disappear. It will be possible for a business to establish computerized information systems whose facilities are dedicated to a particular functional area. An accounting system, for example, would be able to operate independently of a production control system, without many of the conflicts associated with a large data center. It may become common for an organization to have, instead of a single large data processing department, a number of smaller units operating independently in different functional areas. The process of converting raw data into information would then take place within the area generating the data and using the information. These computerized systems would be more responsive to the needs of the users of the information, and would be less limited by the constraints of a centralized

department which must service many users.

The primary technological change which makes this type of decentralized information processing feasible is the availability of low-cost small computers. Minicomputers have been used for several years in dedicated applications relating to process control. More recently minicomputer systems have been developed for use in data processing applications. They may function independently as stand-alone systems or in conjunction with, and as extensions of, large scale machines. As the technology relating to microprocessors and large scale integration develops, the cost of such systems can be expected to decrease as their processing power increases. Thus a business may utilize a number of minicomputers in each of its functional areas to provide more timely and relevant information to its decisions makers than is available with a centralized system. And this could be accomplished at costs which compare favorably with the centralized system.

Minicomputer Networks

Many of the advantages of the centralized data processing structure would be lost with decentralization if it were necessary to maintain duplicate computer files at each minicomputer location. By connecting each of the functionally dedicated systems into a network, each can create and maintain its own unique files and still make these files accessible to other systems in the network.

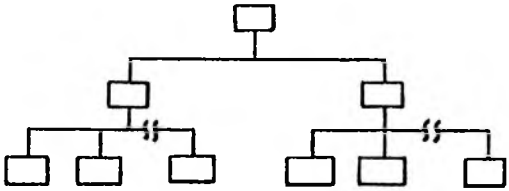
Although certain files may be used by several minicomputers, only one such file need be maintained. Thus, all the data files existing on the network would form a single data base, even though information processing capability is distributed among several locations corresponding to different functional areas of the business.

Computer networking is a technology of recent origin. The first experiments in this area were funded by the Advanced Research Projects Agency of the United States Department of Defense. The result, called ARPANET, was first demonstrated publicly in 1972 and currently links over 100 large scale computers in locations throughout the United States and Europe. Minicomputers have frequently been utilized in networks with large scale machines as front-end processors or communications controllers. However, networks composed entirely of minicomputers are less common. Such an experimental network was constructed at the University of California at Irvine beginning in 1972; others have been implemented by a number of banks and by a building supply company [27] [85].

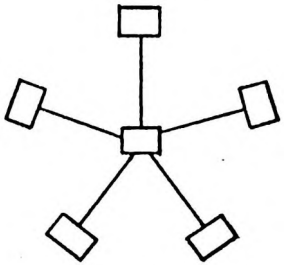
Minicomputer networks may exist in one of three basic configurations: A hierarchical (tree) network, a star network, or a ring (see Figure 3). The hierarchical and star networks are frequently employed when minicomputers are used with large-scale processors. A hierarchical network has been developed in a project at the Institute for Computer Research at the University of Chicago [4]. This

FIGURE 3

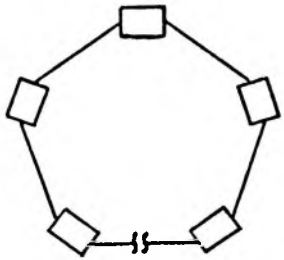
BASIC NETWORK CONFIGURATIONS



HIERARCHICAL



STAR



RING

system consists of three levels. The highest level is an IBM 370/168 which is the University's central computing facility. The intermediate level utilizes a Digital Equipment PDP 11/45 which provides operational and developmental support to the lowest level in the hierarchy. It includes input, output and mass storage facilities which do not exist at the lowest level, and interfaces the lowest level with the central facility. It also may be used for such services as program loading, resource allocation and scheduling, data sorting, and assembling or compiling programs. However, no user programs execute at the intermediate level.

The lowest level of the Chicago network is composed of minicomputers of a variety of types. Each is under control of its user at a number of local sites and is used in dedicated real-time applications relating to the monitoring and control of laboratory experiments.

Star networks are commonly used in business and scientific applications whenever a minicomputer functions as a communications controller or front end processor for a large mainframe computer. In such applications the minicomputer may interface with a number of remote terminals. It serves as a data concentrator and enables more efficient use of the large central processor by accomplishing many of its input and output functions [40]. A star network may also be constructed entirely of minicomputers. In this case the central node processor is

also a minicomputer and provides message switching functions to the other processors in the network. Thus it serves primarily as a communications medium between other computers which operate in dedicated applications.

From a managerial point of view, both hierarchical and tree networks suffer from a major disadvantage. Should a malfunction occur in the node or intermediate levels, all or part of the network is inhibited from functioning in a normal manner. In a star, all message traffic between processors comes to a halt until the node processor is again operational. In a hierarchical network, the support functions provided by the intermediate level processor are not available to the lower level computers connected to it. Thus these forms of distributed processing may not provide the reliability which is necessary in most commercial applications. The type of network which avoids these problems, and thus offers the most advantages for business distributed processing, is the ring network.

A ring consists of a series of processors, each of which is directly connected to only two other processors in the network. For a processor to communicate with a nonadjacent one in the network, as it would do in accessing a remote data file, a message must be relayed from processor to processor around the ring until the destination processor is reached. Then the destination would initiate an acknowledgement which would be relayed back to the originator.

The ring configuration in a distributed processing network offers a number of advantages over the more traditional centralized structure. One advantage, to which references have already been made, is functional dedication. Each minicomputer in a ring could be dedicated to applications concerning a single functional area of the business. Additionally, the ring structure provides for increased reliability. Should a processor in the network become inoperative, that processor can be bypassed and its processing load assumed by another in the ring. Because additional applications can be added to the system simply by adding additional processors to the ring, such a structure also provides increased flexibility. A final advantage, and one which perhaps is primary in organizations highly dependent on computerized systems, is cost. Both in initial cash outlay and in its effects on an organization's efficiency, a network of minicomputers can be expected to cost significantly less than a large multiprocessor of equivalent processing capability [82] [87].

A ring network of minicomputers is, of course, not without its disadvantages. In the most economical minicomputers, logic switching and data access times are longer than in large scale processors; thus computational speed can be expected to be slower. This would be a significant disadvantage in computers used for scientific applications. In most business data processing situations, however, throughput is limited by input/output rates. Thus,

the decreased computational speed is not a serious limitation. Data security is a problem which has not been satisfactorily corrected in distributed networks of minicomputers. It is undesirable to have every file on the network accessible from each processor. With additional experience in network design, methods of avoiding this problem should be developed. Finally, because the concept of distributed processing is a relatively new one, neither sufficient software support nor a breadth of design experience exists for such networks. As more networks are implemented, more software is expected to become available. The lack of design experience in business distributed processing networks is the problem toward which this research is directed.

Research Objectives

As has been indicated, distributed ring networks offer significant advantages in business information processing applications over more centralized structures. However, such networks have been fully implemented only on an experimental basis [2] [29] [47]. An organization considering the installation of a ring network does not have available the experiences of prior users in evaluating the suitability of this structure to its own needs. Whereas the design of centralized data processing systems has become fairly standardized, model applications systems do not exist for distributed ring networks. Nor do organizations have a

basis for determining the minimum hardware configuration required for acceptable network performance. Those organizations currently engaged in the development of ring networks must rely heavily on the advice of equipment vendors, whose objectives may be directed more toward achieving sales rather than optimal resource allocation. A method is needed by which organizations can evaluate alternative systems applications and configurations without the actual purchase and installation of the system. One such method relies on a simulation model of the system.

Using simulation in computer system design is not a new concept. Frequently the operations of special purpose computers, such as those in missile guidance systems, have been analyzed using the technique. Other computer system models, more closely related to commercial applications, will be discussed in Chapter IV. A number of generalized simulation packages exist to aid in the design of large-scale general applications computers [38]. However, these models are not intended for the design of minicomputer networks. A generalized model is needed which can be used by an organization to evaluate minicomputer systems design alternatives. The objectives of this research are to define a set of applications which are common to many businesses, and to develop a simulation model which can be used to evaluate the implementation of these applications on a minicomputer ring network.

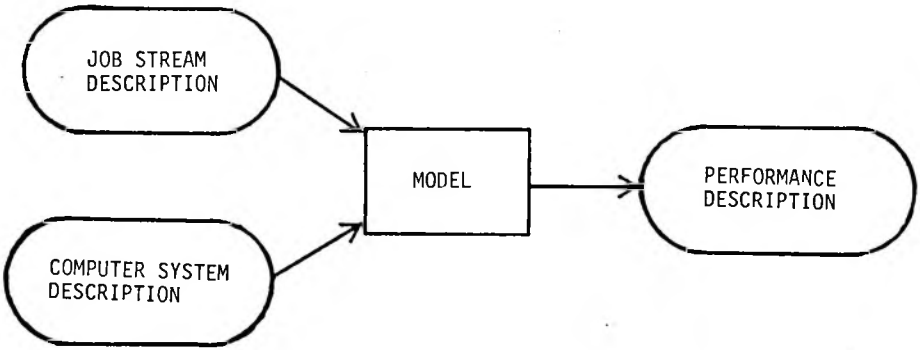
Research Procedure

In developing such a model, it will first be necessary to define a common set of business applications. These represent the uncontrollable variables in the execution of the model. Then it will be necessary to specify a proposed hardware/software configuration on which the applications can be implemented. These are controllable variables, subject to the decision of the organization considering the system. Finally, a set of system evaluation criteria will be designated for measuring the effectiveness of the configuration in implementing the specified applications. The results of the simulation model, as measured according to these criteria, will provide the means of evaluating any particular system under consideration.

Applications which are common to most businesses are those which are normally found in any centralized data processing installation. These include order entry, payroll, general accounting, payables, receivables, purchasing, inventory, and production control. An organization considering implementing them on a distributed system must make a number of decisions. These include the number of minicomputers to employ, which applications should share a processor, and at which processor appropriate data files should be located. These decisions will all affect the desirability of any proposed hardware configuration and data transfer mechanism. Hardware alternatives include the number, types, and location of devices; the type of ring

FIGURE 4

INPUT/OUTPUT RELATIONSHIPS IN A SYSTEM MODEL



interface processor to employ; and the method of data communication to utilize between interface devices.

Computer scientists have devoted a great deal of research to determining the most appropriate system evaluation criteria. There is no universally accepted evaluation procedure. Most researchers agree that one or more of the following measures are of interest to the user. These are: (1) Throughput, which measures the steady state work capacity of the system; (2) Turnaround time, defined as the delay between presentation of input to a system and the receipt of output from it; and (3) Availability, the percentage of time that a system is operating properly [24]. In addition, a cost efficient system is one which exhibits an acceptable level of utilization. Minimum utilization criteria which have been used are 30% for a processor and 70% for a memory device [17].

A simulation model useful for comparing alternative systems design strategies is one in which the controllable variables can be readily altered. These represent the decision alternatives available in the simulated system. A commonly used approach utilizes as inputs to the model a characterization of the job stream and a separate description of the hardware characteristics (see Figure 4). Altering either set of inputs provides comparative performance measures for the system being modeled [89]. Such models which have been developed for use in configuring centralized systems include SCERT (Comress, Inc.), CASE

(Tesdata Systems Corp.), FCSS (Rand Corp), and CSS/II (IBM Corp.). The minicomputer network model in this research was constructed in UNIVAC GPSS 1100.

In summary, ring networks of minicomputers promise to become a cost-effective and efficient means to satisfy the information processing needs of many organizations. As will be discussed further in the next chapter, such a network will avoid many of the disadvantages of a centralized data processing department. A typical business situation will be described in Chapter III for which a minicomputer network is a viable data processing alternative. A simulation methodology is needed by which an organization can evaluate a system without actually implementing it. However, little experience exists in the design of such networks. Chapter IV contains a discussion of the difficulties encountered in building simulation models of computer systems. A network model is presented in Chapter V which overcomes many of these difficulties. Then in Chapters VI and VII, the model is used to design a distributed minicomputer system for the business situation in Chapter III.

CHAPTER II

ORGANIZATIONAL PERSPECTIVES

The arguments both for and against the decentralization of an organization's data processing capability have been widely discussed since the mid 1960's [97] [126] [144]. Some researchers concluded during the latter part of that decade that a distinct trend was emerging toward centralization [36, p. 88] [157, p. 68]. Many companies which have implemented this approach, however, have determined that it is not always a good one [155]. Networks of minicomputers, in which the processing capability is distributed among a number of locations, is considered a viable alternative to the centralized form. It is the objective of this chapter to consider the arguments both for and against a decentralized structure utilizing minicomputers.

Some of the problems associated with a centralized data processing department have already been cited. In general the considerations in evaluating the most appropriate structure fall into one of three categories: Organizational, technical, and financial.

Organizational Considerations

In the centralized data processing structure, a suborganization exists within a company which provides data processing services to the functional areas of the business. A major portion of the information processing exists external to the operating departments. This is considered advantageous in those companies which have a centralized operating philosophy because it provides greater control by top management over the organization. Data are gathered within the operating areas and are summarized by the central computer facility. Information for decision making is then readily available to the decision makers who themselves are centralized.

As an organization grows, the effectiveness of a top management which seeks to control all the activities of the company becomes limited. Most organizations have found that it eventually becomes desirable to disperse decision-making authority by assigning it to operating managers. In this case a number of reasons exist why decentralized data processing is also appropriate.

One of the major advantages of decentralized decision-making is the training it provides for operating managers. Giving such managers control of their own information processing familiarizes them on a first hand basis with the strengths and limitations of computerized systems. This provides valuable training for future top executives who will be heavily dependent on the information

system for making major decisions. It also gives the lower level managers greater autonomy over their operating areas, creating a more distinct profit center. Because there is a more accurate measure for evaluating performance, the managers are provided with increased motivation and greater initiative for improvement [67, p. 12].

Proponents of decentralized decision-making also argue that it results in better decisions. The operating managers are considered to be more capable of making good decisions because of their proximity to the problems and day-to-day activities of the operating units. Yet better decisons result only when adequate information is available to the decision maker. Often this is not the case when the information is provided by an organizational unit external to the operating area.

A common reason why a centralized data processing department may not serve the information user adequately relates to the timeliness of the data. Using conventional data entry via keypunch, the information cycle typically consists of the following actions:

1. Operating area fills in a form,
2. Form is sent to the keypunch room,
3. Keypunch checks, punches, and verifies it,
4. Data are fed into the computer,
5. Invalid data are returned to keypunch and/or the user department for correction,
6. Data are summarized and printed in report form,

7. Report is forwarded to the manager in the user department.

Such a procedure seldom consumes less than one day, and normally requires much longer [141, p. 397]. As a result the information and the files containing it are often out-of-date before the information is available to the operating manager. Using a functionally dedicated minicomputer in the operating area, the data could be recorded, entered in the computer, validated, and summarized simultaneously. It could be reported to the manager more frequently, resulting in more timely information. Furthermore, such an on-line system typically requires less training for an operator, is more accurate, and provides better control as a result of focalized responsibility [87, p. 181] [127, p. 91].

Another reason that centralized data processing may be deficient in a decentralized organization concerns the level of service provided the user. Every user operating area has a unique set of needs which may differ significantly from those of other functional areas in the same company. Production control and engineering data processing applications, for example, are utilized extensively in the daily activities of these functional divisions. Such applications are representative of the overall management control system implemented by a particular manager and may differ depending on managerial style, sophistication, and desires [127, p. 91]. Therefore

such information systems should be expected to differ from factory to factory as well as from other functional areas in the business. When the information processing responsibility is located within the functional area, it is more easily integrated into the operating systems. It can be expected to be more responsive to the manager's needs for information. The result is improved service to the user areas and less conflict between the users and the providers of informational services [67, p. 12]. The sheer size of a centralized data processing organization often makes such service very difficult to achieve [127, p. 93].

The conflicts which often arise between user areas and a centralized data processing facility are a major problem whenever this organizational form is utilized. An operating division which is heavily dependent on the data processing services provided by a central facility easily becomes dissatisfied with that service when it is inadequate or when requests for modifications cannot be readily met. The central facility, on the other hand, is confronted with similar requests from many operating areas. Thus it often is not capable of giving any operating division's needs the priority which that division feels it deserves. The result many times is user resentment and a self-protective attitude on the part of the providers of the service [85, p. 30]. Furthermore, the operating areas normally are forced consumers of the central facility's services. They are not allowed to purchase these services from outside the

organization, and this lack of choice compounds their resentment whenever service breaks down. Experience has shown that as long as the user has no choice in the source of data processing services, the user can be expected to be dissatisfied with that service [127, p. 97].

These conflicts are also often compounded by a lack of goal congruence between the operating departments and the central facility. Operating management typically has goals relating to production levels, cost minimization, or sales achievement. In such situations data processing difficulties are often considered to be a problem of the data processing manager or his staff and not a responsibility of operating management. Similarly, data processing personnel are specialists who seldom have contact with personnel in the user areas. They are often more concerned with their own professional development within their specialty than with profit objectives of an operating division. In a distributed computing environment, where the performance of the information processing specialists are more readily identifiable with the profitability of their functional areas, this lack of goal congruence is minimized.

Some advocates of centralized computing argue that this organizational form provides better utilization of computer personnel. As systems become larger and more complex, qualified systems specialists become rare and more difficult to recruit. Thus, it is argued, a large centralized system not only utilizes such personnel more

efficiently but also makes them easier to attract [68, p. 4] [157, p. 66]. However, a major advantage of smaller, functionally dedicated systems is that they are less complex. The level of expertise required to operate and maintain such systems is not as great as with a more sophisticated one, and a large number of highly qualified analysts and programmers is not required.

From the preceding arguments, it can be seen that a major advantage of distributing the computing function in an organization is the support which it provides for decentralized decision-making. Martin has summarized the issue by saying:

> As an organization becomes large, so it becomes more difficult to control and finds it more difficult to achieve fast reactions. This is true of both an organization that is large geographically, with branches or factories scattered over a wide area, and of one that is large administratively. The geographically dispersed organization needs data transmission links in order to react quickly. The administratively large organization has economically sound reasons for centralizing many of its administrative functions. Centralized purchasing or stock administration, for example, can bring direct financial savings. Centralization, however, can and has in many cases produced a bulky, slow-to-move, bureaucratic administration. The use of computers planned to give the optimum reaction time can help to avoid this. The computer system must carry out the required processing as quickly as it is needed and must give information to management and other staff where and when it is needed. [108, p. 162].

Distributed information systems provide data which are more timely and more relevant. They avoid many of the conflicts associated with the centralized structure. Yet most proponents of centralized systems do not justify their position based on organizational considerations. The

support for centralized computing relies more on grounds which are technical or financial.

Technical Considerations

Technical issues which influence the controversy concerning centralization and decentralization relate to the limitations of minicomputers, the need for control over electronic data processing (EDP) operations, and the complexity of large systems. These will each be discussed in turn.

Minicomputer Limitations

Because of their small size and the history of their development, there are a number of limitations inherent in the use of minicomputers. The largest minicomputers are characterized by logic based on a 16-bit word length. In larger general purpose machines the word length is typically 32 or 36 bits in length. This smaller word size limits the number of locations in main storage which can be addressed, thus effectively limiting the size of any program which can be executed on a minicomputer [153, p. 33]. Although most business applications can be implemented within the word size limitations, small computers cannot be used to handle large simulation and linear programming models or large compilers [115, p. 497]. An organization which utilizes minicomputers for routine business processing and also wishes to use large operations research models must

therefore do so on a time-sharing basis.

Another problem arises from the small size of functionally dedicated minicomputers. Each system must have its own peripherals to provide storage and input/output capability. In a large centralized system, one or more processors will share a number of disk and tape drives, printers, and input units. A centralized system is, therefore, less susceptible to the failure of a peripheral than is a decentralized one where redundancy does not exist. When dedicated minicomputer systems are interconnected in a network many of these problems disappear. Networking allows a minicomputer at one location to access the mass storage devices at another, and provides a degree of reliability comparable to that of a centralized system.

Initially most minicomputer manufacturers were small companies with a limited product line. Most of them have directed their efforts more toward developing their computer hardware than toward providing maintenance and support capabilities for existing systems. As a result, such support has not been consistently available at all installations and from all manufacturers. The user of a minicomputer system has sometimes encountered hardware or software difficulties and found that technical help to remedy the problem is not readily accessible. Although some minicomputer manufacturers have shown remarkable growth, there is still a large number of them which have not matured and are not yet financially stable. Any potential user of

minicomputer systems needs to be aware of the risks associated with obtaining equipment from a manufacturer that is not well established.

Another objection to the concept of functionally dedicated minicomputers relates to the duplication associated with a number of diverse systems. Applications implemented in different operating areas may utilize the same data. It would be difficult and unnecessarily expensive to provide duplicate files at each location. Additionally, minicomputer software is generally not transferable from one manufacturer's machines to those of another. If different types of machines are used in different areas, each would require a separate set of file manipulation and input/output routines. These problems can likewise be avoided by interconnecting similar processors in a network such that a processor could access another's files. To do so would require the establishment of company-wide standards for file design and record layout. The creation of communications protocols would be necessary to allow the inter-processor data communications to take place. In other words, an organization implementing a network of functionally dedicated minicomputers would have to establish a means for controlling its data processing operations company-wide.

Control of EDP Operations

A major argument in support of centralized data

processing operations stems from the greater control which this centralization provides. A data processing department under a single manager can exercise stronger organizational control in a number of areas. These will be referred to as systems management functions. They include [87, p. 53]:

1. Establishing policies, goals, and priorities,
2. Selecting projects, hardware, and languages,
3. Enforcing uniformity of coding systems,
4. Eliminating duplication of programming efforts,
5. Defining common reporting methods and data.

In a functionally dedicated network of minicomputers, the need for such control would be as great, if not greater, than in a fully centralized system. Some mechanism for central control and direction throughout the network would need to be established.

A number of researchers have recognized that most computer-based information systems, like the organizations in which they reside, are neither entirely centralized or entirely decentralized. Instead they are usually a combination. As McFarlan has said:

> A computer based information system entails a broad range of operating and management functions. . . . An organization can elect to perform some subset of these functions centrally and another subset locally. Thus the term "centralization" is meaningless when applied as a generality to information systems. Instead, the concept of centralization must be approached in terms of the specific functions that make up the operation and management of an organization's information system [115, p. 486].

Davis has cited five major strategies in combining

both centralized and decentralized information processing in an organization. They are [35, p. 376]:

1. Central control of all EDP functions,
2. Central advisory function with all the information processing development and operations remaining in the subunits,
3. Central control of hardware and hardware operations with decentralized system development and programming,
4. Central control of all EDP hardware, operations, and programming, with only systems development being decentralized, and
5. Central control of planning, analysis, and programming, with decentralization of hardware.

Most researchers agree that in companies with successful computer-based information systems, the systems management functions are centralized to some degree [3, p. 148] [37, P. 161] [35, P. 379] [115, P. 492]. The controversy which exists, therefore, concerns to what extent hardware, operations, and systems development should be centralized. Arguments for the centralization of hardware and operations are primarily financial and will be discussed in a later section.

Systems development concerns the creation and evolution of applications systems in the organization's operating areas. Dearden has suggested that within any operating area, systems development exists in three stages.

Systems specification determines what information is to be provided and the timing of the information. Data processing implementation specifies the required programs, files, and procedures necessary to provide this information. And finally, in the programming stage the programs and files are actually established [37, p. 157]. As has been discussed previously, organizational considerations dictate that systems specification be performed as much as possible at the operating level. Since the implementation and programming stages rely upon the results of the specification stage, they can also be performed most efficiently at a decentralized level. This can be accomplished provided that the hardware itself is decentralized.

Thus, a major advantage of a distributed minicomputer network is that such a system allows centralized control of the management function. Simultaneously systems hardware, operations, and development can be located within the functional areas of the organization. This provides more timely, useful information to operating management without sacrificing advantages arising from company-wide control of EDP operations.

Complexity

Large multiprocessor systems avoid many of the limitations inherent in the use of minicomputers. However, the advantages of size and flexibility are accompanied by

certain disadvantages resulting from the added complexity of large mainframe computers.

Many managers feel that the lead time needed to integrate such systems into the framework of an organization is disproportionately long. The pace of development is typically slow, and by the time a development project is complete the design is, in many cases, no longer suitable for its application [155, p. 34]. A study group predicting future trends in data processing has expressed concern about the future of such systems:

> It is easy to identify systems that are so large and complex that elapsed time for implementation, using today's development techniques, could well exceed the expected stable life of the system or the organization that needs the system in the first place. The message is clear: Significant progress must be made in applications development if the large, complex applications envisioned today are to be practical (i. e. implementable) tomorrow [42, p. 106].

Minicomputer systems, by nature of their simplicity, are more readily developed and implemented. Furthermore, a network of minicomputers is modular in its structure. Applications can be developed one at a time and added to an already existing network, making it possible to partially implement the network in a comparatively short time period.

The complexity of large-scale multiprocessor systems not only creates difficulties in development, but results in operational shortcomings as well. The raised floors, large capacity air conditioners, and backup power generators required by such systems may require complete reconstruction of the building in which the centralized computer is

located. Dedicated minicomputers, on the other hand, are located within the user operating area. Their size and convenience is such that special building arrangements are usually not required [85, p. 29] [87, p.73] [155, p. 34].

Furthermore, the decentralized minicomputer approach is less susceptible to system failure. In a multiprocessing system, a hardware or software failure will at least degrade system performance for all users, and will in many cases render the entire system unusable for an extended period of time [67, p. 13]. Studies of downtime in large computer systems indicate that about 75% of the diagnosed causes of system failure were due to software problems. Some of these problems were due to software bugs, but most were due to failures of the operating system in allocating and scheduling internal memory and other resources. The minicomputer system is less complex and therefore likely to have fewer errors [87, p. 150]. In addition, failure in a minicomputer network renders only the affected portion of the network unusable. Reliability and functional redundancy in a network are enhanced over the central facility because it is possible to have much smaller backup processors. Likewise, processors can be designed to assume the responsibility of similar processors in the event of node failure with very little degredation of system performance [2] [161, p. 48].

The flexibility of a centralized multiprocessing system is accomplished by its operating system. As such a

system is used for more and more applications, the operating system becomes more and more complicated. The central processor is required to perform an increased number of functions associated with scheduling jobs and input/output transfers. A smaller portion of the machine's capacity is available for user applications [151, p. 93].

The amount of main storage required for operating and control software in a multiprocessing system can be very large. Table 1 indicates the severity of this problem in a centralized system utilizing two IBM 370/165 processors. Approximately 50% of main storage in this system is used by core-resident systems software, many times that much space is reserved for these control programs on mass storage devices [127, p.9].

As can be seen in Table 1, the increased complexity of a multiprocessor creates a large amount of overhead necessary to manage processing tasks. In a network of minicomputers, processors will not be burdened by control software which is not utilized by that application. Such functionally dedicated processors will be less complex and easier to operate and program than a multiprocessor.

As centralized systems become more complex in order to handle a wide variety of applications, the required overhead increases. The result, therefore, is increasing diseconomies of scale. This fact is a significant one in considering those arguments for centralized data processing operations based on financial considerations.

TABLE 1

CORE USAGE

DUAL IBM 370/165 PROCESSORS

(IN KILOBYTES)

CONTROL SOFTWARE	PROCESSOR 1	PROCESSOR 2	% TOTAL
OS	708	694	23
ASP	750	-	13
IMS	530	-	7
ATS	-	166	3
TCAM	-	104	2
TSO	-	128	2
USER	1084	1980	50
TOTAL	3072	3072	100

Financial Considerations

Most of the proponents of centralized data processing have based their arguments on financial considerations. Computer manufacturers have historically priced their product line such that unit cost per computation decreases as the size of the processor on which an application is implemented increases. In the past a single large computer was able to perform the work of a number of small or medium sized ones and do so at less cost. This pricing relationship has been commonly referred to as "Grosch's Law": The throughput capacity of a computer is proportional to the square of its price. Attributed to Herbert Grosch, the relationship was never published by him but has been widely quoted in the industry. It was later confirmed by two studies of IBM's equipment pricing policies [97] [144].

Both studies provided evidence of increasing economies of scale in the IBM 360 line (see Figure 5). Knight's study compared a "cost-effectiveness" ratio to cost for two types of instructions sets on different computers. One set represented an instruction mix typically found in scientific programs and the other a mix representative of commercial applications. Solomon evaluated cost functions for five System/360 models using instruction times, representative programs or "kernals", and typical instruction mixes. The efficiency ratios used, however, were measured in terms of central processor speeds.

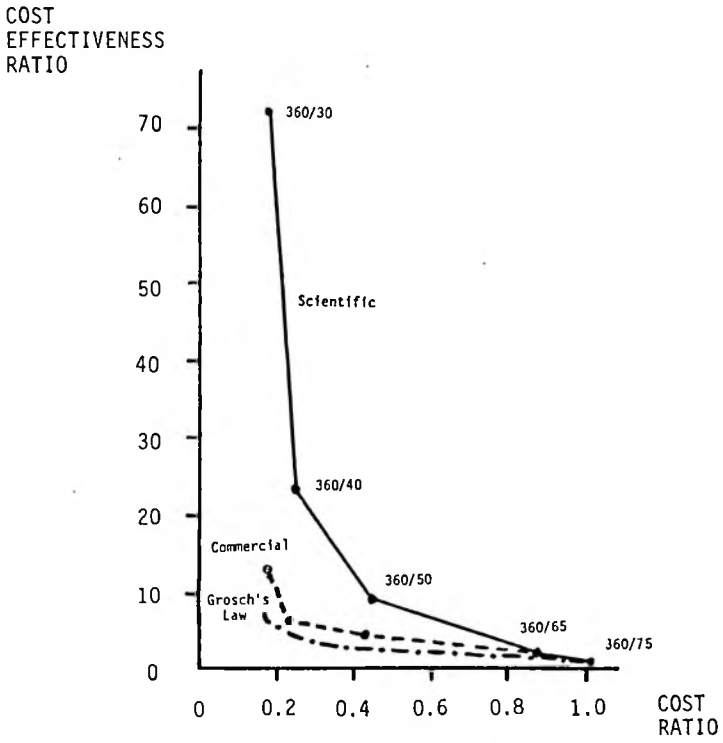
Business applications are more dependent on the input/output capabilities of a system than on its internal processing rates, so the results are not precise indicators of data processing capacity. Although neither study is truly representative of business applications, each has been widely accepted in the industry. Because they unquestionably demonstrate the existence of economies of scale in central processors, the studies have been widely used to justify data processing centralization as a means of achieving cost effectiveness.

Nevertheless economies of scale in central processors do not guarantee economies of scale in the total information system. A meaningful comparison of data processing costs should consider more than just internal computer performance. In comparing system alternatives, consideration should also be given to the costs of systems analysis, programming, operation, and maintenance for both hardware and software. Some researchers have noted evidence of diseconomies of scale in such areas [87, p. 56]. The overhead associated with a large complex processor may in fact obviate any advantages gained by faster processing speeds.

Other costs associated with a data processing system are not easily quantifiable in accounting terms. These include costs resulting from erroneous design specifications, multiple changes in design during system development, poor source document design, and a lack of

FIGURE 5

KNIGHT'S ECONOMIES OF SCALE



Source: Sharpe, p. 320

communications between the designer and the user [67, p. 12]. Data processing costs are typically only a small part of the total cost of the organizational system in which the computer operates. Even though a centralized data processing configuration may minimize per unit processing costs, total systems costs may be increased. An advantage of the distributed processing concept is that the system is more adaptable to the needs of the user. This results in an overall cost savings to the organization even though the savings may be difficult to quantify.

Indeed, technological changes since the studies of Solomon and Knight have led many industry observers to question the existence of economies of scale in per unit processing. Minicomputers provide a great deal of processing capability for a low price. The use of microprocessor based central processors and solid state memory devices can be expected to decrease computer costs even more in the future. Such technological improvements make cost comparisons over time very difficult to make. A decrease in cost is almost always accompanied by an increase in performance, so that computer equipment of one generation does not exist with capabilities identical to those of a previous generation [158]. Considering these performance improvements, one observer has estimated that central processors in 1975 cost approximately one-half of one percent of those in 1960. Furthermore in 1985 processors will cost about twenty percent of those in 1975. The cost

of mass storage devices can be expected to decrease in a similar fashion [151, p. 86].

Although comparisons with large systems are likewise difficult to make, some researchers have concluded that it is possible to achieve adequate capacity for business applications using existing minicomputers. Based on research at Carnegie-Mellon University, Cohen has estimated that a network of minicomputers can provide throughput equivalent to that of a single large processor at 20% less cost [32]. These results are presented in Figure 6. In choosing to decentralize its data processing, First National City Bank of New York replaced its two IBM 370/158 multiprocessors with eight Interdata Model 8/32 computers. The change provided an estimated six-fold increase in computing power at a decrease in initial cost from $3 million to $2.4 Million [82].

It appears that minicomputer technology has decreased the cost of processing to the point where Grosch's Law is no longer applicable. This conclusion has also been reached by other researchers [22, p. 149] [151, P. 93]. Insufficient experience exists in implementing functionally dedicated networks to enable accurate cost predictions for such systems. The technology has not been developed to the point where widespread acceptance is attained. This is particularly true if processing is widely dispersed geographically and data communications costs become a factor. These costs are regulated by government agency and

FIGURE 6

COST COMPARISON

LARGE SCALE VERSUS MINI PROCESSORS

Note: Comparison is based on systems of estimated equivalent throughput capacity.

LARGE SIZE SYSTEM

IBM 370/158:	
CPU & MEMORY	$1,500,000
PERIPHERALS	1,000,000
TOTAL	$2,500,000
NETWORK OF 16 MINICOMPUTERS:	
CPU & MEMORY	$1,000,000
PERIPHERALS	1,000,000
TOTAL	$2,000,000

SAVINGS: $500,000 (20%)

MEDIUM SIZE SYSTEM

IBM 370/145:	
CPU & MEMORY	$ 700,000
PERIPHERALS	600,000
TOTAL	$1,300,000
NETWORK OF 6 MINICOMPUTERS:	
CPU & MEMORY	$ 400,000
PERIPHERALS	600,000
TOTAL	$1,000,000

SAVINGS: $300,000 (23%)

are thus subject to political influences as well as competitive and technological ones [152]. However it is clear that the financial advantages once attributed to the centralized data processing structure can no longer be relied upon. When costs of the total user-information system are considered, distributed minicomputer networks can be expected to compare favorably.

Summary

A decentralized data processing structure utilizing a network of functionally dedicated minicomputers offers a number of advantages which are not present in the centralized structure. Perhaps the foremost is the support that the network provides for decentralized decision-making. The comparative simplicity of minicomputer systems allows them to be implemented within the operating areas of an organization. There they can provide information more suitable to the user's needs and on a more timely basis. Such a system causes less disruption to the organization in which it exists and does not entail many of the overhead costs associated with large-scale multiprocessing systems. As hardware costs continue to decrease and as experience with data communication technology is attained, the financial advantages of centralization are no longer conclusive. Distributed processing networks are gaining wide acceptance in computer-based information systems for business, governmental, and non-profit institutions.

CHAPTER III

A TYPICAL BUSINESS APPLICATION

In the past very little attention has been given to the applications which would be utilized in a distributed business system, or to the effects which the applications would have on the design of the network. Previous research does not provide the information which systems managers need in effectively implementing a distributed information system with minicomputers. A methodology is needed which enables the manager to evaluate expected network performance under conditions determined by an organization's own processing requirements. The methodology should utilize data which is readily available in a business environment and should yield results which are useful in making a wide range of design related decisions.

The purpose of this chapter is to present a fictional organization which is a suitable setting for the implementation of a minicomputer based distributed information system. The company will be described and the informational requirements for its effective management will be discussed. Then the design will be presented of a system that is suggested for satisfying these needs. This system will constitute a typical business application. In Chapter

V a simulation model useful in implementing this application will be presented. In Chapter VI the model will be utilized in the design of a functionally dedicated network.

The Business Environment

The Agee Company was established in a large southwestern city in the years following World War II. Initially a closely held corporation, it has since become publicly held and its stock is traded in the over-the-counter market. Agee is engaged in the design, manufacture, and marketing of conventional and special purpose precision valves. Its approximately 20 customers are primarily in the petroleum, chemical, and natural gas transmission industries.

In a recent year Agee had sales of $20 million and a net income of $1,600,000. Its return on stockholder's equity has averaged 15% over the past few years. The company has 200 employees, of which 150 are paid by the hour. Approximately 100 of these are utilized on the company's three production lines at its main plant. Agee is capable of manufacturing a wide variety of valves, but its current product line includes 25 basic sizes and types. Special purpose valves for certain customers can normally be fabricated from one of the standard types with very little additional production effort.

Agee Company has several competitors in the specialty valve industry, primarily located in the Northeast

or Midwest. Because of its location in a major petrochemical center, Agee has a decided competitive advantage resulting from its proximity to a major portion of the market. With the increasing nationwide emphasis on efficient energy generation and distribution, company management foresees a period of significant growth over the next decade. There is an increasing concern that existing systems and procedures will become inadequate in providing the information necessary for proper management.

Existing company procedures for operational control are essentially those which were established in the first few years of its life. Increased volume has required that they be formalized somewhat, and that responsibility for various tasks be dispersed within the organization. Many operational personnel complain that the large volume of paperwork associated with the manual systems prevents them from devoting sufficient attention to supervisory and managerial tasks. For several years a local computer service bureau has been utilized for routine bookkeeping operations. Yet the data processing services from this source are limited in their scope and have been highly susceptible to system breakdowns. Top management has concluded that the possibility of implementing a computerized information system within the company should be investigated.

Although it is apparent that some improvement in company operating systems is called for, Agee questions the

propriety of implementing a computerized system at the present time. Its employees have no experience and little expertise in data processing management and controls. Thus they are unsure about what type of system would be appropriate. Some officers question that existing volume is sufficient to justify the major expenditures associated with a computerized system. A summary of current transaction volume for existing manual systems is provided in Table 2. All officers agree that any new system should be capable of providing for expected growth without overextending the company's resources. Because Agee is not a large company, it would like to avoid the expense of employing a large staff of analysts, programmers, and key punch operators. Top management takes pride in the degree to which it has delegated operational responsibilities to key supervisors and does not want to implement any changes which will decrease the authority or prestige of these trusted employees. Any proposed system, to be acceptable to the company, should operate within at least most of these constraints.

Informational Requirements

A major objective of the proposed computer based system at Agee Company is to facilitate the flow of information which is currently performed with manual systems. This, it is felt, would eliminate some of the paperwork required of supervisory employees and increase the

TABLE 2

AGEE COMPANY TRANSACTION VOLUME

TRANSACTION TYPE	AVERAGE VOLUME	FREQUENCY
CUSTOMER ORDERS RECEIVED	20	DAY
SHIPMENTS TO CUSTOMERS	20	DAY
CASH PAYMENTS RECEIVED	20	DAY
PURCHASE ORDERS ISSUED	20	DAY
SHIPMENTS FROM SUPPLIERS	20	DAY
VENDOR INVOICES RECEIVED	20	DAY
STORES REQUESTS PRODUCED	50	DAY
CHECKS ISSUED TO VENDORS	30	DAY

quantity of work which can be effectively handled. Some improvement in efficiency could also be expected because a computer based system would require the standardization of procedures which otherwise are performed on an informal basis.

A second objective is to provide additional types of information which are not available from existing manual systems. Because of the speed with which the computer can perform repetitive operations, and of its ability to compactly store large amounts of data, information for decision making can be generated which is not feasible otherwise. And once a system has been designed and installed, new applications can be implemented as previously unrecognized needs become apparent to managers in the company.

Most initial efforts at installing a computer based system begin by mechanizing the routine bookkeeping functions. These operations are performed in a fairly standardized manner regardless of the organization type, and implementing them on a computer increases the capacity of the clerical workforce while decreasing the frequency of errors. Typical applications, already being performed manually at Agee Company, include general ledger, payables, receivables, disbursements, billing, and payroll.

Most businesses are required to keep several types of data relating to their employees. This includes payroll records, tax withholding information, amounts of other

deductions, and personal information of various kinds. An employee data system could maintain much of this data on computer files, thus eliminating many manual ones and facilitating file searches for data retrieval. Such a system would ideally be used in conjunction with Agee's payroll system.

Clerical tasks in other areas of a business are suitable for computer implementation. At Agee Company, new customer orders are entered through its marketing department. This manual procedure is one which can be readily mechanized. An order entry system can provide data which would be useful in the production department for the scheduling of manufacturing operations. Another suitable application is therefore a production scheduling system. A simple program could be created which implements by computer the algorithm already being used by the production manager. Such programs have been shown to be executable within the limitations inherent in the use of minicomputers [137].

Another existing system at Agee which could readily be mechanized is the inventory control system. The computer could maintain data on inventory levels by monitoring the issue of material and supplies from inventory, and the addition of such items to inventory. A product costing system could also utilize data on material issues, as well as data from the payroll system, to provide reliable and timely estimates of product costs.

Information on when to reorder to inventory can be

provided from an easily programmed set of decision rules. These decision rules can be used in conjunction with a purchase order system to notify the purchasing department whenever inventory replenishment is required. The purchase order system could not only provide notice when an order is appropriate, but also print the purchase order and maintain records on inventory items which have been ordered but not received.

These systems are all potential components of a total computer based information sytem. They can be expected to provide Agee Company with information for decision making which is more timely and more accurate than that provided by existing manual systems. However, a major advantage of a computer based system is that it can generate data which would not be practical manually. In this category, additional marketing information is suggested for Agee. Most organizations engaged in marketing have need for data analyzing the characteristics of past sales. For Agee, this data would summarize its sales by product and also by customer. This information could be readily provided by a minicomputer on a periodic basis and could serve as a guide for future marketing efforts.

After a computer based system has been in operation for a period of time, a number of other potential applications will become evident. And when employees observe that the computer based system actually makes their work easier rather than more difficult, additional

applications can he expected to gain support.

A major advantage of a network of dedicated minicomputers, as discussed in Chapter II, is its flexibility. Such a system would allow Agee to implement a limited system initially to avoid overextending company resources. Additional applications could be added over a period of years, thus providing for future growth. Because most minicomputers operate efficiently on an interactive basis, a large data processing staff could be avoided. Since the processors would be dedicated to specific functional areas of the business, supervisory personnel would maintain control over the use of their own applications, thereby maintaining a sense of individual responsibility.

In summary, the need for accurate, timely information exists throughout an organization such as Agee Company. Computer based systems can be used to provide this information in a more acceptable fashion than can a manual one. Management at Agee needs to know how these informational requirements can be met within the company's own business environment. The best alternative which satisfies these needs is a distributed information system utilizing functionally dedicated minicomputers.

A System Design

The following discussion and flowcharts describe a computer based information system that is appropriate for

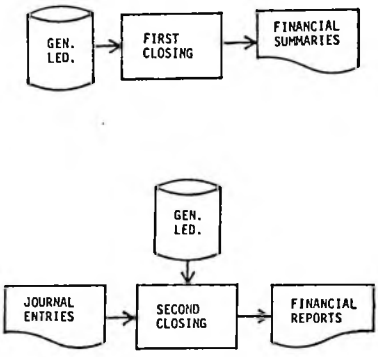
the requirements of Agee Company. It is intended to represent a system typical of applications found in organizations engaged in manufacturing. The flowcharts do not depict the detailed procedures of every program in the system, but rather illustrate the functions which such programs can be expected to perform. The flowcharts are representative of the type which would be utilized in the preliminary design of an information system. They also provide the basis for describing the system to the design model presented in Chapter V. Each jobstream which is a component of the system is discussed.

General Ledger Jobstream

The purpose of the general ledger jobstream is to provide periodic information on the profitability of the company's operations. In the proposed system for Agee Company, accounting reports would be generated on a monthly basis. This procedure is a fairly standardized one, in which nominal accounts are closed to a profit and loss account that reflects the profitability, on a cash basis, of the enterprise. At year end, a second closing jobstream would be utilized to provide for accruals and deferred items.

FIGURE 7

GENERAL LEDGER

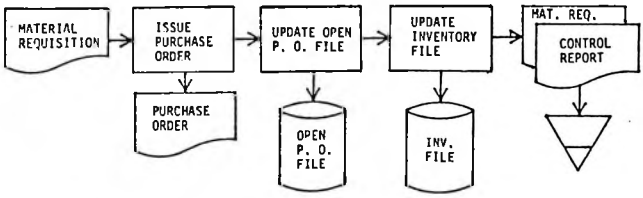


Purchase Order Jobstream

The purchase order jobstream accepts a material requisition, or notice that inventory replenishment is due, from the inventory system. Based on predetermined order quantities, it automatically prints the purchase order and updates a sequential file which contains a list of all unfilled purchase orders. A program also flags the inventory records for the item ordered, indicating that a replenishment has been initiated.

FIGURE 8

PURCHASE ORDER

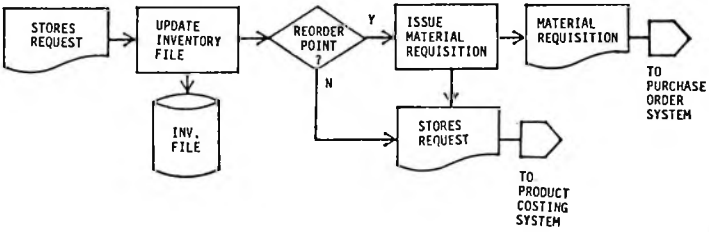


Material Issues Jobstream

This jobstream updates inventory records whenever supplies or raw materials are removed from inventory. At each decrement in inventory, it determines if a replenishment order for the item is appropriate by comparing the quantity in inventory to a preset reorder point. When a replenishment order is needed, a material requisition is produced that initiates action by the purchase order system. Inventory withdrawals which are attributable to a specific product or shop order also provide inputs to the product costing system.

FIGURE 9

MATERIAL ISSUES

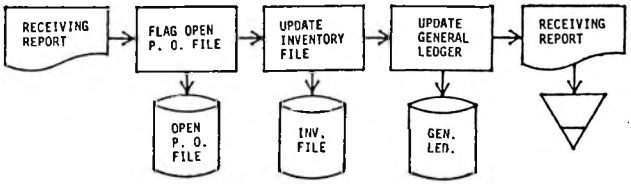


Material Receipts Jobstream

This jobstream updates inventory records whenever supplies or raw materials are received by Agee and are added to stocks on hand. It would be initiated by a notice from the company receiving dock that a shipment has arrived. The open order flag in the inventory file is deleted and the inventory quantity for the received item is incremented. The general ledger account for raw material or supplies inventories is also debited.

FIGURE 10

MATERIAL RECEIPTS

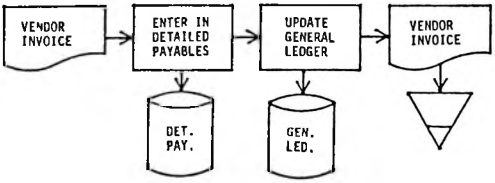


Payables Jobstream

The payables jobstream is initiated upon receipt of an invoice from a vendor. The invoice is vouched, approved for payment, and entered in a detailed payables ledger. The general ledger payables control account is also credited for the amount of the invoice.

FIGURE 11

PAYABLES

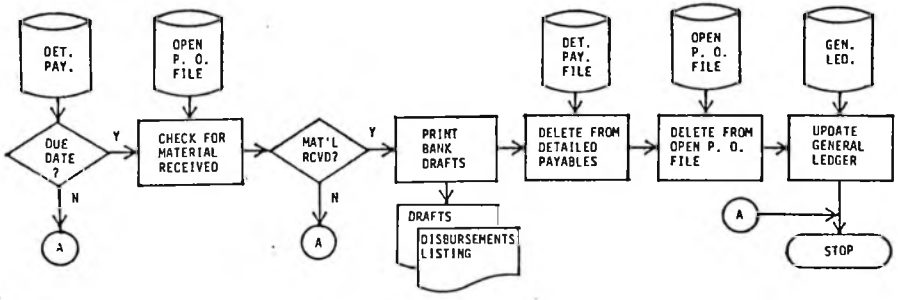


Disbursements Jobstream

The disbursements jobstream is used on a daily basis. The payables file is scanned item by item, and the due date for each item is compared to the current date. When an item is found to be due for payment, a check is performed to see if the item has been received. If it has, a draft is printed in payment for the invoice and appropriate files are updated indicating that the item has been paid for. If either payment is not due or the invoiced item has not been received, payment is deferred until both of these criteria are met.

FIGURE 12

DISBURSEMENTS

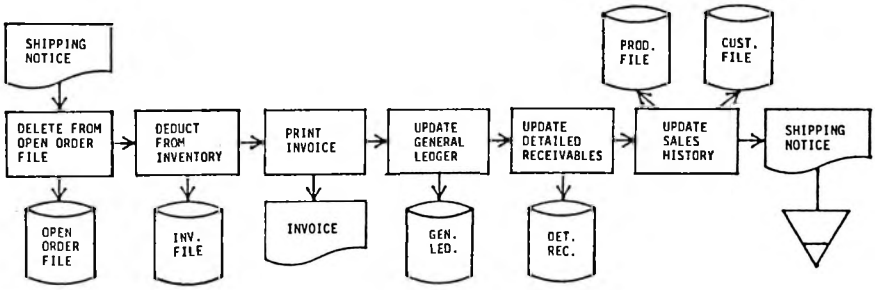


Billing Jobstream

An efficient billing system requires that the customer be promptly invoiced for any product which has been shipped. In the proposed system for Agee, a shipping notice copy, which indicates that a shipment has gone out to a customer, initiates computer processing. The shipment is deducted from inventory and the customer's order is closed. An invoice is printed and the detailed receivables file and general ledger receivables control account are updated. Finally the sale is recorded in customer and product files for reference in the marketing system.

FIGURE 13

BILLING

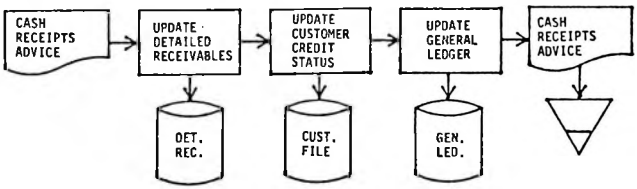


Cash Receipts Jobstream

The cash receipts jobstream is a component of the company's receivables system. When a payment is received from a customer, the accompanying cash remittance advice is sent to the clerk operating the minicomputer. By entering appropriate data at the input console, the processor is caused to register the receipt of cash in the detailed receivables file and the general ledger control and cash accounts. The customer's entry in the customer file is flagged for later use as a credit check in the order entry system.

FIGURE 14

CASH RECEIPTS

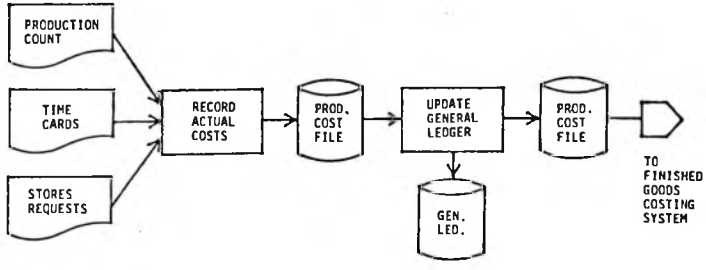


Work in Process Jobstream

Computer processing for the product costing system is specified in general terms in order to allow any one of a number of cost accounting methods. The first jobstream is used to compute actual costs for partially finished goods and estimate a value for work in process inventories. It utilizes production counts, time cards, and stores requests to estimate these costs. This jobstream would be run weekly and would provide cost information on each shop order not completed during the week.

FIGURE 15

WORK IN PROCESS COSTING

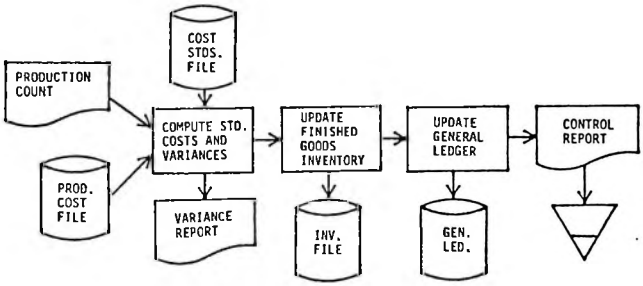


Finished Goods Costing Jobstream

The second product costing jobstream is also run weekly and estimates costs for shop orders completed. It utilizes production counts, production standards, and previously generated in process costs to compute values for finished goods and variances associated with the production process. It also uses the production counts to update finished goods inventory records.

FIGURE 16

FINISHED GOODS COSTING

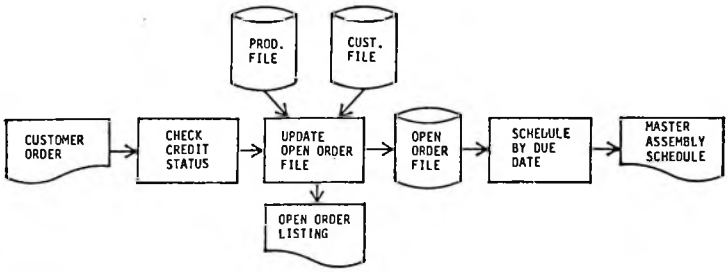


Order Entry Jobstream

Computer processing begins in the order entry jobstream when an order is received from a customer. Appropriate data is entered into the system and a credit check is performed. Then the requested items are added to an open order file and an order acknowledgement is printed for the customer. When a day's orders have been entered in the system, an open order listing and updated master assembly schedule are prepared.

FIGURE 17

ORDER ENTRY

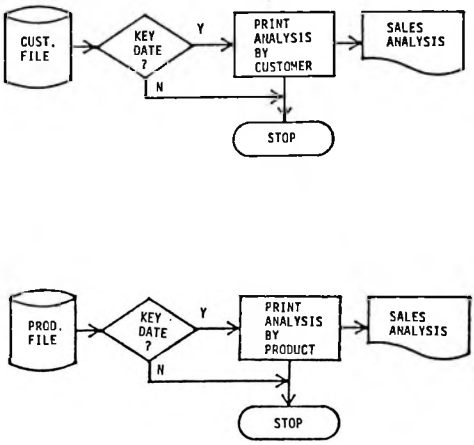


Sales Analysis Jobstream

The sales analysis jobstream provides historical summaries of sales both by customer and by product. Such information could be produced at any time it is required by marketing personnel. A key date is entered in the system, and all sales which have occurred since that date are listed and totaled either by customer or by product.

FIGURE 18

SALES ANALYSIS

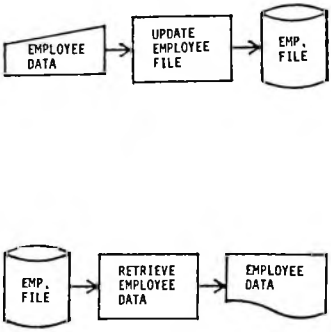


Employee Data Jobstream

This jobstream, which is a component of a personnel system, also is available for use whenever required. Employee records are maintained on mass storage and are modified or data retrieved on an inquiry basis.

FIGURE 19

EMPLOYEE DATA

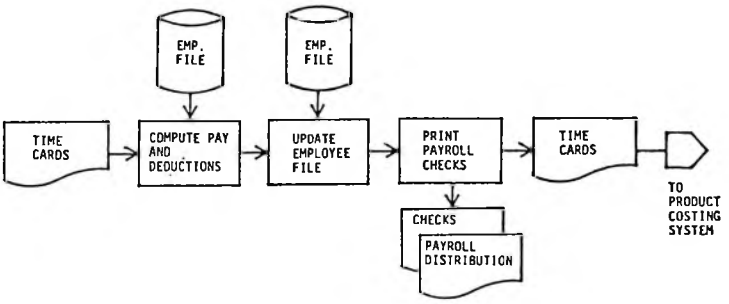


Payroll Jobstream

The payroll jobstream is a fairly standardized application in most businesses. In Agee Company, it would be run on a weekly basis. Hours worked for each employee are entered in the system from time cards. Pay and deductions are computed and the employee's personnel records are updated. Finally the payroll checks are printed along with a control report showing the payroll distribution. The time cards are also used as an input to the product costing system.

FIGURE 20

PAYROLL

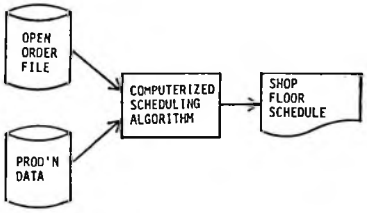


Scheduling Jobstream

Production scheduling is specified in general terms to accommodate any algorithm in use at Agee Company. Such an algorithm would rely on data accumulated in the open order file and also on predetermined production rates and capacities. The algorithm would then produce a shop floor schedule on a weekly basis.

FIGURE 21

SCHEDULING



System Files

The data files required by the system, and the jobstreams accessing them, are summarized in Table 3.

Summary

Agee Company represents a typical organization for which a distributed network of minicomputers presents a viable data processing alternative. The system design presented for Agee does not represent an ideal or all-inclusive system but rather one which might be expected to exist in a business of this type. Some applications have been kept purposefully simple. This was necessary because all programs are restricted in size due to the limitations of minicomputers discussed in Chapter II. Furthermore, applications are not presented at a highly detailed level, but only at a level required for specifying each application to the simulation model referred to in Chapter I. Applications are normally specified to a greater degree of detail whenever system evaluation is taking place prior to implementation.

TABLE 3

AGEE COMPANY SYSTEM FILES

FILE NAME	JOBSTREAMS ACCESSING
EMPLOYEE	EMPLOYEE DATA PAYROLL
OPEN PURCHASE ORDER	PURCHASE ORDER MATERIAL RECEIPTS DISBURSEMENTS
INVENTORY	PURCHASE ORDER MATERIAL ISSUES PRODUCT COSTING MATERIAL RECEIPTS
DETAILED PAYABLES	PAYABLES DISBURSEMENTS
OPEN ORDER	ORDER ENTRY BILLING SCHEDULING
PRODUCT	ORDER ENTRY BILLING SALES ANALYSIS
CUSTOMER	ORDER ENTRY BILLING SALES ANALYSIS CASH RECEIPTS
DETAILED RECEIVABLES	CASH RECEIPTS BILLING
PRODUCTION DATA	SCHEDULING
GENERAL LEDGER	PRODUCT COSTING MATERIAL RECEIPTS PAYABLES DISBURSEMENTS BILLING CASH RECEIPTS
STANDARD COSTS	PRODUCT COSTING
PRODUCT COSTS	PRODUCT COSTING

CHAPTER IV

COMPUTER SYSTEM SIMULATION

System simulation has been described as "a process in which real world data . . . is processed through a model which reproduces the operating environment" [147, p. 455]. Because it permits evaluation of decision alternatives without requiring alterations to a system, simulation has been proven an extremely valuable procedure in analyzing a wide variety of situations. Computer scientists have recognized for a number of years the potential advantages of utilizing simulation models of computer systems. For an extensive bibliography of early research in the area see MacDougall, p. 204. As has been discussed in Chapter I, organizations wishing to implement a network of minicomputers do not have a wide range of experience on which to base design decisions. Such a distributed network is a relatively new concept in data processing and errors in design or development could be costly and difficult to correct. A design methodology which relies on simulation may be expected to avoid many of the shortcomings of a trial and error approach to systems development.

The simulation process is subject to some unique difficulties when it is applied to computer systems. These

difficulties place limitations on the manner in which computer system models may be effectively used in aiding managerial decisions. As a result, most such research has been highly technical in its nature and deals with engineering rather than managerial problems. It is the purpose of this chapter to discuss the difficulties encountered in creating a computer network model which will be useful in a managerial context. These problems will have a major impact on the model described in Chapter V and utilized in Chapters VI and VII for the design of the Agee Company system. A survey of prior research in developing simulation models of computer systems will also be presented.

Difficulties

If management is to use simulation as a tool in developing an effective distributed information system, the decision makers should be aware of the advantages and limitations of the simulation process. Simulation models of complex systems are characterized by the large amount of analysis and coding required for their development. As a result they are often expensive both to develop and to utilize after completion of the model. The expense is a commonly cited disadvantage to the use of the procedure [77, p. 470].

Complexity

The problems of size and expense are increased when the simulation model is that of a computer system. Computers are much more complex than most systems involving humans. They exist in a wide variety of types, and the level of understanding required for adequate modeling dictates that the analyst have a high level of expertise in the system being modeled. This fact has caused many experienced simulation model-builders to avoid undertakings which involve computer systems. In the past most attempts at modeling such systems have been made, not by simulation specialists, but by computer specialists with an interest in simulation. As large-scale computer systems become more complex, this trend can be expected to continue. The ability to accurately model such a computer system requires that the model-builder have a highly specialized set of interests and qualifications.

Level of Aggregation

Additional difficulties arise in utilizing commonly available simulation languages to represent computer systems. Most such languages are event-based. A simulation clock is advanced from event to event at increments of a basic time unit specified in the model by the analyst. In computers, many events take place requiring time intervals measured in billionths of a second. Models of such systems which are constructed at a very detailed level are

consequently extremely large and inordinately expensive to run. For example, a detailed simulation of an entire computer system at the machine instruction level could easily produce over one billion events in a one minute simulated time period [146, p. 137]. Given the computational speeds of present-day computers, such models are not economically feasible. The analyst is required to construct his model at a higher level of aggregation in order to place a reasonable bound on the number of events generated by each run. This higher level causes a resulting decrease in the accuracy of the output of the model. Perhaps the most difficult task facing an analyst modeling a computer system is to achieve a satisfactory tradeoff between the level of detail in the model and the cost of its implementation. Fine levels of detail can only be achieved by narrowing the scope of the model to represent only a portion of the entire system.

Teorey has identified three levels of aggregation which may exist in models of computer systems [146, p. 138]. They are:

1. Gross functional descriptions,
2. Detailed functional descriptions,
3. Machine instruction execution descriptions.

Gross functional descriptions are the most general type of computer system model. Only the major components of the actual system are depicted, with such attributes as capacity, service times, and assumed probability

distributions. Detailed functional descriptions exist at a lower level of aggregation but still require estimatino descriptors of system processes. A number of special purpose languages have been developed for modeling computer systems at this level. They include CSS, developed from GPSS by the IBM Corporation, and a language based on Simscript called ECSS. Machine instruction execution level models attempt to represent each machine language operation being executed by the computer. They have been successfully used in analyzing the performance of small processors in relatively simple applications such as military guidance systems.

Models incorporating progressively finer levels of detail utilize smaller basic time units and require greater simulation run times to implement. Models of processes constructed at the machine instruction level, in fact, typically require greater run times than do the execution of the processes being simulated. These relationships are illustrated in Table 4.

Data Sources

Another problem which exists in developing models of computer systems, as in all simulation experiments, is the difficulty of obtaining appropriate data. Data sources which have been used in constructing models of computer systems include observations of a system in operation, published performance specifications, and assumptions based

TABLE 4

SIMULATION CHARACTERISTICS

AT DIFFERENT LEVELS OF AGGREGATION

LEVEL	BASIC TIME UNIT	RATIO OF RUN TIME TO REAL TIME
GROSS FUNCTIONAL DESCRIPTION	50 - 500 MILLISECONDS	10 - 50 TIMES FASTER
DETAILED FUNCTIONAL DESCRIPTION	1 - 10 MILLISECONDS	20 TIMES SLOWER TO 10 TIMES FASTER
MACHINE INSTRUCTION EXECUTION	LESS THAN 10 MICRO-SECONDS	MORE THAN 20 TIMES SLOWER

Source: Teorey and Merten, p. 138.

on reasonable estimates.

The best source of data in creating a simulation model of a computer system is observation of the system itself. Unfortunately a system of the kind being modeled is not always available. And even when an existing system can be monitored, the kinds of data needed are still often difficult to obtain. Some researchers have attempted to use computer-usage accounting data as a source for model input parameters [20] [121]. Other studies have utilized software or hardware monitors to provide statistics on system operation [86] [102]. A suitably formulated model can rely on data from software monitors such as IBM's System Measurement Facility (SMF). Such computer programs have been a common source of data for simulation studies [23] [148]. Software monitors tend to degrade the performance of the system in which they are utilized. As a result the data obtained from them is not truly indicative of the performance of the system without the monitor. Hardware monitors, on the other hand, provide a means of collecting performance data without affecting the system in which they are used. A hardware monitor is a physical device which can be attached to a computer in order to record specific parameters describing its operation. Typically the types of information gathered are not as appropriate for most simulations as that provided by a software monitor. A type of hardware monitor useful for performance measurement in computer networks has been developed at the National Bureau

of Standards. It requires the use of a large-scale processor to interpret and analyze the data recorded by a data probe [129].

In spite of their limitations, software and hardware monitors are widely used as data sources for computer system simulation. Situations which lend themselves to these methods are also those in which the resultng models can be readily validated. Thus many models based on data from direct observation have been shown to be adequate representations of the actual systems.

In many cases it is not possible to obtain data from an already existing system. If the modeled system does not exist, as is the case in designing prototypes, other data sources must be found. There are also instances where the system exists but all the required data cannot be obtained from monitors or accounting records. In such situations many analysts have utilized performance characteristics which are published by computer equipment manufacturers [13] [162]. A common use of such data is in modeling the performance of mass storage devices [108] [148]. Successful applications of the method also include detailed level models of specific components of a large system, or of small special purpose ones [80] [124]. As is the case when data are obtained from direct observation, the utilization of published performance characteristics can provide useful results when it is used appropriately in the simulation model. In most cases such published characteristics are not

intended specifically for use in simulation, and inappropriate use may produce misleading results or incorrect conclusions about the system being modeled. An advantage is that such data may be used in situations where no other data is available.

Computer system simulation has been utilized in numerous instances where the right kind of data simply does not exist. Such situations occur when the model is at a high level of aggregation, so that data from system monitors and published specifications are not appropriate. Then it may be necessary to make assumptions, which are the analyst's subjective evaluation of estimates which are reasonable. For example, the rate and sequence which many processes follow in a computer system model are highly randomized. This has led some researchers to assume arrival and service rate distributions described by the Poisson distribution [17] [18] [59]. Very little research has been done in establishing the validity of these assumptions, although observations have indicated that Erlang, lognormal, and geometric distributions may be more suitable when the model is created at a high level of detail [62]. Other analysts have based their estimates on past observation of workload and system characteristics or on user interviews [13] [121]. Estimated data values always contain an error factor, and this error is carried through the simulation process and reflected in its results. Therefore, it is important that the simulation's experimental procedure be

carefully designed to minimize the effects of the error. Conclusions drawn from utilizing the model should not be such that they are highly contingent on the accuracy of assumed data input.

Obtaining suitable data is always a problem when constructing a simulation model. The problem is compounded when simulating computer systems because of the wide variety of processes which occur and the very small time periods which many of the processes require. This makes even a small error in an assumed value a significant proportion of the total process time. In order to properly utilize the results of such a model, the existence of data error should be recognized and considered before conclusions are drawn from the model.

Misinterpretation of Results

In order to make a computer system model economically feasible, aggregations must be made in its construction which entail approximations. The results of the model are therefore also approximations. Another source of problems in utilizing simulation models of computer systems arises from a misinterpretation of the accuracy and validity of the results. It has been estimated that with great care and skill, simulation of computers can achieve accuracies of the order of 80% [131, p. 5]. Many models, however, produce results which are less accurate. Any conclusions drawn from the use of such a model must be

reached only after proper consideration is given to the model's limitations. For example, one common use of the generalized simulation packages mentioned in Chapter I is to guide purchasing decisions. Equipment from alternative vendors is compared by modeling its operation on a given data processing job load. The creators of the model must accurately represent differences in computer architecture between the various manufacturers. An error of only 20% in the representation of only one of the alternative systems could have a radical effect on the results of the comparison. In this manner faulty conclusions can be drawn and poor decisions made even though the modeling error is relatively small. Nevertheless simulation models can produce useful results if the objectives are suitably defined and the model developed at a level of detail appropriate for these objectives.

Objectives of Simulation

It is apparent that levels of aggregation and data collection methods which are appropriate for some applications of simulation may not be appropriate in others. McCredie and Schlesinger have correctly pointed out that attempting simulations "capable of answering almost any reasonable question about the system . . . must be paid for by large investments in personnel and computer time" [114, p. 201]. In developing the model consideration must be given to the objectives which the manager hopes to achieve

in the modeling process. The model can then be developed at a level of aggregation appropriate for those objectives.

A management information system is never really completed, but rather goes through a repetitive sequence of events. This sequence, sometimes called the system life cycle, is repeated every five to ten years and consists of a number of stages [35, p. 413] [38, p. 99]. These stages are summarized in Table 5.

Simulation models of computer systems have been particularly valuable in the system planning phase. By modeling the performance of a conceptual workload on a specific class system, a manager can evaluate the feasibility of that system for his own applications. At such a high level of detail most data will be provided by estimate, yet only order-of-magnitude accuracy is required so a large percentage error is not significant. A model can be further used by altering its parameters to represent a number of configurations. Ranges of performance effectiveness could be established for each alternative, thus aiding in equipment evaluation and procurement decisions. Once again a high degree of accuracy is not required. Comparisons based on ranges could be used to eliminate alternatives which are obviously not advisable. Only in the final equipment selection, in which alternatives are likely to be closely matched in performance characteristics, is a high level of accuracy important. At this point in the procurement process, if the simulation

TABLE 5

MANAGEMENT INFORMATION SYSTEM LIFE CYCLE

I. System Planning Phase

- Feasibility
- Equipment Evaluation
- Procurement
- Preliminary Design

II. System Development Phase

- Detailed Design
- Programming

III. System Installation Phase

- Testing
- Revision
- Conversion

IV. System Review Phase

- Modernization
- Reconfiguration
- Expansion

model cannot be refined to produce results of sufficient accuracy then the decision should be made based on other criteria. By comparing alternatives, the simulation model could also aid in the preliminary software design of the information system.

Most of the generalized simulation packages referenced in Chapter I have been applied successfully in the planning phase of the systems life cycle [38, p. 98] [76]. A number of other researchers have developed simulation models which were applied in feasibility analysis, equipment evaluation, and preliminary system design [13] [121]. Computer system simulation has established itself as a significant aid to the information system manager in the planning phase.

During the systems development phase simulation models have been frequently used in providing support for the detailed design process. They are typically used for predicting the effects of specific design decisions, or for monitoring the development of a system. They have been applied at the subsystem or component level to assess the implications of such items as multiplexers, disk drives, or controllers. Models can aid in determining the optimum location of data and programs in main and peripheral storage. Such models necessarily exist at a lower level of detail. As a result they are most valuable in the design of smaller, special purpose systems or in addressing specific problems encountered in developing large systems [43] [46]

[80].

In the installation phase, the design decisions reached during the previous phases of the life cycle are implemented. The operating systems and initial applications are made operational, tested, and revised as necessary. Finally, conversion to the new system from a manual one or a previous computerized one is performed. This phase does not require numerous decisions of significant impact as do the previous ones, so simulation is seldom utilized as a decision-making tool during installation.

Simulation models have been of significant value at a number of data processing installations during the system review phase. Most of these applications have been directed toward achieving improvements in system performance. Sometimes referred to as "tuning" the system, this phase entails efforts to modernize or reconfigure it in some fashion. Simulation models can be used to justify expansion or replacement of facilities based on system performance when subjected to expected increases in future workload [86]. They have also been used in identifying components of the hardware/software system which limit its overall performance, thereby directing expansion efforts toward areas which hold the most promise for improvement [20] [103] [148]. By identifying points of system saturation or circumstances under which resource contention occurs, the configuration can be altered to increase its effectiveness. Additionally, simulation has been utilized to establish

possible redistributions of the workload and thus justify elimination of hardware [44].

Successful applications of computer system modeling have been most profound in this area because models are constructed of existing systems. Accurate data is most readily available during the review phase because it can be obtained from direct observation. Once a model for a system is developed during this phase, it can be modified as system changes occur and used over an extended period of time.

It can be seen that during the life cycle of a management information system, simulation models may be employed to achieve a number of diverse objectives. Most of the applications discussed may be grouped in one of five categories according to their objectives. They are:

1. Feasibility analysis,
2. Procurement decision making,
3. Performance projection,
4. Design support,
5. Improving system performance.

Models for different objectives require development at different levels of detail, and require data of different degrees of accuracy. When the model is constructed and the simulation experiments designed in a manner appropriate for the objectives of the model, the procedure is an extremely valuable tool for evaluating alternatives relating to information system design.

Computer Network Models

Simulation models have been less widely used in the development of computer networks. Most network models already in existence have been created to analyze mission oriented systems such as those used by the military, or to investigate special problems associated with large scale computer networks. Very little research has been done in building models of minicomputer networks for use in business data processing applications.

Models of mission oriented networks have been developed by Pasahow, Moses, and a number of other researchers. The research by E. J. Pasahow was done at the Naval Electronics Laboratory in San Diego, California [124]. A model was created in GPSS of a network of microprocessors interconnected by one or more interprocessor buses. The network itself is a component in a weapons system. The model allowed system designers to determine the effect on backlog queues of increasing the number of buses. It also allowed the isolation of the most highly used processors for replacement by more powerful ones. A model is in development at the Georgia Institute of Technology by Moss, Hammond, and Donaldson for the United States Air Force. It is a highly detailed representation of a digital communications system which includes not only processors but also phase modulated transmitting and receiving equipment. An initial objective for the study is to formulate concepts for the modeling of such systems, and these may eventually

be adaptable to future business applications [119].

Models relying on both simulation and queuing theory which have been developed by a private firm have been described by Kershenbaum [88]. The models are oriented toward the planning, design and development of large telecommunications networks with processing located at a single site. By using analytical models to describe certain component subsystems, such simulations can be created at a high level of aggregation and still claim a relatively high degree of accuracy in the results produced. The concept is that propounded by Teorey and Merten and referred to as "two level simulation" [146]. Models of this type can be expected to produce better results only if the mathematical assumptions made in the low level models are valid, and if appropriate input data exists at a high level of precision.

Kleinrock has done extensive work in applying analytical models to computer networks [92] [94] [95]. He has also used simulation models to evaluate communications protocols in packet-switched networks such as ARPANET [93]. This model has been quite valuable in considering design alternatives for experimental networks of large-scale computers but is inappropriate for those involving minicomputers. The large amount of overhead required for ARPANET type protocols make their efficacy for business applications questionable [96].

Bowdon, Mamrak, and Salz used simulation as a tool for evaluating alternative methods of improving system

performance in network computers. Their model was developed for a single IBM 360/75 and extended to represent a network of three such processors. Simulation experiments were devised to evaluate alternative priority assignment algorithms and predict the effects of load-leveling between different central processors. The model, however, was not intended to be used in configuring a network or for applications involving minicomputers [17] [18].

Two other models are being developed with the objective of providing tools useful in the design of computer networks. The model of Segal and White will ultimately represent a national network interconnecting several universities, colleges, and research institutions. The network will serve to extend the processing capabilities of participating organizations. The simulation model is additionally designed to assist administrators in understanding the advantages and disadvantages of joining the network [138].

A network simulation model is being developed by Linsenmeyer and Ligomenides at the University of Maryland [104]. It requires the specification of hardware, software, and job description statistics at a high level of detail. Their goal is to be able to predict delays in a network due to processor job execution, network communication, and queues. In creating the model, they have assumed that all processors operate in batch mode and that each processor is equally accessible to all jobs. These assumptions will not

be typically valid ones in a functionally dedicated minicomputer network where programs will be normally resident in certain processors, and where a significant amount of processing will be in interactive or real-time modes. The strength of their approach is its flexibility; however it requires a type of data which normally is not available during the planning and development phases of the information systems life cycle.

Summary

In summary, numerous researchers have recognized the need for evaluating network design alternatives prior to the installation of such a system. Much has been achieved in developing simulation methodologies which overcome the peculiar problems associated with modeling computer systems. Yet very little has been accomplished in devising means to apply the technology to the specific informational needs of organizations. Managers who wish to achieve the advantages of decentralization inherent in the practice of distributed computing do not have access to simulation models which will aid them in this objective. A significant need exists for network models which can aid in applying recently developed data communications methods to commonly existing business and organizational requirements.

CHAPTER V

A GENERALIZED RING NETWORK SIMULATOR

A simulation model is a useful tool for evaluating possible changes in the design of a system without requiring changes in the system itself. During the system planning phase of the management information system life cycle, such a model is useful in aiding in the preliminary design of the system. At this stage the system does not exist and decisions of long-term consequence must be made regarding the form which it will take. Simulation models are expensive to develop, however. If the tool is to be a practical one in developing information systems, it must be flexible enough to describe a number of different systems in a variety of business applications. A generalized model, capable of representing different applications through proper specification of model parameters, is a viable approach because it can be modified and reused in the design of other installations.

The purpose of this chapter is to describe such a model. First a general description of the type of installation for which this model is applicable will be presented. Then the manner in which the characteristics of an installation are specified to the model is discussed.

Finally the construction of the model itself is described along with a method of interpreting its output.

General Description

A generalized simulation model should be flexible enough to be applicable in a number of situations, and yet specific enough to provide accurate and useful information. Thus it should adequately represent a particular type of system without being restricted to just one application of that type.

The objective of the model created in this study is to serve as a tool in the design of minicomputer networks in business applications. The jobs performed by the processors are therefore assumed to be those which are commonly found in business information systems. This precludes the use of the model to represent minicomputers in real-time, process control applications as they are often utilized in industry. A further assumption is that the processors are interconnected in a ring network. The advantages and disadvantages of this topology have been discussed in Chapter I.

The Operating System

Network operating systems are a subject of continuing research among computer scientists, and several of them which are implementable in a ring topology have been developed. Associated with the operating system is a

network protocol, which ensures that messages between processors are formatted in a fashion that permits interpretation of the message by all of them. An interprocessor message is a string of binary digits, or bits, which is transmitted through a medium such as cables, microwave radio links, or communications satellites. In most systems the message is accompanied by a header and a trailer, two sequences of bits appended to the message at its start and at its end. The header and trailer delineate the actual text being transmitted and provide control information to the processors in the network. This includes data such as the destination processor, the processor initiating the message, check bits or totals, and perhaps the message length. The entire structure is referred to as a packet, and a lengthy message may be segmented into several packets which are transmitted independently.

It is evident that such network operating systems can become very complex, much more so than is practical to implement using minicomputers. The type of operating system assumed in this model is that developed by researchers at the University of California at Irvine. This type, sometimes referred to as a ring switched network, utilizes shift registers to interface the processors with the ring. Each processor has an associated shift register so that packets move from one register to another around the ring. When a message packet is present at a processor's register, the processor decodes the header field containing the

destination address. If the packet has reached its destination, it is removed from the ring. Otherwise it is transmitted to the next register where the procedure is repeated. One processor is designated as the control unit, to ensure that any packet does not remain on the ring indefinitely.

A major advantage of such an operating system is its simplicity. It has been shown to perform adequately in minicomputer networks. The method has not, however, been adopted by minicomputer manufacturers, and is not implementable in other than a ring configuration. However, its comparatively low cost and reliability make it suited for functionally dedicated networks in which the processors are not widely separated.

The Peripheral Devices

In a functionally dedicated network each processor provides data processing support for a particular segment of an organization. As a result each is required to have its own set of peripheral equipment. In a minimum configuration, each processor location would be expected to have an input device, an output device, and a direct access mass storage device. Some sites might also be equipped with a tape drive if long term storage is needed for its files.

In developing the model, it was desired that the methodology be able to specify a minimum cost configuration which satisfies an organization's needs. This configuration

was assumed to include at each site a processor, an input/output console with display, a low speed printer and some form of disk storage. Each peripheral device is connected to a single processor and the processors are connected through shift register interfaces. This single site configuration is illustrated in Figure 22. The entire ring network was assumed to contain a maximum of ten sites.

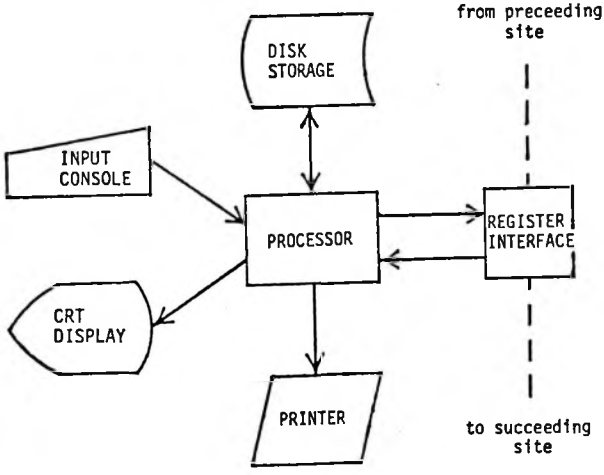
The characteristics of the peripheral devices affect the overall performance of each site and of the network as a whole. In most new systems the device types are chosen by management when configuring the network, and are thus decision variables in the design of the system. The model allows an analyst to specify printer, disk, and input console characteristics. The effects of total system performance due to changes in the peripheral types can thereby be predicted.

The Processor

In a functionally dedicated network, each processor is utilized for one or a few closely related applications. And unlike large multiprocessing systems, most minicomputers are serial processors. Only one job is resident in main storage and being executed at any point in time. This fact makes a generalized model of a minicomputer system considerably less complex to construct than an equivalent model of a systems using multiprocessors. Furthermore, most business applications programs are file processors and are

FIGURE 22

SINGLE SITE CONFIGURATION



therefore relatively straightforward in their operation. It is possible to build reasonably accurate models depicting computer processing in business applications without a detailed representation of the internal workings of the central processor. In doing so, such a model is more flexible because it can be modified easily and used to represent a wider range of processor types.

As was pointed out in Chapter IV, the level of detail appropriate for a model depends on its objectives. The model created in this study is intended for use in evaluating system configuration alternatives, but not for evaluating processors or predicting processor performance. By representing the ring network at a high level of aggregation, this model is more flexible in its use and less expensive to develop and utilize. Given the objectives of the model, the resulting accuracy is adequate for its intended applications. In Chapter VI it will be used to evaluate a number of design alternatives for which interprocessor data communications, but not internal processor operations, are critical.

The model uses a single data variable, an estimated compute time, to represent the time required for the actions of the processor between file accesses. This value can be predicted with reasonable accuracy from results provided by software monitors, and is the same technique used in a model of an on-line system by Traub and Zachman [148, p. 19]. The advantages of this approach far outweigh the disadvantages,

provided that undue reliance is not placed on those model output statistics which are highly dependent on processor timings.

The model processor representation is also highly aggregated in that main storage core limitations are not considered. For any applications program, the amount of core storage required for its execution is extremely difficult to estimate with sufficient accuracy before the program is coded and executed in a processor. It is felt that the inaccuracies associated with such estimates would make any output from a simulation model relating to core utilization unusable. Because the model depicts a serial processing environment, core utilization by applications programs does not significantly affect system performance and is not a serious concern as it would be in a multiprocessing system.

Data Input

The model utilizes two types of data. One, the computer system description, specifies to the model relevant characteristics of the computer hardware which constitutes the physical network. The other data type is the job stream description. It is the mechanism by which the anticipated data processing load is introduced into the model. The model then provides statistics describing the operation of the hardware under conditions of the specified workload. By changing either of the two data types, changes in the

performance of the network can be predicted arising from alternatives in either hardware or operating environment.

Computer System Description

The model was created using a common simulation language, the UNIVAC General Purpose Systems Simulator (GPSS). Hardware characteristics are introduced by means of GPSS "variable" and "function" statements. This method allows each hardware characteristic to be specified only once in the model, and then referred to in later references by its mnemonic, or statement label. Changes to the model representing changes to the hardware itself are thereby facilitated. A list of the input data which constitutes the computer system description is provided in Table 6. All times are specified in milliseconds.

The types of data required by the model are readily available from manufacturers published data or other sources. Representative processing times (PROC.TIME) have already been discussed. Published hardware specifications provide data on channel transfer rates (CHAN.TIM), input transfer rates (XFER.TIME), comparison times (COM.TIME), disk rotation and seek times (ROT.TIME and SEEK.TIM), and printing speeds (PRT.LINE) [41]. The number of processors in the network (NO.PROCS) is a design variable subject to managerial control. The remaining input data items are the interface read-in time (RIP.TIME), the message packet size (PAK.SIZE), and the time required for keyboard entry at a

TABLE 6

COMPUTER SYSTEM DESCRIPTION

INPUT DATA

DATA MNEMONIC	CONCEPTUAL INTERPRETATION
PROCTIME	a representative CPU processing time for one data block
CHAN.TIM	the data transfer rate between mass and main storage, for one block of data
XFER.TIM	the time required to read one block of data from console input buffer to main storage
COM.TIME	time required for the CPU to execute a data comparison
NO.PROCS	the number of processors in the ring
PAK.SIZE	the message packet size in blocks
RIP.TIME	the time required for a ring interface to read in an entire message packet
ROT.TIME	time required for a disk rotation
SEEK.TIM	time required for a disk arm to access a desired cylinder
INP.TIME	time required for data input at a console
PRT.LINE	estimated time for outputing one line at a printer

console (INP.TIME). They may be accurately estimated from observations on existing systems. They are also to a limited degree subject to managerial control.

A significant advantage offered by this model is that data for its hardware description is readily available to a systems manager. Furthermore, the data can be easily altered, enabling the model to represent a wide range of hardware configurations without major reprogramming effort. In its method for specifying the computer system description, the model is a generalized one and can be used to evaluate several systems alternatives. Similar advantages exist relative to the job stream description.

Job Stream Description

The job stream description specifies to the model the expected data processing workload. For the model to be useful in making systems design decisions, this data should be obtainable from an existing business environment. By restricting the type of computer program to those commonly found in business applications, the model created in this study utilizes data which is readily available to characterize the processing requirements.

Job stream descriptions are specified to this model using GPSS transaction parameters. A single transaction is used to represent any data processing job stream. Each GPSS transaction defines six specifications containing a total of 31 parameters which are used to describe the processing

operations constituting the job stream. A specification may call for one of five operations, or file actions. They are (1) read from an input console, (2) read from mass storage, (3) write to mass storage, (4) print a line on a printer, and (5) no processing. These operations may be initiated from any processor site in the network for any other processor site using a file designation parameter. Parameters used as switches also may require a comparison, as is often used in business file processing, and that printing be initiated. The last parameter in each specification indicates the number of transactions, or repetitions of the file action. A summary of one GPSS transaction's parameters is provided in Table 7, and Table 8 illustrates the specification parameter codes as interpreted by the model.

The model is executed by first running a smaller GPSS program. This Jobtape Write program generates GPSS transactions describing the system to be simulated in the manner just described, and stores them on a computer tape. Then in actual execution these transactions are read into the model from the tape and the simulation procedure begins. In this fashion a complete job stream description is produced in a form interpretable by GPSS prior to execution of the system model itself. Job stream characteristics can readily be altered by changing specification parameters and executing the Jobtape Write program.

As a GPSS transaction is read into the model, it

TABLE 7

JOB STREAM DESCRIPTION

GPSS TRANSACTION PARAMETERS

PARAMETER NUMBER	MNEMONIC	CONCEPTUAL INTERPRETATION
1	JOB.NAME	IDENTIFIER FOR THE JOB STREAM
2	SP.FA1	FIRST SPECIFICATION FILE ACTION CODE
3	SP.FS1	FIRST SPECIFICATION FILE DESIGNATION CODE
4	SP.CI1	FIRST SPECIFICATION COMPARE INDICATOR CODE
5	SP.PI1	FIRST SPECIFICATION PRINT INDICATOR CODE
6	SP.TRAN1	FIRST SPECIFICATION NUMBER OF TRANSACTIONS
7 -11	SP.--2	SECOND SPECIFICATION
12-16	SP.--3	THIRD SPECIFICATION
17-21	SP.--4	FOURTH SPECIFICATION
22-26	SP.--5	FIFTH SPECIFICATION
27-31	SP.--6	SIXTH SPECIFICATION

TABLE 8

JOB STREAM DESCRIPTION

PARAMETER SPECIFICATION CODES

File Action Codes (SP.FA-):

0 - No processing
1 - Read input
2 - Read mass storage
3 - Write mass storage
4 - Print

File Designation Codes (SP.FS-):

Digit 1 - 10 denoting processor
at which the file resides

Compare Indicator Codes (SP.CI-):

0 - No comparison
1 - Comparison after read

Print Indicator Codes (SP.PI-):

0 - No printing
1 - Print one line
after comparison

defines a new set of five parameters, called the "active specification parameters". The active specification is used by the model to determine which file action code is to be implemented and the processor on which it is to take place. The model first assigns specification 1 to active status. Once appropriate operations have taken place for the required number of repetitions, then specification 2 is assigned to active status. This procedure is continued until all six specifications have been implemented. In this manner the model simulates the actions of a minicomputer which processes in batch mode a file of transactions entered interactively.

This approach to job stream characterization is accurate as long as processor functions can be represented by one of the five file action codes. Most business data processing applications are in this category. It would not be appropriate for representing compute bound applications such as linear programming or simulation models which are sometimes used by businesses for planning purposes. But as was pointed out in Chapter II, models of this type are seldom implemented on minicomputers. Thus this shortcoming is not a serious one for a model of a minicomputer network.

Model Construction

The simulation model consists of four GPSS macros and a control segment. A macro is used to represent each of the four operations specified by the file action codes in

the GPSS transaction parameters. They are the Print macro, the Input macro, the Read Disk macro, and the Write Disk macro. The control segment is used to interpret job stream specifications and to represent the actions of the processors and the ring interface devices. In GPSS each printer, input console, disk device, and processor are represented as facilities. Each ring interface device is represented as a storage.

Because the model can be used to represent up to ten of each of these components, its size was greatly reduced and logic simplified by indexing them. Another parameter, a pointer, is assigned to each transaction during model execution. The value of the pointer parameter may range between 1 and the number of processors in the modeled network. All references to any component are made in terms of the current value of the pointer, which is used in determining the index. Conceptually, the pointer represents the site location at which a data processing job is being executed. Each peripheral device is indexed by the value of pointer plus some constant. This allows the unique specification by means of indexes for all consoles, printers, and mass storage devices in the system. Because the maximum number of each of the components in the network is ten, the constant added to the value of pointer is always a multiple of ten. This scheme produces GPSS facility and queue indexes which are summarized in Table 9.

TABLE 9

MODEL CONSTRUCTION

SYSTEM COMPONENT DESIGNATIONS

GPSS REPRESENTATION	RANGE OF INDEX VALUES	CONCEPTUAL INTERPRETATION
Facilities:	1-10	Processors
	11-20	Channels
	21-30	Disk Devices
	31-40	Input Consoles
	41-50	Printers
Queues:	1-10	Ring Input Buffers
	11-20	Ring Output Buffers
	21-30	Channel Queues
	31-40	Printer Queues

Macro Construction

In GPSS, a macro is used to replicate a series of statements within the model without recoding them each time they are required. Macros were used to represent the operations specified by file action codes. The Print and Input macros are very simple, allowing a GPSS transaction to hold the appropriate device for the specified length of time. In the Read Disk macro, the GPSS transaction utilizes a channel to gain access to a mass storage device. The device is held a length of time required for a seek and a disk rotation, and the channel is utilized for returning the data to the processor. The Write Disk macro is similar except that two file accesses are required. Since a disk is first read before it is written on, the first access in the macro represents a read and the second a write. Flowcharts for these macros are provided in Figures 23 through 26.

Control Segment Construction

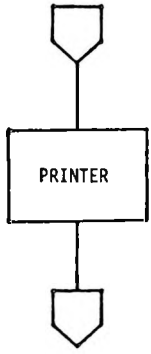
The control program is the most complex segment of the model and performs three essential functions. It interprets the job stream description specifications and directs the model through the appropriate simulated operations. It represents the actions of the network processors in performing data transaction comparisons and input/output operations. And finally it models the interprocessor communications through the ring.

The control segment reads the first GPSS

FIGURE 23

MACRO CONSTRUCTION

PRINT MACRO



from Control Segment

Time: PRT.LINE

to Control Segment

FIGURE 24

MACRO CONSTRUCTION

INPUT MACRO

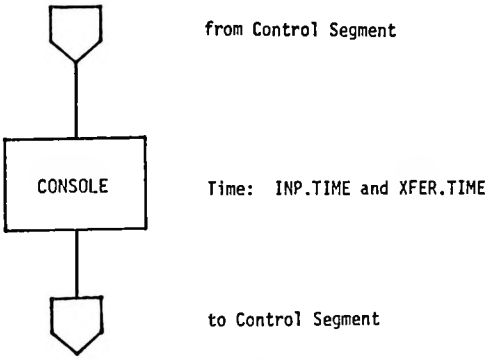


FIGURE 25

MACRO CONSTRUCTION

READ DISK MACRO

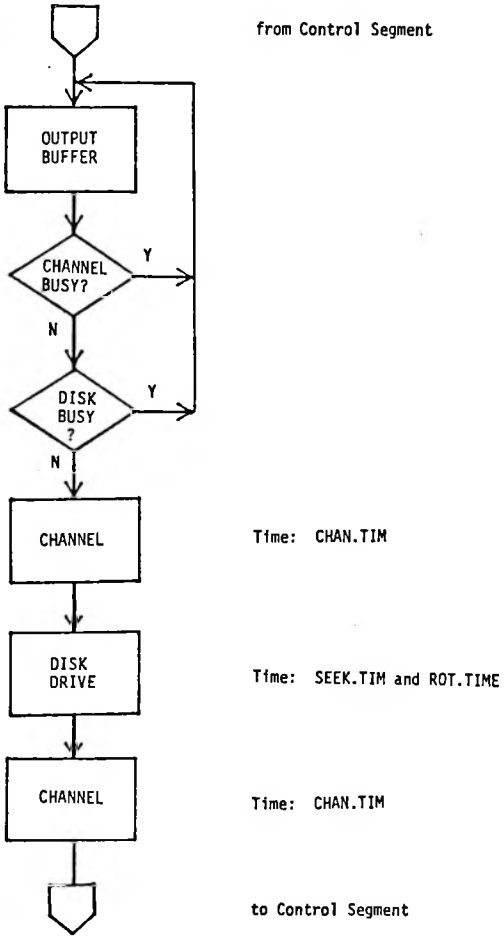
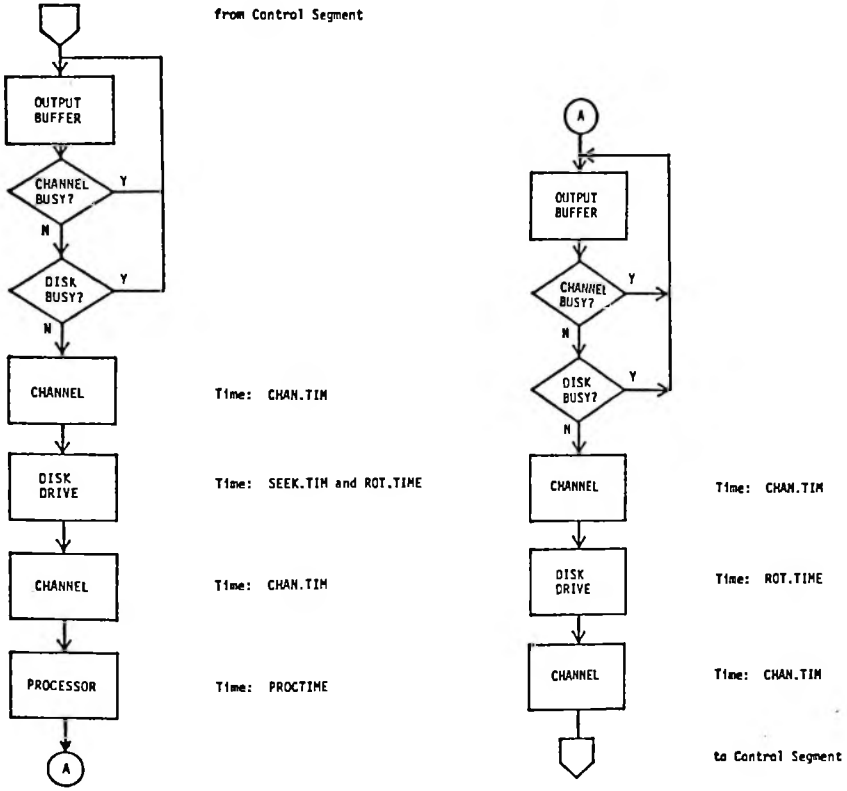


FIGURE 26

MACRO CONSTRUCTION

WRITE DISK MACRO



transaction, representing a data processing job, from tape. It then assigns the first job description specification to active status and sets the value of pointer equal to the first file specification. This is conceptually equivalent to assuming that, in the minicomputer network, a job will always be initiated from the processor at which the job's programs are stored in memory. The assumption is quite reasonable in a functionally dedicated network.

The model then begins to simulate computer processing with the specific nature of the processing determined by the specification's file action code. When processing for the first specification is complete, the second of the six specifications in the GPSS transaction is assigned to active status. Processing is then simulated which is specified by the second file action code. The process is repeated by the control segment until all six specifications have been assigned to active status and completed. After this occurs the GPSS transaction is terminated, representing completion of the data processing job stream.

The procedure just described is that of the interpretation and processor functions of the control segment. These are summarized by the flowchart in Figure 27.

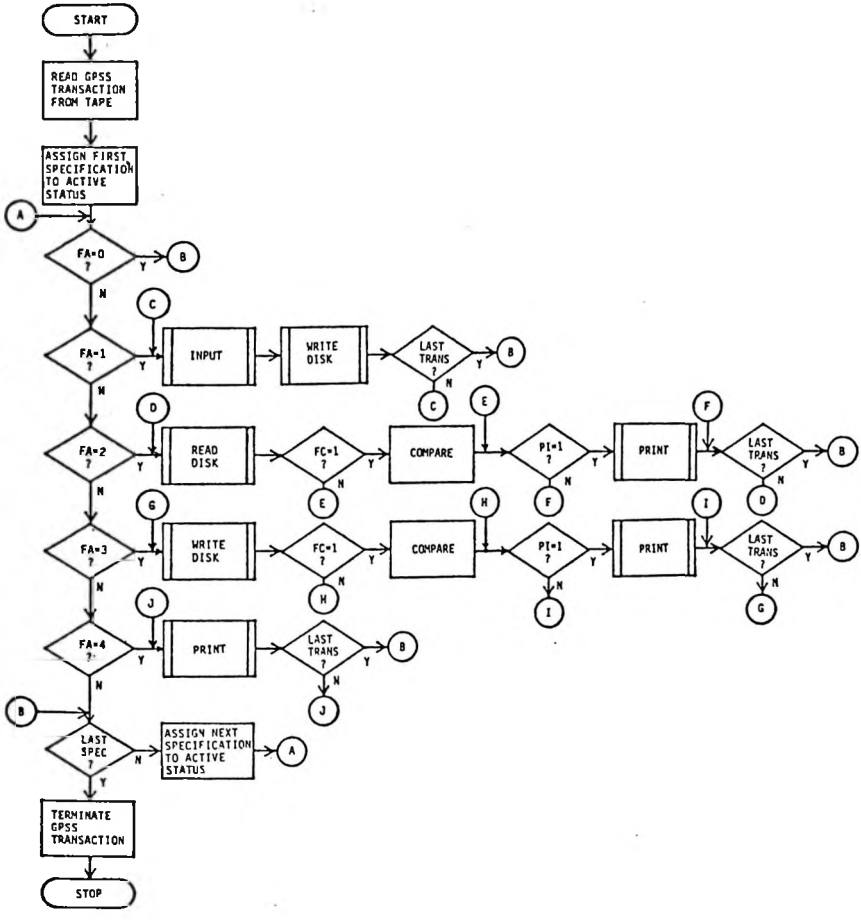
The control segment also is used to model interprocessor communication. Each job stream description specification contains a parameter referred to as a file

FIGURE 27

MODEL CONSTRUCTION

CONTROL SEGMENT INTERPRETATION

AND PROCESSOR FUNCTIONS



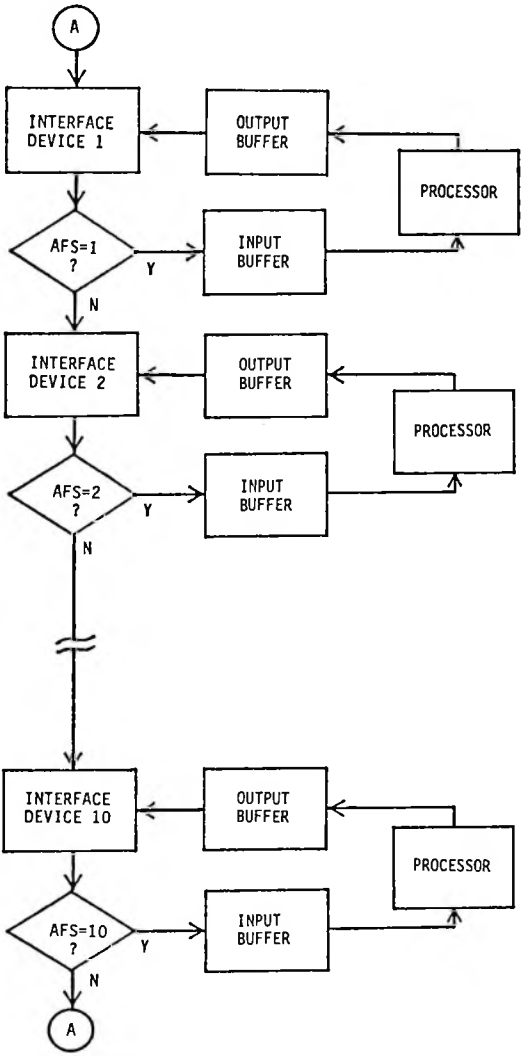
designation, denoted SP.FS- in the model. This parameter designates a processor on which operations specified by the file action code are to take place. At any point in simulated time, the processor site at which a job is being executed is determined by the value of the pointer parameter. Before processing is begun, the control segment compares the value of pointer to the active file designation parameter. If they are the same, then processing is to continue at the job's current site. If they differ, processing must continue at a different site. The GPSS transaction enters the ring at the interface device indexed by pointer. The nointer parameter is then incremented until it is equal to the active file designation parameter, at which point processing can begin. This procedure is conceptually equivalent to a message packet moving around the ring until the processor site specified in the header field is located. It is illustrated by the flowchart in Figure 28.

The model is constructed to monitor and predict execution times for a predetermined job load on a specified ring network configuration. By making minor changes to the input data a wide range of design alternatives can be evaluated. The model does not predict main storage requirements for system applications nor does it require specification of minicomputer main storage and mass storage capacities. The model's emphasis on timings is a result of the performance measures chosen for evaluating a

FIGURE 28

MODEL CONSTRUCTION

CONTROL SEGMENT COMMUNICATIONS FUNCTION



minicomputer network. These performance measures were referred to in Chapter I and can be established from analysis of the model's output.

Model Output

Because the model is written in GPSS, it produces all the standard output of that simulation language. This form of output can provide many valuable insights into the operation of a minicomputer ring network. Nevertheless, to establish measures of computer performance which are meaningful to the information system user, a certain amount of analysis is required of the standard GPSS output.

Performance Measures

In the introductory chapter three performance measures for computer systems were cited which have been widely accepted in the industry. They were availability, thoughput, and turnaround time. In addition it was suggested that a measure of facility utilization would also be useful.

Availability may be measured by determining the percentage of time a system is functioning in useable condition. Conversely it may be measured by monitoring system downtime, the percentage of time that it is inoperable. Downtime in computer systems generally arises from two sources, hardware malfunction and operating system software failures. Engineers often use statistics such as

the mean time between failures (MTBF) and the mean time to repair (MTTR) to indicate the reliability of hardware. Although it may be possible to introduce these statistics as parameters in simulation experiments, no attempt to do so was made in constructing this model. Hardware malfunctions are relatively uncommon in equipment provided by reputable manufacturers [87, p. 150].

Software failures, on the other hand, are a more common source of system downtime. Unfortunately these failures are extremely difficult to predict, and software failure rates can best be determined from historically observed data. Because minicomputer networking technology is relatively new, this type of data is not widely available. For these reasons the performance measures established for this study are throughput, turnaround time, and utilization.

Throughput has been described as the steady state work capacity of a system. It is related to the speed of processing a particular workload. This measure could be expected to vary as changes in the characteristics of the workload occur. The wide range of workload compositions which exist in data processing applications have made truly representative throughput measures difficult to specify.

In contrast, turnaround time is comparatively easy to determine. It is the delay between the presentation of input to a system and the receipt of output from it. Turnaround time is perhaps the most meaningful performance

measure from the user's point of view.

The third measure used in this study is utilization. It is the fraction of time available that a system, processor, or peripheral device is actually in use. Typically high utilization rates can be expected when turnaround time is high, and conversely low turnaround time accompanies low utilization. From the point of view of the organization purchasing a data processing system, high utilization may seem desirable because it indicates efficient use of existing facilities. However, it does not necessarily indicate efficient allocation of all company resources if the ability of personnel to get work through the system is limited.

Standard GPSS Output

The GPSS model's standard output is highly informative, and can be used to predict the performance measures just described. The elapsed clock time for the simulation experiment is indicative of the amount of time required to process the entire workload. Utilization figures are given for all components in the system as a fraction of elapsed simulation time. Accompanying data tabulates the number of times a component in the system is accessed and the average time required for each access. The model uses GPSS queue blocks to represent system buffers, and for them maximum and average queue lengths are presented. The tabulate feature of GPSS is a convenient

means of providing nonstandard data and of observing frequency distributions. Nevertheless the model output must be translated into terms meaningful to the system manager if the model is to be an effective design tool.

The model's standard output has been modified to produce statistics on turnaround time directly. This modification produces a table showing the elapsed time to completion of all job streams processed. An average turnaround time is also calculated, which is a measure of the overall effectiveness of the system in processing a given workload.

For a given workload composition, a measure of throughput can be calculated from the elapsed simulation time for the completion of the simulation run. A convenient unit of measure for throughput is data processing transactions per eight hour day. Data processing transactions include all those items which constitute the daily workload, such as cash receipts, customer orders entered, and production jobs scheduled. The total number of transactions is taken from the job stream description discussed in a previous section. Throughput can be calculated from the relation:

$$T = \frac{N_t}{T_s} \cdot 2.88 \cdot 10^7$$

Where T = throughput for a given workload

N_t = number of data processing transactions in the workload

T_s = elapsed simulation time.

Utilization may be computed for each component in the system from statistics included in the standard output. A satisfactory measure of system utilization, then, is the maximum utilization observed for any component. The appropriate relationships are defined by:

$$U_i = \frac{A_i \ T_s}{2.88 \cdot 10^7}$$

$$U_s = \max_i (U_i)$$

Where U_i = utilization of component i

A_i = average utilization for component i from standard GPSS output

U_s = system utilization.

Summary

A model has been described which is a useful managerial tool in the design of one type of distributed information sytem. It can be used to evaluate a wide range of design alternatives in implementing a ring network of functionally dedicated minicomputers. The significant feature of this model, which differentiates it from other network models, is its user orientation. Data required to implement the model is readily available to any organization considering a system of this type. The performance of several specific configurations can be evaluated under a projected workload of the kind normally found in data

processing applications, and the results compared. Finally, the performance measures produced are in the form most meaningful to users of the system.

In the next chapter, the use of the model in the design of a distributed information system will be demonstrated.

CHAPTER VI

DESIGN PROCEDURE

The Generalized Ring Network Simulator promises to be a useful device for aiding in the design of minicomputer networks for business applications. It utilizes data which is readily available to a systems manager or designer, and provides performance measures which are relevant to the needs of the users of the system. Yet the aggregations necessary for simulating computer systems requires that an analyst exercise care in drawing conclusions from a model's output. A simulation model can only provide useful information in evaluating the design alternatives for which it is intended to be used.

The purpose of this chapter is to describe a procedure by which the Generalized Ring Network Simulator may be used in making major decisions concerning the design of such a network. First some of the design alternatives for which use of the model is appropriate will be described. Heuristic procedures will be presented that facilitate implementation of the model in evaluating these alternatives. Finally the model will be utilized in the design of a minicomputer network for the typical business application presented in Chapter III.

Design Alternatives

In designing any computer based system, attention must be directed to both the hardware and the software alternatives which are available. These alternatives are constrained by certain organizational and financial limitations peculiar to the business enterprise implementing the system. The Simulator can be used to provide answers to three questions encountered by a system designer when implementing a distributed system. They are:

(1) How extensive should the network be?

(2) Where should each applications program and data file be located in the network?

(3) What types of peripheral devices should be used to provide satisfactory system performance?

The manner in which each of these questions can be answered will be discussed.

Network Size

The size of a minicomputer network is determined by the number of processors in it. Ideally a minicomputer based distributed information system will have a processor located at each decision center in the organization. Each decision maker would then have direct access and control over the information processing functions which influence decisions. Often, however, limitations on the organization's financial resources prevent such an extensive

network. Furthermore, the processing workload at a decision center may not be sufficient to justify the expense of equipment dedicated entirely to that center. The system designer is required to evaluate a tradeoff between the costs and the benefits associated with each possible network size.

In a network composed of a number of identical single site configurations, hardware costs can be expected to increase in approximately linear fashion as the number of sites increases. If system performance measures are user oriented, then the benefits associated with a network configuration will vary in proportion to the performance of the network. And since the benefits associated with a computer based information system are usually difficult to quantify, surrogate criteria need to be established for evaluating the cost-benefit tradeoff. Two such criteria are network size and network performance.

For a given workload, there are two conflicting factors which will affect the performance of a network as its size increases. The first will be referred to as "processor contention". In a small network several different applications will be sharing a single processor. Because most minicomputers do not operate in multiprocessing mode, an application cannot be executed until another application currently using the processor is complete. This contention between different applications, or jobstreams, for a processor will decrease the overall throughput of the

system and increase turnaround time for the user. As the number of processors in the network increases, performance will improve because applications are more widely distributed and processor contention is decreased.

The second factor which affects performance will be referred to as "ring contention". In a small network, a single processor site will have resident several system files. Programs also resident at this processor can access these files without utilizing the communications features provided by the interconnections in the ring. As the number of sites is increased, the files will become more widely distributed. Any applications program will be required to utilize data communications via the ring more frequently to access these files. The capacity of the ring to transfer requests for data then becomes a limiting factor on the performance of the network. Applications programs begin to contend for the communications capabilities of the ring rather than the file accessing capabilities of the processors.

The two conflicting factors, processor contention and ring contention, make system performance difficult to predict analytically. The relationship between network size and performance cannot be expected to follow a simple linear relationship, but rather a polynomial one resulting from an interaction of the two factors. The Generalized Ring Network Simulator can provide satisfactory performance predictions in spite of the difficulties introduced by these factors,

and can therefore be used to evaluate the size-performance and cost-benefit tradeoff. The procedure for determining this tradeoff is a heuristic one relying on multiple executions of the Simulator.

A Network Size Heuristic

The most cost-efficient number of processors in a network can be determined using a network size heuristic algorithm. This algorithm, summarized in Figure 29, attempts to optimize network size by locating that number of processors at which the marginal improvement in performance becomes negative. This point can be observed if a measure of network performance such as throughput is graphed on the ordinate and the network size on the abscissa. The marginal improvement in throughput is the first derivative of the equation of this line, and becomes negative when there is a change in the slope of the line.

Since the use of additional processors incurs additional expenditures for equipment, the rate of change of network cost is always positive as network size increases. Therefore when the marginal improvement in throughput becomes negative as a processor is added to the network, the marginal change in cost always exceeds the marginal performance improvement, and hence the optimum network size has been attained. This argument, based on microeconomic theory, is the justification for the network size heuristic algorithm [75, p. 71] [113, P. 189]. Similar arguments hold

FIGURE 29

NETWORK SIZE ALGORITHM

1. Determine the largest possible number of processors in the network, based on organizational constraints.

2. Execute the Generalized Ring Network Simulator for this maximum number of processors. Record the predicted throughput and turnaround time for the network.

3. Decrease the number of processors by one and execute the Simulator, recording the resulting performance measures.

4. Continue to execute the Simulator, each time decreasing the network size by one processor and recording the results. Cease when a network of two processors has been evaluated.

5. Graph the two performance measures as a function of the number of processors in the ring.

6. The most cost-efficient number of processors is that where either (1) the slope of the throughput line becomes negative, or (2) the slope of the turnaround time line becomes positive.

7. If the two performance measures provide contradictory results, subjective factors may be used to eliminate one of the two remaining alternatives.

when turnaround time is used as a performance measure, except that improved performance accompanies a decrease in turnaround time. When the marginal change in turnaround time is positive, the marginal improvement in performance is negative.

In the type of experiment just described, the decision or controllable variable is the number of processors in the network. The uncontrollable variables include the hardware description of the network devices and the characteristics of the data processing workload. The endogeneous variables are the performance measures defined by the model: Throughput, turnaround time, and utilization. Results of employing the network size algorithm will be reported in Charter VII.

File Locations

Another question that must be addressed in designing a distributed system of minicomputers is concerned with the software configuration of the network. Its software system consists of a number of applications programs and the files which these programs utilize. Each program and file resides in storage at a minimum of one node of the network. A system designer must decide which files and programs to assign in residence at each site.

One alternative in distributed processing networks is to replicate some or all of the files at each processor. This practice has been successfully employed in networks of

large-scale processors such as that developed at Celanese [27]. However such a replicated data base requires that periodically all copies of any file be updated to make them identical. In a small network of large-scale processors the replicated data base is a good solution. But in a network of several small processors, the overhead associated with maintaining identical files soon makes the procedure infeasible. A better approach for a minicomputer network is a partitioned data base.

In a partitioned data base each system file is resident at a single network node. When any program resident at a node needs to access a file resident at a different node, it does so using the communications facilities of the network. The problems associated with maintaining identical files at different locations are thereby avoided. In minicomputers, where it is desirable to keep operating system complexity at a minimum, the replicated data base is preferable. For this reason the Simulator assumes that unique software and files will be maintained at each node.

With a partitioned data base the system designer still must decide at which site to locate each system file and applications program. The best file location distribution is highly dependent on the characteristics of the data processing workload. The Generalized Ping Network Simulator, because of its ability to characterize expected workloads in evaluating alternative system designs, is a

valuable tool in choosing a file location distribution for a minicomputer network.

A File Location Heuristic

A basic concept of distributed processing requires that program execution take place within the affected operating segment of the business. The programs which constitute an applications jobstream should be resident at the processor site which utilizes the output from the jobstream. This fact specifies the location of all program files in the network.

In order to minimize the amount of data communications required, thereby avoiding ring contention, each file in the software system should be located at the processor site where it is utilized by an applications jobstream. But because some data files are accessed by more than one jobstream, and because each data file is unique, some data communication will almost always be required. This heuristic approach attempts to locate each data file at a site which maximizes overall system performance. It does so by determining those data file distribution alternatives which minimize the effects of both ring and processor contention.

The algorithm, summarized in Figure 30, begins by assuming an initial distribution. This assignment is made so that each data file in the system is located at a processor site where at least one applications jobstream

FIGURE 30

FILE LOCATION ALGORITHM

1. Obtain an initial assignment by placing each file at a processor where a jobstream accessing that file is located.

2. For each jobstream, determine those files accessed by the jobstream which are not resident at the same processor as the jobstream. These are candidates to be moved to a new location.

3. Execute the Simulator using the initial file assignment.

4. From the output of the Simulator, consider each processor in order of decreasing utilization. A file at that processor which is a candidate for relocation may remain a candidate if:

   A. It is accessed by a jobstream at a processor with a lower utilization, and

   B. There is more than one file at the processor location.

5. Eliminate those files as candidates for relocation which do not meet these two criteria.

6. Evaluate all remaining candidates for file location using the Simulator. Adopt a file relocation which improves mean turnaround time and/or throughput.

accessing that file is located. This provides a file configuration which minimizes ring contention in the network. Then possible file relocations are considered which will move a data file from a processor of higher to one of lower utilization. Each possible relocation could be expected to decrease processor contention, although possibly increasing contention for the communications capabilities of the ring. The Simulator is executed for the initial assignment and for those feasible file relocations specified by the algorithm. That file location scheme which produces the best performance measures is chosen for implementation in the network.

In the type of experiment just described, the controllable variable is the network's file distribution. Uncontrollable variables include the number of processors in the network, the devices' hardware characteristics, and the data processing workload. Endogenous variables are again the three performance measures defined by the model.

Equipment Choices

The Simulator can not only be used in evaluating network configuration alternatives, but it also can be implemented in the equipment selection process. Each processor site has been assumed to have a basic configuration consisting of processor, input console, printer, and mass storage device. As was pointed out in the previous chapter, the model is not at an appropriate level

of detail for making processor comparisons. It can nevertheless be used in predicting the effect on overall system performance of various peripheral device choices. By altering the hardware description in the input data, the characteristics of several types of printers and mass storage devices may be introduced into the Simulator on separate model executions. The results of the choices can then be compared.

The use of the Simulator for this purpose aids in the cost-benefit analysis of the peripheral devices. It is widely recognized that a more expensive line printer, for example, will generate more pages of printout than a serial printer in an equal length of time. In most situations it is not clear that this added printer speed will result in a significant performance improvement in the entire system. The two alternatives may be implemented within a simulation model. This will provide the decision maker with information helpful in determining if the performance improvement is worth the added cost of the peripheral. This and other equipment choice alternatives are explored in an actual application and the results reported in the next chapter.

In this context, the controllable variables are the hardware descriptions for the equipment at a single site. Uncontrollable variables are the file distribution, the number of processors, and the software characteristics of the workload. As in the other two design experiments, the

endogenous variables are throughput, turnaround time, and utilization.

An Application of the Procedure

In Chapter III a suitable business context for the implementation of a functionally dedicated minicomputer network was described. The procedure outlined in this chapter has been utilized in the design of such a system for that company. The applications of the heuristic algorithms will be discussed in this section and the results of the procedure described in Chapter VII.

System Description

A maximum network size of ten minicomputers was assumed for Agee Company. An initial assignment of the ten sites was made to several functional applications areas of the business based on organizational considerations. The appropriate jobstreams from Chapter III were then assigned to the processor at each applications area. Using the initial file assignment criteria of the file location algorithm, a file distribution was established for the ten processor network. The result was a single system topology, the performance of which could be predicted using the Simulator. This initial network topology is summarized in Table 10.

The use of the Simulator requires two types of data, a hardware description and a software description. The

TABLE 10

INITIAL NETWORK TOPOLOGY

TEN PROCESSORS

PROCESSOR NUMBER	APPLICATION AREA	RESIDENT JOBSTREAMS	RESIDENT FILES
1	Accounting	General Ledger Product Costing	General Ledger Standard Costs Product Costs
2	Production	Scheduling	Production Data
3	Marketing	Order Entry Sales Analysis	Open Order Product Customer
4	Purchasing	Purchase Order	Open Purchase Order
5	Payables	Payables	Detailed Payables
6	Personnel	Employee Data Payroll	Employee
7	Billing	Billing	none
8	Cash Receipts	Cash Receipts	Detailed Receivables
9	Inventory	Material Issues Material Receipts	Inventory
10	Disbursements	Disbursements	none

single site configuration was initially assumed to consist of an input console, a Digital Equipment Corporation Model PDP/8 processor, a DEC Model RK8 Disk Drive, and a Teletype Model ASR33 printer. Since the factor which limits input rates in an interactive system is normally the user's rate of entry, no specific input console was specified. The Teletype terminal could in fact be used for this purpose. From data provided by the manufacturer, appropriate performance characteristics were calculated and used for the hardware description. These data values are presented in Table 11.

Obtaining the appropriate job stream description was more complex, but the necessary data is included in Chapter III. Each job stream designed for the company was analyzed and the number of file accesses for each determined. A file access is represented by one file action specification within the model. The remaining parameters in a specification indicate the processor at which the file access is required, and whether or not the file access resulted in a data comparison or printing. The processor number is found from the initial system topology, and the requirements for comparing and printing are obtained from jobstream flowcharts. The number of business data transactions to be processed is obtained from observations of past volume as summarized in Table 2. All fifteen jobstreams could be characterized in this fashion.

In implementing the design procedure, two types of

TABLE 11

AGEE COMPANY HARDWARE DESCRIPTION

DATA MNEMONIC	DATA VALUE
PROCTIME	7
CHAN.TIM	4
XFER.TIM	11
COM.TIME	1
NO.PROCS	10
PAK.SIZE	1
RIP.TIME	1 X PAK.SIZE
ROT.TIME	Uniformly distributed between 5 and 40
SEEK.TIM	Uniformly distributed between 1 and 134
INP.TIME	Normally distributed mean 67584 standard deviation 20000
PRT.LINE	Normally distributed mean 5000 standard deviation 1000

Note: Basic time unit is one millisecond.

tests were made. Eight of the jobstreams were those which would be run on a daily basis, and six others would be run once each week. Another, the general ledger closing run, would normally be used only once each month. The first type of test evaluated the performance of the network in processing only the daily jobstreams. The second type represented the fourteen weekly jobstreams. Because the general ledger run occurs relatively infrequently, it was not considered suitable for inclusion in the jobstream description.

Model input to describe one jobstream is included in Table 12. A complete listing of the system jobstream description for a ten processor network is provided in Appendix 3.

System Tests

Once hardware and software descriptions had been introduced to the Simulator, it was executed and the performance measures computed. The jobstream description and model were then modified in order to represent a nine processor configuration. This required that one application area be elimirated from the network, and that its data processing functions be combined with those of another applications area. The Simulator was executed, its performance measures noted, and the model and jobstream description again modified. The procedure was continued until a two processor network had been modeled on both the

TABLE 12

EXAMPLE JOB STREAM CHARACTERIZATION

PURCHASE ORDER JOBSTREAM

PARAMETER NO.	PARAMETER	VALUE
1	JOB.NAME	2
2	SP.FA1	1
3	SP.FS1	4
4	SP.CI1	0
5	SP.PI1	1
6	SP.TRAN1	20
7	SP.FA2	3
8	SP.FS2	4
9	SP.CI2	0
10	SP.PI2	0
11	SP.TRAN2	20
12	SP.FA3	3
13	SP.FS3	1
14	SP.CI3	0
15	SP.PI3	0
16	SP.TRAN3	20
17	SP.FA4	4
18	SP.FS4	4
19	SP.CI4	0
20	SP.PI4	1
21	SP.TRAN4	60
22	SP.FA5	0
23	SP.FS5	0
24	SP.CI5	0
25	SP.PI5	0
26	SP.TRAN5	0
27	SP.FA6	0
28	SP.FS6	0
29	SP.CI6	0
30	SP.PI6	0
31	SP.TRAN6	0

daily and weekly jobstreams.

As each new network size was evaluated, a new file distribution was created. In each case the distribution was determined in the same manner as that for the initial ten processor network. Each system file was located at a processor site where jobstreams accessing that file were located. When an optimum network size was found, the file location algorithm was employed. This heuristic tested alternative file locations within the network to arrive at an acceptable system topology.

Once the network size and file locations had been determined, possible peripheral equipment changes were evaluated. As alternatives to the Teletype terminal two types of printers were considered. The Digital Equipment Model LA 180 serial printer and the Model LE8 line printer cost considerably more than the Teletype. Yet they are also faster and offer the capability of printing on special forms such as blank invoices and purchase orders. Another option for Agee is to use a different mass storage device. A lower cost form, the Digital Equipment Model RX8/11 Floppy Disk System was considered and the resulting change in performance noted. The modifications to the input data required for these tests are shown in Tables 13, 14, and 15.

Summary

The design procedure described in this chapter can provide a system designer with a wide range of useful

TABLE 13

AGEE COMPANY SERIAL PRINTER EVALUATION

HARDWARE DESCRIPTION

DATA MNEMONIC	DATA VALUE
PROCTIME	7
CHAN.TIM	4
XFER.TIM	11
COM.TIME	1
NO.PROCS	6*
PAK.SIZE	1
RIP.TIME	1 X PAK.SIZE
ROT.TIME	Uniformly distributed between 5 and 40
SEEK.TIM	Uniformly distributed between 1 and 134
INP.TIME	Normally distributed mean 67584 standard deviation 20000
PRT.LINE	Normally distributed mean 450 standard deviation 110*

* denotes changes from Table 11.

TABLE 14

AGEE COMPANY LINE PRINTER EVALUATION

HARDWARE DESCRIPTION

DATA MNEMONIC	DATA VALUE
PROCTIME	7
CHAN.TIM	4
XFER.TIM	11
COM.TIME	1
NO.PROCS	6*
PAK.SIZE	1
RIP.TIME	1 X PAK.SIZE
ROT.TIME	Uniformly distributed between 5 and 40
SEEK.TIM	Uniformly distributed between 1 and 134
INP.TIME	Normally distributed mean 67584 standard deviation 20000
PRT.LINE	Uniformly distributed between 346 and 348*

* denotes changes from Table 11.

TABLE 15

AGEE COMPANY FLOPPY DISK EVALUATION

HARDWARE DESCRIPTION

DATA MNEMONIC	DATA VALUE
PROCTIME	7
CHAN.TIM	22*
XFER.TIM	11
COM.TIME	1
NO.PROCS	6*
PAK.SIZE	1
RIP.TIME	1 X PAK.SIZE
ROT.TIME	Uniformly distributed between 5 and 166*
SEEK.TIM	Uniformly distributed between 1 and 169*
INP.TIME	Normally distributed mean 67584 standard deviation 20000
PRT.LINE	Normally distributed mean 450 standard deviation 110*

* denotes changes from Table 11.

information concerning a distributed minicomputer system's performance. However the model's level of aggregation is such that it will not produce results of extreme accuracy. Nor are the procedures outlined intended to determine the optimum network configuration. The network size algorithm requires the assumption of some set of file locations in the system. Similarly the file location algorithm assumes a predetermined network size. Although each may produce a locally optimum configuration with respect to its objective, the two together do not necessarily produce a global optimum, a best possible configuration for the entire system. They should, however, produce a satisfactory design and a reasonable estimate of the level of performance which can be expected from it.

CHAPTER VII

RESULTS AND CONCLUSIONS

As has been discussed in Chapters I and II, the centralized data processing structure has frequently been found deficient in providing the type of information processing support required by many organizations. Technological changes have made economically feasible a decentralized structure utilizing minicomputers, in which processing takes place within the information user's operating unit. A business situation has been described in Chapter III for which a network of minicomputers is a viable data processing alternative. Simulation models have been shown to be valuable tools in planning and designing computer based systems. The Generalized Ring Network Simulator, discussed in Chapter V, has been created specifically to aid in the design of minicomputer networks for situations such as that described. A design procedure has been presented which utilizes the Simulator to compare some of the alternatives available in implementing a distributed business system.

It is the purpose of this chapter to illustrate the results of applying the procedure to the typical business application of Chapter III. Based on the output of the

Simulator, suggestions will be made concerning the form which the minicomputer based system should take. Some conclusions will be drawn on the efficacy of using simulation as a design tool for the establishment of minicomputer systems in business applications.

Design Procedure Results

The Generalized Ring Network Simulator has been utilized to provide answers to the three questions encountered in the implementation of a minicomputer network. It is an aid in determiring the most cost-efficient number of processors to employ. It has been used to evalutate the effects of alternative file locations within the system. And it has been used to predict the effects on system performance of altering the basic single site configuration with changes in peripheral devices. The results of applying the suggested design procedures to Agee Company will now be presented.

Network Size

The network size algorithm was used to determine the minimum number of processors which would effectively support Agee's anticipated data processing workload. The maximum network size was assumed to be ten processors. The algorithm was employed for a lightly loaded system consisting of eight daily jobstreams, and for a heavily loaded system of fourteen weekly jobstreams. The

constitution of each load is shown in Figure 31.

The heuristic algorithm requires that the model be executed for decreasing network sizes and that the changes in performance be noted. These results are summarized in Table 16 and Table 17. Turnaround time and throughput have been graphed as a function of network size in Figures 32 through 35.

The effects of processor and ring contention are evident for both the lightly loaded and the heavily loaded systems. In Figures 32 and 33 a performance peak exists for network sizes of five and six processors. Below this size processor contention causes increased turnaround time and decreased throughput. As network size increases above these values, performance is degraded until a major discontinuity in the performance curves occurs. This discontinuity, at a network size of nine, results from the very lightly loaded system. Average processor utilization is only about five per cent at this point so that very little processor contention exists. Because the network is large in comparison to the processing load, there is also little ring contention. It may also be noted that a major decrease in maximum processor utilization occurs when the network size is increased from eight to nine processors.

In executing the daily jobstreams only eight of the twelve system files are utilized. The file locations accessed by these jobstreams were identical when the network size was either five or six processors. Hence identical

FIGURE 31

AGEE COMPANY DATA PROCESSING WORKLOADS

WORKLOAD	JOBSTREAMS INCLUDED
Daily	Purchase Order Material Issues Material Receipts Payables Disbursements Billing Cash Receipts Order Entry
Weekly	daily jobsreams plus: Work in Process Costing Finished Goods Costing Sales Analysis Employee Data Payroll Scheduling

TABLE 16

NETWORK PERFORMANCE MEASURES

EIGHT DAILY JOBS

NUMBER OF PROCESSORS	TURNAROUND TIME (MINUTES) AVG.	TURNAROUND TIME (MINUTES) MAX.	PROCESSOR UTILIZATION (%) AVG.	PROCESSOR UTILIZATION (%) MAX.	THROUGHPUT (TRANS/DAY)
10	44.4	79.8	4	16	1444
9	45.5	82.3	5	16	1400
8	76.8	108.8	6	23	1059
7	83.8	106.3	7	22	1084
6	91.6	104.1	9	22	1106
5	91.6	104.1	9	22	1106
4	100.6	112.5	10	23	1024
3	100.6	112.4	17	23	1025
2	127.9	131.0	26	27	879

FIGURE 32

NETWORK PERFORMANCE

TURNAROUND TIME VERSUS SIZE

EIGHT DAILY JOBSTREAMS

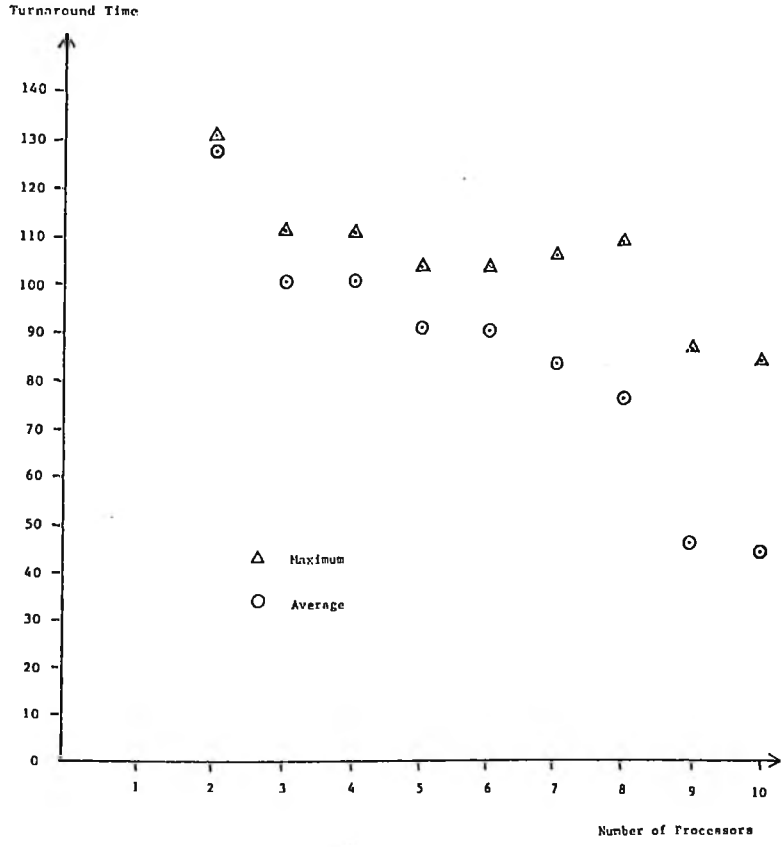
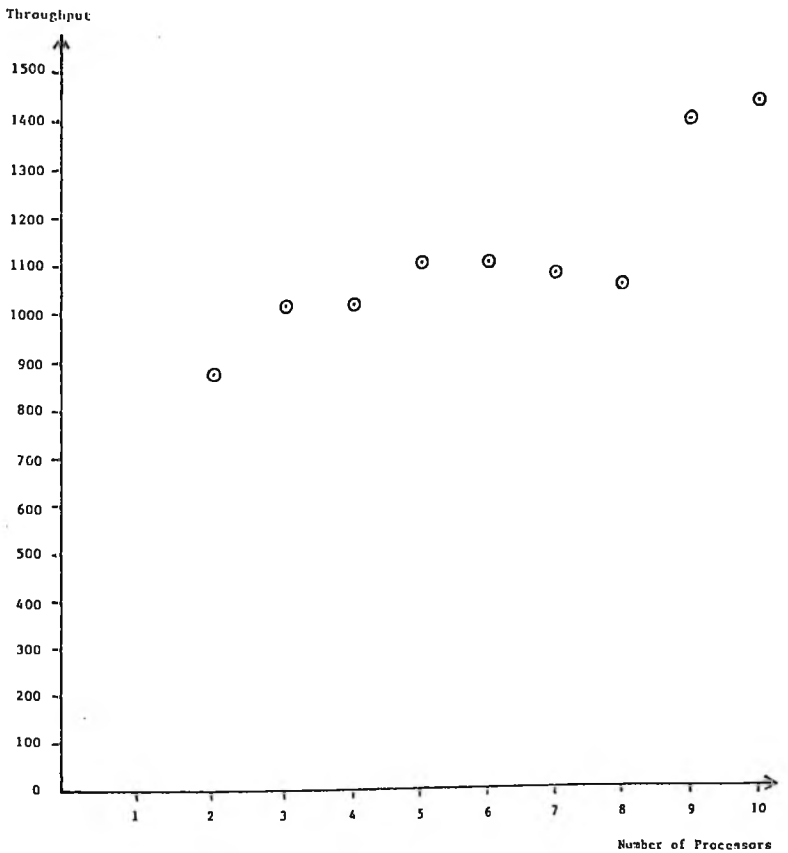


FIGURE 33

NETWORK PERFORMANCE

THROUGHPUT VERSUS SIZE

EIGHT DAILY JOBSTREAMS



performance characteristics were observed for these alternatives. Because daily jobstreams do not require the use of the entire network, and because they load the system only lightly, the optimum network size can better be determined from analysis utilizing the more heavily loaded weekly jobstream applications.

Fourteen of the fifteen applications jobstreams designed for Agee Company are executed on a weekly basis. The result of applying the network size algorithm for the weekly load is shown in Table 17. No significant improvement in turnaround time and throughput is evidenced when the network size is extended beyond six processors. Ring contention results in a gradual degradation in throughput and maximum turnaround time as the network size is increased beyond seven. In smaller networks processor contention causes a significant decrease in performance. These results are illustrated graphically in Figure 34 and Figure 35.

The network size algorithm stipulates that the optimum network size has occurred when a unit increase or decrease in size fails to result in an improvement in performance. Using this criterion and the results of Figures 34 and 35, the optimum size network for Agee Company would utilize six processors.

File Location

The management information system designed for Agee

TABLE 17

NETWORK PERFORMANCE MEASURES

FOURTEEN WEEKLY JOBSTREAMS

NUMBER OF PROCESSORS	TURNAROUND TIME (MINUTES) AVG.	TURNAROUND TIME (MINUTES) MAX.	PROCESSOR UTILIZATION (%) AVG.	PROCESSOR UTILIZATION (%) MAX.	THROUGHPUT (TRANS/DAY)
9	117.5	202.6	15	42	924
8	121.5	200.0	17	42	936
7	120.8	199.1	20	41	940
6	125.1	200.0	23	42	936
5	140.3	216.0	26	45	867
4	200.8	284.5	32	59	658
3	230.6	291.9	43	61	641
2	299.7	310.3	64	65	603

FIGURE 34

NETWORK PERFORMANCE

TURNAROUND TIME VERSUS SIZE

FOURTEEN WEEKLY JOBSTREAMS

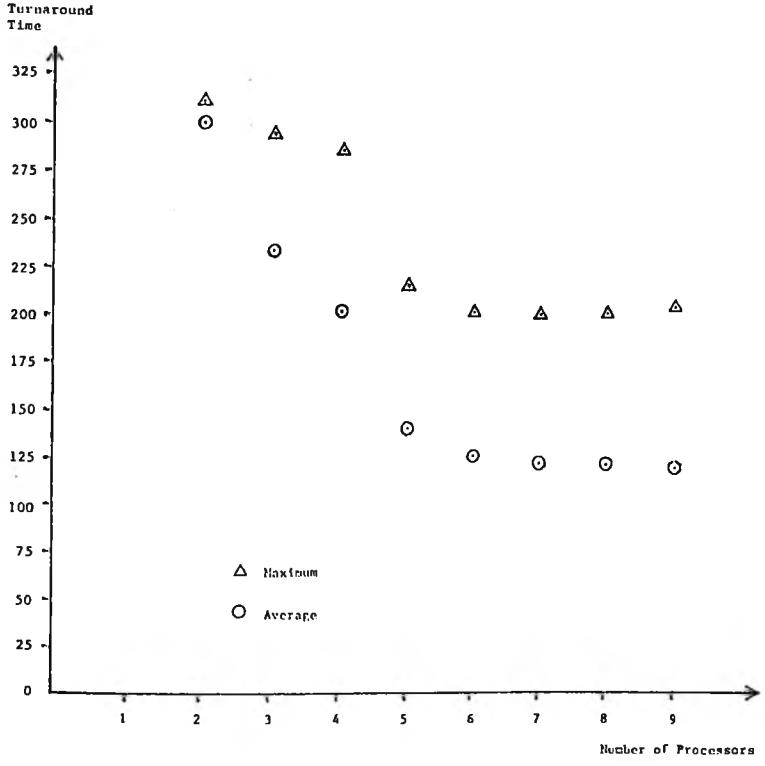
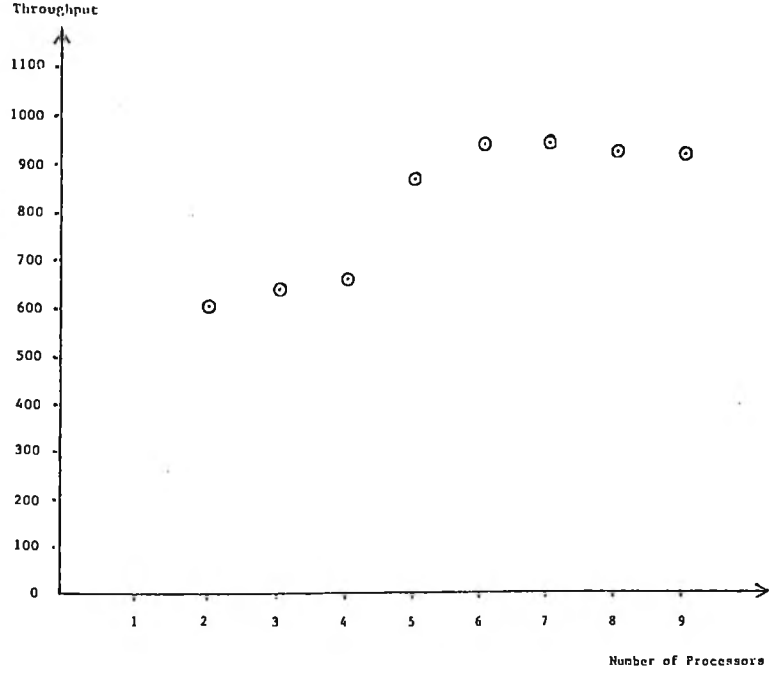


FIGURE 35

NETWORK PERFORMANCE

THROUGHPUT VERSUS SIZE

FOURTEEN WEEKLY JOBSTREAMS



Company in Chapter III requires that a total of twelve system files be established. The physical locations of these files within the network will vary depending on the number of processors employed. In utilizing the network size algorithm a file location scheme, or file distribution, was assumed for each possible network size. After determining an optimum size, the file distribution assumed for that size should be tested for possible improvements. The file location algorithm provides a mechanism for evaluating these alternative file locations.

Files were initially assigned to the six processors in the Agee network in order to keep them in proximity to the jobstreams accessing them. The twelve files were assigned to applications areas as is summarized in Table 18.

The file location algorithm requires that this initial file distribution be examined to determine candidate files for possible relocation. A candidate for relocation is a file which is accessed by a jobstream not resident at the file's initial location. For the Agee network the candidates are as follows:

1. Files for relocation to processor 1:

   Inventory from processor 4,

   Open Order from processor 3,

   Product from processor 3,

   Customer from processor 3.

2. Files for relocation to processor 2:

   Inventory from processor 4,

TABLE 18

INITIAL NETWORK TOPOLOGY

SIX PROCESSORS

PROCESSOR NUMBER	APPLICATION AREA	RESIDENT JOBSTREAMS	RESIDENT FILES
1	Accounting	General Ledger Billing Cash Receipts	Detailed Receivables General Ledger
2	Production	Product Costing Scheduling	Standard Costs Product Costs Production Data
3	Marketing	Order Entry Sales Analysis	Open Order Product Customer
4	Purchasing	Purchase Order Material Issues Material Receipts	Open Purchase Order Inventory
5	Payables	Payables Disbursements	Detailed Payables
6	Personnel	Employee Data Payroll	Employee

Open Order from processor 3,

General Ledger from processor 1,

3. Files for relocation to processor 5:

Open Purchase Order from processor 4,

General Ledger from processor 1.

When the Simulator was executed for the initial file assignment, the following processor utilizations were observed.

Processor 1 - 19.4%

Processor 2 - 36.7%

Processor 3 - 8.4%

Processor 4 - 25.3%

Processor 5 - 7.7%

Processor 6 - 41.6%

A possible file relocation can be expected to decrease processor contention only if the file is moved to a processor of lower utilization. It can be seen that of the original nine candidates for relocation, only three meet this utilization criterion. They are as follows.

1. File for relocation to processor 1:

Inventory from processor 4.

2. Files for relocation to processor 5:

Open Purchase Order from processor 4,

General Ledger from processor 1.

The three remaining candidates specify three possible file distributions for the Agee network which can be evaluated using the Simulator. The results of this evaluation are

summarized in Table 19.

All three of the alternative file distributions produce an improvement in average turnaround time. However in each case the turnaround time for the most lengthy jobstream is increased, and a slight decrease in throughput is observed. Of the remaining candidates for relocation, the one which provides the greatest improvement in average turnaround time entails moving the General Ledger account file. It would be relocated from processor 1, in the accounting applications area, to processor 5 which is in the payables area. This change creates the optimal network topology presented in Table 20.

Peripheral Equipment Choices

In evaluating alternative network sizes and file location schemes, a specific hardware configuration for each processor site was assumed. Once the optimal number of processors and file distribution had been found, the Simulator was used to test for possible changes in this single site configuration. Three such changes were evaluated for Agee Company. Considered were the substitution of a serial printer and a line printer for the Teletype assumed in the original configuration, and the replacement of the disk drive with a floppy disk system. These tests required three additional executions of the Simulator. The results of these tests are summarized in Table 21.

TABLE 19

FILE RELOCATION TESTS

RELOCATION ALTERNATIVE	TURNAROUND TIME (MINUTES) AVG.	MAX.	THROUGHPUT (TRANS/DAY)
General Ledger From processor 1 To processor 5	121.4	202.5	924
Open Purchase Order from processor 4 To processor 5	123.9	202.4	925
Inventory From processor 4 To processor 1	123.4	202.4	925
Initial File Distribution	125.1	200.0	936

Note: Results are those for a network of six processors, with disk drive and Teletype printer.

TABLE 20

OPTIMAL NETWORK TOPOLOGY

SIX PROCESSORS

PROCESSOR NUMBER	APPLICATION AREA	RESIDENT JOBSTREAMS	RESIDENT FILES
1	Accounting	General Ledger Billing Cash Receipts	Detailed Receivables
2	Production	Product Costing Scheduling	Standard Costs Product Costs Production Data
3	Marketing	Order Entry Sales Analysis	Open Order Product Customer
4	Purchasing	Purchase Order Material Issues Material Receipts	Open Purchase Order Inventory
5	Payables	Payables Disbursements	Detailed Payables General Ledger
6	Personnel	Employee Data Payroll	Employee

TABLE 21

PERIPHERAL DEVICE SELECTION TESTS

DEVICE ALTERNATIVES	TURNAROUND TIME (MINUTES) AVG.	MAX.	THROUGHPUT (TRANS/DAY)
Teletype printer Disk drive	121.4	202.5	924
Serial printer Disk drive	115.9	189.6	987
Line printer Disk drive	115.3	190.0	985
Serial printer Floppy disk	117.0	189.9	986

Note: Results are those when the optimal network topology is employed.

It was found that the use of either the serial printer or the line printer would provide a noticeable improvement in system performance. The difference in average turnaround time between the two is probably not sufficient to justify the additional cost of a higher speed line printer. Incorporating a serial printer with a floppy disk system at each site caused only a slight degradation in performance from that when a disk drive was employed. Thus in application areas with a light processing workload, Agee may choose to utilize the slower floppy disk system without producing a significant decrease in system turnaround time. With more heavily utilized processors the disk drive and possibly the line printer would be appropriate.

Sensitivity to Workload Changes

A major concern of management at Agee Company was the ability of existing systems to provide for anticipated future growth. Any proposed computer based system should not only give adequate user service at current demand levels, but should also be capable of processing larger workloads as the company expands its operations. The Simulator was used in predicting the response of the proposed six processor network to such increases in the data processing workload.

Two increased workloads were considered. The performance of the system was evaluated at a transaction level equal to twice and to three times the current level.

It was observed that at twice the current level average turnaround time for a jobstream increased to approximately four hours, and the most lengthy jobstream required six and one-third hours for completion. At triple the current level average turnaround time was just under six hours. In this case, however, one jobstream would require in excess of nine hours of processing. Since a normal workday is eight hours in length, triple the current workload approximately represents the level of saturation at which the system performance becomes unacceptable.

Management may conclude that the six processor network utilizing serial printers and disk drives will be sufficient until processing requirements increase to approximately three times those presently experienced. At that time, Agee should consider expanding the network by adding one or more processors. The Simulator would again be a useful device for evaluating alternative file locations in the expanded system.

Validation

An important part of any simulation study is the validation process. The user requires assurance that a model is an adequate representation of the system being modeled. In ideal circumstances, it is possible to validate a model by comparing its output to observed values of endogeneous variables from the actual system. And where simulation has been utilized to evaluate possible

reconfigurations of existing computer systems, this method has been used successfully. Yet in the planning phase of a system design project, validation in this fashion is not possible. The model's parameters, structure, and output must be examined for reasonableness and creditability. It is also desirable that the sensitivity of the model's results to data assumptions be determined.

As was discussed in Chapter IV, a computer system simulation model may be valid for some purposes while being inappropriate for others. The Generalized Ring Network Simulator exists at a high level of aggregation relative to individual processor operations and is therefore not appropriate for making processor comparisons. Provided that the aggregations made in depicting processor functions do not significantly affect the conclusions drawn from the model, it will still be useful for evaluating other design alternatives. A form of parametric programming was used to determine the effects of these aggregations.

All processor functions were represented by a single data variable, designated PROCTIME. This variable represents the time required for the actions of a processor between file accesses. A value of seven milliseconds was assumed for PROCTIME in the analysis of the Agee system. By varying the value of this parameter on successive executions of the Simulator, the sensitivity of the model to this assumption could be determined.

The model was executed for three additional values of

PROCTIME, ranging between four and fourteen milliseconds. All other model parameters were held constant and used to depict a six processor network with serial printers and disk drives. The current level of weekly processing requirements was also assumed. Maximum variations in average turnaround time, maximum turnaround time, and throughput were all within one-half percent of those output values produced in the original analysis. Thus the conclusions drawn from the model are relatively insensitive to the processing time assumption as long as reasonable values are chosen for it.

The manner in which the Simulator represents system components and component interfaces is described in detail in Chapter V. The file access, input, output and communications functions are depicted at an accurate level of detail. Test data were created and used to trace the GPSS transaction flow within the model. Careful analysis of the results of these tests indicate that the model represents the system as desired. Aggregations are made in describing central processor functions, but these do not significantly affect the output obtained from the model. Results indicate that average turnaround time for a data processing jobstream on a six processor network is about one and one-half hours for daily applications and two hours for weekly ones. These results are reasonable and do not conflict with observed characteristics at data processing installations with equivalent workloads. The Generalized Ring Network Simulator therefore provides an adequate

representation of a miricomputer ring network and is useful for evaluating many of the alternatives faced by the designer of such a system.

Interpretation of Results

A major difficulty arising from the use of simulation models of computer systems is the tendency to attribute an unjustified degree of accuracy to the results. The degree of reliance which should be placed on the output values produced by the Simulator is of concern to any system designer using it.

As was discussed in Chapter IV, a carefully constructed computer system simulation model may be expected to have an accuracy of about eighty percent. Any performance projections made by a model such as the Simulator are subject to this limitation. Because of aggregations made in the development of the model, the performance of an actual system can be expected to vary somewhat from the values predicted. Furthermore, the hardware descriptions utliized by the model rely on data generated from probability distributions. The statistical properties of the random number sequences used for this purpose may produce inaccuracies in the resulting performance measures. Inaccuracies from this source were estimated by executing the Simulator for the optimal system design using different random number sequences. Variations in performance measures were observed ranging between one

and four percent as the sequence was changed. This error is not considered a significant one. The results should therefore be of sufficient accuracy that they will be of use in the planning phase of the management information system life cycle.

Of more significance is the value of the Simulator in comparing design alternatives. A model's performance projections may differ significantly from the true values, and yet the model can merit a very high reliance in comparing two or more like alternatives. As long as the aggregations made in model construction do not affect the representation of these alternatives in different ways, the model can be expected to provide highly accurate results in predicting the relative desirability of each of them. Because of this fact, aggregations made in the Simulator concerning central processor functions do not affect the alternative network sizes, file location schemes, or peripheral device choices. The model is a highly reliable one in making comparisons between alternatives, but is less reliable in making projections of performance.

In any actual business situation, the model will be used in the context of a set of financial constraints peculiar to the organization. Design alternatives which are found to be best using the Simulator may not be feasible financially. Although a seven processor network was found to produce the best performance for Agee Company, a six processor network was recommended for them. The incremental

improvement in performance did not justify the additional cost of the seventh processor. Similarly Agee management may conclude that, based on funds available for information system development, a still smaller network will give adequate performance at a more acceptable cost. The advantage of utilizing the Simulator is not that it gives answers, but rather that it provides guidelines to management in evaluating the cost and benefit tradeoff.

The model must additionally be applied within a set of organizational constraints. Peculiarities of a particular business may dictate that more than the optimum number of processors be employed. If a decision center which requires data processing service is remotely located, a processor may be required for it even though overall system performance is decreased by it. And management may desire that certain application areas be created or eliminated from the network to the detriment of overall performance. In this fashion organizational constraints may affect desion variables such as size, applications jobstream and file locations, and even hardware. An advantage of the Simulator is that it provides management with a measure of the cost, in terms of degraded processing service, of satisfying such organizational requirements. The major strength of a distributed network is that it allows the information processing system to be modified in this fashion. The objective in designing a minicomputer network is to optimize the operations of the functional business

unit, rather than to optimize the operations of the hardware providing the informational services.

Conclusions

Distributed networks of minicomputers will become a cost-effective and efficient means to satisfy the information processing needs of many organizations. Such a network will overcome many of the limitations inherent in the more traditional centralized data processing structure. However very little experience exists within the business world in developing networks of this type.

A methodology has been presented for the design of distributed business systems which utilize minicomputers. The methodology employs a model, the Generalized Ring Network Simulator, to evaluate alternatives available to a system designer. The model utilizes data which is readily available in a business environment, and evaluates alternatives based on criteria which are of the most significance to the information system user. The Simulator can thus be used to overcome the difficulties resulting from the lack of experience in business minicomputer network design.

The use of the methodology has been demonstrated in a typical business application. A company's informational needs were described and a software system designed for satisfying these needs. This software system was characterized to the Simulator, and several hardware and

software design alternatives for it were considered. A heuristic procedure was employed to establish an optimum network configuration for this application.

A simulation model such as the Generalized Ring Network Simulator can be an extremely valuable tool in the design of minicomputer networks. When executed carefully and interpreted properly, the model can be of significant aid in the planning phase of the information system life cycle. In this manner the methodology presented here can contribute to achieving the ultimate goal of all management information systems: More relevant, more timely, and more accurate information for the decsion makers in an organization.

BIBLIOGRAPHY

1 Abramson, Norman, and Franklin F. Kuo, eds., Computer Communication Networks, Engilewood Cliffs, N. J.: Prentice-Hall, 1973.

2 Acree, John, and Arthur Lynch, "Ping Network Architecture Supports Distributed Processing," Data Communications, (March/April 1976).

3 Adams, Donald L., "Data Processing Management," Management Controls 16,7(July 1969): 148-153.

4 Ashenhurst, R. L., And R. H. Vonderohe, "A Hierarchical Network," Datamation, 21,2 (February 1975): 40-44.

5 Auerbach Editorial Staff, "What is Network Architecture?," Computer Decisions, 8,6 (June 1976): 24-33.

6 Baer, Jean-Loup, "Models for the Design, Simulation, and Performance of Distributed-Function Architecture," Computer, 7,3(March 1974): 25-30.

7 Barber, D. L. A., "The European Computer Network Project," in Winkler, Stanley, ed., Computer Communications: Impacts and Implications, Proceedings of the First International Conference on Computer Communication, Washington, D. C., October 24-26, 1972, pp. 192-200.

8 Becker, Hal P., Functional Analysis of Information Networks, New York: Wiley, 1973.

9 Bell, C. Gordon, "More Power by Networking," IEEE Spectrum, 11,2(February 1974): 40-45.

10 Bell, C. G., A. N. Habermann, J. McCredie, R. Rutledge, and W. Wulf, "Computer Networks," Computer (September/October 1971): 14-23.

11 Bell, Thomas E., "Objectives and Problems in Simulating Computers," Proceedings of the Fall Joint Computer Conference, 1972, AFIPS Press, Montvale, N. J., 287-297.

12 Bell, Thomas E.; Boehm, Barry W.; And Jeffrey, S., Eds., Computer Performance Evaluation: Report of the 1973 NBS/ACM Workshop, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, D. C., 1975.

13 Beus, H. Lynn, "The Obsolescence of Look-Ahead Buffering," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder, Colorado, August 12-14, 1975, 77-87.

14 Booth, Grayce M., "The Use of Distributed Data Bases in Information Networks," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-26, 1972, 371-376.

15 --------, Functional Analysis of Information Processing, New York: Wiley, 1973.

16 --------, "Distributed Information Systems," Proceedings of the 1976 National Computer Conference, AFIPS Press, Montvale, N. J., 789-794.

17 Bowdon, Edward K., Sr., Sandra A. Mamrak, and Fred P. Salz, "Performance Evaluation in Network Computers," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 67-75.

18 Bowdon, Edward K., Sr., Sandra A. Mamrak, and Fred P. Salz, "Simulation - A Tool for Performance Evaluation in Network Computers," Proceedings of the National Computer Conference, 1973, AFIPS Press, Montvale, N. J., 121-131.

19 Boylan, David T., "DBMS for Minis," Computer Decisions, 8,1(January 1976): 66-69.

20 Brauch, Charles J., "Simulation of a Large Computing Facility as a Multi-product Firm," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder, Colorado, August 12-14, 1975, 243-249.

21 Bray, Olin H., "Distributed Data Base Considerations," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 121-127.

22 Burnett, Gerald L., And Richard L. Nolan, "At Last, Major Roles for Minicomputers," Harvard Business Review, (May-June 1975), 148-156.

23 Byrne, Thomas A., Alan V. Piercey, and Frank L. Myers, "Prototype II: A Job Selection Simulation Model," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 3-14.

24 Calingaert, Peter, "System Performance Evaluation: Survey and Appraisal," Communications of the ACM, 10,1(January 1967): 12-18.

25 Carr, C. Stephen, "Host-Host Communication in the ARPA Network," Proceedings of the Spring Joint Computer Conference, 1970, AFIPS Press, Montvale, N. J., 589-592.

26 Carren, Donald M. "Multiple Minis for Information Management," Datamation, 21,9(September 1975): 54-58.

27 Champine, G. A., "Six Approaches to Distributed Databases," Datamation, 23,4(May 1977): 69-72.

28 Chow, John V., "What You Need to Know About DBMS," Journal of Systems Management 26,5(May 1975): 22-27.

29 Chou, Wushow, "Computer Communications Networks - The Parts Make Up the Whole," Proceedings of the National Computer Conference, 1975, AFIPS Press, Montvale, N. J., 119-128.

30 Chu, W. W., "Optimal File Allocation on a Multiple Computer System," IEEE Transactions on Computers, C-18,10(October 1969): 885-889.

31 ---------, "Performance of File Directory Systems for Data Bases in Star and Distributed Networks," Proceedings of the National Computer Conference, 1976, AFIPS Press, Montvale, N. J., 577-587.

32 Cohen, Ellis, "Symmetric Multi-mini-processors: A Better Way to Go?," Computer Decisions, (January 1973): 16-20.

33 Conrath, David W., "Measuring the Computer's Impact on Organizational Structure," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-25, 1972, 68-73.

34 Crowther, W. R., F. E. Heart, A. A. McKenzie, J. M. McQuillan, and D. C. Walden, "Issues in Packet Switching Network Design," Proceedings of the 1975 National Computer Conference, AFIPS Press, Montvale, N. J., 161-166.

35 Davis, Gordon, Management Information Systems, Conceptual Foundations, Structure, and Development, New York: McGraw-Hill, 1974.

36 Dean, Neil J., "The Computer Comes of Age," Harvard Business Review, (January-February 1968): 83-91.

37 Dearden, John, Computers in Business Management, Homewood, Ill.: Dow Jones-Irwin, 1966.

38 Dearden, John, F. Warren McFarlan, and William M. Zani, Managing Computer Based Information Systems, Homewood, Ill.: Richard D. Irwin, 1971.

39 Despres, Remi F., "A Packet Switching Network With Graceful Saturated Operation," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-25, 1972, 345-351.

40 Digital Equipment Corporation, Introduction to Minicomputer Networks, Maynard, Massachusetts, 1974.

41 ---------, PDP8/E & PDP8/M Small Computer Handbook, Maynard, Mass., 1972.

42 Dollotta, T. A , et. al., Data Processing in 1980-1985, New York: John Wiley & Sons, 1976.

43 Dumas, K., "The Effect of Program Segmentation on Job Completion Times in a Multiprocessor Computing System," Digest of the Second Conference on Applications of Simulation, New York, December 1968, pp. 77-78.

44 Enger, Norman L., "Simulation Case Studies," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 53-63.

45 Enslow, P. H. Jr., "Non-Technical Issues in Network Design: Economic, Legal, Social, and Other Considerations," Computer 6,8(August 1973): 21-30.

46 Fagan, B. M., And Henry Kleine, "A Simulation Model of a Message Switching System," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 4-6, 1974, 63-73.

47 Farber, David J., "A Survey of Computer Networks," Datamation, 18,4(April 1972): 36-39.

48 --------, "The Distributed Computer System," COMPCON '73 Digest, Seventh Annual IEEE Computer Society Annual Conference, February 27 - March 1, 1973, 31-34.

49 --------"Software Considerations in Distributed Architectures," Computer, 7,3(March 1974): 31-35.

50 --------, "A Ring Network," Datamation, 21,2 (February 1975): 44-46.

51 Farber, David J., And Frank R. Heinrich, "The Structure of a Distributed Computer System - The Distributed File System," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-25, 1972, 364-370.

52 Farber, David J., And K. Larson, "The Structure of a Distributed Computer System - The Communications System," Proceedings of the Symposium on Computer-Communications Networks and Teletraffic, Microwave Research Institute of the Polytechnic Institute of Brooklyn, April 1972, 21-27.

53 --------, "A Prototype Implementation of a Protocol for Network Security," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 33-36.

54 Feistel, H., W. A. Notz, and J. L. Smith, "Some Cryptographic Techniques for Machine-to-machine Data Communications," Proceedings of the IEEE, 63,11(November 1975): 1545-1554.

55 Floam, Gary, "Putting a Data Base on a Mini," Datamation, 22,6(June 1976): 97-100.

56 Flynn, Robert L., "A Brief History of Data Base Management," Datamation, 20,8(August 1974): 71-77.

57 Foster, John D., "Distributive Processing for Banking," Datamation, 22,7(July 1976): 54-56.

58 Frank, Howard, and Lynn Hopewell, "Network Reliability," Datamation, 20,8(August 1974): 85-91.

59 Franklin, Mark, and David Looney, "Simulation of a Computer System With Single and Dual Density Disks," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 259-267.

60 Franta, William R., And Kurt Malz, "A Framework for Investigating a Class of Distributed Processing Algorithms," IEEE Transactions on Systems, Man, and Cybernetics, SMC-7,8(August 1977): 597-604.

61 Fraser, A. G., "A Virtual Channel Network," Datamation, 21,2(February 1975): 51-56.

62 Fuchs, E., And P. E. Jackson, "Estimates of Random Variables for Certain Computer Communications Traffic Models," Communications of the ACM, 13,12(December 1970): 752-757.

63 Fulton, David L., Claude Overstreet, and Richard T. Thomas, "The Design of a Minicomputer Network Operating System," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 6-10.

64 Galbraith, Jay R., Organizational Design: An Information Processing View, Reading, Mass.: Addison-Wesley, 1974.

65 Gerke, Stephen R., "Performance Projection and Evaluation for a Transaction Oriented System," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 4-6, 1974, 77-83.

66 Glinka, L. R., P. M. Brush, and A. J. Ungar, "Design, Through Simulation, of a Multiple Access Information System," Proceedings of the 1967 Fall Joint Computer Conference, Washington, D. C.: Thompson Book Co., 1967, 437-447.

67 Glaser, George, "The EDP People Problem," Data Processing Digest, 15,11(November 1969): 1-14.

68 --------, "The Centralization vs. Decentralization Issue," Data Base, 2,3(Fall/Winter 1970): 3-13.

69 Gray, James P., And Charles R. Blair, "IBM's, Systems Network Architecture," Datamation, 21,4(April 1975): 51-56.

70 Hammons, Lawrence, "A Computer for Every Manager," Automation, (July 1970): 42-45.

71 Haussmann, F. W., And H. Schultz Kistler, Modeling for Computer Center Planning, IBM Systems Journal, 10,4(April 1971): 305-324.

72 Heart, F. E., R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The Interface Message Processor for the ARPA Network," Proceedings of the 1970 Spring Joint Computer Conference, AFIPS Press, Montvale, N. J., 551-567.

73 Heinrich, Frank R., And David J. Kaufman, "A Centralized Approach to Computer Network Security," Proceedings of the 1976 National Computer Conference, AFIPS Press, Montvale, N. J., 85-90.

74 Helgeson, W. B., "Distributed Data Bases- Networks Bring D. P. To Source of Data," Data Communications User, (December 1975): 39-41.

75 Henderson, James M., And Richard E. Quandt, Microeconomic Theory, New York: McGraw-Hill, 1958.

76 Herman, Donald J., "SCERT: A Computer Evaluation Tool," Datamation, 13,2(February 1967): 26-28.

77 Hiller, Frederick S., And Gerald J. Lieberman, Introduction to Operations Research, First Edition, San Francisco: Holden-Day, 1967.

78 Hunt, Pearson, "Fallacy of the One Big Brain," Harvard Business Review (July-August 1966): 84-90.

79 Hunter, John J., "Distributing a Database," Computer Decisions, 8,6(June 1976): 36-40.

80 Hutchinson, George K., "The Use of Micro Level Simulation in the Design of a Computer Supervisory System," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 243-254.

81 Huynh, D., H. Kobayashi, and F. F. Kuo, "Design Issues for Mixed Media Packet Switching Networks," Proceedings of the 1976 National Computer Conference, AFIPS Press, Montvale, N. J., 541-549.

82 "Interdata at Citibank," Datamation, (July 1975): 126

83 Jensen, Flemming J., "Centralization or Decentralization in Banking," Datamation, 22,7(July 1976): 60

84 Joseph, Earl C., "Innovations in Heterogeneous and Homogeneous Distributed Function Architecture," Computer 3,7(March 1974): 17-24.

85 Kalogeras, C. M., "Centralized vs. Local DP Organization," Journal of Systems Management, (March 1977): 28-31.

86 Kaspar, Hans and Donald S. Miller, "Job and Jobstream Modeling in the TRW Timeshare System Simulation," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 4-6, 1974, 95-121.

87 Kenny, Donald P., Minicomputers: Low-Cost Computer Power for Management, New York: AMACOM, A Division of American Management Associations, 1973.

88 Kershenbaum, Aaron, "Tools for Planning and Designing Data Communications Networks," Proceedings of the 1974 National Computer Conference, AFIPS Press, Montvale, N. J.: 583-591.

89 Kimbleton, Stephen R., "The Role of Computer System Models in Performance Evaluation," Communications of the ACM, 15,7(July 1972): 586-590.

90 --------, "Performance Evaluation - A Structured Approach," Proceedings of the 1972 Spring Joint Computer Conference, AFIPS Press, Montvale,N. J., 411-416.

90 Kimbleton, Stephen R., And Richard L. Mandell, "A Perspective on Network Operating Systems," Proceedings of the 1976 National Computer Conference, AFIPS Press, Montvale, N. J., 551-559.

92 Kleinrock, Leonard, Communications Nets: Stochastic Message Flow and Delay, New York: McGraw-Hill, 1964.

93 --------, "Analytical and Simulation Methods in Computer Network Design," Proceedings of the 1970 Spring Joint Computer Conference, AFIPS Press, Montvale, N. J., 569-579.

94 --------, "On Communications and Networks," IEEE Transactions on Computers, C-25,12(December 1976): 1326-1335.

95 --------, Queuing Systems, New York: Wiley, 1975.

96 Kleinrock, Leonard, William E. Naylor, and Holger Opderbeck, "A Study of Line Overhead in the ARPANET," Communications of the ACM, 19,1(January 1976): 3-13.

97 Knight, Kenneth, "Evolving Computer Performance 1962-1967," Datamation, 14,1(January 1968): 31-35.

98 Knottek, Nancy E., "Selecting a Distributed Processing System," Computer Decisions, 8,6(June 1976): 42-44.

99 Lane, M. G., "Supporting Synchronous Line Protocols With Minicomputers," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronic Engineers, Inc., 30-32.

100 Lennon, William J., "A Mini-Computer Network for Support of Real-Time Research," Proceedings of the ACM Annual Conference, 2(November 1974): 595-604.

101 --------, "A User Oriented Mini-computer Network," Proceedings of the Eleventh IEEE Computer Society Conference, September 9-11, 1976, 133-136.

102 Lennon, William J., Eric L. Baatz, Sandra L. Colussi, and Kenneth E. Kinnear, Jr., "Using a Distributed Mini-Computer Network to Automate a Biochemical Laboratory," Proceedings of the ACM Interface Meeting on Programming Systems in the Small Processor Environment, New Orleans, La., March 4-6, 1976.

103 Lichtic, Bernard A., "Simulation is 'Unreal'," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder, Colorado, August 12-14, 1975, 213-219.

104 Linsenmeyer, G., And P. Ligomenides, "A General Computer Network Model," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 155-161.

105 Loomis, Mary Snuggs, "A Model for Data Base Distribution," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 162-169.

106 "Major Bank Peplaces Large Mainframes With Minicomputers," Datamation, 22,8(August 1976): 106.

107 March, James, and Herbert Simon, Organizations, New York: John Wiley & Sons, 1958.

108 Martin, James, Design of Real-Time Computer Systems, Englewood Cliffs, N. J.: Prentice-Hall, 1967.

109 --------, Teleprocessing Network Organization, Englewood Cliffs, N. J.: Prentice-Hall, 1970.

110 --------, Future Developments in Telecommunications, Englewood Cliffs, N. J.: Prentice-Hall, 1971..

111 --------, Introduction to Teleprocessing, Englewood Cliffs, N. J.: Prentice-Hall, 1972.

112 MacDougall, M. H., "Computer System Simulation: An Introduction," Computing Surveys, 2,3(September 1970): 191-209.

113 Mansfield, Edwin, Microeconomics Theory and Applications, New York: Norton, 1970.

114 McCredie, J. W., And S. J. Schlesinger, "A Modular Simulation of TSS/360," Proceedings of the Fourth Conference on Applications of Simulation, New York, December 1970, pp. 201-206.

115 McFarlan, F. Warren, Richard L. Nolan, and David P. Morton, Information Systems Administration, New York: Holt, Rinehart, & Winston, 1973.

116 Metcalfe, Robert M., And David P. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Communications of the ACM, 19,7(July 1976): 395-403.

117 McKenzie, A. A., B. P. Cosell, J. M. McQuillan, and M. J. Thorpe, "The Network Control Center for the ARPA Network," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-26, 1972, 185-191.

118 Mills, D. L., "An Overview of the Distributed Computer Network," Proceedings of the 1976 National Computer Conference, AFIPS Press, Montvale, N. J., 523-531.

119 Moss, Richard W., Joe L. Hammond, and Ernest E. Donaldson, "Interactive Communications Modeling Study," Washington, D. C.: National Technical Information Service, United States Department of Commerce, 1976.

120 National Science Foundation, Symposium on Modeling and Analysis of Data Networks: A Report, Washington, D. C., March 18-19, 1976.

121 Newkirk, Ross T., And James K. Mullin, "Simulated Demand in Large-Scale Batch Processor Acquisitions: Role and Justification," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 4-6, 1974, 187-194.

122 Oldehoeft, A. E., And M. H. Halstead, "Maximum Computing Power and Cost Factors in the Centralization Problem," Communications of the ACM, 15,2(February 1972): 94-96.

123 Palmer, Ian, Data Base Systems: A Practical Reference, Wellesley, Mass.: Q. E. D. Information Sciences, Inc., 1975.

124 Pasahow, E. J., "Simulation of a Real-Time Microprocessor Network," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder, Colorado, August 12-14, 1975, 67-71.

125 Patrick, Robert L., "Decentralizing Hardware and Disrersing Responsibility," Datamation, 22,5(May 1975): 79-84.

126 Perlman, Justin A., "Centralization vs. Decentralization," Datamation, 11,9(September 1965): 24-28.

127 Reynolds, Carl H., "Issues in Centralization," Datamation, 23,3(March 1977): 91-100.

128 Roberts, Lawrence G., And Barry D. Wessler, "Computer Network Development to Achieve Resource Sharing," Proceedings of the 1970 Spring Joint Computer Conference, AFIPS Press, Montvale,N. J., 543-549.

129 Rosenthal, Robert, Don E. Rippy, and Helen M. Wood, "The Network Measurement Machine- A Data Collection Device for Measuring the Performance and Utilization of Computer Networks," Washington, D. C.: National Technical Information Service, United States Department of Commerce, 1976.

130 Ross, Ronald G., "Data Base Management Systems: An Overview of Data Base Technology," Government Data Systems, 4,6(December 1975/ January 1976): 22-27.

131 Roth, Paul F., "Simulation of Computers: A Tutorial Introduction," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder Colorado, August 12-14, 1975, 3-5.

132 Russell, Richard M., "Approaches to Network Design," Computer Decisions, 8,6(June 1976): 20-22.

133 Rustin, Randall, Computer Networks, Englewood Cliffs, N. J.: Prentice-Hall, 1972.

134 Sanders, Ray W., And Vinton G. Cerf, "Compatibility or Chaos in Communication," Datamation, 22,3(March 1976): 50-55.

135 Schussel, George, "When Not to Use a Data Base," Datamation, 21,11(November 1975): 82-98.

136 Schwartz, Mischa, Computer Communications Network Design and Analysis, Englewood Cliffs, N. J.: Prentice-Hall, 1977.

137 Schwengels, Forrest V., "A Heuristic Job Shop Scheduling Algorithm for a Minicomputer,", Unpublished Masters Thesis in Management Science, Graduate School of Business, University of Alabama, 1976.

138 Segal, Ronald, and Norman White, "A Simulation Model of a National Computer Resource-Sharing Network for Education and Research," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 142-147.

139 Siler, Kenneth F., "A Stochastic Evaluation Model for Database Organizations in Data Retrieval Systems," Communications of the ACM, 19,2(February 1976): 84-95.

140 Simonette, Ilario, "Ring In Distributed Computing," Computer Decisions, 8,1(January 1976): 20-21.

141 Singer, C. R. M., "The User Department and the Computer," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-26, 1972, 397-398.

142 Smith, B. T., "Mixed Computer Networks: Benefits, Problems, and Guidelines," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-26, 1972, 201-209.

143 Sobolewski, J. S., "Programmable Communications Processors," Proceedings of the International Conference on Computer Communication, Washington, D. C., October 24-26, 1972, 380-389.

144 Solomon, Martin B., "Economies of Scale in the IBM System/360," Communications of the ACM, 9,6(June 1966): 435-440.

145 Teorey, Toby J., "Validation Criteria for Computer System Simulations," Proceedings of the Symposium on the Simulation of Computer Systems, Boulder, Colorado, August 12-14, 1975, 161-172.

146 Teorey, Toby J., And Alan G. Merten, "Considerations on the Level of Detail in Simulation," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 137-143.

147 Thierauf, Robert J., And Robert C. Klekamp, Decision Making Through Operations Research, New York: Wiley, 1975.

148 Traub, Arthur C., And William F. Zachmann, "A GPSS Model of a Complex On-Line Computer System," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 19-20, 1973, 17-27.

149 Turn, Rein, Computers in the 1980's, New York: Columbia University Press, 1974.

150 Visarescu, Dan C., And Chelsea C. White, "Modeling and Dynamic Control of Multicomputer Networks," Trends and Applications 1976: Computer Networks, New York: Institute of Electrical and Electronics Engineers, Inc., 1976, 42-47.

151 Wagner, Frank v., "Is Decentralization Inevitable?," Datamation, 22,11(November 1976): 86-97.

152 Walker, Phillip M., "Regulatory Developments in Data Communications - The Past Five Years," Proceedings of the 1972 Spring Joint Computer Conference, AFIPS Press, Montvale, N. J., 593-604.

153 Weitzman, Cay, Minicomputer Systems, Structure, Implementation, and Application, Englewood Cliffs, N. J.: Prentice-Hall, 1974.

154 "Where Minicomputers Do All of the Job," Business Week, (July 14, 1973): 38-39.

155 White, Robert R., "A Business Approach to Improving Productivity in the Services Sector," address presented to the Financial Systems Marketing Seminar of NCR, Miami, Florida, March 7, 1975. Reprinted in Infosystems, (August 1975): 34.

156 Willis, Ronald R., "Structured Model Development Techniques," Proceedings of the Symposium on the Simulation of Computer Systems, Gaithersburg, Md., June 4-6, 1974, 133-144.

157 Withington, Frederic G., "Data Processing's Evolving Place in the Organization," Datamation, 15,6(June 1969): 58-68.

158 --------,"Beyond 1984: A Technology Forecast," Datamation, 21,1(January 1975): 54-73.

159 --------, "Structural Revolution in the Computer Industry," Datamation, 22,1(January 1976): 68.

160 Wulf, W. A., And C. G. Bell, "C.mmp - A Multi-miniprocessor," Proceedings of the 1972 Fall Joint Computer Conference, Part II, AFIPS Press, Montvale, N. J., 765-777.

161 Wulf, William, and Roy Levin, "A Local Network," Datamation, 21,2(February 1975): 47-50.

162 Wyatt. Joe B., And Vincent I. Polley, "Network Interface Systems - An Evaluation by Simulation," Proceedings of the 1976 National Computer Conference, AFIPS Press, Montvale, N. J., 511-521.

163 Yasaki, Edward K., "Microprocessors in System Design," Datamation, 22,2(February 1976): 71-76.

APPENDIX 1

NETWORK SIMULATOR LISTING

```
@JEGPSS,I NETWORK
 GPSS 4.0    -08/28-19:57-(000)

      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      *                                                                     *
      *                 GENERALIZED RING NETWORK SIMULATOR                  *
      *                                                                     *
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      *
      *
                    JOB
                    ORDER,S 10
                    ORDER,Q 40
                    ORDER,F 50
                    ORDER,P JOB.NAME,SP.FA1,SP.FS1,SP.CI1,SP.PI1,SP.TRAN1,
     +SP.FA2,SP.FS2,SP.CI2,SP.PI2,SP.TRAN2,SP.FA3,SP.FS3,SP.CI3,
     +SP.PI3,SP.TRAN3,SP.FA4,SP.FS4,SP.CI4,SP.PI4,SP.TRAN4,
     +SP.FA5,SP.FS5,SP.CI5,SP.PI5,SP.TRAN5,SP.FA6,SP.FS6,SP.CI6,
     +SP.PI6,SP.TRAN6
                  S(1)       CAPACITY          1
                  S(2)       CAPACITY          1
                  S(3)       CAPACITY          1
                  S(4)       CAPACITY          1
                  S(5)       CAPACITY          1
                  S(6)       CAPACITY          1
                  S(7)       CAPACITY          1
                  S(8)       CAPACITY          1
                  S(9)       CAPACITY          1
                  S(10)      CAPACITY          1
     * TA.MIN IS TURNAROUND TIME IN MINUTES
      TA.MIN     VARIABLE      FS1/60000
     * JA.HRS IS JOB ARRIVAL TIMES IN HOURS
      JA.HRS     VARIABLE      C$1/3600000
     * PROCTIME IS AN APPROXIMATE CPU PROCESSING TIME
      PROCTIME   VARIABLE      10
     * CHAN.TIM IS THE TIME REQUIRED FOR A CHANNEL TRANSFER
      CHAN.TIM   VARIABLE      4
     * XFER.TIME IS THE TIME REQUIRED TO READ FROM INPUT BUFFER TO CORE
      XFER.TIM   VARIABLE      11
     * COM.TIME IS THE PROCESSOR TIME REQUIRED FOR A COMPARISON
      COM.TIME   VARIABLE      1
     * NO.PROCS IS THE NUMBER OF PROCESSORS IN THE RING
      NO.PROCS   VARIABLE      6
     * PAK.SIZE IS THE MESSAGE PACKET SIZE IN BITS
      PAK.SIZE   VARIABLE      1
     * RIP.TIME IS THE RING INTERFACE PROCESSOR READ-IN TIME
      RIP.TIME   VARIABLE      V$PAK.SIZE*1.
     * ROT.TIME IS THE TIME REQUIRED FOR A DISK ROTATION
      ROT.TIME   FUNCTION,UNI RF$8,5,40
     * SEEK.TIM ESTIMATES DISK ARM SEEK TIMES
      SEEK.TIM   FUNCTION,UNI RF$6,1,134
     * INP.TIME IS THE CONSOLE DATA INPUT TIME
      INP.TIME   FUNCTION,NOR RF$4,67584,20000
     * PRT.LINE ESTIMATES PRINTER TIME FOR ONE LINE
      PRT.LINE   FUNCTION,UNI RF$5,346,348
```

```
55      * LOOK.UP ESTIMATES THE DIRECTORY TABLE LOOK-UP TIME
56       LOOK.UP    FUNCTION,UNI RFS7,1,11
57      * JOB.TRAK TABLE SUMMARIZES FILE ACTIONS PERFORMED
58       JOB.TRAK   TABLE          P$SP.AFA,0,1,7
59      * TURNAROUND TABLE HAS STATISTICS ON JOB COMPLETION TIMES
60       TURN.ARD   TABLE          V$TA.MIN,1,1,480
61      *JOB.ARRV TABLE DESCRIBES JOB INTERARRIVAL TIMES TO THE SYSTEM
62       JOB.ARRV   TABLE          V$JA.HRS,1,1,9
63      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
64      *                                                               *
65      *                              P R I N T                        *
66      *                                                               *
67      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
68      * THIS MACRO SIMULATES THE PRINTING OF ONE LINE OF DATA ON A LINE *
69      * PRINTER.  IT IS CALLED BY THE 'PROCESSOR' MACRO WHENEVER A JOB  $
70      * SPECIFICATION CALLS FOR PRINTING.  DATA INPUT IS 'PRT.LINE',    *
71      * THE AMOUNT OF TIME REQUIRED TO PRINT A LINE.                    $
72      *
73      *
74      PRINT      STARTMACRO
75                  QUEUE          Q(P$POINTER+30)
76                  SEIZE          F(P$POINTER+40) TIME(FN$PRT.LINE)
77                  RELEASE        F(P$POINTER+40)
78                  ENDMACRO
79      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
80      *                                                               *
81      *                              I N P U T                        *
82      *                                                               *
83      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
84      * THIS MACRO SIMULATES ON-LINE DATA ENTRY INTO A DATA FILE WHICH  *
85      * IS SUBSEQUENTLY PROCESSED IN BATCH MODE.  A DATA TRANSACTION IS *
86      * READ INTO AN INPUT BUFFER FROM A CONSOLE INPUT DEVICE.  ONCE ALL*
87      * DATA TRANSACTIONS HAVE BEEN READ, CONTROL IS RETURNED TO THE    *
88      * 'PROCESSOR' MACRO WHICH WRITES THE FILE ON DISK.                *
89      *
90      *
91      INPUT      STARTMACRO
92                  QUEUE          CONSOLEQ
93                  SEIZE          F(P$POINTER+30) TIME(FN$.ND.TIME)
94                  HOLD           F(P$POINTER+30) TIME(V$XFER.TIM)
95                  RELEASE        F(P$POINTER+30)
96                  ENDMACRO
97      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
98      *                                                               *
99      *                            WRITE DISK                         *
100     *                                                               *
101     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
102     * THIS MACRO SIMULATES THE ACTION OF A PROCESSOR IN WRITING A DISK*
103     * FILE.  WHEN A JOB CALLS FOR A FILE UPDATE, THE 'PROCESSOR' MACRO*
104     * CALLS THIS SEGMENT.  THE CPU SEIZES THE CHANNEL, MOVES THE DISK *
105     * ARM TO THE APPROPRIATE CYLINDER, AND WAITS FOR DISK ROTATION.  A*
106     * DATA ITEM IS READ INTO CORE, AND A SIMILAR PROCEDURE IS FOLLOWED*
107     * AS THE UPDATED DATA ITEM IS WRITTEN TO DISK.  THE VARIABLES ARE:*
108     * (1) CHAN.TIM - READ/WRITE TIME FOR THE CHANNEL, (2) SEEK.TIM -  *
109     * TIME FOR MOVEMENT OF DISK ARM (ASSUMED UNIFORMLY DISTRIBUTED)   *
110     * (3) ROT.TIME - THE DISK ROTATION TIME.                          *
111     *
```

```
112       *
113       WRITDISK  STARTMACRO
114                 QUEUE         Q(PSPOINTER+20)
115                 GATE          NU,F(PSPOINTER+10)
116                 GATE          NU,F(PSPOINTER+20)
117                 SEIZE         F(PSPOINTER+10) TIME(VSCHAN.TIM)
118                 RELEASE       F(PSPOINTER+10)
119                 SEIZE         F(PSPOINTER+20) TIME(FNSSEEK.TIM)
120                 HOLD          F(PSPOINTER+20) TIME(FNSROT.TIME)
121                 RELEASE       F(PSPOINTER+20)
122                 QUEUE         Q(PSPOINTER+20)
123                 SEIZE         F(PSPOINTER+10) TIME(VSCHAN.TIM)
124                 RELEASE       F(PSPOINTER+10)
125                 SEIZE         F(PSPOINTER)    TIME(VSPROCTIME)
126                 RELEASE       F(PSPOINTER)
127                 QUEUE         Q(PSPOINTER+20)
128                 GATE          NU,F(PSPOINTER+10)
129                 GATE          NU,F(PSPOINTER+20)
130                 SEIZE         F(PSPOINTER+10) TIME(VSCHAN.TIM)
131                 RELEASE       F(PSPOINTER+10)
132                 SEIZE         F(PSPOINTER+20) TIME(FNSROT.TIME)
133                 RELEASE       F(PSPOINTER+20)
134                 QUEUE         Q(PSPOINTER+20)
135                 SEIZE         F(PSPOINTER+10) TIME(VSCHAN.TIM)
136                 RELEASE       F(PSPOINTER+10)
137                 ENDMACRO
138       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
139       *                                                                     *
140       *                              READ DISK                              *
141       *                                                                     *
142       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
143       * THIS MACRO SIMULATES THE ACTION OF A PROCESSOR IN READING A DISK    *
144       * FILE.  WHEN A JOB REQUESTS A DATA ITEM FROM A FILE, THE CPU         *
145       * CALLS THIS SEGMENT.  THE CPU SEIZES A CHANNEL AND MOVES THE         *
146       * FILE ARM TO THE APPROPRIATE CYLINDER, AND THEN WAITS FOR DISK       *
147       * ROTATION.  THE DATA IS THEN TRANSFERRED TO THE CPU.  THE            *
148       * VARIABLES USED IN THE MODEL ARE: (1) CHAN.TIM - READ/WRITE TIME     *
149       * FOR THE CHANNEL, (2) SEEK.TIM - TIME FOR MOVEMENT OF THE DISK       *
150       * ARM (ASSUMED UNIFORMLY DISTRIBUTED), AND (3) ROT.TIME - THE         *
151       * DISK ROTATION TIME.                                                 *
152       *
153       *
154       READDISK  STARTMACRO
155                 QUEUE         Q(PSPOINTER+20)
156                 GATE          NU,F(PSPOINTER+10)
157                 GATE          NU,F(PSPOINTER+20)
158                 SEIZE         F(PSPOINTER+10) TIME(VSCHAN.TIM)
159                 RELEASE       F(PSPOINTER+10)
160                 SEIZE         F(PSPOINTER+20) TIME(FNSSEEK.TIM)
161                 HOLD          F(PSPOINTER+20) TIME(FNSROT.TIME)
162                 RELEASE       F(PSPOINTER+20)
163                 QUEUE         Q(PSPOINTER+20)
164                 SEIZE         F(PSPOINTER+10) TIME(VSCHAN.TIM)
165                 RELEASE       F(PSPOINTER+10)
166                 ENDMACRO
167       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
168       *                                                                     *
```

```
*                                  C O N T R O L                                *
*                                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* THIS IS THE CALLING SEGMENT WHICH DIRECTS THE OPERATION OF THE               *
* SIMULATOR.  A TRANSACTION REPRESENTING A DATA PROCESSING JOB IS              *
* READ FROM A JOBTAPE.  SPECIFICATION PARAMETERS, DENOTED BY THE               *
* PREFIX 'SP.', DETERMINE THE STEPS THE SIMULATOR WILL TAKE.                   *
* BASED ON THE VALUES OF THESE PARAMETERS, A PROCESSOR WILL (1)                *
* COMMUNICATE WITH ANOTHER PROCESSOR VIA THE 'RING' MACRO, (2)                 *
* ACCEPT DATA FROM AN INPUT DEVICE, (3) READ A DISK FILE, OR (4)               *
* WRITE TO A DISK FILE.  THE FOLLOWING INDEXES ARE USED:                       *
*   QUEUES              1 - 10  RING INPUT BUFFERS
*                      11 - 20  RING OUTPUT BUFFERS
*                      21 - 30  CHANNEL QUEUES
*                      31 - 40  PRINTER QUEUES
*   FACILITIES:         1 - 10  PROCESSORS
*                      11 - 20  CHANNELS
*                      21 - 30  DISKS
*                      31 - 40  INPUT CONSOLES
*                      41 - 50  PRINTERS
*
*
                JOBTAPE,B     1,1
                TABULATE      JOB.ARRV
      ASSN.1    ASSIGN        SP.AFA,P$SP.FA1
                ASSIGN        SP.AFS,P$SP.FS1
                ASSIGN        SP.ACI,P$SP.CI1
                ASSIGN        SP.API,P$SP.PI1
                ASSIGN        SP.NTRAN,P$SP.TRAN1
                ASSIGN        POINTER,P$SP.FS1
                TABULATE      JOB.TRAK
                MARK
* * * BEGIN PROCESSOR CALL 1 * * * * * * * * * * * * * *
                SEIZE         F(P$POINTER)
      B1READSP  ADVANCE       GOTO(B1FA0,B1FA1)
      B1FA0     COMPARE       P$SP.AFA EQ 0
                ADVANCE       GOTO(B1NEXTSP)
      B1FA1     ADVANCE       GOTO(B1CFA1,B1FA2)
      B1CFA1    COMPARE       P$SP.AFA EQ 1
                ASSIGN        TRANS.CT,0
      B1LOOP1   HOLD          F(P$POINTER)   TIME(V$PROCTIME)
                INPUT
                WRITDISK
                ASSIGN        TRANS.CT,P$TRANS.CT+1
                ADVANCE       GOTO(B1NEXT1,B1LAST1)
      B1NEXT1   COMPARE       P$SP.NTRAN GT P$TRANS.CT
                ADVANCE       GOTO(B1LOOP1)
      B1LAST1   ADVANCE       GOTO(B1NEXTSP)
      B1FA2     ADVANCE       GOTO(B1CFA2,B1FA3)
      B1CFA2    COMPARE       P$SP.AFA EQ 2
                ASSIGN        TRANS.CT,0
      B1LOOP2   HOLD          F(P$POINTER)   TIME(V$PROCTIME)
                READDISK
                ADVANCE       GOTO(B1CCI2,B1CPI2,B1CNT2)
      B1CCI2    COMPARE       P$SP.ACI EQ 1
                ADVANCE       TIME(V$COM.TIME) GOTO(500:B1PRT2,B1CNT2)
      B1CPI2    COMPARE       P$SP.API EQ 1
```

```
226   70  B1PRT2     ADVANCE
227   71             PRINT
228   74  B1CNT2     ASSIGN      TRANS.CT,PSTRANS.CT+1
229   75             ADVANCE     GOTO(B1NEXT2,B1LAST2)
230   76  B1NEXT2    COMPARE     PSSP.NTRAN GT PSTRANS.CT
231   77             ADVANCE     GOTO(B1LOOP2)
232   78  B1LAST2    ADVANCE     GOTO(B1NEXTSP)
233   79  B1FA3      ADVANCE     GOTO(B1CFA3,B1FA4)
234   80  B1CFA3     COMPARE     PSSP.AFA EQ 3
235   81             ASSIGN      TRANS.CT,0
236   82  B1LOOP3    HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
237   83             WRITDISK
238  106             ADVANCE     GOTO(B1CCI3,B1CPI3,B1CNT3)
239  107  B1CCI3     COMPARE     PSSP.ACI EQ 1
240  108             ADVANCE     TIME(VSCOM.TIME) GOTO(SGO:B1PRNT3,B1CNT3)
241  109  B1CPI3     COMPARE     PSSP.API EQ 1
242  110  B1PRNT3    ADVANCE
243  111             PRINT
244  114  B1CNT3     ASSIGN      TRANS.CT,PSTRANS.CT+1
245  115             ADVANCE     GOTO(B1NEXT3,B1LAST3)
246  116  B1NEXT3    COMPARE     PSSP.NTRAN GT PSTRANS.CT
247  117             ADVANCE     GOTO(B1LOOP3)
248  118  B1LAST3    ADVANCE     GOTO(B1NEXTSP)
249  119  B1FA4      ADVANCE     GOTO(B1CFA4,B1NEXTSP)
250  120  B1CFA4     COMPARE     PSSP.AFA EQ 4
251  121             ASSIGN      TRANS.CT,0
252  122  B1LOOP4    HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
253  123             ADVANCE     GOTO(B1CCI4,B1CPI4,B1CNT4)
254  124  B1CCI4     COMPARE     PSSP.ACI EQ 1
255  125             ADVANCE     TIME(VSCOM.TIME) GOTO(SGO:B1PRNT4,B1CNT4)
256  126  B1CPI4     COMPARE     PSSP.API EQ 1
257  127  B1PRNT4    ADVANCE
258  128             PRINT
259  131             ADVANCE     GOTO(B1CNT4)
260  132  B1CNT4     ASSIGN      TRANS.CT,PSTRANS.CT+1
261  133             ADVANCE     GOTO(B1NEXT4,B1LAST4)
262  134  B1NEXT4    COMPARE     PSSP.NTRAN GT PSTRANS.CT
263  135             ADVANCE     GOTO(B1LOOP4)
264  136  B1LAST4    ADVANCE     GOTO(B1NEXTSP)
265  137  B1NEXTSP   RELEASE     F(PSPOINTER)
266       * * * END PROCESSOR CALL 1 * * * * * * * * * * * * * * * *
267  138  ASSN.2     ASSIGN      SP.AFA,PSSP.FA?
268  139             ASSIGN      SP.AFS,PSSP.FS2
269  140             ASSIGN      SP.ACI,PSSP.CI?
270  141             ASSIGN      SP.ACI,PSSP.PI?
271  142             ASSIGN      SP.NTRAN,PSSP.TRAN?
272  143             TABULATE    JOB.TRAK
273  144             ADVANCE     GOTO(SKIP.2,SPEC.2,WAIT.2)
274  145  SKIP.2     COMPARE     PSSP.AFA EQ 0
275  146             ADVANCE     GOTO(ASSN.3)
276  147  SPEC.2     COMPARE     PIPOINTER EQ PSSP.AFS TIME(FNSLOOK.UP)
277       * * * BEGIN PROCESSOR CALL 2 * * * * * * * * * * * * * * *
278  148             SEIZE       F(PIPOINTER)
279  149  B2READSP   ADVANCE     GOTO(B2CFA1,B2FA2)
280  150  B2CFA1     COMPARE     PSSP.AFA EQ 1
281  151             ASSIGN      TRANS.CT,0
282  152  B2LOOP1    HOLD        F(PSPOINTER)   TIME(V1PROCTIME)
```

```
283  153              INPUT
284  157              WRITDISK
285  160              ASSIGN      TRANS.CT,PSTRANS.CT+1
286  181              ADVANCE     GOTO(B2NEXT1,B2LAST1)
287  182  B2NEXT1     COMPARE     PSSP.NTRAN GT PSTRANS.CT
288  183              ADVANCE     GOTO(B2LOOP1)
289  184  B2LAST1     ADVANCE     GOTO(B2NEXTSP)
290  185  B2FA2       ADVANCE     GOTO(B2CFA2,B2FA3)
291  186  B2CFA2      COMPARE     PSSP.AFA EQ 2
292  187              ASSIGN      TRANS.CT,0
293  188  B2LOOP2     HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
294  189              READDISK
295  200              ADVANCE     GOTO(B2CCI2,B2CPI2,B2CNT2)
296  201  B2CCI2      COMPARE     PSSP.ACI EQ 1
297  202              ADVANCE     TIME(VSCOM.TIME) GOTO(500:B2PRT2,B2CNT2)
298  203  B2CPI2      COMPARE     PSSP.API EQ 1
299  204  B2PRT2      ADVANCE
300  205              PRINT
301  208  B2CNT2      ASSIGN      TRANS.CT,PSTRANS.CT+1
302  209              ADVANCE     GOTO(B2NEXT2,B2LAST2)
303  210  B2NEXT2     COMPARE     PSSP.NTRAN GT PSTRANS.CT
304  211              ADVANCE     GOTO(B2LOOP2)
305  212  B2LAST2     ADVANCE     GOTO(B2NEXTSP)
306  213  B2FA3       ADVANCE     GOTO(B2CFA3,B2FA4)
307  214  B2CFA3      COMPARE     PSSP.AFA EQ 3
308  215              ASSIGN      TRANS.CT,0
309  216  B2LOOP3     HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
310  217              WRITDISK
311  240              ADVANCE     GOTO(B2CCI3,B2CPI3,B2CNT3)
312  241  B2CCI3      COMPARE     PSSP.ACI EQ 1
313  242              ADVANCE     TIME(VSCOM.TIME) GOTO(500:B2PRNT3,B2CNT3)
314  243  B2CPI3      COMPARE     PSSP.API EQ 1
315  244  B2PRNT3     ADVANCE
316  245              PRINT
317  248  B2CNT3      ASSIGN      TRANS.CT,PSTRANS.CT+1
318  249              ADVANCE     GOTO(B2NEXT3,B2LAST3)
319  250  B2NEXT3     COMPARE     PSSP.NTRAN GT PSTRANS.CT
320  251              ADVANCE     GOTO(B2LOOP3)
321  252  B2LAST3     ADVANCE     GOTO(B2NEXTSP)
322  253  B2FA4       ADVANCE     GOTO(B2CFA4,B2NEXTSP)
323  254  B2CFA4      COMPARE     PSSP.AFA EQ 4
324  255              ASSIGN      TRANS.CT,0
325  256  B2LOOP4     HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
326  257              ADVANCE     GOTO(B2CCI4,B2CPI4,B2CNT4)
327  258  B2CCI4      COMPARE     PSSP.ACI EQ 1
328  259              ADVANCE     TIME(VSCOM.TIME) GOTO(500:B2PRNT4,B2CNT4)
329  260  B2CPI4      COMPARE     PSSP.API EQ 1
330  261  B2PRNT4     ADVANCE
331  262              PRINT
332  265              ADVANCE     GOTO(B2CNT4)
333  266  B2CNT4      ASSIGN      TRANS.CT,PSTRANS.CT+1
334  267              ADVANCE     GOTO(B2NEXT4,B2LAST4)
335  268  B2NEXT4     COMPARE     PSSP.NTRAN GT PSTRANS.CT
336  269              ADVANCE     GOTO(B2LOOP4)
337  270  B2LAST4     ADVANCE     GOTO(B2NEXTSP)
338  271  B2NEXTSP    RELEASE     F(PSPOINTER)
339       * * * END PROCESSOR CALL 2 * * * * * * * * * * * * * * *
```

```
340  272                ADVANCE       GOTO(ASSN.3)
341  273  WAIT.2        QUEUE         Q(PSPOINTER+10)
342       * * * BEGIN RING CALL 1 * * * * * * * * * * * * * * * * * *
343  274                SAVEX         RING.CTR,0
344  275  B1BEGIN       ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
345  276  B1RIP0        LEAVE         S(PSPOINTER)   GOTO(B1COMP1,B1RIP1,B1RGLOOP)
346  277  B1COMP1       COMPARE       PSPOINTER EQ P$SP.AFS
347  278                ADVANCE       GOTO(B1ENDRNG)
348  279  B1RIP1        COMPARE       PSPOINTER LT VSNO.PROCS
349  280                ASSIGN        POINTER,PSPOINTER+1
350  281                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
351  282                LEAVE         S(PSPOINTER)   GOTO(B1COMP2,B1RIP2,B1RGLOOP)
352  283  B1COMP2       COMPARE       PSPOINTER EQ P$SP.AFS
353  284                ADVANCE       GOTO(B1ENDRNG)
354  285  B1RIP2        COMPARE       PSPOINTER LT VSNO.PROCS
355  286                ASSIGN        POINTER,PSPOINTER+1
356  287                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
357  288                LEAVE         S(PSPOINTER)   GOTO(B1COMP3,B1RIP3,B1RGLOOP)
358  289  B1COMP3       COMPARE       PSPOINTER EQ P$SP.AFS
359  290                ADVANCE       GOTO(B1ENDRNG)
360  291  B1RIP3        COMPARE       PSPOINTER LT VSNO.PROCS
361  292                ASSIGN        POINTER,PSPOINTER+1
362  293                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
363  294                LEAVE         S(PSPOINTER)   GOTO(B1COMP4,B1RIP4,B1RGLOOP)
364  295  B1COMP4       COMPARE       PSPOINTER EQ P$SP.AFS
365  296                ADVANCE       GOTO(B1ENDRNG)
366  297  B1RIP4        COMPARE       PSPOINTER LT VSNO.PROCS
367  298                ASSIGN        POINTER,PSPOINTER+1
368  299                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
369  300                LEAVE         S(PSPOINTER)   GOTO(B1COMP5,B1RIP5,B1RGLOOP)
370  301  B1COMP5       COMPARE       PSPOINTER EQ P$SP.AFS
371  302                ADVANCE       GOTO(B1ENDRNG)
372  303  B1RIP5        COMPARE       PSPOINTER LT VSNO.PROCS
373  304                ASSIGN        POINTER,PSPOINTER+1
374  305                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
375  306                LEAVE         S(PSPOINTER)   GOTO(B1COMP6,B1ENDCMP,B1RGLOOP)
376  307  B1COMP6       COMPARE       PSPOINTER EQ P$SP.AFS
377  308                ADVANCE       GOTO(B1ENDRNG)
378  309  B1RIP6        COMPARE       PSPOINTER LT VSNO.PROCS
379  310                ASSIGN        POINTER,PSPOINTER+1
380  311                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
381  312                LEAVE         S(PSPOINTER)   GOTO(B1COMP7,B1ENDCMP,B1RGLOOP)
382  313  B1COMP7       COMPARE       PSPOINTER EQ P$SP.AFS
383  314                ADVANCE       GOTO(B1ENDRNG)
384  315  B1RIP7        COMPARE       PSPOINTER LT VSNO.PROCS
385  316                ASSIGN        POINTER,PSPOINTER+1
386  317                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
387  318                LEAVE         S(PSPOINTER)   GOTO(B1COMP8,B1ENDCMP,B1RGLOOP)
388  319  B1COMP8       COMPARE       PSPOINTER EQ P$SP.AFS
389  320                ADVANCE       GOTO(B1ENDRNG)
390  321  B1RIP8        COMPARE       PSPOINTER LT VSNO.PROCS
391  322                ASSIGN        POINTER,PSPOINTER+1
392  323                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
393  324                LEAVE         S(PSPOINTER)   GOTO(B1COMP9,B1ENDCMP,B1RGLOOP)
394  325  B1COMP9       COMPARE       PSPOINTER EQ P$SP.AFS
395  326                ADVANCE       GOTO(B1ENDRNG)
396  327  B1RIP9        COMPARE       PSPOINTER LT VSNO.PROCS
```

```
397  328                ASSIGN        POINTER,P$POINTER+1
398  329                ENTER         S(P$POINTER)   TIME(V$RIP.TIME)
399  330                LEAVE         S(P$POINTER)   GOTO(B1COMP10,B1ENDCMP,B1RGLOOP)
400  331   B1COMP10     COMPARE       P$POINTER EQ P$SP.AFS
401  332                ADVANCE       GOTO(B1ENDRNG)
402  333                SAVEX         RING.CTR,X$RING.CTR+1
403  334                ADVANCE       GOTO(B1ENDCMP,B1RGLOOP)
404  335   B1ENDCMP     COMPARE       X$RING.CTR GE 7
405  336                TERMINATE,R
406  337   B1RGLOOP     ASSIGN        POINTER,1      GOTO(B1BEGIN)
407  338   B1ENDRNG     ADVANCE
408       * * * END RING CALL 1 * * * * * * * * * * * * * * * * * * *
409  339                QUEUE         S(P$POINTER)
410       * * * BEGIN PROCESSOR CALL 3 (REMOTE) * * * * * * * * * *
411  340                SEIZE         F(P$POINTER)
412  341   B3READSP     ADVANCE       GOTO(B3FA0,B3FA1)
413  342   B3FA0        COMPARE       P$SP.AFA EQ 0
414  343                ADVANCE       GOTO(B3NEXTSP)
415  344   B3FA1        ADVANCE       GOTO(B3CFA1,B3FA2)
416  345   B3CFA1       COMPARE       P$SP.AFA EQ 1
417  346                ASSIGN        TRANS.CT,0
418  347   B3LOOP1      HOLD          F(P$POINTER)   TIME(V$PROCTIME)
419  348                INPUT
420  352                WRITDISK
421  375                ASSIGN        TRANS.CT,P$TRANS.CT+1
422  376                ADVANCE       GOTO(B3NEXT1,B3LAST1)
423  377   B3NEXT1      COMPARE       P$SP.NTRAN GT P$TRANS.CT
424  378                ADVANCE       GOTO(B3LOOP1)
425  379   B3LAST1      ADVANCE       GOTO(B3NEXTSP)
426  380   B3FA2        ADVANCE       GOTO(B3CFA2,B3FA3)
427  381   B3CFA2       COMPARE       P$SP.AFA EQ 2
428  382                ASSIGN        TRANS.CT,0
429  383   B3LOOP2      HOLD          F(P$POINTER)   TIME(V$PROCTIME)
430  384                READDISK
431  395                ADVANCE       GOTO(B3CCI2,B3CPI2,B3CNT2)
432  396   B3CCI2       COMPARE       P$SP.ACI EQ 1
433  397                ADVANCE       TIME(V$COM.TIME) GOTO(S$O:B3PRT2,B3CNT2)
434  398   B3CPI2       COMPARE       P$SP.API EQ 1
435  399   B3PRT2       ADVANCE
436  400                PRINT
437  403   B3CNT2       ASSIGN        TRANS.CT,P$TRANS.CT+1
438  404                ADVANCE       GOTO(B3NEXT2,B3LAST2)
439  405   B3NEXT2      COMPARE       P$SP.NTRAN GT P$TRANS.CT
440  406                ADVANCE       GOTO(B3LOOP2)
441  407   B3LAST2      ADVANCE       GOTO(B3NEXTSP)
442  408   B3FA3        ADVANCE       GOTO(B3CFA3,B3FA4)
443  409   B3CFA3       COMPARE       P$SP.AFA EQ 3
444  410                ASSIGN        TRANS.CT,0
445  411   B3LOOP3      HOLD          F(P$POINTER)   TIME(V$PROCTIME)
446  412                WRITDISK
447  435                ADVANCE       GOTO(B3CCI3,B3CPI3,B3CNT3)
448  436   B3CCI3       COMPARE       P$SP.ACI EQ 1
449  437                ADVANCE       TIME(V$COM.TIME) GOTO(S$O:B3PRNT3,B3CNT3)
450  438   B3CPI3       COMPARE       P$SP.API EQ 1
451  439   B3PRNT3      ADVANCE
452  440                PRINT
453  443   B3CNT3       ASSIGN        TRANS.CT,P$TRANS.CT+1
```

```
454  444                ADVANCE       GOTO(B3NEXT3,B3LAST3)
455  445  B3NEXT3       COMPARE       PSSP.NTRAN GT PSTRANS.CT
456  446                ADVANCE       GOTO(B3LOOP3)
457  447  B3LAST3       ADVANCE       GOTO(B3NEXTSP)
458  448  B3FA4         ADVANCE       GOTO(B3CFA4,B3NEXTSP)
459  449  B3CFA4        COMPARE       PSSP.AFA EQ 4
460  450                ASSIGN        TRANS.CT,0
461  451  B3LOOP4       HOLD          F(PSPOINTER)   TIME(V$PROCTIME)
462  452                ADVANCE       GOTO(B3CCI4,B3CPI4,B3CNT4)
463  453  B3CCI4        COMPARE       PSSP.ACI EQ 1
464  454                ADVANCE       TIME(V$COM.TIME) GOTO(500:B3PRNT4,B3CNT4)
465  455  B3CPI4        COMPARE       PSSP.API EQ 1
466  456  B3PRNT4       ADVANCE
467  457                PRINT
468  460                ADVANCE       GOTO(B3CNT4)
469  461  B3CNT4        ASSIGN        TRANS.CT,PSTRANS.CT+1
470  462                ADVANCE       GOTO(B3NEXT4,B3LAST4)
471  463  B3NEXT4       COMPARE       PSSP.NTRAN GT PSTRANS.CT
472  464                ADVANCE       GOTO(B3LOOP4)
473  465  B3LAST4       ADVANCE       GOTO(B3NEXT[illegible])
474  466  B3NEXTSP      RELEASE       F(PSPOINTER)
475       * * * END PROCESSOR CALL 3 * * * * * * * * * * * * * * * * *
476  467                ASSIGN        SP.AFS,PSSP.FS1
477  468                QUEUE         Q(PSPOINTER+1,0)
478       * * * BEGIN RING CALL 2 * * * * * * * * * * * * * * * * * *
479  469                SAVEX         RING.CTR,0
480  470  B2BEGIN       ENTER         S(PSPOINTER)   TIME(V$RIP.TIME)
481  471  B2RIP0        LEAVE         S(PSPOINTER)   GOTO(B2COMP1,B2RIP1,B2RGLOOP)
482  472  B2COMP1       COMPARE       PSPOINTER EQ PSSP.AFS
483  473                ADVANCE       GOTO(B2ENDRNG)
484  474  B2RIP1        COMPARE       PSPOINTER LT V$NO.PROCS
485  475                ASSIGN        POINTER,PSPOINTER+1
486  476                ENTER         S(PSPOINTER)   TIME(V$RIP.TIME)
487  477                LEAVE         S(PSPOINTER)   GOTO(B2COMP2,B2RIP2,B2RGLOOP)
488  478  B2COMP2       COMPARE       PSPOINTER EQ PSSP.AFS
489  479                ADVANCE       GOTO(B2ENDRNG)
490  480  B2RIP2        COMPARE       PSPOINTER LT V$NO.PROCS
491  481                ASSIGN        POINTER,PSPOINTER+1
492  482                ENTER         S(PSPOINTER)   TIME(V$RIP.TIME)
493  483                LEAVE         S(PSPOINTER)   GOTO(B2COMP3,B2RIP3,B2RGLOOP)
494  484  B2COMP3       COMPARE       PSPOINTER EQ PSSP.AFS
495  485                ADVANCE       GOTO(B2ENDRNG)
496  486  B2RIP3        COMPARE       PSPOINTER LT V$NO.PROCS
497  487                ASSIGN        POINTER,PSPOINTER+1
498  488                ENTER         S(PSPOINTER)   TIME(V$RIP.TIME)
499  489                LEAVE         S(PSPOINTER)   GOTO(B2COMP4,B2RIP4,B2RGLOOP)
500  490  B2COMP4       COMPARE       PSPOINTER EQ PSSP.AFS
501  491                ADVANCE       GOTO(B2ENDRNG)
502  492  B2RIP4        COMPARE       PSPOINTER LT V$NO.PROCS
503  493                ASSIGN        POINTER,PSPOINTER+1
504  494                ENTER         S(PSPOINTER)   TIME(V$RIP.TIME)
505  495                LEAVE         S(PSPOINTER)   GOTO(B2COMP5,B2RIP5,B2RGLOOP)
506  496  B2COMP5       COMPARE       PSPOINTER EQ PSSP.AFS
507  497                ADVANCE       GOTO(B2ENDRNG)
508  498  B2RIP5        COMPARE       PSPOINTER LT V$NO.PROCS
509  499                ASSIGN        POINTER,PSPOINTER+1
510  500                ENTER         S(PSPOINTER)   TIME(V$RIP.TIME)
```

```
511   501                 LEAVE         S(P1POINTER)   GOTO(B2COMP6,B2RIP6,B2RGLOOP)
512   502   B2COMP6       COMPARE       P1POINTER EQ P1SP.AFS
513   503                 ADVANCE       GOTO(B2ENDRNG)
514   504   B2RIP6        COMPARE       P1POINTER LT V1NO.PROCS
515   505                 ASSIGN        POINTER,P1POINTER+1
516   506                 ENTER         S(P1POINTER)   TIME(V1PIP.TIME)
517   507                 LEAVE         S(P1POINTER)   GOTO(B2COMP7,B2ENDCMP,B2RGLOOP)
518   508   B2COMP7       COMPARE       P1POINTER EQ P1SP.AFS
519   509                 ADVANCE       GOTO(B2ENDRNG)
520   510   B2RIP7        COMPARE       P1POINTER LT V1NO.PROCS
521   511                 ASSIGN        POINTER,P1POINTER+1
522   512                 ENTER         S(P1POINTER)   TIME(V1PIP.TIME)
523   513                 LEAVE         S(P1POINTER)   GOTO(B2COMP8,B2ENDCMP,B2RGLOOP)
524   514   B2COMP8       COMPARE       P1POINTER EQ P1SP.AFS
525   515                 ADVANCE       GOTO(B2ENDRNG)
526   516   B2RIP8        COMPARE       P1POINTER LT V1NO.PROCS
527   517                 ASSIGN        POINTER,P1POINTER+1
528   518                 ENTER         S(P1POINTER)   TIME(V1PIP.TIME)
529   519                 LEAVE         S(P1POINTER)   GOTO(B2COMP9,B2ENDCMP,B2RGLOOP)
530   520   B2COMP9       COMPARE       P1POINTER EQ P1SP.AFS
531   521                 ADVANCE       GOTO(B2ENDRNG)
532   522   B2RIP9        COMPARE       P1POINTER LT V1NO.PROCS
533   523                 ASSIGN        POINTER,P1POINTER+1
534   524                 ENTER         S(P1POINTER)   TIME(V1PIP.TIME)
535   525                 LEAVE         S(P1POINTER)   GOTO(B2COMP10,B2ENDCMP,B2RGLOOP)
536   526   B2COMP10      COMPARE       P1POINTER EQ P1SP.AFS
537   527                 ADVANCE       GOTO(B2ENDRNG)
538   528                 SAVEX         RING.CTR,X1RING.CTR+1
539   529                 ADVANCE       GOTO(B2ENDCMP,B2RGLOOP)
540   530   B2ENDCMP      COMPARE       X1RING.CTR GE 7
541   531                 TERMINATE,R
542   532   B2RGLOOP      ASSIGN        POINTER,1      GOTO(B2BEGIN)
543   533   B2ENDRNG      ADVANCE
544         * * * END RING CALL 2 * * * * * * * * * * * * * * * * * * *
545   534                 QUEUE         Q(P1SPOINTER)
546   535                 SEIZE         F(P1SPOINTER)   TIME(V1SPROCTIME)
547   536                 RELEASE       F(P1SPOINTER)
548   537   ASSN.3        ASSIGN        SP.AFS,P1SP.FS3
549   538                 ASSIGN        SP.AFA,P1SP.FA3
550   539                 ASSIGN        SP.ACI,P1SP.CI3
551   540                 ASSIGN        SP.API,P1SP.PI3
552   541                 ASSIGN        SP.ATRAN,P1SP.TRAN3
553   542                 TABULATE      JOB.TRAK
554   543                 ADVANCE       GOTO(SKIP.3,SPEC.3,WAIT.3)
555   544   SKIP.3        COMPARE       P1SP.AFA EQ 0
556   545                 ADVANCE       GOTO(ASSN.4)
557   546   SPEC.3        COMPARE       P1POINTER EQ P1SP.AFS TIME(FNSLOOK.UP)
558         * * * BEGIN PROCESSOR CALL 4 * * * * * * * * * * * * * * *
559   547                 SEIZE         F(P1POINTER)
560   548   B4READSP      ADVANCE       GOTO(B4CFA1,B4FA2)
561   549   B4CFA1        COMPARE       P1SP.AFA EQ 1
562   550                 ASSIGN        TRANS.CT,0
563   551   B4LOOP1       HOLD          F(P1POINTER)   TIME(V1SPROCTIME)
564   552                 INPUT
565   556                 WRITDISK
566   579                 ASSIGN        TRANS.CT,P1TRANS.CT+1
567   580                 ADVANCE       GOTO(B4NEXT1,B4LAST1)
```

```
568  581  B4NEXT1   COMPARE      PSSP.NTRAN GT PSTRANS.CT
569  582            ADVANCE      GOTO(B4LOOP1)
570  583  B4LAST1   ADVANCE      GOTO(B4NEXTSP)
571  584  B4FA2     ADVANCE      GOTO(B4CFA2,B4FA3)
572  585  B4CFA2    COMPARE      PSSP.AFA EQ 2
573  586            ASSIGN       TRANS.CT,0
574  587  B4LOOP2   HOLD         F(PSPOINTER)   TIME(VSPROCTIME)
575  588            READDISK
576  599            ADVANCE      GOTO(B4CCI2,B4CPI2,B4CNT2)
577  600  B4CCI2    COMPARE      PSSP.ACI EQ 1
578  601            ADVANCE      TIME(VSCOM.TIME) GOTO(500:B4PRT2,B4CNT2)
579  602  B4CPI2    COMPARE      PSSP.API EQ 1
580  603  B4PRT2    ADVANCE
581  604            PRINT
582  607  B4CNT2    ASSIGN       TRANS.CT,PSTRANS.CT+1
583  608            ADVANCE      GOTO(B4NEXT2,B4LAST2)
584  609  B4NEXT2   COMPARE      PSSP.NTRAN GT PSTRANS.CT
585  610            ADVANCE      GOTO(B4LOOP2)
586  611  B4LAST2   ADVANCE      GOTO(B4NEXTSP)
587  612  B4FA3     ADVANCE      GOTO(B4CFA3,B4FA4)
588  613  B4CFA3    COMPARE      PSSP.AFA EQ 3
589  614            ASSIGN       TRANS.CT,0
590  615  B4LOOP3   HOLD         F(PSPOINTER)   TIME(VSPROCTIME)
591  616            WRITDISK
592  639            ADVANCE      GOTO(B4CCI3,B4CPI3,B4CNT3)
593  640  B4CCI3    COMPARE      PSSP.ACI EQ 1
594  641            ADVANCE      TIME(VSCOM.TIME) GOTO(500:B4PRNT3,B4CNT3)
595  642  B4CPI3    COMPARE      PSSP.API EQ 1
596  643  B4PRNT3   ADVANCE
597  644            PRINT
598  647  B4CNT3    ASSIGN       TRANS.CT,PSTRANS.CT+1
599  648            ADVANCE      GOTO(B4NEXT3,B4LAST3)
600  649  B4NEXT3   COMPARE      PSSP.NTRAN GT PSTRANS.CT
601  650            ADVANCE      GOTO(B4LOOP3)
602  651  B4LAST3   ADVANCE      GOTO(B4NEXTSP)
603  652  B4FA4     ADVANCE      GOTO(B4CFA4,B4NEXTSP)
604  653  B4CFA4    COMPARE      PSSP.AFA EQ 4
605  654            ASSIGN       TRANS.CT,0
606  655  B4LOOP4   HOLD         F(PSPOINTER)   TIME(VSPROCTIME)
607  656            ADVANCE      GOTO(B4CCI4,B4CPI4,B4CNT4)
608  657  B4CCI4    COMPARE      PSSP.ACI EQ 1
609  658            ADVANCE      TIME(VSCOM.TIME) GOTO(500:B4PRNT4,B4CNT4)
610  659  B4CPI4    COMPARE      PSSP.API EQ 1
611  660  B4PRNT4   ADVANCE
612  661            PRINT
613  664            ADVANCE      GOTO(B4CNT4)
614  665  B4CNT4    ASSIGN       TRANS.CT,PSTRANS.CT+1
615  666            ADVANCE      GOTO(B4NEXT4,B4LAST4)
616  667  B4NEXT4   COMPARE      PSSP.NTRAN GT PSTRANS.CT
617  668            ADVANCE      GOTO(B4LOOP4)
618  669  B4LAST4   ADVANCE      GOTO(B4NEXTSP)
619  670  B4NEXTSP  RELEASE      F(PSPOINTER)
620       * * * END PROCESSOR CALL 4 * * * * * * * * * * * * * * * *
621  671            ADVANCE      GOTO(ASSN.4)
622  672  WAIT.3    QUEUE        Q(PSPOINTER+10)
623       * * * BEGIN RING CALL 3 * * * * * * * * * * * * * * * * * *
624  673            SAVEX        RING.CTR,0
```

```
625  674  B3BEGIN  ENTER      S(PSPOINTER)   TIME(VSRIP.TIME)
626  675  B3RIP0   LEAVE      S(PSPOINTER)   GOTO(B3COMP1,B3RIP1,B3RGLOOP)
627  676  B3COMP1  COMPARE    PSPOINTER EQ PSSP.AFS
628  677           ADVANCE     GOTO(B3ENDRNG)
629  678  B3RIP1   COMPARE    PSPOINTER LT VSNO.PROCS
630  679           ASSIGN      POINTER,PSPOINTER+1
631  680           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
632  681           LEAVE       S(PSPOINTER)   GOTO(B3COMP2,B3RIP2,B3RGLOOP)
633  682  B3COMP2  COMPARE    PSPOINTER EQ PSSP.AFS
634  683           ADVANCE     GOTO(B3ENDRNG)
635  684  B3RIP2   COMPARE    PSPOINTER LT VSNO.PROCS
636  685           ASSIGN      POINTER,PSPOINTER+1
637  686           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
638  687           LEAVE       S(PSPOINTER)   GOTO(B3COMP3,B3RIP3,B3RGLOOP)
639  688  B3COMP3  COMPARE    PSPOINTER EQ PSSP.AFS
640  689           ADVANCE     GOTO(B3ENDRNG)
641  690  B3RIP3   COMPARE    PSPOINTER LT VSNO.PROCS
642  691           ASSIGN      POINTER,PSPOINTER+1
643  692           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
644  693           LEAVE       S(PSPOINTER)   GOTO(B3COMP4,B3RIP4,B3RGLOOP)
645  694  B3COMP4  COMPARE    PSPOINTER EQ PSSP.AFS
646  695           ADVANCE     GOTO(B3ENDRNG)
647  696  B3RIP4   COMPARE    PSPOINTER LT VSNO.PROCS
648  697           ASSIGN      POINTER,PSPOINTER+1
649  698           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
650  699           LEAVE       S(PSPOINTER)   GOTO(B3COMP5,B3RIP5,B3RGLOOP)
651  700  B3COMP5  COMPARE    PSPOINTER EQ PSSP.AFS
652  701           ADVANCE     GOTO(B3ENDRNG)
653  702  B3RIP5   COMPARE    PSPOINTER LT VSNO.PROCS
654  703           ASSIGN      POINTER,PSPOINTER+1
655  704           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
656  705           LEAVE       S(PSPOINTER)   GOTO(B3COMP6,B3RIP6,B3RGLOOP)
657  706  B3COMP6  COMPARE    PSPOINTER EQ PSSP.AFS
658  707           ADVANCE     GOTO(B3ENDRNG)
659  708  B3RIP6   COMPARE    PSPOINTER LT VSNO.PROCS
660  709           ASSIGN      POINTER,PSPOINTER+1
661  710           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
662  711           LEAVE       S(PSPOINTER)   GOTO(B3COMP7,B3ENDCMP,B3RGLOOP)
663  712  B3COMP7  COMPARE    PSPOINTER EQ PSSP.AFS
664  713           ADVANCE     GOTO(B3ENDRNG)
665  714  B3RIP7   COMPARE    PSPOINTER LT VSNO.PROCS
666  715           ASSIGN      POINTER,PSPOINTER+1
667  716           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
668  717           LEAVE       S(PSPOINTER)   GOTO(B3COMP8,B3ENDCMP,B3RGLOOP)
669  718  B3COMP8  COMPARE    PSPOINTER EQ PSSP.AFS
670  719           ADVANCE     GOTO(B3ENDRNG)
671  720  B3RIP8   COMPARE    PSPOINTER LT VSNO.PROCS
672  721           ASSIGN      POINTER,PSPOINTER+1
673  722           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
674  723           LEAVE       S(PSPOINTER)   GOTO(B3COMP9,B3ENDCMP,B3RGLOOP)
675  724  B3COMP9  COMPARE    PSPOINTER EQ PSSP.AFS
676  725           ADVANCE     GOTO(B3ENDRNG)
677  726  B3RIP9   COMPARE    PSPOINTER LT VSNO.PROCS
678  727           ASSIGN      POINTER,PSPOINTER+1
679  728           ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
680  729           LEAVE       S(PSPOINTER)   GOTO(B3COMP10,B3ENDCMP,B3RGLOOP)
681  730  B3COMP10 COMPARE    PSPOINTER EQ PSSP.AFS
```

```
682  731                ADVANCE        GOTO(B3ENDRNG)
683  732                SAVEX          RING.CTR,XRING.CTR+1
684  733                ADVANCE        GOTO(B3ENDCMP,B3RGLOOP)
685  734  B3ENDCMP      COMPARE        XRING.CTR GE 7
686  735                TERMINATE,R
687  736  B3RGLOOP      ASSIGN         POINTER,1       GOTO(B3BEGIN)
688  737  B3ENDRNG      ADVANCE
689       * * * END RING CALL 3 * * * * * * * * * * * * * * * * * * *
690  738                QUEUE          Q(PSPOINTER)
691  739                SEIZE          F(PSPOINTER)
692       * * * BEGIN PROCESSOR CALL 5 (REMOTE) * * * * * * * * * * * * * * * *
693  740  B5READSP      ADVANCE        GOTO(B5FA0,B5FA1)
694  741  B5FA0         COMPARE        PSSP.AFA EQ 0
695  742                ADVANCE        GOTO(B5NEXTSP)
696  743  B5FA1         ADVANCE        GOTO(B5CFA1,B5FA2)
697  744  B5CFA1        COMPARE        PSSP.AFA EQ 1
698  745                ASSIGN         TRANS.CT,0
699  746  B5LOOP1       HOLD           F(PSPOINTER)    TIME(VSPROCTIME)
700  747                INPUT
701  751                WRITDISK
702  774                ASSIGN         TRANS.CT,PSTRANS.CT+1
703  775                ADVANCE        GOTO(B5NEXT1,B5LAST1)
704  776  B5NEXT1       COMPARE        PSSP.NTRAN GT PSTRANS.CT
705  777                ADVANCE        GOTO(B5LOOP1)
706  778  B5LAST1       ADVANCE        GOTO(B5NEXTSP)
707  779  B5FA2         ADVANCE        GOTO(B5CFA2,B5FA3)
708  780  B5CFA2        COMPARE        PSSP.AFA EQ 2
709  781                ASSIGN         TRANS.CT,0
710  782  B5LOOP2       HOLD           F(PSPOINTER)    TIME(VSPROCTIME)
711  783                READDISK
712  794                ADVANCE        GOTO(B5CCI2,B5CPI2,B5CNT2)
713  795  B5CCI2        COMPARE        PSSP.ACI EQ 1
714  796                ADVANCE        TIME(VSCOM.TIME) GOTO(500:B5PRT2,B5CNT2)
715  797  B5CPI2        COMPARE        PSSP.API EQ 1
716  798  B5PRT2        ADVANCE
717  799                PRINT
718  802  B5CNT2        ASSIGN         TRANS.CT,PSTRANS.CT+1
719  803                ADVANCE        GOTO(B5NEXT2,B5LAST2)
720  804  B5NEXT2       COMPARE        PSSP.NTRAN GT PSTRANS.CT
721  805                ADVANCE        GOTO(B5LOOP2)
722  806  B5LAST2       ADVANCE        GOTO(B5NEXTSP)
723  807  B5FA3         ADVANCE        GOTO(B5CFA3,B5FA4)
724  808  B5CFA3        COMPARE        PSSP.AFA EQ 3
725  809                ASSIGN         TRANS.CT,0
726  810  B5LOOP3       HOLD           F(PSPOINTER)    TIME(VSPROCTIME)
727  811                WRITDISK
728  834                ADVANCE        GOTO(B5CCI3,B5CPI3,B5CNT3)
729  835  B5CCI3        COMPARE        PSSP.ACI EQ 1
730  836                ADVANCE        TIME(VSCOM.TIME) GOTO(500:B5PRNT3,B5CNT3)
731  837  B5CPI3        COMPARE        PSSP.API EQ 1
732  838  B5PRNT3       ADVANCE
733  839                PRINT
734  842  B5CNT3        ASSIGN         TRANS.CT,PSTRANS.CT+1
735  843                ADVANCE        GOTO(B5NEXT3,B5LAST3)
736  844  B5NEXT3       COMPARE        PSSP.NTRAN GT PSTRANS.CT
737  845                ADVANCE        GOTO(B5LOOP3)
738  846  B5LAST3       ADVANCE        GOTO(B5NEXTSP)
```

```
739  847  B5FA4     ADVANCE     GOTO(B5CFA4,B5NEXTSP)
740  848  B5CFA4    COMPARE     P$SP.AFA EQ 4
741  849             ASSIGN      TRANS.CT,0
742  850  B5LOOP4   HOLD        F(P$POINTER)   TIME(V$PROCTIME)
743  851             ADVANCE     GOTO(B5CCI4,B5CPI4,B5CNT4)
744  852  B5CCI4    COMPARE     P$SP.ACI EQ 1
745  853             ADVANCE     TIME(V$CON.TIME) GOTO(B5CC:B5PRNT4,B5CNT4)
746  854  B5CPI4    COMPARE     P$SP.API EQ 1
747  855  B5PRNT4   ADVANCE
748  856             PRINT
749  859             ADVANCE     GOTO(B5CNT4)
750  860  B5CNT4    ASSIGN      TRANS.CT,P$TRANS.CT+1
751  861             ADVANCE     GOTO(B5NEXT4,B5LAST4)
752  862  B5NEXT4   COMPARE     P$SP.NTRAN GT P$TRANS.CT
753  863             ADVANCE     GOTO(B5LOOP4)
754  864  B5LAST4   ADVANCE     GOTO(B5NEXTSP)
755  865  B5NEXTSP  RELEASE     F(P$POINTER)
756       * * * END PROCESSOR CALL 5 * * * * * * * * * * * * * * *
757  866             ASSIGN      SP.AFS,P$SP.FS1
758  867             QUEUE       Q(P$POINTER [..)
759       * * * BEGIN RING CALL 4 * * * * * * * * * * * * * * * *
760  868             SAVEX       RING.CTR,0
761  869  B4BEGIN   ENTER       S(P$POINTER)   TIME(V$RIP.TIME)
762  870  B4RIP1    LEAVE       S(P$POINTER)   GOTO(B4COMP1,B4RIP1,B4RGLOOP)
763  871  B4COMP1   COMPARE     P$POINTER EQ P$SP.AFS
764  872             ADVANCE     GOTO(B4ENDRNG)
765  873  B4RIP1    COMPARE     P$POINTER LT V$NO.PROCS
766  874             ASSIGN      POINTER,P$POINTER+1
767  875             ENTER       S(P$POINTER)   TIME(V$RIP.TIME)
768  876             LEAVE       S(P$POINTER)   GOTO(B4COMP2,B4RIP2,B4RGLOOP)
769  877  B4COMP2   COMPARE     P$POINTER EQ P$SP.AFS
770  878             ADVANCE     GOTO(B4ENDRNG)
771  879  B4RIP2    COMPARE     P$POINTER LT V$NO.PROCS
772  880             ASSIGN      POINTER,P$POINTER+1
773  881             ENTER       S(P$POINTER)   TIME(V$RIP.TIME)
774  882             LEAVE       S(P$POINTER)   GOTO(B4COMP3,B4RIP3,B4RGLOOP)
775  883  B4COMP3   COMPARE     P$POINTER EQ P$SP.AFS
776  884             ADVANCE     GOTO(B4ENDRNG)
777  885  B4RIP3    COMPARE     P$POINTER LT V$NO.PROCS
778  886             ASSIGN      POINTER,P$POINTER+1
779  887             ENTER       S(P$POINTER)   TIME(V$RIP.TIME)
780  888             LEAVE       S(P$POINTER)   GOTO(B4COMP4,B4RIP4,B4RGLOOP)
781  889  B4COMP4   COMPARE     P$POINTER EQ P$SP.AFS
782  890             ADVANCE     GOTO(B4ENDRNG)
783  891  B4RIP4    COMPARE     P$POINTER LT V$NO.PROCS
784  892             ASSIGN      POINTER,P$POINTER+1
785  893             ENTER       S(P$POINTER)   TIME(V$RIP.TIME)
786  894             LEAVE       S(P$POINTER)   GOTO(B4COMP5,B4RIP5,B4RGLOOP)
787  895  B4COMP5   COMPARE     P$POINTER EQ P$SP.AFS
788  896             ADVANCE     GOTO(B4ENDRNG)
789  897  B4RIP5    COMPARE     P$POINTER LT V$NO.PROCS
790  898             ASSIGN      POINTER,P$POINTER+1
791  899             ENTER       S(P$POINTER)   TIME(V$RIP.TIME)
792  900             LEAVE       S(P$POINTER)   GOTO(B4COMP6,B4RIP6,B4RGLOOP)
793  901  B4COMP6   COMPARE     P$POINTER EQ P$SP.AFS
794  902             ADVANCE     GOTO(B4ENDRNG)
795  903  B4RIP6    COMPARE     P$POINTER LT V$NO.PROCS
```

```
796  904                ASSIGN        POINTER,PSPOINTER+1
797  905                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
798  906                LEAVE         S(PSPOINTER)   GOTO(B4COMP7,B4ENDCMP,B4RGLOOP)
799  907  B4COMP7       COMPARE       PSPOINTER EQ PSSP.AFS
800  908                ADVANCE       GOTO(B4ENDRNG)
801  909  B4RIP7        COMPARE       PSPOINTER LT VSNO.PROCS
802  910                ASSIGN        POINTER,PSPOINTER+1
803  911                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
804  912                LEAVE         S(PSPOINTER)   GOTO(B4COMP8,B4ENDCMP,B4RGLOOP)
805  913  B4COMP8       COMPARE       PSPOINTER EQ PSSP.AFS
806  914                ADVANCE       GOTO(B4ENDRNG)
807  915  B4RIP8        COMPARE       PSPOINTER LT VSNO.PROCS
808  916                ASSIGN        POINTER,PSPOINTER+1
809  917                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
810  918                LEAVE         S(PSPOINTER)   GOTO(B4COMP9,B4ENDCMP,B4RGLOOP)
811  919  B4COMP9       COMPARE       PSPOINTER EQ PSSP.AFS
812  920                ADVANCE       GOTO(B4ENDRNG)
813  921  B4RIP9        COMPARE       PSPOINTER LT VSNO.PROCS
814  922                ASSIGN        POINTER,PSPOINTER+1
815  923                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
816  924                LEAVE         S(PSPOINTER)   GOTO(B4COMP10,B4ENDCMP,B4RGLOOP)
817  925  B4COMP10      COMPARE       PSPOINTER EQ PSSP.AFS
818  926                ADVANCE       GOTO(B4ENDRNG)
819  927                SAVEX         RING.CTR,XSRING.CTR+1
820  928                ADVANCE       GOTO(B4ENDCMP,B4RGLOOP)
821  929  B4ENDCMP      COMPARE       XSRING.CTR GE 7
822  930                TERMINATE,R
823  931  B4RGLOOP      ASSIGN        POINTER,1      GOTO(B4BEGIN)
824  932  B4ENDRNG      ADVANCE
825       * * * END RING CALL 4 * * * * * * * * * * * * * * * * * * *
826  933                QUEUE         Q(PSPOINTER)
827  934                SEIZE         F(PSPOINTER)   TIME(VSPROCTIME)
828  935                RELEASE       F(PSPOINTER)
829  936  ASSN.4        ASSIGN        SP.AFA,PSSP.FA4
830  937                ASSIGN        SP.AFS,PSSP.FS4
831  938                ASSIGN        SP.ACI,PSSP.CI4
832  939                ASSIGN        SP.API,PSSP.PI4
833  940                ASSIGN        SP.XTRAN,PSSP.TRAN4
834  941                TABULATE      JOB.TRAK
835  942                ADVANCE       GOTO(SKIP.4,SPEC.4,*WAIT.4)
836  943  SKIP.4        COMPARE       PSSP.AFA EQ 0
837  944                ADVANCE       GOTO(ASSN.5)
838  945  SPEC.4        COMPARE       PSPOINTER EQ PSSP.AFS TIME(FNSLOOK.UP)
839       * * * BEGIN PROCESSOR CALL 6 * * * * * * * * * * * * * * * *
840  946                SEIZE         F(PSPOINTER)
841  947  B6READSP      ADVANCE       GOTO(B6CFA1,B6FA2)
842  948  B6CFA1        COMPARE       PSSP.AFA EQ 1
843  949                ASSIGN        TRANS.CT,0
844  950  B6LOOP1       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
845  951                INPUT
846  955                WRITDISK
847  978                ASSIGN        TRANS.CT,PSTRANS.CT+1
848  979                ADVANCE       GOTO(B6NEXT1,B6LAST1)
849  980  B6NEXT1       COMPARE       PSSP.XTRAN GT PSTRANS.CT
850  981                ADVANCE       GOTO(B6LOOP1)
851  982  B6LAST1       ADVANCE       GOTO(B6NEXTSP)
852  983  B6FA2         ADVANCE       GOTO(B6CFA2,B6FA3)
```

```
853  984  B6CFA2    COMPARE     PSSP.AFA EQ 2
854  985             ASSIGN      TRANS.CT,C
855  986  B6LOOP2    HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
856  987             READDISK
857  988             ADVANCE     GOTO(B6CCI2,B6CPI2,B6CNT2)
858  989  B6CCI2     COMPARE     PSSP.ACI EQ 1
859  1000            ADVANCE     TIME(VSCOM.TIME) GOTO(SUU:B6PRT2,B6CNT2)
860  1001 B6CPI2     COMPARE     PSSP.API EQ 1
861  1002 B6PRT2     ADVANCE
862  1003            PRINT
863  1006 B6CNT2     ASSIGN      TRANS.CT,PSTRANS.CT+1
864  1007            ADVANCE     GOTO(B6NEXT2,B6LAST2)
865  1008 B6NEXT2    COMPARE     PSSP.NTRAN GT PSTRANS.CT
866  1009            ADVANCE     GOTO(B6LOOP2)
867  1010 B6LAST2    ADVANCE     GOTO(B6NEXTSP)
868  1011 B6FA3      ADVANCE     GOTO(B6CFA3,B6FA4)
869  1012 B6CFA3     COMPARE     PSSP.AFA EQ 3
870  1013            ASSIGN      TRANS.CT,C
871  1014 B6LOOP3    HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
872  1015            WRITDISK
873  1038            ADVANCE     GOTO(B6CCI3,B6CPI3,B6CNT3)
874  1039 B6CCI3     COMPARE     PSSP.ACI EQ 1
875  1040            ADVANCE     TIME(VSCOM.TIME) GOTO(500:B6PRNT3,B6CNT3)
876  1041 B6CPI3     COMPARE     PSSP.API EQ 1
877  1042 B6PRNT3    ADVANCE
878  1043            PRINT
879  1046 B6CNT3     ASSIGN      TRANS.CT,PSTRANS.CT+1
880  1047            ADVANCE     GOTO(B6NEXT3,B6LAST3)
881  1048 B6NEXT3    COMPARE     PSSP.NTRAN GT PSTRANS.CT
882  1049            ADVANCE     GOTO(B6LOOP3)
883  1050 B6LAST3    ADVANCE     GOTO(B6NEXTSP)
884  1051 B6FA4      ADVANCE     GOTO(B6CFA4,B6NEXTSP)
885  1052 B6CFA4     COMPARE     PSSP.AFA EQ 4
886  1053            ASSIGN      TRANS.CT,C
887  1054 B6LOOP4    HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
888  1055            ADVANCE     GOTO(B6CCI4,B6CPI4,B6CNT4)
889  1056 B6CCI4     COMPARE     PSSP.ACI EQ 1
890  1057            ADVANCE     TIME(VSCOM.TIME) GOTO(500:B6PRNT4,B6CNT4)
891  1058 B6CPI4     COMPARE     PSSP.API EQ 1
892  1059 B6PRNT4    ADVANCE
893  1060            PRINT
894  1063            ADVANCE     GOTO(B6CNT4)
895  1064 B6CNT4     ASSIGN      TRANS.CT,PSTRANS.CT+1
896  1065            ADVANCE     GOTO(B6NEXT4,B6LAST4)
897  1066 B6NEXT4    COMPARE     PSSP.NTRAN GT PSTRANS.CT
898  1067            ADVANCE     GOTO(B6LOOP4)
899  1068 B6LAST4    ADVANCE     GOTO(B6NEXTSP)
900  1069 B6NEXTSP   RELEASE     F(PSPOINTER)
901          * * * END PROCESSOR CALL 6 * * * * * * * * * * * * * * * *
902  1070            ADVANCE     GOTO(ASSN.5)
903  1071 WAIT.4     QUEUE       S(PSPOINTER+10)
904          * * * BEGIN RING CALL 5 * * * * * * * * * * * * * * * * *
905  1072            SAVEX       RING.CTR,3
906  1073 B5BEGIN    ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
907  1074 B5RIP3     LEAVE       S(PSPOINTER)   GOTO(B5COMP1,B5RIP1,B5RGLOOP)
908  1075 B5COMP1    COMPARE     PSPOINTER EQ PSSP.AFS
909  1076            ADVANCE     GOTO(B5ENDRNG)
```

```
910 1077  @SRIP1     COMPARE     PSPOINTER LT VSNO.PROCS
911 1078              ASSIGN      POINTER,PSPOINTER+1
912 1079              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
913 1080              LEAVE       S(PSPOINTER)   GOTO(@SCOMP2,@SRIP2,@SRGLOOP)
914 1081  @SCOMP2     COMPARE     PSPOINTER EQ PSSP.AFS
915 1082              ADVANCE     GOTO(@SENDRNG)
916 1083  @SRIP2      COMPARE     PSPOINTER LT VSNO.PROCS
917 1084              ASSIGN      POINTER,PSPOINTER+1
918 1085              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
919 1086              LEAVE       S(PSPOINTER)   GOTO(@SCOMP3,@SRIP3,@SRGLOOP)
920 1087  @SCOMP3     COMPARE     PSPOINTER EQ PSSP.AFS
921 1088              ADVANCE     GOTO(@SENDRNG)
922 1089  @SRIP3      COMPARE     PSPOINTER LT VSNO.PROCS
923 1090              ASSIGN      POINTER,PSPOINTER+1
924 1091              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
925 1092              LEAVE       S(PSPOINTER)   GOTO(@SCOMP4,@SRIP4,@SRGLOOP)
926 1093  @SCOMP4     COMPARE     PSPOINTER EQ PSSP.AFS
927 1094              ADVANCE     GOTO(@SENDRNG)
928 1095  @SRIP4      COMPARE     PSPOINTER LT VSNO.PROCS
929 1096              ASSIGN      POINTER,PSPOINTER+1
930 1097              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
931 1098              LEAVE       S(PSPOINTER)   GOTO(@SCOMP5,@SRIP5,@SRGLOOP)
932 1099  @SCOMP5     COMPARE     PSPOINTER EQ PSSP.AFS
933 1100              ADVANCE     GOTO(@SENDRNG)
934 1101  @SRIP5      COMPARE     PSPOINTER LT VSNO.PROCS
935 1102              ASSIGN      POINTER,PSPOINTER+1
936 1103              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
937 1104              LEAVE       S(PSPOINTER)   GOTO(@SCOMP6,@SRIP6,@SRGLOOP)
938 1105  @SCOMP6     COMPARE     PSPOINTER EQ PSSP.AFS
939 1106              ADVANCE     GOTO(@SENDRNG)
940 1107  @SRIP6      COMPARE     PSPOINTER LT VSNO.PROCS
941 1108              ASSIGN      POINTER,PSPOINTER+1
942 1109              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
943 1110              LEAVE       S(PSPOINTER)   GOTO(@SCOMP7,@SENDCMP,@SRGLOOP)
944 1111  @SCOMP7     COMPARE     PSPOINTER EQ PSSP.AFS
945 1112              ADVANCE     GOTO(@SENDRNG)
946 1113  @SRIP7      COMPARE     PSPOINTER LT VSNO.PROCS
947 1114              ASSIGN      POINTER,PSPOINTER+1
948 1115              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
949 1116              LEAVE       S(PSPOINTER)   GOTO(@SCOMP8,@SENDCMP,@SRGLOOP)
950 1117  @SCOMP8     COMPARE     PSPOINTER EQ PSSP.AFS
951 1118              ADVANCE     GOTO(@SENDRNG)
952 1119  @SRIP8      COMPARE     PSPOINTER LT VSNO.PROCS
953 1120              ASSIGN      POINTER,PSPOINTER+1
954 1121              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
955 1122              LEAVE       S(PSPOINTER)   GOTO(@SCOMP9,@SENDCMP,@SRGLOOP)
956 1123  @SCOMP9     COMPARE     PSPOINTER EQ PSSP.AFS
957 1124              ADVANCE     GOTO(@SENDRNG)
958 1125  @SRIP9      COMPARE     PSPOINTER LT VSNO.PROCS
959 1126              ASSIGN      POINTER,PSPOINTER+1
960 1127              ENTER       S(PSPOINTER)   TIME(VSRIP.TIME)
961 1128              LEAVE       S(PSPOINTER)   GOTO(@SCOMP10,@SENDCMP,@SRGLOOP)
962 1129  @SCOMP10    COMPARE     PSPOINTER EQ PSSP.AFS
963 1130              ADVANCE     GOTO(@SENDRNG)
964 1131              SAVEX       ARING.CTR,ARING.CTR+1
965 1132              ADVANCE     GOTO(@SENDCMP,PSPGLOOP)
966 1133  @SENDCMP    COMPARE     ARING.CTR GE 7
```

```
 967 1134                TERMINATE,R
 968 1135  B5RGLOOP      ASSIGN        POINTER,1      GOTO(B5BEGIN)
 969 1136  B5ENDRNG      ADVANCE
 97_       * * * END RING CALL 5 * * * * * * * * * * * * * * * * * *
 971 1137                GOODBYE       Q(PSPOINTER)
 972       * * * BEGIN PROCESSOR CALL 7 (REMOTE) * * * * * * * * *
 973 1138                SEIZE         F(PSPOINTER)
 974 1139  B7READSP      ADVANCE       GOTO(B7FA0,B7FA1)
 975 1140  B7FA0         COMPARE       PSSP.AFA EQ 0
 976 1141                ADVANCE       GOTO(B7NEXTSP)
 977 1142  B7FA1         ADVANCE       GOTO(B7CFA1,B7FA2)
 978 1143  B7CFA1        COMPARE       PSSP.AFA EQ 1
 979 1144                ASSIGN        TRANS.CT,0
 980 1145  B7LOOP1       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
 981 1146                INPUT
 982 1150                WRITDISK
 983 1173                ASSIGN        TRANS.CT,PSTRANS.CT+1
 984 1174                ADVANCE       GOTO(B7NEXT1,B7LAST1)
 985 1175  B7NEXT1       COMPARE       PSSP.NTRAN GT PSTRANS.CT
 986 1176                ADVANCE       GOTO(B7LOOP1.
 987 1177  B7LAST1       ADVANCE       GOTO(B7NEXTSP)
 988 1178  B7FA2         ADVANCE       GOTO(B7CFA2,B7FA3)
 989 1179  B7CFA2        COMPARE       PSSP.AFA EQ 2
 990 1180                ASSIGN        TRANS.CT,0
 991 1181  B7LOOP2       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
 992 1182                READDISK
 993 1193                ADVANCE       GOTO(B7CCI2,B7CPI2,B7CNT2)
 994 1194  B7CCI2        COMPARE       PSSP.ACI EQ 1
 995 1195                ADVANCE       TIME(VSCOM.TIME) GOTO(500:B7PRT2,B7CNT2)
 996 1196  B7CPI2        COMPARE       PSSP.API EQ 1
 997 1197  B7PRT2        ADVANCE
 998 1198                PRINT
 999 1201  B7CNT2        ASSIGN        TRANS.CT,PSTRANS.CT+1
1000 1202                ADVANCE       GOTO(B7NEXT2,B7LAST2)
1001 1203  B7NEXT2       COMPARE       PSSP.NTRAN GT PSTRANS.CT
1002 1204                ADVANCE       GOTO(B7LOOP2)
1003 1205  B7LAST2       ADVANCE       GOTO(B7NEXTSP)
1004 1206  B7FA3         ADVANCE       GOTO(B7CFA3,B7FA4)
1005 1207  B7CFA3        COMPARE       PSSP.AFA EQ 3
1006 1208                ASSIGN        TRANS.CT,0
1007 1209  B7LOOP3       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
1008 1210                WRITDISK
1009 1233                ADVANCE       GOTO(B7CCI3,B7CPI3,B7CNT3)
1010 1234  B7CCI3        COMPARE       PSSP.ACI EQ 1
1011 1235                ADVANCE       TIME(VSCOM.TIME) GOTO(500:B7PRT3,B7CNT3)
1012 1236  B7CPI3        COMPARE       PSSP.API EQ 1
1013 1237  B7PRT3        ADVANCE
1014 1238                PRINT
1015 1241  B7CNT3        ASSIGN        TRANS.CT,PSTRANS.CT+1
1016 1242                ADVANCE       GOTO(B7NEXT3,B7LAST3)
1017 1243  B7NEXT3       COMPARE       PSSP.NTRAN GT PSTRANS.CT
1018 1244                ADVANCE       GOTO(B7LOOP3)
1019 1245  B7LAST3       ADVANCE       GOTO(B7NEXTSP)
1020 1246  B7FA4         ADVANCE       GOTO(B7CFA4,B7NEXTSP)
1021 1247  B7CFA4        COMPARE       PSSP.AFA EQ 4
1022 1248                ASSIGN        TRANS.CT,0
1023 1249  B7LOOP4       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
```

```
1024 1250                ADVANCE       GOTO(B7CCI4,B7CPI4,B7CNT4)
1025 1251   B7CCI4       COMPARE       PSSP.ACI EQ 1
1026 1252                ADVANCE       TIME(VSCOM.TIME) GOTO(5,0:B7PRNT4,B7CNT4)
1027 1253   B7CPI4       COMPARE       PSSP.API EQ 1
1028 1254   B7PRNT4      ADVANCE
1029 1255                PRINT
1030 1256                ADVANCE       GOTO(B7CNT4)
1031 1259   B7CNT4       ASSIGN        TRANS.CT,PSTRANS.CT+1
1032 1260                ADVANCE       GOTO(B7NEXT4,B7LAST4)
1033 1261   B7NEXT4      COMPARE       PSSP.NTRAN GT PSTRANS.CT
1034 1262                ADVANCE       GOTO(B7LOOP4)
1035 1263   B7LAST4      ADVANCE       GOTO(B7NEXTSP)
1036 1264   B7NEXTSP     RELEASE       F(PSPOINTER)
1037                * * * END PROCESSOR CALL 7 * * * * * * * * * * * * * * * *
1038 1265                ASSIGN        SP.AFS,PSSP.FS1
1039 1266                QUEUE         *(PSPOINTER+1)
1040                * * * BEGIN RING CALL 6 * * * * * * * * * * * * * * * * *
1041 1267                SAVEX         RING.CTR,0
1042 1268   B6BEGIN      ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1043 1269   B6RIP0       LEAVE         S(PSPOINTER)   GOTO(B6COMP1,B6RIP1,B6RGLOOP)
1044 1270   B6COMP1      COMPARE       PSPOINTER EQ PSSP.AFS
1045 1271                ADVANCE       GOTO(B6ENDRNG)
1046 1272   B6RIP1       COMPARE       PSPOINTER LT VSNO.PROCS
1047 1273                ASSIGN        POINTER,PSPOINTER+1
1048 1274                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1049 1275                LEAVE         S(PSPOINTER)   GOTO(B6COMP2,B6RIP2,B6RGLOOP)
1050 1276   B6COMP2      COMPARE       PSPOINTER EQ PSSP.AFS
1051 1277                ADVANCE       GOTO(B6ENDRNG)
1052 1278   B6RIP2       COMPARE       PSPOINTER LT VSNO.PROCS
1053 1279                ASSIGN        POINTER,PSPOINTER+1
1054 1280                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1055 1281                LEAVE         S(PSPOINTER)   GOTO(B6COMP3,B6RIP3,B6RGLOOP)
1056 1282   B6COMP3      COMPARE       PSPOINTER EQ PSSP.AFS
1057 1283                ADVANCE       GOTO(B6ENDRNG)
1058 1284   B6RIP3       COMPARE       PSPOINTER LT VSNO.PROCS
1059 1285                ASSIGN        POINTER,PSPOINTER+1
1060 1286                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1061 1287                LEAVE         S(PSPOINTER)   GOTO(B6COMP4,B6RIP4,B6RGLOOP)
1062 1288   B6COMP4      COMPARE       PSPOINTER EQ PSSP.AFS
1063 1289                ADVANCE       GOTO(B6ENDRNG)
1064 1290   B6RIP4       COMPARE       PSPOINTER LT VSNO.PROCS
1065 1291                ASSIGN        POINTER,PSPOINTER+1
1066 1292                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1067 1293                LEAVE         S(PSPOINTER)   GOTO(B6COMP5,B6RIP5,B6RGLOOP)
1068 1294   B6COMP5      COMPARE       PSPOINTER EQ PSSP.AFS
1069 1295                ADVANCE       GOTO(B6ENDRNG)
1070 1296   B6RIP5       COMPARE       PSPOINTER LT VSNO.PROCS
1071 1297                ASSIGN        POINTER,PSPOINTER+1
1072 1298                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1073 1299                LEAVE         S(PSPOINTER)   GOTO(B6COMP6,B6RIP6,B6RGLOOP)
1074 1300   B6COMP6      COMPARE       PSPOINTER EQ PSSP.AFS
1075 1301                ADVANCE       GOTO(B6ENDRNG)
1076 1302   B6RIP6       COMPARE       PSPOINTER LT VSNO.PROCS
1077 1303                ASSIGN        POINTER,PSPOINTER+1
1078 1304                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1079 1305                LEAVE         S(PSPOINTER)   GOTO(B6COMP7,B6ENDCMP,B6RGLOOP)
1080 1306   B6COMP7      COMPARE       PSPOINTER EQ PSSP.AFS
```

```
1081 1307                ADVANCE        GOTO(B6ENDRNG)
1082 1308   B6RIP7       COMPARE        PIPOINTER LT VSNO.PROCS
1083 1309                ASSIGN         POINTER,PIPOINTER+1
1084 1310                ENTER          S(PIPOINTER)    TIME(VSRIP.TIME)
1085 1311                LEAVE          S(PSPOINTER)    GOTO(B6COMP8,B6ENDCMP,B6RGLOOP)
1086 1312   B6COMP8      COMPARE        PSPOINTER EQ PSSP.AFS
1087 1313                ADVANCE        GOTO(B6ENDRNG)
1088 1314   B6RIP8       COMPARE        PIPOINTER LT VSNO.PROCS
1089 1315                ASSIGN         POINTER,PIPOINTER+1
1090 1316                ENTER          S(PIPOINTER)    TIME(VSRIP.TIME)
1091 1317                LEAVE          S(PSPOINTER)    GOTO(B6COMP9,B6ENDCMP,B6RGLOOP)
1092 1318   B6COMP9      COMPARE        PSPOINTER EQ PSSP.AFS
1093 1319                ADVANCE        GOTO(B6ENDRNG)
1094 1320   B6RIP9       COMPARE        PSPOINTER LT VSNO.PROCS
1095 1321                ASSIGN         POINTER,PSPOINTER+1
1096 1322                ENTER          S(PSPOINTER)    TIME(VSRIP.TIME)
1097 1323                LEAVE          S(PSPOINTER)    GOTO(B6COMP10,B6ENDCMP,B6RGLOOP)
1098 1324   B6COMP10     COMPARE        PIPOINTER EQ PSSP.AFS
1099 1325                ADVANCE        GOTO(B6ENDRNG)
1100 1326                SAVEX          RING.CTR,XSRING.CTR+1
1101 1327                ADVANCE        GOTO(B6ENDCMP,B6RGLOOP)
1102 1328   B6ENDCMP     COMPARE        XSRING.CTR GE 7
1103 1329                TERMINATE,R
1104 1330   B6RGLOOP     ASSIGN         POINTER,1       GOTO(B6BEGIN)
1105 1331   B6ENDRNG     ADVANCE
1106                 * * * END PING CALL 6 * * * * * * * * * * * * * * * * * *
1107 1332                QUEUE          Q(PSPOINTER)
1108 1333                SEIZE          F(PSPOINTER)    TIME(VSPROCTIME)
1109 1334                RELEASE        F(PSPOINTER)
1110 1335   ASSN.5       ASSIGN         SP.AFA,PSSP.FA5
1111 1336                ASSIGN         SP.AFS,PSSP.FS5
1112 1337                ASSIGN         SP.ACI,PSSP.CI5
1113 1338                ASSIGN         SP.API,PSSP.PI5
1114 1339                ASSIGN         SP.NTRAN,PSSP.TRAN5
1115 1340                TABULATE       JOB.TRAK
1116 1341                ADVANCE        GOTO(SKIP.5,SPEC.5,WAIT.5)
1117 1342   SKIP.5       COMPARE        PSSP.AFA EQ 0
1118 1343                ADVANCE        GOTO(ASSN.6)
1119 1344   SPEC.5       COMPARE        PSPOINTER EQ PSSP.AFS TIME(FNSLOOK.UP)
1120                 * * * START PROCESSOR CALL 8 * * * * * * * * * * * * * * *
1121 1345                SEIZE          F(PSPOINTER)
1122 1346   B8READSP     ADVANCE        GOTO(B8CFA1,B8FA2)
1123 1347   B8CFA1       COMPARE        PSSP.AFA EQ 1
1124 1348                ASSIGN         TRANS.CT,0
1125 1349   B8LOOP1      HOLD           F(PSPOINTER)    TIME(VSPROCTIME)
1126 1350                INPUT
1127 1354                WRITDISK
1128 1377                ASSIGN         TRANS.CT,PSTRANS.CT+1
1129 1378                ADVANCE        GOTO(B8NEXT1,B8LAST1)
1130 1379   B8NEXT1      COMPARE        PSSP.NTRAN GT PSTRANS.CT
1131 1380                ADVANCE        GOTO(B8LOOP1)
1132 1381   B8LAST1      ADVANCE        GOTO(B8NEXTSP)
1133 1382   B8FA2        ADVANCE        GOTO(B8CFA2,B8FA3)
1134 1383   B8CFA2       COMPARE        PSSP.AFA EQ 2
1135 1384                ASSIGN         TRANS.CT,1
1136 1385   B8LOOP2      HOLD           F(PSPOINTER)    TIME(VSPROCTIME)
1137 1386                READDISK
```

```
1138 1397                ADVANCE       GOTO(B8CCI2,B8CPI2,B8CNT2)
1139 1398  B8CCI2        COMPARE       PSSP.ACI EQ 1
1140 1399                ADVANCE       TIME(VSCOM.TIME) GOTO(5[illegible]B8PRT2,B8CNT2)
1141 1400  B8CPI2        COMPARE       PSSP.API EQ 1
1142 1401  B8PRT2        ADVANCE
1143 1402                PRINT
1144 1403  B8CNT2        ASSIGN        TRANS.CT,PSTRANS.CT+1
1145 1404                ADVANCE       GOTO(B8NEXT2,B8LAST2)
1146 1405  B8NEXT2       COMPARE       PSSP.NTRAN GT PSTRANS.CT
1147 1406                ADVANCE       GOTO(B8LOOP2)
1148 1407  B8LAST2       ADVANCE       GOTO(B2NEXTSP)
1149 1408  B8FA3         ADVANCE       GOTO(B8CFA3,B8FA4)
1150 1409  B8CFA3        COMPARE       PSSP.AFA EQ 3
1151 1410                ASSIGN        TRANS.CT,0
1152 1411  B8LOOP3       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
1153 1412                WRITDISK
1154 1413                ADVANCE       GOTO(B8CCI3,B8CPI3,B8CNT3)
1155 1414  B8CCI3        COMPARE       PSSP.ACI EQ 1
1156 1415                ADVANCE       TIME(VSCOM.TIME) GOTO(5[illegible]B8PRNT3,B8CNT3)
1157 1416  B8CPI3        COMPARE       PSSP.API EQ 1
1158 1417  B8PRNT3       ADVANCE
1159 1418                PRINT
1160 1419  B8CNT3        ASSIGN        TRANS.CT,PSTRANS.CT+1
1161 1420                ADVANCE       GOTO(B8NEXT3,B8LAST3)
1162 1421  B8NEXT3       COMPARE       PSSP.NTRAN GT PSTRANS.CT
1163 1422                ADVANCE       GOTO(B8LOOP3)
1164 1423  B8LAST3       ADVANCE       GOTO(B8NEXTSP)
1165 1424  B8FA4         ADVANCE       GOTO(B8CFA4,B8NEXTSP)
1166 1425  B8CFA4        COMPARE       PSSP.AFA EQ 4
1167 1426                ASSIGN        TRANS.CT,0
1168 1427  B8LOOP4       HOLD          F(PSPOINTER)   TIME(VSPROCTIME)
1169 1428                ADVANCE       GOTO(B8CCI4,B8CPI4,B8CNT4)
1170 1429  B8CCI4        COMPARE       PSSP.ACI EQ 1
1171 1430                ADVANCE       TIME(VSCOM.TIME) GOTO(5[illegible]B8PRNT4,B8CNT4)
1172 1431  B8CPI4        COMPARE       PSSP.API EQ 1
1173 1432  B8PRNT4       ADVANCE
1174 1433                PRINT
1175 1434                ADVANCE       GOTO(B8CNT4)
1176 1435  B8CNT4        ASSIGN        TRANS.CT,PSTRANS.CT+1
1177 1436                ADVANCE       GOTO(B8NEXT4,B8LAST4)
1178 1437  B8NEXT4       COMPARE       PSSP.NTRAN GT PSTRANS.CT
1179 1438                ADVANCE       GOTO(B8LOOP4)
1180 1439  B8LAST4       ADVANCE       GOTO(B2NEXTSP)
1181 1440  B8NEXTSP      RELEASE       F(PSPOINTER)
1182       * * * END PROCESSOR CALL 8 * * * * * * * * * * * * * * * * *
1183 1441                ADVANCE       GOTO(ASSN.6)
1184 1442  WAIT.5        QUEUE         2(PSPOINTER+10)
1185       * * * BEGIN RING CALL 7 * * * * * * * * * * * * * * * * * *
1186 1443                SAVEX         RING.CTR,0
1187 1444  B7BEGIN       ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1188 1445  B7RIP0        LEAVE         S(PSPOINTER)   GOTO(B7COMP1,B7RIP1,B7RGLOOP)
1189 1446  B7COMP1       COMPARE       PSPOINTER EQ PSSP.AFS
1190 1447                ADVANCE       GOTO(B7ENDRNG)
1191 1448  B7RIP1        COMPARE       PSPOINTER LT VSNO.PROCS
1192 1449                ASSIGN        POINTER,PSPOINTER+1
1193 1450                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1194 1451                LEAVE         S(PSPOINTER)   GOTO(B7COMP2,B7RIP2,B7RGLOOP)
```

```
1195 1480  B7COMP2   COMPARE      P$POINTER EQ P$SP.AFS
1196 1481            ADVANCE      GOTO(B7ENDRNG)
1197 1482  B7RIP2    COMPARE      P$POINTER LT V$NO.PROCS
1198 1483            ASSIGN       POINTER,P$POINTER+1
1199 1484            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1200 1485            LEAVE        S(P$POINTER)   GOTO(B7COMP3,B7RIP3,B7RGLOOP)
1201 1486  B7COMP3   COMPARE      P$POINTER EQ P$SP.AFS
1202 1487            ADVANCE      GOTO(B7ENDRNG)
1203 1488  B7RIP3    COMPARE      P$POINTER LT V$NO.PROCS
1204 1489            ASSIGN       POINTER,P$POINTER+1
1205 1490            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1206 1491            LEAVE        S(P$POINTER)   GOTO(B7COMP4,B7RIP4,B7RGLOOP)
1207 1492  B7COMP4   COMPARE      P$POINTER EQ P$SP.AFS
1208 1493            ADVANCE      GOTO(B7ENDRNG)
1209 1494  B7RIP4    COMPARE      P$POINTER LT V$NO.PROCS
1210 1495            ASSIGN       POINTER,P$POINTER+1
1211 1496            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1212 1497            LEAVE        S(P$POINTER)   GOTO(B7COMP5,B7RIP5,B7RGLOOP)
1213 1498  B7COMP5   COMPARE      P$POINTER EQ P$SP.AFS
1214 1499            ADVANCE      GOTO(B7ENDRNG)
1215 1500  B7RIP5    COMPARE      P$POINTER LT V$NO.PROCS
1216 1501            ASSIGN       POINTER,P$POINTER+1
1217 1502            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1218 1503            LEAVE        S(P$POINTER)   GOTO(B7COMP6,B7RIP6,B7RGLOOP)
1219 1504  B7COMP6   COMPARE      P$POINTER EQ P$SP.AFS
1220 1505            ADVANCE      GOTO(B7ENDRNG)
1221 1506  B7RIP6    COMPARE      P$POINTER LT V$NO.PROCS
1222 1507            ASSIGN       POINTER,P$POINTER+1
1223 1508            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1224 1509            LEAVE        S(P$POINTER)   GOTO(B7COMP7,B7ENDCMP,B7RGLOOP)
1225 1510  B7COMP7   COMPARE      P$POINTER EQ P$SP.AFS
1226 1511            ADVANCE      GOTO(B7ENDRNG)
1227 1512  B7RIP7    COMPARE      P$POINTER LT V$NO.PROCS
1228 1513            ASSIGN       POINTER,P$POINTER+1
1229 1514            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1230 1515            LEAVE        S(P$POINTER)   GOTO(B7COMP8,B7ENDCMP,B7RGLOOP)
1231 1516  B7COMP8   COMPARE      P$POINTER EQ P$SP.AFS
1232 1517            ADVANCE      GOTO(B7ENDRNG)
1233 1518  B7RIP8    COMPARE      P$POINTER LT V$NO.PROCS
1234 1519            ASSIGN       POINTER,P$POINTER+1
1235 1520            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1236 1521            LEAVE        S(P$POINTER)   GOTO(B7COMP9,B7ENDCMP,B7RGLOOP)
1237 1522  B7COMP9   COMPARE      P$POINTER EQ P$SP.AFS
1238 1523            ADVANCE      GOTO(B7ENDRNG)
1239 1524  B7RIP9    COMPARE      P$POINTER LT V$NO.PROCS
1240 1525            ASSIGN       POINTER,P$POINTER+1
1241 1526            ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1242 1527            LEAVE        S(P$POINTER)   GOTO(B7COMP10,B7ENDCMP,B7RGLOOP)
1243 1528  B7COMP10  COMPARE      P$POINTER EQ P$SP.AFS
1244 1529            ADVANCE      GOTO(B7ENDRNG)
1245 1530            SAVEX        RING.CTR,X$RING.CTR+1
1246 1531            ADVANCE      GOTO(B7ENDCMP,B7RGLOOP)
1247 1532  B7ENDCMP  COMPARE      X$RING.CTR GE 7
1248 1533            TERMINATE,R
1249 1534  B7RGLOOP  ASSIGN       POINTER,1      GOTO(B7BEGIN)
1250 1535  B7ENDRNG  ADVANCE
1251       * * * END RING CALL 7 * * * * * * * * * * * * * * * * * *
```

```
1252 1536                QUEUE          3(PSPOINTER)
1253          * * * START PROCESSOR CALL 9 (REMOTE) * * * * * * * * *
1254 1537                SEIZE          F(P1POINTER)
1255 1538  B9READSP      ADVANCE        GOTO(B9FA0,B9FA1)
1256 1539  B9FA0         COMPARE        P1SP.AFA EQ 0
1257 1540                ADVANCE        GOTO(B9NEXTSP)
1258 1541  B9FA1         ADVANCE        GOTO(B9CFA1,B9FA2)
1259 1542  B9CFA1        COMPARE        P1SP.AFA EQ 1
1260 1543                ASSIGN         TRANS.CT,0
1261 1544  B9LOOP1       HOLD           F(P1POINTER)   TIME(V1PROCTIME)
1262 1545                INPUT
1263 1549                WRITDISK
1264 1572                ASSIGN         TRANS.CT,P1TRANS.CT+1
1265 1573                ADVANCE        GOTO(B9NEXT1,B9LAST1)
1266 1574  B9NEXT1       COMPARE        P1SP.NTRAN GT P1TRANS.CT
1267 1575                ADVANCE        GOTO(B9LOOP1)
1268 1576  B9LAST1       ADVANCE        GOTO(B9NEXTSP)
1269 1577  B9FA2         ADVANCE        GOTO(B9CFA2,B9FA3)
1270 1578  B9CFA2        COMPARE        P1SP.AFA EQ 2
1271 1579                ASSIGN         TRANS.CT,0
1272 1580  B9LOOP2       HOLD           F(P1POINTER)   TIME(V1PROCTIME)
1273 1581                READDISK
1274 1592                ADVANCE        GOTO(B9CCI2,B9CPI2,B9CNT2)
1275 1593  B9CCI2        COMPARE        P1SP.ACI EQ 1
1276 1594                ADVANCE        TIME(V1COM.TIME) GOTO(500:B9PRT2,B9CNT2)
1277 1595  B9CPI2        COMPARE        P1SP.API EQ 1
1278 1596  B9PRT2        ADVANCE
1279 1597                PRINT
1280 1600  B9CNT2        ASSIGN         TRANS.CT,P1TRANS.CT+1
1281 1601                ADVANCE        GOTO(B9NEXT2,B9LAST2)
1282 1602  B9NEXT2       COMPARE        P1SP.NTRAN GT P1TRANS.CT
1283 1603                ADVANCE        GOTO(B9LOOP2)
1284 1604  B9LAST2       ADVANCE        GOTO(B9NEXTSP)
1285 1605  B9FA3         ADVANCE        GOTO(B9CFA3,B9FA4)
1286 1606  B9CFA3        COMPARE        P1SP.AFA EQ 3
1287 1607                ASSIGN         TRANS.CT,0
1288 1608  B9LOOP3       HOLD           F(P1POINTER)   TIME(V1PROCTIME)
1289 1609                WRITDISK
1290 1632                ADVANCE        GOTO(B9CCI3,B9CPI3,B9CNT3)
1291 1633  B9CCI3        COMPARE        P1SP.ACI EQ 1
1292 1634                ADVANCE        TIME(V1COM.TIME) GOTO(500:B9PRNT3,B9CNT3)
1293 1635  B9CPI3        COMPARE        P1SP.API EQ 1
1294 1636  B9PRNT3       ADVANCE
1295 1637                PRINT
1296 1640  B9CNT3        ASSIGN         TRANS.CT,P1TRANS.CT+1
1297 1641                ADVANCE        GOTO(B9NEXT3,B9LAST3)
1298 1642  B9NEXT3       COMPARE        P1SP.NTRAN GT P1TRANS.CT
1299 1643                ADVANCE        GOTO(B9LOOP3)
1300 1644  B9LAST3       ADVANCE        GOTO(B9NEXTSP)
1301 1645  B9FA4         ADVANCE        GOTO(B9CFA4,B9NEXTSP)
1302 1646  B9CFA4        COMPARE        P1SP.AFA EQ 4
1303 1647                ASSIGN         TRANS.CT,0
1304 1648  B9LOOP4       HOLD           F(P1POINTER)   TIME(V1PROCTIME)
1305 1649                ADVANCE        GOTO(B9CCI4,B9CPI4,B9CNT4)
1306 1650  B9CCI4        COMPARE        P1SP.ACI EQ 1
1307 1651                ADVANCE        TIME(V1COM.TIME) GOTO(500:B9PRNT4,B9CNT4)
1308 1652  B9CPI4        COMPARE        P1SP.API EQ 1
```

```
1309 1653  B9PRNT4    ADVANCE
1310 1654              PRINT
1311 1657              ADVANCE       GOTO(B9CNT4)
1312 1658  B9CNT4     ASSIGN        TRANS.CT,PSTRANS.CT+1
1313 1659              ADVANCE       GOTO(B9NEXT4,B9LAST4)
1314 1660  B9NEXT4    COMPARE       PSSP.NTRAN GT PSTRANS.CT
1315 1661              ADVANCE       GOTO(B9LOOP4)
1316 1662  B9LAST4    ADVANCE       GOTO(B9NEXTSP)
1317 1663  B9NEXTSP   RELEASE       F(PSPOINTER)
1318       * * * END PROCESSOR CALL 9 * * * * * * * * * * * * * * *
1319 1664              ASSIGN        SP.AFS,PSSP.FS1
1320 1665              QUEUE         Q(PSPOINTER+10)
1321       * * * BEGIN RING CALL 8 * * * * * * * * * * * * * * * * *
1322 1666              SAVEX         RING.CTR,3
1323 1667  B8BEGIN    ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1324 1668  B8RIP0     LEAVE         S(PSPOINTER)   GOTO(B8ECOMP1,B8RIP1,B8RGLOOP)
1325 1669  B8COMP1    COMPARE       PSPOINTER EQ PSSP.AFS
1326 1670              ADVANCE       GOTO(B8ENDRNG)
1327 1671  B8RIP1     COMPARE       PSPOINTER LT VSNO.PROCS
1328 1672              ASSIGN        POINTER,PSPOI..TER+1
1329 1673              ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1330 1674              LEAVE         S(PSPOINTER)   GOTO(B8COMP2,B8RIP2,B8RGLOOP)
1331 1675  B8COMP2    COMPARE       PSPOINTER EQ PSSP.AFS
1332 1676              ADVANCE       GOTO(B8ENDRNG)
1333 1677  B8RIP2     COMPARE       PSPOINTER LT VSNO.PROCS
1334 1678              ASSIGN        POINTER,PSPOINTER+1
1335 1679              ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1336 1680              LEAVE         S(PSPOINTER)   GOTO(B8COMP3,B8RIP3,B8RGLOOP)
1337 1681  B8COMP3    COMPARE       PSPOINTER EQ PSSP.AFS
1338 1682              ADVANCE       GOTO(B8ENDRNG)
1339 1683  B8RIP3     COMPARE       PSPOINTER LT VSNO.PROCS
1340 1684              ASSIGN        POINTER,PSPOINTER+1
1341 1685              ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1342 1686              LEAVE         S(PSPOINTER)   GOTO(B8COMP4,B8RIP4,B8RGLOOP)
1343 1687  B8COMP4    COMPARE       PSPOINTER EQ PSSP.AFS
1344 1688              ADVANCE       GOTO(B8ENDRNG)
1345 1689  B8RIP4     COMPARE       PSPOINTER LT VSNO.PROCS
1346 1690              ASSIGN        POINTER,PSPOINTER+1
1347 1691              ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1348 1692              LEAVE         S(PSPOINTER)   GOTO(B8COMP5,B8RIP5,B8RGLOOP)
1349 1693  B8COMP5    COMPARE       PSPOINTER EQ PSSP.AFS
1350 1694              ADVANCE       GOTO(B8ENDRNG)
1351 1695  B8RIP5     COMPARE       PSPOINTER LT VSNO.PROCS
1352 1696              ASSIGN        POINTER,PSPOINTER+1
1353 1697              ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1354 1698              LEAVE         S(PSPOINTER)   GOTO(B8COMP6,B8RIP6,B8RGLOOP)
1355 1699  B8COMP6    COMPARE       PSPOINTER EQ PSSP.AFS
1356 1700              ADVANCE       GOTO(B8ENDRNG)
1357 1701  B8RIP6     COMPARE       PSPOINTER LT VSNO.PROCS
1358 1702              ASSIGN        POINTER,PSPOINTER+1
1359 1703              ENTER         S(PSPOINTER)   TIME(VSPIP.TIME)
1360 1704              LEAVE         S(PSPOINTER)   GOTO(B8COMP7,B8ENDCMP,B8RGLOOP)
1361 1705  B8COMP7    COMPARE       PSPOINTER EQ PSSP.AFS
1362 1706              ADVANCE       GOTO(B8ENDRNG)
1363 1707  B8RIP7     COMPARE       PSPOINTER LT VSNO.PROCS
1364 1708              ASSIGN        POINTER,PSPOINTER+1
1365 1709              ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
```

```
1366 1710                LEAVE          S(PSPOINTER)   GOTO(B8COMP8,B8ENDCMP,B8RGLOOP)
1367 1711   B8COMP8      COMPARE        PSPOINTER EQ PSSP.AFS
1368 1712                ADVANCE        GOTO(B8ENDRNG)
1369 1713   B8RIP8       COMPARE        PSPOINTER LT VSNO.PROCS
1370 1714                ASSIGN         PCINTER,PSPOINTER+1
1371 1715                ENTER          S(PSPOINTER)   TIME(VSRIP.TIME)
1372 1716                LEAVE          S(PSPOINTER)   GOTO(B8COMP9,B8ENDCMP,B8RGLOOP)
1373 1717   B8COMP9      COMPARE        PSPOINTER EQ PSSP.AFS
1374 1718                ADVANCE        GOTO(B8ENDRNG)
1375 1719   B8RIP9       COMPARE        PSPOINTER LT VSNO.PROCS
1376 1720                ASSIGN         POINTER,PSPOINTER+1
1377 1721                ENTER          S(PSPOINTER)   TIME(VSRIP.TIME)
1378 1722                LEAVE          S(PSPOINTER)   GOTO(B8COMP10,B8ENDCMP,B8RGLOOP)
1379 1723   B8COMP10     COMPARE        PSPOINTER EQ PSSP.AFS
1380 1724                ADVANCE        GOTO(B8ENDRNG)
1381 1725                SAVEX          RING.CTR,XSRING.CTR+1
1382 1726                ADVANCE        GOTO(B8ENDCMP,B8RGLOOP)
1383 1727   B8ENDCMP     COMPARE        XSRING.CTR GE 7
1384 1728                TERMINATE,R
1385 1729   B8RGLOOP     ASSIGN         POINTER,1      GOTO(B8BEGIN)
1386 1730   B8ENDRNG     ADVANCE
1387        * * * END RING CALL 8 * * * * * * * * * * * * * * * * * *
1388 1731                QUEUE          Q(PSPOINTER)
1389 1732                SEIZE          F(PSPOINTER)   TIME(VSPROCTIME)
1390 1733                RELEASE        F(PSPOINTER)
1391 1734   ASSN.6       ASSIGN         SP.AFA,PSSP.FA6
1392 1735                ASSIGN         SP.AFS,PSSP.FS6
1393 1736                ASSIGN         SP.ACI,PSSP.CI6
1394 1737                ASSIGN         SP.API,PSSP.PI6
1395 1738                ASSIGN         SP.ATRAN,PSSP TRAN6
1396 1739                TABULATE       JOB.TRAK
1397 1740                ADVANCE        GOTO(SKIP.6,SPEC.6,WAIT.6)
1398 1741   SKIP.6       COMPARE        PSSP.AFA EQ 0
1399 1742                ADVANCE        GOTO(END.JOB)
1400 1743   SPEC.6       COMPARE        PSPOINTER EQ PSSP.AFS TIME(FASLOOK.UP)
1401        * * * BEGIN PROCESSOR CALL 10 * * * * * * * * * * * * * *
1402 1744                SEIZE          F(PSPOINTER)
1403 1745   B10READS     ADVANCE        GOTO(B10CFA1,B10FA2)
1404 1746   B10CFA1      COMPARE        PSSP.AFA EQ 1
1405 1747                ASSIGN         TRANS.CT,0
1406 1748   B10LOOP1     HOLD           F(PSPOINTER)   TIME(VSPROCTIME)
1407 1749                INPUT
1408 1753                WRITDISK
1409 1776                ASSIGN         TRANS.CT,PSTRANS.CT+1
1410 1777                ADVANCE        GOTO(B10NEXT1,B10LAST1)
1411 1778   B10NEXT1     COMPARE        PSSP.NTRAN GT PSTRANS.CT
1412 1779                ADVANCE        GOTO(B10LOOP1)
1413 1780   B10LAST1     ADVANCE        GOTO(B10NXTSP)
1414 1781   B10FA2       ADVANCE        GOTO(B10CFA2,B10FA3)
1415 1782   B10CFA2      COMPARE        PSSP.AFA EQ 2
1416 1783                ASSIGN         TRANS.CT,0
1417 1784   B10LOOP2     HOLD           F(PSPOINTER)   TIME(VSPROCTIME)
1418 1785                READDISK
1419 1786                ADVANCE        GOTO(B10CCI2,B10CPI2,B10CNT2)
1420 1787   B10CCI2      COMPARE        PSSP.ACI EQ 1
1421 1788                ADVANCE        TIME(VSCOM.TIME) GOTO(SOO:B10CPRT2,B10CNT2)
1422 1789   B10CPI2      COMPARE        PSSP.API EQ 1
```

```
1423 1800  B10PRT2    ADVANCE
1424 1801             PRINT
1425 1804  B10CNT2    ASSIGN      TRANS.CT,PSTRANS.CT+1
1426 1805             ADVANCE     GOTO(B10NEXT2,B10LAST2)
1427 1806  B10NEXT2   COMPARE     PSSP.NTRAN GT PSTRANS.CT
1428 1807             ADVANCE     GOTO(B10LOOP2)
1429 1808  B10LAST2   ADVANCE     GOTO(B10NXTSP)
1430 1809  B10FA3     ADVANCE     GOTO(B10CFA3,B10FA4)
1431 1810  B10CFA3    COMPARE     PSSP.AFA EQ 3
1432 1811             ASSIGN      TRANS.CT,0
1433 1812  B10LOOP3   HOLD        F(PIPOINTER)   TIME(VSPROCTIME)
1434 1813             WRITDISK
1435 1836             ADVANCE     GOTO(B10CCI3,B10CPI3,B10CNT3)
1436 1837  B10CCI3    COMPARE     PSSP.ACI EQ 1
1437 1838             ADVANCE     TIME(VSCOM.TIME) GOTO(500:B10PRNT3,B10CNT3)
1438 1839  B10CPI3    COMPARE     PSSP.API EQ 1
1439 1840  B10PRNT3   ADVANCE
1440 1841             PRINT
1441 1844  B10CNT3    ASSIGN      TRANS.CT,PSTRANS.CT+1
1442 1845             ADVANCE     GOTO(B10NEXT3,B10LAST3)
1443 1846  B10NEXT3   COMPARE     PSSP.NTRAN GT PSTRANS.CT
1444 1847             ADVANCE     GOTO(B10LOOP3)
1445 1848  B10LAST3   ADVANCE     GOTO(B10NXTSP)
1446 1849  B10FA4     ADVANCE     GOTO(B10CFA4,B10NXTSP)
1447 1850  B10CFA4    COMPARE     PSSP.AFA EQ 4
1448 1851             ASSIGN      TRANS.CT,0
1449 1852  B10LOOP4   HOLD        F(PSPOINTER)   TIME(VSPROCTIME)
1450 1853             ADVANCE     GOTO(B10CCI4,B10CPI4,B10CNT4)
1451 1854  B10CCI4    COMPARE     PSSP.ACI EQ 1
1452 1855             ADVANCE     TIME(VSCOM.TIME) GOTO(500:B10PRNT4,B10CNT4)
1453 1856  B10CPI4    COMPARE     PSSP.API EQ 1
1454 1857  B10PRNT4   ADVANCE
1455 1858             PRINT
1456 1861             ADVANCE     GOTO(B10CNT4)
1457 1862  B10CNT4    ASSIGN      TRANS.CT,PSTRANS.CT+1
1458 1863             ADVANCE     GOTO(B10NEXT4,B10LAST4)
1459 1864  B10NEXT4   COMPARE     PSSP.NTRAN GT PSTRANS.CT
1460 1865             ADVANCE     GOTO(B10LOOP4)
1461 1866  B10LAST4   ADVANCE     GOTO(B10NXTSP)
1462 1867  B10NXTSP   RELEASE     F(PIPOINTER)
1463       * * * END PROCESSOR CALL 10 * * * * * * * * * * * * * * *
1464 1868             ADVANCE     GOTO(END.JOB)
1465 1869  WAIT.6     QUEUE       C(PSPOINTER+10)
1466       * * * BEGIN RING CALL 9 * * * * * * * * * * * * * * * * *
1467 1870             SAVEX       RING.CTR,3
1468 1871  B9BEGIN    ENTER       S(PIPOINTER)   TIME(VSRIP.TIME)
1469 1872  B9RIP0     LEAVE       S(PIPOINTER)   GOTO(B9COMP1,B9RIP1,B9RGLOOP)
1470 1873  B9COMP1    COMPARE     PIPOINTER EQ PSSP.AFS
1471 1874             ADVANCE     GOTO(B9ENDRNG)
1472 1875  B9RIP1     COMPARE     PIPOINTER LT VSNO.PROCS
1473 1876             ASSIGN      POINTER,PIPOINTER+1
1474 1877             ENTER       S(PIPOINTER)   TIME(VSRIP.TIME)
1475 1878             LEAVE       S(PIPOINTER)   GOTO(B9COMP2,B9RIP2,B9RGLOOP)
1476 1879  B9COMP2    COMPARE     PIPOINTER EQ PSSP.AFS
1477 1880             ADVANCE     GOTO(B9ENDRNG)
1478 1881  B9RIP2     COMPARE     PIPOINTER LT VSNO.PROCS
1479 1882             ASSIGN      POINTER,PIPOINTER+1
```

```
1880 1883                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1881 1884                LEAVE        S(P$POINTER)   GOTO(B9COMP3,B9RIP3,B9RGLOOP)
1882 1885   B9COMP3      COMPARE      P$POINTER EQ P$SP.AFS
1883 1886                ADVANCE      GOTO(B9ENDRNG)
1884 1887   B9RIP3       COMPARE      P$POINTER LT V$NO.PROCS
1885 1888                ASSIGN       POINTER,P$POINTER+1
1886 1889                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1887 1890                LEAVE        S(P$POINTER)   GOTO(B9COMP4,B9RIP4,B9RGLOOP)
1888 1891   B9COMP4      COMPARE      P$POINTER EQ P$SP.AFS
1889 1892                ADVANCE      GOTO(B9ENDRNG)
1890 1893   B9RIP4       COMPARE      P$POINTER LT V$NO.PROCS
1891 1894                ASSIGN       POINTER,P$POINTER+1
1892 1895                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1893 1896                LEAVE        S(P$POINTER)   GOTO(B9COMP5,B9RIP5,B9RGLOOP)
1894 1897   B9COMP5      COMPARE      P$POINTER EQ P$SP.AFS
1895 1898                ADVANCE      GOTO(B9ENDRNG)
1896 1899   B9RIP5       COMPARE      P$POINTER LT V$NO.PROCS
1897 1900                ASSIGN       POINTER,P$POINTER+1
1898 1901                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1899 1902                LEAVE        S(P$POINTER)   GOTO(B9COMP6,B9RIP6,B9RGLOOP)
1900 1903   B9COMP6      COMPARE      P$POINTER EQ P$SP.AFS
1901 1904                ADVANCE      GOTO(B9ENDRNG)
1902 1905   B9RIP6       COMPARE      P$POINTER LT V$NO.PROCS
1903 1906                ASSIGN       POINTER,P$POINTER+1
1904 1907                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1905 1908                LEAVE        S(P$POINTER)   GOTO(B9COMP7,B9ENDCMP,B9RGLOOP)
1906 1909   B9COMP7      COMPARE      P$POINTER EQ P$SP.AFS
1907 1910                ADVANCE      GOTO(B9ENDRNG)
1908 1911   B9RIP7       COMPARE      P$POINTER LT V$NO.PROCS
1909 1912                ASSIGN       POINTER,P$POINTER+1
1910 1913                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1911 1914                LEAVE        S(P$POINTER)   GOTO(B9COMP8,B9ENDCMP,B9RGLOOP)
1912 1915   B9COMP8      COMPARE      P$POINTER EQ P$SP.AFS
1913 1916                ADVANCE      GOTO(B9ENDRNG)
1914 1917   B9RIP8       COMPARE      P$POINTER LT V$NO.PROCS
1915 1918                ASSIGN       POINTER,P$POINTER+1
1916 1919                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1917 1920                LEAVE        S(P$POINTER)   GOTO(B9COMP9,B9ENDCMP,B9RGLOOP)
1918 1921   B9COMP9      COMPARE      P$POINTER EQ P$SP.AFS
1919 1922                ADVANCE      GOTO(B9ENDRNG)
1920 1923   B9RIP9       COMPARE      P$POINTER LT V$NO.PROCS
1921 1924                ASSIGN       POINTER,P$POINTER+1
1922 1925                ENTER        S(P$POINTER)   TIME(V$RIP.TIME)
1923 1926                LEAVE        S(P$POINTER)   GOTO(B9COMP10,B9ENDCMP,B9RGLOOP)
1924 1927   B9COMP10     COMPARE      P$POINTER EQ P$SP.AFS
1925 1928                ADVANCE      GOTO(B9ENDRNG)
1926 1929                SAVEX        RING.CTR,X$RING.CTR+1
1927 1930                ADVANCE      GOTO(B9ENDCMP,B9RGLOOP)
1928 1931   B9ENDCMP     COMPARE      X$RING.CTR GE 7
1929 1932                TERMINATE,R
1930 1933   B9RGLOOP     ASSIGN       POINTER,1      GOTO(B9BEGIN)
1931 1934   B9ENDRNG     ADVANCE
1932        * * * END RING CALL 9 * * * * * * * * * * * * * * * * * *
1933 1935                QUEUE        Q(P$POINTER)
1934        * * * START PROCESSOR CALL 11 (REMOTE) * * * * * * * *
1935 1936                SEIZE        F(P$POINTER)
1936 1937   B11READSP    ADVANCE      GOTO(B11FA0,B11FA1)
```

```
1537 1938  B11FA0     COMPARE     P$SP.AFA EQ 0
1538 1939              ADVANCE     GOTO(B11NXTSP)
1539 1940  B11FA1     ADVANCE     GOTO(B1CFA1,B11FA2)
1540 1941  B11CFA1    COMPARE     P$SP.AFA EQ 1
1541 1942              ASSIGN      TRANS.CT,0
1542 1943  B11LOOP1   HOLD        F(P$POINTER)   TIME(V$PROCTIME)
1543 1944              INPUT
1544 1948              WRITDISK
1545 1971              ASSIGN      TRANS.CT,P$TRANS.CT+1
1546 1972              ADVANCE     GOTO(B11NEXT1,B11LAST1)
1547 1973  B11NEXT1   COMPARE     P$SP.NTRAN GT P$TRANS.CT
1548 1974              ADVANCE     GOTO(B11LOOP1)
1549 1975  B11LAST1   ADVANCE     GOTO(B11NXTSP)
1550 1976  B11FA2     ADVANCE     GOTO(B11CFA2,B11FA3)
1551 1977  B11CFA2    COMPARE     P$SP.AFA EQ 2
1552 1978              ASSIGN      TRANS.CT,0
1553 1979  B11LOOP2   HOLD        F(P$POINTER)   TIME(V$PROCTIME)
1554 1980              READDISK
1555 1991              ADVANCE     GOTO(B11CCI2,B11CPI2,B11CNT2)
1556 1992  B11CCI2    COMPARE     P$SP.ACI EQ 1
1557 1993              ADVANCE     TIME(V$COM.TIME) GOTO(500:B11PRT2,B11CNT2)
1558 1994  B11CPI2    COMPARE     P$SP.API EQ 1
1559 1995  B11PRT2    ADVANCE
1560 1996              PRINT
1561 1999  B11CNT2    ASSIGN      TRANS.CT,P$TRANS.CT+1
1562 2000              ADVANCE     GOTO(B11NEXT2,B11LAST2)
1563 2001  B11NEXT2   COMPARE     P$SP.NTRAN GT P$TRANS.CT
1564 2002              ADVANCE     GOTO(B11LOOP2)
1565 2003  B11LAST2   ADVANCE     GOTO(B11NXTSP)
1566 2004  B11FA3     ADVANCE     GOTO(B11CFA3,B11FA4)
1567 2005  B11CFA3    COMPARE     P$SP.AFA EQ 3
1568 2006              ASSIGN      TRANS.CT,0
1569 2007  B11LOOP3   HOLD        F(P$POINTER)   TIME(V$PROCTIME)
1570 2008              WRITDISK
1571 2031              ADVANCE     GOTO(B11CCI3,B11CPI3,B11CNT3)
1572 2032  B11CCI3    COMPARE     P$SP.ACI EQ 1
1573 2033              ADVANCE     TIME(V$COM.TIME) GOTO(500:B11PRT3,B11CNT3)
1574 2034  B11CPI3    COMPARE     P$SP.API EQ 1
1575 2035  B11PRNT3   ADVANCE
1576 2036              PRINT
1577 2039  B11CNT3    ASSIGN      TRANS.CT,P$TRANS.CT+1
1578 2040              ADVANCE     GOTO(B11NEXT3,B11LAST3)
1579 2041  B11NEXT3   COMPARE     P$SP.NTRAN GT P$TRANS.CT
1580 2042              ADVANCE     GOTO(B11LOOP3)
1581 2043  B11LAST3   ADVANCE     GOTO(B11NXTSP)
1582 2044  B11FA4     ADVANCE     GOTO(B11CFA4,B11NXTSP)
1583 2045  B11CFA4    COMPARE     P$SP.AFA EQ 4
1584 2046              ASSIGN      TRANS.CT,0
1585 2047  B11LOOP4   HOLD        F(P$POINTER)   TIME(V$PROCTIME)
1586 2048              ADVANCE     GOTO(B11CCI4,B11CPI4,B11CNT4)
1587 2049  B11CCI4    COMPARE     P$SP.ACI EQ 1
1588 2050              ADVANCE     TIME(V$COM.TIME) GOTO(500:B11PRT4,B11CNT4)
1589 2051  B11CPI4    COMPARE     P$SP.API EQ 1
1590 2052  B11PRNT4   ADVANCE
1591 2053              PRINT
1592 2056              ADVANCE     GOTO(B11CNT4)
1593 2057  B11CNT4    ASSIGN      TRANS.CT,P$TRANS.CT+1
```

```
1594 2058                ADVANCE       GOTO(B11NEXT4,B11LAST4)
1595 2059  B11NEXT4       COMPARE       PSSP.NTRAN GT PSTRANS.CT
1596 2060                ADVANCE       GOTO(B11LOOP4)
1597 2061  B11LAST4       ADVANCE       GOTO(B11NXTSP)
1598 2062  B11NXTSP       RELEASE       F(PSPOINTER)
1599          * * * END PROCESSOR CALL 11 (REMOTE) * * * * * * * * *
1600 2063                ASSIGN        SP.AFS,PSSP.FS1
1601 2064                QUEUE         2(PSPOINTER+10)
1602          * * * BEGIN RING CALL 10 * * * * * * * * * * * * * * *
1603 2065                SAVEX         RING.CTR,0
1604 2066  B10BEGIN       ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1605 2067  B10RIP0        LEAVE         S(PSPOINTER)   GOTO(B10COMP1,B10RIP1,B10RLOOP)
1606 2068  B10COMP1       COMPARE       PSPOINTER EQ PSSP.AFS
1607 2069                ADVANCE       GOTO(B10EDRNG)
1608 2070  B10RIP1        COMPARE       PSPOINTER LT VSNO.PROCS
1609 2071                ASSIGN        POINTER,PSPOINTER+1
1610 2072                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1611 2073                LEAVE         S(PSPOINTER)   GOTO(B10COMP2,B10RIP2,B10RLOOP)
1612 2074  B10COMP2       COMPARE       PSPOINTER EQ PSSP.AFS
1613 2075                ADVANCE       GOTO(B10EDRNG)
1614 2076  B10RIP2        COMPARE       PSPOINTER LT VSNO.PROCS
1615 2077                ASSIGN        POINTER,PSPOINTER+1
1616 2078                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1617 2079                LEAVE         S(PSPOINTER)   GOTO(B10COMP3,B10RIP3,B10RLOOP)
1618 2080  B10COMP3       COMPARE       PSPOINTER EQ PSSP.AFS
1619 2081                ADVANCE       GOTO(B10EDRNG)
1620 2082  B10RIP3        COMPARE       PSPOINTER LT VSNO.PROCS
1621 2083                ASSIGN        POINTER,PSPOINTER+1
1622 2084                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1623 2085                LEAVE         S(PSPOINTER)   GOTO(B10COMP4,B10RIP4,B10RLOOP)
1624 2086  B10COMP4       COMPARE       PSPOINTER EQ PSSP.AFS
1625 2087                ADVANCE       GOTO(B10EDRNG)
1626 2088  B10RIP4        COMPARE       PSPOINTER LT VSNO.PROCS
1627 2089                ASSIGN        POINTER,PSPOINTER+1
1628 2090                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1629 2091                LEAVE         S(PSPOINTER)   GOTO(B10COMP5,B10RIP5,B10RLOOP)
1630 2092  B10COMP5       COMPARE       PSPOINTER EQ PSSP.AFS
1631 2093                ADVANCE       GOTO(B10EDRNG)
1632 2094  B10RIP5        COMPARE       PSPOINTER LT VSNO.PROCS
1633 2095                ASSIGN        POINTER,PSPOINTER+1
1634 2096                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1635 2097                LEAVE         S(PSPOINTER)   GOTO(B10COMP6,B10RIP6,B10RLOOP)
1636 2098  B10COMP6       COMPARE       PSPOINTER EQ PSSP.AFS
1637 2099                ADVANCE       GOTO(B10EDRNG)
1638 2100  B10RIP6        COMPARE       PSPOINTER LT VSNO.PROCS
1639 2101                ASSIGN        POINTER,PSPOINTER+1
1640 2102                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1641 2103                LEAVE         S(PSPOINTER)   GOTO(B10COMP7,B10EDCMP,B10RLOOP)
1642 2104  B10COMP7       COMPARE       PSPOINTER EQ PSSP.AFS
1643 2105                ADVANCE       GOTO(B10EDRNG)
1644 2106  B10RIP7        COMPARE       PSPOINTER LT VSNO.PROCS
1645 2107                ASSIGN        POINTER,PSPOINTER+1
1646 2108                ENTER         S(PSPOINTER)   TIME(VSRIP.TIME)
1647 2109                LEAVE         S(PSPOINTER)   GOTO(B10COMP8,B10EDCMP,B10RLOOP)
1648 2110  B10COMP8       COMPARE       PSPOINTER EQ PSSP.AFS
1649 2111                ADVANCE       GOTO(B10EDRNG)
1650 2112  B10RIP8        COMPARE       PSPOINTER LT VSNO.PROCS
```

```
1651 2113                ASSIGN          POINTER,PSPOINTER+1
1652 2114                ENTER           S(PSPOINTER)    TIME(VSRIP.TIME)
1653 2115                LEAVE           S(PSPOINTER)    GOTO(B1OCOMP9,B1LEDCMP,B1ORLOOP)
1654 2116  B1OCOMP9      COMPARE         PSPOINTER EQ PSSP.AFS
1655 2117                ADVANCE         GOTO(B1OEDRNG)
1656 2118  B1ORIP9       COMPARE         PSPOINTER LT VSNO.PROCS
1657 2119                ASSIGN          POINTER,PSPOINTER+1
1658 2120                ENTER           S(PSPOINTER)    TIME(VSRIP.TIME)
1659 2121                LEAVE           S(PSPOINTER)    GOTO(B1OCOMP10,B1OEDCMP,B1ORLOOP)
1660 2122  B1OCOMP10     COMPARE         PSPOINTER EQ PSSP.AFS
1661 2123                ADVANCE         GOTO(B1OEDRNG)
1662 2124                SAVEX           RING.CTR,XSRING.CTR+1
1663 2125                ADVANCE         GOTO(B1OEDCMP,B1ORLOOP)
1664 2126  B1OEDCMP      COMPARE         XSRING.CTR GE 7
1665 2127                TERMINATE,R
1666 2128  B1ORLOOP      ASSIGN          POINTER,1       GOTO(B1OBEGIN)
1667 2129  B1OEDRNG      ADVANCE
1668       * * * END RING CALL 10 * * * * * * * * * * * * * * * * * *
1669 2130                QUEUE           Q(PSPOINTER)
1670 2131                SEIZE           F(PSPOINTER)    TIME(VSRIP.TIME)
1671 2132                RELEASE         F(PSPOINTER)
1672 2133  END.JOB       ADVANCE
1673 2134                TABULATE TURN.ARD
1674 2135                TERMINATE,R
1675                      START          14


QUEUES:
CONSOLEQ


PARAMETERS:
JOB.NAME        SP.FA1          SP.FS1          SP.CI1          SP.PI1
SP.TRAN1        SP.FA2          SP.FS2          SP.CI2          SP.PI2
SP.TRAN2        SP.FA3          SP.FS3          SP.CI3          SP.PI3
SP.TRAN3        SP.FA4          SP.FS4          SP.CI4          SP.PI4
SP.TRAN4        SP.FA5          SP.FS5          SP.CI5          SP.PI5
SP.TRAN5        SP.FA6          SP.FS6          SP.CI6          SP.PI6
SP.TRAN6        SP.AFA          SP.AFS          SP.ACI          SP.API
SP.NTRAN        POINTER         TRANS.CT


TABLES:
JOB.TRAK        TURN.ARD        JOB.ARRV


SAVEXS:
RING.CTR


FUNCTIONS:
ROT.TIME        SEEK.TIM        INP.TIME        PRT.LINE        LOOK.UP


VARIABLES:
TA.MIN          JA.HRS          PROCTIME        CHAN.TIM        XFER.TIM
```

COM.TIME NO.PROCS PAK.SIZE RIP.TIME

BLOCKS:

3 ASSN.1	12 B1READSP	13 B1FA0
15 B1FA1	16 B1CFA1	18 B1LOOP1
48 B1NEXT1	50 B1LAST1	51 B1FA2
52 B1CFA2	54 B1LOOP2	67 B1CCI2
69 B1CPI2	70 B1PRT2	74 B1CNT2
76 B1NEXT2	78 B1LAST2	79 B1FA3
80 B1CFA3	82 B1LOOP3	107 B1CCI3
109 B1CPI3	110 B1PRNT3	114 B1CNT3
116 B1NEXT3	118 B1LAST3	119 B1FA4
120 B1CFA4	122 B1LOOP4	124 B1CCI4
126 B1CPI4	127 B1PRNT4	132 B1CNT4
134 B1NEXT4	136 B1LAST4	137 B1NEXTSP
139 ASSN.2	145 SKIP.2	147 SPEC.2
149 B2READSP	150 B2CFA1	152 B2LOOP1
182 B2NEXT1	184 B2LAST1	185 B2FA2
186 B2CFA2	188 B2LOOP2	201 B2CCI2
203 B2CPI2	204 B2PRT2	208 B2CNT2
210 B2NEXT2	212 B2LAST2	213 B2FA3
214 B2CFA3	216 B2LOOP3	241 B2CCI3
243 B2CPI3	244 B2PRNT3	248 B2CNT3
250 B2NEXT3	252 B2LAST3	253 B2FA4
254 B2CFA4	256 B2LOOP4	258 B2CCI4
260 B2CPI4	261 B2PRNT4	266 B2CNT4
268 B2NEXT4	270 B2LAST4	271 B2NEXTSP
273 WAIT.2	275 B1BEGIN	276 B1RIP0
277 B1COMP1	279 B1RIP1	283 B1COMP2
285 B1RIP2	289 B1COMP3	291 B1RIP3
295 B1COMP4	297 B1RIP4	301 B1COMP5
303 B1RIP5	307 B1COMP6	309 B1RIP6
313 B1COMP7	315 B1RIP7	319 B1COMP8
321 B1RIP8	325 B1COMP9	327 B1RIP9
331 B1COMP10	335 B1ENDCMP	337 B1RGLOOP
338 B1ENDRNG	341 B3READSP	342 B3FA0
344 B3FA1	345 B3CFA1	347 B3LOOP1
377 B3NEXT1	379 B3LAST1	380 B3FA2
381 B3CFA2	383 B3LOOP2	396 B3CCI2
398 B3CPI2	399 B3PRT2	403 B3CNT2
405 B3NEXT2	407 B3LAST2	408 B3FA3
409 B3CFA3	411 B3LOOP3	436 B3CCI3
438 B3CPI3	439 B3PRNT3	443 B3CNT3
445 B3NEXT3	447 B3LAST3	448 B3FA4
449 B3CFA4	451 B3LOOP4	453 B3CCI4
455 B3CPI4	456 B3PRNT4	461 B3CNT4
463 B3NEXT4	465 B3LAST4	466 B3NEXTSP
470 B2BEGIN	471 B2RIP0	472 B2COMP1
474 B2RIP1	478 B2COMP2	480 B2RIP2
484 B2COMP3	486 B2RIP3	490 B2COMP4
492 B2RIP4	496 B2COMP5	498 B2RIP5
502 B2COMP6	504 B2RIP6	508 B2COMP7
510 B2RIP7	514 B2COMP8	515 B2RIP8
520 B2COMP9	522 B2RIP9	526 B2COMP10
530 B2ENDCMP	532 B2RGLOOP	533 B2ENDRNG
537 ASSN.3	544 SKIP.3	545 SPEC.3

```
543 B4READSP
561 B4NEXT1
585 B4CFA2
602 B4CPI2
629 B4NEXT2
615 B4CFA3
642 B4CPI3
649 B4NEXT3
653 B4CFA4
659 B4CPI4
667 B4NEXT4
672 WAIT.3
676 B3COMP1
684 B3RIP2
694 B3COMP4
702 B3RIP5
712 B3COMP7
720 B3RIP8
730 B3COMP10
737 B3ENDRNG
743 B5FA1
776 B5NEXT1
780 B5CFA2
797 B5CPI2
804 B5NEXT2
809 B5CFA3
837 B5CPI3
844 B5NEXT3
848 B5CFA4
854 B5CPI4
862 B5NEXT4
869 B4BEGIN
873 B4RIP1
883 B4COMP3
891 B4RIP4
901 B4COMP6
909 B4RIP7
919 B4COMP9
929 B4ENDCMP
935 ASSN.4
947 B6READSP
987 B6NEXT1
964 B6CFA2
1001 B6CPI2
1003 B6NEXT2
1012 B6CFA3
1041 B6CPI3
1043 B6NEXT3
1052 B6CFA4
1058 B6CPI4
1066 B6NEXT4
1071 WAIT.4
1075 B5COMP1
1083 B5RIP2
1093 B5COMP4
1101 B5RIP5
1111 B5COMP7
549 B4CFA1
583 B4LAST1
597 B4LOOP2
603 B4PRT2
611 B4LAST2
615 B4LOOP3
643 B4PRNT3
651 B4LAST3
655 B4LOOP4
660 B4PRNT4
667 B4LAST4
674 B3BEGIN
673 B3RIP1
688 B3COMP3
696 B3RIP4
706 B3COMP6
714 B3RIP7
724 B3COMP9
734 B3ENDCMP
740 B5READSP
744 B5CFA1
778 B5LAST1
732 B5LOOP2
798 B5PRT2
606 B5LAST2
613 B5LOOP3
638 B5PRNT3
646 B5LAST3
650 B5LOOP4
655 B5PRNT4
664 B5LAST4
670 B4RIP0
677 B4COMP2
685 B4RIP3
695 B4COMP5
903 B4RIP6
913 B4COMP8
921 B4RIP9
931 B4RGLOOP
943 SKIP.4
948 B6CFA1
932 B6LAST1
936 B6LOOP2
1002 B6PRT2
1010 B6LAST2
1014 B6LOOP3
1042 B6PRNT3
1050 B6LAST3
1054 B6LOOP4
1059 B6PRNT4
1066 B6LAST4
1073 B5BEGIN
1077 B5RIP1
1087 B5COMP3
1095 B5RIP4
1105 B5COMP6
1113 B5RIP7
551 B4LOOP1
564 B4FA2
600 B4CCI2
607 B4CNT2
612 B4FA3
640 B4CCI3
647 B4CNT3
652 B4FA4
657 B4CCI4
665 B4CNT4
670 B4NEXTSP
675 B3RIPC
682 B3COMP2
693 B3RIP3
700 B3COMP5
708 B3RIP6
718 B3COMP8
726 B3RIP9
736 B3RGLOOP
741 B5FA1
746 B5LOOP1
779 B5FA2
795 B5CCI2
802 B5CNT2
807 B5FA3
835 B5CCI3
842 B5CNT3
847 B5FA4
852 B5CCI4
860 B5CNT4
865 B5NEXTSP
871 B4COMP1
879 B4RIP2
889 B4COMP4
897 B4RIP5
907 B4COMP7
915 B4RIP8
925 B4COMP10
932 B4ENDRNG
945 SPEC.4
953 B6LOOP1
983 B6FA2
999 B6CCI2
1006 B6CNT2
1011 B6FA3
1039 B6CCI3
1046 B6CNT3
1051 B6FA4
1056 B6CCI4
1064 B6CNT4
1069 B6NEXTSP
1074 B5RIPC
1081 B5COMP2
1089 B5RIP3
1099 B5COMP5
1107 B5RIP6
1117 B5COMP8
```

```
1119 B5RIP6
1129 B5COMP10
1136 B5ENDRNG
1142 B7FA1
1175 B7NEXT1
1179 B7CFA2
1196 B7CPI2
1203 B7NEXT2
1207 B7CFA3
1236 B7CPI3
1243 B7NEXT3
1247 B7CFA4
1253 B7CPI4
1261 B7NEXT4
1268 B6BEGIN
1272 B6RIP1
1282 B6COMP3
1290 B6RIP4
1300 B6COMP6
1309 B6RIP7
1319 B6COMP9
1329 B6ENDCMP
1335 ASSN.5
1345 B5READSP
1379 B6NEXT1
1383 B6CFA2
1400 B6CPI2
1407 B6NEXT2
1411 B6CFA3
1442 B6CPI3
1447 B6NEXT3
1451 B6CFA4
1457 B6CPI4
1465 B6NEXT4
1479 WAIT.5
1474 B7COMP1
1482 B7RIP2
1492 B7COMP4
1500 B7RIP5
1510 B7COMP7
1518 B7RIP6
1528 B7COMP10
1535 B7ENDRNG
1541 B9FA1
1574 B9NEXT1
1578 B9CFA2
1595 B9CPI2
1602 B9NEXT2
1606 B9CFA3
1635 B9CPI3
1642 B9NEXT3
1646 B9CFA4
1652 B9CPI4
1660 B9NEXT4
1667 B8BLGIN
1671 B8RIP1
1681 B8COMP3
1123 B5COMP9
1133 B5ENDCMP
1137 B7READSP
1143 B7CFA1
1177 B7LAST1
1181 B7LOOP2
1197 B7PRT2
1235 B7LAST2
1239 B7LOOP3
1237 B7PRNT3
1245 B7LAST3
1249 B7LOOP4
1254 B7PRNT4
1263 B7LAST4
1269 B6RIP0
1276 B6COMP2
1284 B6RIP3
1294 B6COMP5
1302 B6RIP6
1312 B6COMP8
1320 B6RIP9
1330 B6RGLOOP
1342 SKIP.5
1347 B8CFA1
1381 B8LAST1
1385 B8LOOP2
1401 B8PRT2
1409 B8LAST2
1413 B8LOOP3
1441 B8PRNT3
1449 B8LAST3
1453 B8LOOP4
1456 B8PRNT4
1467 B8LAST4
1472 B7BEGIN
1476 B7RIP1
1480 B7COMP3
1494 B7RIP4
1504 B7COMP6
1512 B7RIP7
1522 B7COMP9
1532 B7ENDCMP
1538 B9READSP
1542 B9CFA1
1576 B9LAST1
1580 B9LOOP2
1596 B9PRT2
1604 B9LAST2
1608 B9LOOP3
1636 B9PRNT3
1644 B9LAST3
1648 B9LOOP4
1653 B9PRNT4
1662 B9LAST4
1668 B8RIP0
1675 B8COMP2
1683 B8RIP3
1125 B5RIP9
1135 B5RGLOOP
1140 B7FA3
1145 B7LOOP1
1175 B7FA2
1194 B7CCI2
1201 B7CNT2
1206 B7FA3
1234 B7CCI3
1241 B7CNT3
1246 B7FA4
1251 B7CCI4
1259 B7CNT4
1264 B7NEXTSP
1273 B6COMP1
1279 B6RIP2
1288 B6COMP4
1296 B6RIP5
1306 B6COMP7
1314 B6RIP8
1324 B6COMP10
1331 B6ENDRNG
1344 SPEC.5
1349 B8LOOP1
1382 B8FA2
1395 B8CCI2
1405 B8CNT2
1410 B8FA3
1438 B8CCI3
1445 B8CNT3
1452 B8FA4
1455 B8CCI4
1463 B8CNT4
1469 B8NEXTSP
1473 B7RIP0
1490 B7COMP2
1488 B7RIP3
1498 B7COMP5
1506 B7RIP6
1516 B7COMP8
1524 B7RIP9
1534 B7RGLOOP
1539 B9FA2
1544 B9LOOP1
1577 B9FA2
1593 B9CCI2
1600 B9CNT2
1605 B9FA3
1633 B9CCI3
1640 B9CNT3
1645 B9FA4
1650 B9CCI4
1658 B9CNT4
1663 B9NEXTSP
1669 B8COMP1
1677 B8RIP2
1687 B8COMP4
```

```
1689 B8RIP4          1693 B8COMP5          1695 B8RIP5
1699 B8COMP6          1701 B8RIP6          1705 B8COMP7
1707 B8RIP7          1711 B8COMP8          1713 B8RIP8
1717 B8COMP9          1719 B8RIP9          1723 B8COMP10
1727 B8ENDCMP         1729 B8RGLOOP        1733 B8ENDRNG
1734 ASSN.6           1741 SKIP.6          1743 SPEC.6
1745 B11READS         1746 B10CFA1         1749 B10LOOP1
1773 B10NEXT1         1780 B10LAST1        1781 B10FA2
1782 B10CFA2          1784 B10LOOP2        1797 B10CCI2
1799 B10CPI2          1800 B10PRT2         1804 B10CNT2
1806 B10NEXT2         1808 B10LAST2        1809 B10FA3
1811 B10CFA3          1812 B10LOOP3        1837 B10CCI3
1839 B10CPI3          1840 B10PRNT3        1844 B10CNT3
1846 B10NEXT3         1846 B10LAST3        1849 B10FA4
1850 B10CFA4          1852 B10LOOP4        1854 B10CCI4
1856 B10CPI4          1857 B10PRNT4        1862 B10CNT4
1864 B10NEXT4         1866 B10LAST4        1867 B10NXTSP
1869 WAIT.5           1871 B9BEGIN         1872 B9RIP1
1873 B9COMP1          1875 B9RIP1          1879 B9COMP2
1881 B9RIP2           1885 B9COMP3         1887 B9RIP3
1891 B9COMP4          1893 B9RIP4          1897 B9COMP5
1899 B9RIP5           1903 B9COMP6         1905 B9RIP6
1909 B9COMP7          1911 B9RIP7          1915 B9COMP8
1917 B9RIP8           1921 B9COMP9         1923 B9RIP9
1927 B9COMP10         1931 B9ENDCMP        1933 B9RGLOOP
1934 B9ENDRNG         1937 B11READSP       1938 B11FA6
1940 B11FA1           1941 B11CFA1         1943 B11LOOP1
1973 B11NEXT1         1975 B11LAST1        1976 B11FA2
1977 B11CFA2          1979 B11LOOP2        1992 B11CCI2
1994 B11CPI2          1995 B11PRT2         1999 B11CNT2
2001 B11NEXT2         2003 B11LAST2        2004 B11FA3
2005 B11CFA3          2007 B11LOOP3        2032 B11CCI3
2034 B11CPI3          2035 B11PRNT3        2039 B11CNT3
2041 B11NEXT3         2043 B11LAST3        2044 B11FA4
2045 B11CFA4          2047 B11LOOP4        2049 B11CCI4
2051 B11CPI4          2052 B11PRNT4        2057 B11CNT4
2059 B11NEXT4         2061 B11LAST4        2062 B11NXTSP
2066 B10BEGIN         2067 B10RIP1         2068 B10COMP1
2070 B10RIP1          2074 B10COMP2        2076 B10RIP2
2080 B10COMP3         2082 B10RIP3         2086 B10COMP4
2088 B10RIP4          2092 B10COMP5        2094 B10RIP5
2098 B10COMP6         2100 B10RIP6         2104 B10COMP7
2106 B10RIP7          2110 B10COMP8        2112 B10RIP8
2116 B10COMP9         2118 B10RIP9         2122 B10COMP10
2126 B10ENDCMP        2126 B10RLOOP        2129 B10ENDRNG
2133 END.JOB
```

NUMBER OF TRANSACTIONS ALLOWED: 411

APPENDIX 2

SAMPLE MODEL OUTPUT

RELATIVE CLOCK TIME 11402963

ABSOLUTE CLOCK TIME 11402963

TERMINATION COUNT 14

BLOCK #	CURR TRAN	TOTAL TRAN	BLOCK #	CURR TRAN	TOTAL TRAN	BLOCK #	CURR TRAN	TOTAL TRAN	BLOCK #	CURR TRAN	TOTAL TRAN	BLOCK #	CURR TRAN	TOTAL TRAN
1	0	14	2	0	14	3	0	14	4	0	14	5	0	14
6	0	14	7	0	14	8	0	14	9	0	14	10	0	14
11	0	14	12	0	14	13	0	7	14	0	0	15	0	14
16	0	11	17	0	11	18	0	386	19	0	386	20	0	366
21	0	386	22	0	386	23	0	386	24	0	386	25	0	386
26	0	386	27	0	386	28	0	386	29	0	386	30	0	386
31	0	386	32	0	386	33	0	386	34	0	386	35	0	386
36	0	386	37	0	386	38	0	386	39	0	386	40	0	386
41	0	386	42	0	386	43	0	386	44	0	386	45	0	386
46	0	386	47	0	386	48	0	375	49	0	375	50	0	11
51	0	3	52	0	3	53	0	3	54	0	330	55	0	330
56	0	330	57	0	330	58	0	330	59	0	330	60	0	330
61	0	330	62	0	330	63	0	330	64	0	330	65	0	330
66	0	330	67	0	330	68	0	320	69	0	0	70	0	168
71	0	163	72	0	168	73	0	168	74	0	330	75	0	330
76	0	327	77	0	327	78	0	3	79	0	0	80	0	0
81	0	0	82	0	0	83	0	0	84	0	0	85	0	0
86	0	0	87	0	0	88	0	0	89	0	0	90	0	0
91	0	0	92	0	0	93	0	0	94	0	0	95	0	0
96	0	0	97	0	0	98	0	0	99	0	0	100	0	0
101	0	0	102	0	0	103	0	0	104	0	0	105	0	0
106	0	0	107	0	0	108	0	0	109	0	0	110	0	0
111	0	0	112	0	0	113	0	0	114	0	0	115	0	0
116	0	0	117	0	0	118	0	0	119	0	0	120	0	0
121	0	0	122	0	0	123	0	0	124	0	0	125	0	0
126	0	0	127	0	0	128	0	0	129	0	0	130	0	0
131	0	0	132	0	0	133	0	0	134	0	0	135	0	0
136	0	0	137	0	14	138	0	14	139	0	14	140	0	14
141	0	14	142	0	14	143	0	14	144	0	14	145	0	0
146	0	3	147	0	11	148	0	11	149	0	11	150	0	1
151	0	1	152	0	100	153	0	100	154	0	100	155	0	100
156	0	100	157	0	100	158	0	100	159	0	100	160	0	100
161	0	100	162	0	100	163	0	100	164	0	100	165	0	100
166	0	100	167	0	100	168	0	100	169	0	100	170	0	100
171	0	100	172	0	100	173	0	100	174	0	100	175	0	100
176	0	100	177	0	100	178	0	100	179	0	100	180	0	100
181	0	100	182	0	99	183	0	99	184	0	1	185	0	10
186	0	5	187	0	5	188	0	208	189	0	208	190	0	208
191	0	208	192	0	208	193	0	208	194	0	208	195	0	208
196	0	208	197	0	208	198	0	208	199	0	208	200	0	208
201	0	45	202	0	45	203	0	5	204	0	14	205	0	14
206	0	14	207	0	14	208	0	208	209	0	208	210	0	208
211	0	208	212	0	5	213	0	5	214	0	5	215	0	5
216	0	130	217	0	130	218	0	130	219	0	130	220	0	130
221	0	130	222	0	130	223	0	130	224	0	130	225	0	130
226	0	130	227	0	130	228	0	130	229	0	130	230	0	130
231	0	130	232	0	130	233	0	130	234	0	130	235	0	130
236	0	130	237	0	130	238	0	130	239	0	130	240	0	130
241	0	0	242	0	0	243	0	70	244	0	70	245	0	70
246	0	70	247	0	70	248	0	130	249	0	130	250	0	125

1961			1962	0	0	1963	0	0	1964	0	0	1965	0	0
1966			1967	0	0	1968	0	0	1969	0	0	1970	0	0
1971			1972	0	0	1973	0	0	1974	0	0	1975	0	0
1976		2	1977	0	0	1978	0	0	1979	0	0	1980	0	0
1981		0	1982	0	0	1983	0	0	1984	0	0	1985	0	0
1986		0	1987	0	0	1988	0	0	1989	0	0	1990	0	0
1991		0	1992	0	0	1993	0	0	1994	0	0	1995	0	0
1995		0	1997	0	0	1998	0	0	1999	0	0	2000	0	0
2001		0	2002	0	0	2003	0	0	2004	0	2	2005	0	2
2006		2	2007	0	2	2008	0	2	2009	0	2	2010	0	2
2011		2	2012	0	2	2013	0	2	2014	0	2	2015	0	2
2016		2	2017	0	2	2018	0	2	2019	0	2	2020	0	2
2021		2	2022	0	2	2023	0	2	2024	0	2	2025	0	2
2026		2	2027	0	2	2028	0	2	2029	0	2	2030	0	2
2031		2	2032	0	0	2033	0	2	2034	0	2	2035	0	2
2036		2	2037	0	2	2038	0	2	2039	0	2	2040	0	2
2041		2	2042	0	0	2043	0	2	2044	0	0	2045	0	0
2046		0	2047	0	0	2048	0	0	2049	0	0	2050	0	0
2051		0	2052	0	0	2053	0	0	2054	0	0	2055	0	0
2056		0	2057	0	0	2058	0	0	2059	0	0	2060	0	0
2061		0	2062	0	2	2063	0	2	2064	0	2	2065	0	2
2066		4	2067	0	4	2068	0	4	2069	0	1	2070	0	3
2071		3	2072	0	3	2073	0	3	2074	0	1	2075	0	1
2076		0	2077	0	0	2078	0	0	2079	0	0	2080	0	0
2081		0	2082	0	0	2083	0	0	2084	0	0	2085	0	0
2086		0	2087	0	0	2088	0	0	2089	0	0	2090	0	0
2091		0	2092	0	0	2093	0	0	2094	0	0	2095	0	0
2096		0	2097	0	0	2098	0	0	2099	0	0	2100	0	0
2101		0	2102	0	0	2103	0	0	2104	0	0	2105	0	0
2106		0	2107	0	0	2108	0	0	2109	0	0	2110	0	0
2111		0	2112	0	0	2113	0	0	2114	0	0	2115	0	0
2116		0	2117	0	0	2118	0	0	2119	0	0	2120	0	0
2121		0	2122	0	0	2123	0	0	2124	0	0	2125	0	0
2126		0	2127	0	0	2128	0	0	2129	0	2	2130	0	2
2131		2	2132	0	2	2133	0	14	2134	0	14	2135	0	14

SAVEX NAME	SAVEX VALUE	SAVEX NAME	SAVEX VALUE	SAVEX NAME	SAVEX VALUE	SAVEX NAME	SAVEX VALUE
FACILITY	3						

FACILITY NAME	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRANS
F (1)	.4644	329	16736.27
F (2)	.9102	541	19184.69
F (3)	.1421	660	2383.35
F (4)	.5696	741	8764.99
F (5)	.1305	467	3187.41
F (6)	1.0000	956	11927.75
F (7)	.0000	0	.00
F (8)	.0000	0	.00
F (9)	.0000	0	.00
F (10)	.0000	0	.00
F (11)	.0002	640	4.00
F (12)	.0004	1068	4.00
F (13)	.0005	1336	4.00
F (14)	.0005	1332	4.00
F (15)	.0003	958	4.00

F (16)	.0006	1600	4.00
F (17)	.0000	0	.00
F (18)	.0000	0	.00
F (19)	.0000	0	.00
F (20)	.0000	0	.00
F (21)	.0015	480	36.64
F (22)	.0030	869	36.98
F (23)	.0036	1056	36.47
F (24)	.0036	1039	39.38
F (25)	.0032	831	44.09
F (26)	.0043	1280	36.57
F (27)	.0000	0	.00
F (28)	.0000	0	.00
F (29)	.0000	0	.00
F (30)	.0000	0	.00
F (31)	.4824	160	34376.45
F (32)	.9063	312	33123.43
F (33)	.1325	40	37766.02
F (34)	.5587	180	35393.06
F (35)	.1216	40	34657.10
F (36)	.9897	340	33193.00
F (37)	.0000	0	.00
F (38)	.0000	0	.00
F (39)	.0000	0	.00
F (40)	.0000	0	.00
F (41)	.0000	0	.00
F (42)	.0001	3	346.00
F (43)	.0050	165	347.06
F (44)	.0060	203	346.97
F (45)	.0050	164	347.05
F (46)	.0046	150	346.95
F (47)	.0000	0	.00
F (48)	.0000	0	.00
F (49)	.0000	0	.00
F (50)	.0000	0	.00

STORAGE NAME	MAXIMUM CONTENTS	AVERAGE CONTENTS	MAXIMUM CAPACITY	AVERAGE CAPACITY	AVERAGE UTILIZATION	TOTAL ENTRIES	TOTAL TRANS	AVERAGE ENT/TRANS	AVERAGE TIME/ENT	CURRENT CONTENTS
S (1)	1	.00	1	1.00	.0000	20	20	1.00	1.00	0
S (2)	1	.00	1	1.00	.0000	19	19	1.00	1.00	0
S (3)	1	.00	1	1.00	.0000	18	18	1.00	1.00	0
S (4)	1	.00	1	1.00	.0000	20	20	1.00	1.00	0
S (5)	1	.00	1	1.00	.0000	21	21	1.00	1.00	0
S (6)	1	.00	1	1.00	.0000	14	14	1.00	1.00	0
S (7)	0	.00	1	1.00	.0000	0	0	.00	.00	0
S (8)	0	.00	1	1.00	.0000	0	0	.00	.00	0
S (9)	0	.00	1	1.00	.0000	0	0	.00	.00	0
S (10)	0	.00	1	1.00	.0000	0	0	.00	.00	0

GUIDE NAME	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	ZEROS PERCENT	AV. TIME/ENT (ALL)	AV. TIME/ENT (NON ZERO)	CURRENT CONTENTS	TABLE NAME
Q (1)	1	.00	6	6	100.00	.00	.00	0	
Q (2)	2	.83	5	1	20.00	*********	*********	0	
Q (3)	1	.10	4	3	75.00	293673.50	*********	0	
Q (4)	2	.64	6	3	50.00	*********	*********	0	
Q (5)	1	.00	7	7	100.00	.00	.00	0	
Q (6)	0	.00	0	0	.00	.00	.00	0	

5 (7)	0	.00	0	0	.00	.00	.00	0
6 (8)	0	.00	0	0	.00	.00	.00	0
6 (9)	0	.00	0	0	.00	.00	.00	0
8 (10)	0	.00	0	0	.00	.00	.00	0
9 (11)	1	.00	5	5	100.00	.00	.00	0
8 (12)	1	.00	5	5	100.00	.00	.00	0
8 (13)	1	.00	4	4	100.00	.00	.00	0
9 (14)	1	.00	6	6	100.00	.00	.00	0
6 (15)	1	.00	7	7	100.00	.00	.00	0
8 (16)	0	.00	0	0	.00	.00	.00	0
6 (17)	0	.00	0	0	.00	.00	.00	0
8 (18)	0	.00	0	0	.00	.00	.00	0
8 (19)	0	.00	0	0	.00	.00	.00	0
8 (20)	0	.00	0	0	.00	.00	.00	0
9 (21)	1	.00	640	640	100.00	.00	.00	0
6 (22)	1	.00	1048	1048	100.00	.00	.00	0
8 (23)	1	.00	1336	1336	100.00	.00	.00	0
9 (24)	1	.00	1352	1352	100.00	.00	.00	0
9 (25)	1	.00	908	908	100.00	.00	.00	0
8 (26)	1	.00	1600	1600	100.00	.00	.00	0
8 (27)	0	.00	0	0	.00	.00	.00	0
8 (28)	0	.00	0	0	.00	.00	.00	0
9 (29)	0	.00	0	0	.00	.00	.00	0
8 (30)	0	.00	0	0	.00	.00	.00	0
8 (31)	0	.00	0	0	.00	.00	.00	0
8 (32)	1	.00	3	3	100.00	.00	.00	0
8 (33)	1	.00	165	165	100.00	.00	.00	0
8 (34)	1	.00	203	203	100.00	.00	.00	0
8 (35)	1	.00	164	164	100.00	.00	.00	0
8 (36)	1	.00	150	150	100.00	.00	.00	0
8 (37)	0	.00	0	0	.00	.00	.00	0
8 (38)	0	.00	0	0	.00	.00	.00	0
8 (39)	0	.00	0	0	.00	.00	.00	0
8 (40)	0	.00	0	0	.00	.00	.00	0
OVERFLOW	1	.00	536	536	100.00	.00	.00	0

TABLE NAME	NON-WEIGHTED NO. OF ENTRIES	NON-WEIGHTED SUM OF ARGUMENTS	NON-WEIGHTED MEAN ARGUMENT	NON-WEIGHTED STD. DEV.	WEIGHTED NO. OF ENTRIES	WEIGHTED SUM OF ARGUMENTS	WEIGHTED MEAN ARGUMENT	WEIGHTED STD. DEV.
JOB.TIME	34	131.0000	1.560	1.322	84	131.0000	1.560	1.322
TURNARD	14	1614.9494	115.354	57.202	14	1614.9494	115.354	57.202
JOB.ARRV	14	.0000	.000	.000	14	.0000	.000	.000

TABLE NAME: JOB.TIME

UPPER LIMIT	OBSERVED FREQUENCY	PERCENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
1.000	22	33.33	33.33	66.67	.000	-1.160
2.000	13	19.68	48.81	51.19	.641	-.423
3.000	11	16.67	65.48	34.52	1.282	.333
4.000	20	30.95	96.43	3.57	1.924	1.090
5.000	1	3.57	100.00	.00	2.565	1.847

REMAINING FREQUENCIES ARE ALL ZERO

TABLE NAME: TURNARD

UPPER LIMIT	OBSERVED FREQUENCY	PERCENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN

172.0000	0	.00	64.29	35.71	1.491	.990
173.0000	3	21.43	85.71	14.29	1.500	1.008
174.0000	0	.00	85.71	14.29	1.508	1.025
175.0000	0	.00	85.71	14.29	1.517	1.043
176.0000	0	.00	85.71	14.29	1.526	1.060
177.0000	0	.00	85.71	14.29	1.534	1.078
178.0000	0	.00	85.71	14.29	1.543	1.095
179.0000	0	.00	85.71	14.29	1.552	1.113
180.0000	0	.00	85.71	14.29	1.560	1.130
181.0000	0	.00	85.71	14.29	1.569	1.148
182.0000	0	.00	85.71	14.29	1.578	1.165
183.0000	0	.00	85.71	14.29	1.586	1.183
184.0000	0	.00	85.71	14.29	1.595	1.200
185.0000	0	.00	85.71	14.29	1.604	1.218
186.0000	0	.00	85.71	14.29	1.612	1.235
187.0000	0	.00	85.71	14.29	1.621	1.253
188.0000	0	.00	85.71	14.29	1.630	1.270
189.0000	1	7.14	92.86	7.14	1.638	1.287
190.0000	0	.00	92.86	7.14	1.647	1.305
191.0000	1	7.14	100.00	.00	1.656	1.322

REMAINING FREQUENCIES ARE ALL ZERO

TABLE NAME: JOB.ARRV

UPPER LIMIT	OBSERVED FREQUENCY	PERCENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
1.0000	14	100.00	100.00	.00	******	893047.922

REMAINING FREQUENCIES ARE ALL ZERO

RANDOM GENERATOR	RANDOM MULTIPLIER	RANDOM INCREMENT	RANDOM SEED
1	1220703125	0	99606606569
2	314159265	2713681829	33419926553
3	4713681829	3141992653	27138251829
4	1.00449E+11	7261.67585	46617933758
5	1.72E+10	7261.67585	45613771690
6	3.51757E+13	7261.67585	1607732436
7	4.5E+12	35981229	28697666525
8	1.7E+11	4.92E+14	29181741765
9	2.43E+12	6591	22430762251
10	621444377	92111326	621444377

APPENDIX 3

JOBTAPE WRITE PROGRAM LISTING

```
$UOA*GPSS.GPSS
 GPSS 4.0    -08/05-11:38-(000)

    1       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    2       *                                                               *
    3       *                          TEST 15 - 6 PROCS                    *
    4       *                                                               *
    5       * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    6       * THIS SEGMENT WRITES A GPSS TRANSACTION ON JOBTAPE WHICH REPRE- *
    7       * SENTS A 6 PROCESSOR WEEKLY JOB STREAM.  THE NUMBER OF          *
    8       * TRANSACTIONS TO BE PROCESSED IS RECORDED IN 'SP.TRAN-'.        *
    9                 JOB
   10                 JOB.IVAL      VARIABLE       1
   11     1           GENERATE      0,1
   12     2           ASSIGN        JOB.NAME,2
   13     3           ASSIGN        SP.FA1,1
   14     4           ASSIGN        SP.FS1,4
   15     5           ASSIGN        SP.CI1,0
   16     6           ASSIGN        SP.PI1,1
   17     7           ASSIGN        SP.TRAN1,20
   18     8           ASSIGN        SP.FA2,3
   19     9           ASSIGN        SP.FS2,4
   20    10           ASSIGN        SP.CI2,0
   21    11           ASSIGN        SP.PI2,0
   22    12           ASSIGN        SP.TRAN2,20
   23    13           ASSIGN        SP.FA3,3
   24    14           ASSIGN        SP.FS3,2
   25    15           ASSIGN        SP.CI3,0
   26    16           ASSIGN        SP.PI3,0
   27    17           ASSIGN        SP.TRAN3,20
   28    18           ASSIGN        SP.FA4,4
   29    19           ASSIGN        SP.FS4,4
   30    20           ASSIGN        SP.CI4,0
   31    21           ASSIGN        SP.PI4,1
   32    22           ASSIGN        SP.TRAN4,60
   33    23           ASSIGN        SP.FA5,0
   34    24           ASSIGN        SP.FS5,1
   35    25           ASSIGN        SP.CI5,0
   36    26           ASSIGN        SP.PI5,0
   37    27           ASSIGN        SP.TRAN5,0
   38    28           ASSIGN        SP.FA6,0
   39    29           ASSIGN        SP.FS6,1
   40    30           ASSIGN        SP.CI6,0
   41    31           ASSIGN        SP.PI6,0
   42    32           ASSIGN        SP.TRAN6,0
   43    33           ADVANCE       TIME(V$JOB.IVAL)
   44    34           WRITE,B
   45    35           ASSIGN        JOB.NAME,3
   46    36           ASSIGN        SP.FA1,1
   47    37           ASSIGN        SP.FS1,4
   48    38           ASSIGN        SP.CI1,0
   49    39           ASSIGN        SP.PI1,1
   50    40           ASSIGN        SP.TRAN1,50
   51    41           ASSIGN        SP.FA2,3
   52    42           ASSIGN        SP.FS2,4
   53    43           ASSIGN        SP.CI2,0
   54    44           ASSIGN        SP.PI2,0
```

```
55   45          ASSIGN      SP.TRAN2,50
56   46          ASSIGN      SP.FA3,2
57   47          ASSIGN      SP.FS3,4
58   48          ASSIGN      SP.CI3,1
59   49          ASSIGN      SP.PI3,1
60   50          ASSIGN      SP.TRAN3,50
61   51          ASSIGN      SP.FA4,6
62   52          ASSIGN      SP.FS4,2
63   53          ASSIGN      SP.CI4,0
64   54          ASSIGN      SP.PI4,1
65   55          ASSIGN      SP.TRAN4,0
66   56          ASSIGN      SP.FA5,0
67   57          ASSIGN      SP.FS5,0
68   58          ASSIGN      SP.CI5,0
69   59          ASSIGN      SP.PI5,0
70   60          ASSIGN      SP.TRAN5,0
71   61          ASSIGN      SP.FA6,0
72   62          ASSIGN      SP.FS6,0
73   63          ASSIGN      SP.CI6,0
74   64          ASSIGN      SP.PI6,0
75   65          ASSIGN      SP.TRAN6,0
76   66          ADVANCE     TIME(V$JOB.IVAL)
77   67          WRITE,8
78   68          ASSIGN      JOB.NAME,4
79   69          ASSIGN      SP.FA1,1
80   70          ASSIGN      SP.FS1,4
81   71          ASSIGN      SP.CI1,0
82   72          ASSIGN      SP.PI1,0
83   73          ASSIGN      SP.TRAN1,20
84   74          ASSIGN      SP.FA2,3
85   75          ASSIGN      SP.FS2,4
86   76          ASSIGN      SP.CI2,0
87   77          ASSIGN      SP.PI2,0
88   78          ASSIGN      SP.TRAN2,20
89   79          ASSIGN      SP.FA3,3
90   80          ASSIGN      SP.FS3,4
91   81          ASSIGN      SP.CI3,0
92   82          ASSIGN      SP.PI3,0
93   83          ASSIGN      SP.TRAN3,20
94   84          ASSIGN      SP.FA4,3
95   85          ASSIGN      SP.FS4,1
96   86          ASSIGN      SP.CI4,0
97   87          ASSIGN      SP.PI4,1
98   88          ASSIGN      SP.TRAN4,1
99   89          ASSIGN      SP.FA5,0
100  90          ASSIGN      SP.FS5,0
101  91          ASSIGN      SP.CI5,0
102  92          ASSIGN      SP.PI5,0
103  93          ASSIGN      SP.TRAN5,0
104  94          ASSIGN      SP.FA6,0
105  95          ASSIGN      SP.FS6,0
106  96          ASSIGN      SP.CI6,0
107  97          ASSIGN      SP.PI6,0
108  98          ASSIGN      SP.TRAN6,0
109  99          ADVANCE     TIME(V$JOB.IVAL)
110  100         WRITE,8
111  101         ASSIGN      JOB.NAME,5
```

```
112  102          ASSIGN      SP.FA1,1
113  103          ASSIGN      SP.FS1,5
114  104          ASSIGN      SP.CI1,0
115  105          ASSIGN      SP.PI1,0
116  106          ASSIGN      SP.TRAN1,20
117  107          ASSIGN      SP.FA2,3
118  108          ASSIGN      SP.FS2,5
119  109          ASSIGN      SP.CI2,0
120  110          ASSIGN      SP.PI2,0
121  111          ASSIGN      SP.TRAN2,20
122  112          ASSIGN      SP.FA3,3
123  113          ASSIGN      SP.FS3,1
124  114          ASSIGN      SP.CI3,0
125  115          ASSIGN      SP.PI3,1
126  116          ASSIGN      SP.TRAN3,1
127  117          ASSIGN      SP.FA4,0
128  118          ASSIGN      SP.FS4,0
129  119          ASSIGN      SP.CI4,0
130  120          ASSIGN      SP.PI4,0
131  121          ASSIGN      SP.TRAN4,0
132  122          ASSIGN      SP.FA5,0
133  123          ASSIGN      SP.FS5,0
134  124          ASSIGN      SP.CI5,0
135  125          ASSIGN      SP.PI5,0
136  126          ASSIGN      SP.TRAN5,0
137  127          ASSIGN      SP.FA6,0
138  128          ASSIGN      SP.FS6,0
139  129          ASSIGN      SP.CI6,0
140  130          ASSIGN      SP.PI6,0
141  131          ASSIGN      SP.TRAN6,0
142  132          ADVANCE     TIME(V.JOB.IVAL)
143  133          WRITE,8
144  134          ASSIGN      JOB.NAME,6
145  135          ASSIGN      SP.FA1,2
146  136          ASSIGN      SP.FS1,5
147  137          ASSIGN      SP.CI1,1
148  138          ASSIGN      SP.PI1,0
149  139          ASSIGN      SP.TRAN1,300
150  140          ASSIGN      SP.FA2,2
151  141          ASSIGN      SP.FS2,4
152  142          ASSIGN      SP.CI2,1
153  143          ASSIGN      SP.PI2,1
154  144          ASSIGN      SP.TRAN2,30
155  145          ASSIGN      SP.FA3,3
156  146          ASSIGN      SP.FS3,4
157  147          ASSIGN      SP.CI3,0
158  148          ASSIGN      SP.PI3,0
159  149          ASSIGN      SP.TRAN3,30
160  150          ASSIGN      SP.FA4,3
161  151          ASSIGN      SP.FS4,5
162  152          ASSIGN      SP.CI4,0
163  153          ASSIGN      SP.PI4,0
164  154          ASSIGN      SP.TRAN4,30
165  155          ASSIGN      SP.FA5,3
166  156          ASSIGN      SP.FS5,1
167  157          ASSIGN      SP.CI5,0
168  158          ASSIGN      SP.PI5,1
```

```
169  159          ASSIGN      SP.TRA,5,1
170  160          ASSIGN      SP.FA6,0
171  161          ASSIGN      SP.FS6,0
172  162          ASSIGN      SP.CI6,0
173  163          ASSIGN      SP.PI6,0
174  164          ASSIGN      SP.TRA,6,0
175  165          ADVANCE     TIME(V,JOB.IVAL)
176  166          WRITE,8
177  167          ASSIGN      JOB.NAME,7
178  168          ASSIGN      SP.FA1,1
179  169          ASSIGN      SP.FS1,1
180  170          ASSIGN      SP.CI1,0
181  171          ASSIGN      SP.PI0,1
182  172          ASSIGN      SP.TRA,1,60
183  173          ASSIGN      SP.FA2,3
184  174          ASSIGN      SP.FS2,4
185  175          ASSIGN      SP.CI2,0
186  176          ASSIGN      SP.PI2,1
187  177          ASSIGN      SP.TRA,2,60
188  178          ASSIGN      SP.FA3,3
189  179          ASSIGN      SP.FS3,1
190  180          ASSIGN      SP.CI3,0
191  181          ASSIGN      SP.PI3,0
192  182          ASSIGN      SP.TRA,3,60
193  183          ASSIGN      SP.FA4,3
194  184          ASSIGN      SP.FS4,3
195  185          ASSIGN      SP.CI4,0
196  186          ASSIGN      SP.PI4,0
197  187          ASSIGN      SP.TRA,4,60
198  188          ASSIGN      SP.FA5,3
199  189          ASSIGN      SP.FS5,3
200  190          ASSIGN      SP.CI5,0
201  191          ASSIGN      SP.PI5,0
202  192          ASSIGN      SP.TRA,5,60
203  193          ASSIGN      SP.FA6,3
204  194          ASSIGN      SP.FS6,1
205  195          ASSIGN      SP.CI6,0
206  196          ASSIGN      SP.PI6,1
207  197          ASSIGN      SP.TRA,6,1
208  198          ADVANCE     TIME(V,JOB.IVAL)
209  199          WRITE,8
210  200          ASSIGN      JOB.NAME,8
211  201          ASSIGN      SP.FA1,1
212  202          ASSIGN      SP.FS1,1
213  203          ASSIGN      SP.CI1,0
214  204          ASSIGN      SP.PI1,0
215  205          ASSIGN      SP.TRA,1,20
216  206          ASSIGN      SP.FA2,3
217  207          ASSIGN      SP.FS2,1
218  208          ASSIGN      SP.CI2,0
219  209          ASSIGN      SP.PI2,0
220  210          ASSIGN      SP.TRA,2,20
221  211          ASSIGN      SP.FA3,3
222  212          ASSIGN      SP.FS3,3
223  213          ASSIGN      SP.CI3,0
224  214          ASSIGN      SP.PI3,0
225  215          ASSIGN      SP.TRA,3,20
```

```
225  216          ASSIGN      SP.FA4,3
227  217          ASSIGN      SP.FS4,1
228  218          ASSIGN      SP.CI4,0
229  219          ASSIGN      SP.PI4,1
230  220          ASSIGN      SP.TRANS4,1
231  221          ASSIGN      SP.FA5,0
232  222          ASSIGN      SP.FS5,0
233  223          ASSIGN      SP.CI5,0
234  224          ASSIGN      SP.PI5,0
235  225          ASSIGN      SP.TRANS5,0
236  226          ASSIGN      SP.FA6,0
237  227          ASSIGN      SP.FS6,0
238  228          ASSIGN      SP.CI6,0
239  229          ASSIGN      SP.PI6,0
240  230          ASSIGN      SP.TRANS6,0
241  231          ADVANCE     TIME(V$JOB.IVAL)
242  232          WRITE,8
243  233          ASSIGN      JOB.NAME,9
244  234          ASSIGN      SP.FA1,1
245  235          ASSIGN      SP.FS1,2
246  236          ASSIGN      SP.CI1,0
247  237          ASSIGN      SP.PI1,0
248  238          ASSIGN      SP.TRANS1,3
249  239          ASSIGN      SP.FA2,1
250  240          ASSIGN      SP.FS2,2
251  241          ASSIGN      SP.CI2,0
252  242          ASSIGN      SP.PI2,0
253  243          ASSIGN      SP.TRANS2,100
254  244          ASSIGN      SP.FA3,1
255  245          ASSIGN      SP.FS3,2
256  246          ASSIGN      SP.CI3,0
257  247          ASSIGN      SP.PI3,0
258  248          ASSIGN      SP.TRANS3,50
259  249          ASSIGN      SP.FA4,2
260  250          ASSIGN      SP.FS4,2
261  251          ASSIGN      SP.CI4,0
262  252          ASSIGN      SP.PI4,0
263  253          ASSIGN      SP.TRANS4,153
264  254          ASSIGN      SP.FA5,3
265  255          ASSIGN      SP.FS5,2
266  256          ASSIGN      SP.CI5,0
267  257          ASSIGN      SP.PI5,0
268  258          ASSIGN      SP.TRANS5,3
269  259          ASSIGN      SP.FA6,3
270  260          ASSIGN      SP.FS6,1
271  261          ASSIGN      SP.CI6,0
272  262          ASSIGN      SP.PI6,1
273  263          ASSIGN      SP.TRANS6,1
274  264          ADVANCE     TIME(V$JOB.IVAL)
275  265          WRITE,8
276  266          ASSIGN      JOB.NAME,10
277  267          ASSIGN      SP.FA1,1
278  268          ASSIGN      SP.FS1,2
279  269          ASSIGN      SP.CI1,0
280  270          ASSIGN      SP.PI1,0
281  271          ASSIGN      SP.TRANS1,3
282  272          ASSIGN      SP.FA2,2
```

```
283  273          ASSIGN      SP.FS2,2
284  274          ASSIGN      SP.CI2,0
285  275          ASSIGN      SP.PI2,0
286  276          ASSIGN      SP.TRAN2,3
287  277          ASSIGN      SP.FA3,3
288  278          ASSIGN      SP.FS3,4
289  279          ASSIGN      SP.CI3,0
290  280          ASSIGN      SP.PI3,0
291  281          ASSIGN      SP.TRAN3,3
292  282          ASSIGN      SP.FA4,3
293  283          ASSIGN      SP.FS4,1
294  284          ASSIGN      SP.CI4,0
295  285          ASSIGN      SP.PI4,1
296  286          ASSIGN      SP.TRAN4,1
297  287          ASSIGN      SP.FA5,0
298  288          ASSIGN      SP.FS5,0
299  289          ASSIGN      SP.CI5,0
300  290          ASSIGN      SP.PI5,0
301  291          ASSIGN      SP.TRAN5,0
302  292          ASSIGN      SP.FA6,0
303  293          ASSIGN      SP.FS6,0
304  294          ASSIGN      SP.CI6,0
305  295          ASSIGN      SP.PI6,0
306  296          ASSIGN      SP.TRAN6,0
307  297          ADVANCE     TIME(V$JOB.IVAL)
308  298          WRITE,8
309  299          ASSIGN      JOB.NAME,11
310  300          ASSIGN      SP.FA1,1
311  301          ASSIGN      SP.FS1,3
312  302          ASSIGN      SP.CI1,0
313  303          ASSIGN      SP.PI1,0
314  304          ASSIGN      SP.TRAN1,20
315  305          ASSIGN      SP.FA2,2
316  306          ASSIGN      SP.FS2,3
317  307          ASSIGN      SP.CI2,0
318  308          ASSIGN      SP.PI2,0
319  309          ASSIGN      SP.TRAN2,20
320  310          ASSIGN      SP.FA3,2
321  311          ASSIGN      SP.FS3,3
322  312          ASSIGN      SP.CI3,0
323  313          ASSIGN      SP.PI3,0
324  314          ASSIGN      SP.TRAN3,20
325  315          ASSIGN      SP.FA4,2
326  316          ASSIGN      SP.FS4,3
327  317          ASSIGN      SP.CI4,0
328  318          ASSIGN      SP.PI4,1
329  319          ASSIGN      SP.TRAN4,20
330  320          ASSIGN      SP.FA5,3
331  321          ASSIGN      SP.FS5,3
332  322          ASSIGN      SP.CI5,0
333  323          ASSIGN      SP.PI5,1
334  324          ASSIGN      SP.TRAN5,20
335  325          ASSIGN      SP.FA6,3
336  326          ASSIGN      SP.FS6,3
337  327          ASSIGN      SP.CI6,0
338  328          ASSIGN      SP.PI6,1
339  329          ASSIGN      SP.TRAN6,100
```

```
340  330          ADVANCE      TIME(V$JOB.IVAL)
341  331          WRITE,3
342  332          ASSIGN       JOB.NAME,12
343  333          ASSIGN       SP.FA1,2
344  334          ASSIGN       SP.FS1,3
345  335          ASSIGN       SP.CI1,1
346  336          ASSIGN       SP.PI1,1
347  337          ASSIGN       SP.TRAN1,20
348  338          ASSIGN       SP.FA2,2
349  339          ASSIGN       SP.FS2,3
350  340          ASSIGN       SP.CI2,1
351  341          ASSIGN       SP.PI2,1
352  342          ASSIGN       SP.TRAN2,25
353  343          ASSIGN       SP.FA3,0
354  344          ASSIGN       SP.FS3,0
355  345          ASSIGN       SP.CI3,0
356  346          ASSIGN       SP.PI3,0
357  347          ASSIGN       SP.TRAN3,0
358  348          ASSIGN       SP.FA4,0
359  349          ASSIGN       SP.FS4,0
360  350          ASSIGN       SP.CI4,0
361  351          ASSIGN       SP.PI4,0
362  352          ASSIGN       SP.TRAN4,0
363  353          ASSIGN       SP.FA5,0
364  354          ASSIGN       SP.FS5,0
365  355          ASSIGN       SP.CI5,0
366  356          ASSIGN       SP.PI5,0
367  357          ASSIGN       SP.TRAN5,0
368  358          ASSIGN       SP.FA6,0
369  359          ASSIGN       SP.FS6,0
370  360          ASSIGN       SP.CI6,0
371  361          ASSIGN       SP.PI6,0
372  362          ASSIGN       SP.TRAN6,0
373  363          ADVANCE      TIME(V$JOB.IVAL)
374  364          WRITE,8
375  365          ASSIGN       JOB.NAME,13
376  366          ASSIGN       SP.FA1,1
377  367          ASSIGN       SP.FS1,6
378  368          ASSIGN       SP.CI1,0
379  369          ASSIGN       SP.PI1,0
380  370          ASSIGN       SP.TRAN1,20
381  371          ASSIGN       SP.FA2,2
382  372          ASSIGN       SP.FS2,6
383  373          ASSIGN       SP.CI2,0
384  374          ASSIGN       SP.PI2,0
385  375          ASSIGN       SP.TRAN2,10
386  376          ASSIGN       SP.FA3,0
387  377          ASSIGN       SP.FS3,6
388  378          ASSIGN       SP.CI3,0
389  379          ASSIGN       SP.PI3,0
390  380          ASSIGN       SP.TRAN3,10
391  381          ASSIGN       SP.FA4,0
392  382          ASSIGN       SP.FS4,0
393  383          ASSIGN       SP.CI4,0
394  384          ASSIGN       SP.PI4,0
395  385          ASSIGN       SP.TRAN4,0
396  386          ASSIGN       SP.FA5,0
```

```
397  387          ASSIGN    SP.FS5,0
398  388          ASSIGN    SP.CI5,0
399  389          ASSIGN    SP.PI5,0
400  390          ASSIGN    SP.TRA5,0
401  391          ASSIGN    SP.FA6,0
402  392          ASSIGN    SP.FS6,0
403  393          ASSIGN    SP.CI6,0
404  394          ASSIGN    SP.PI6,0
405  395          ASSIGN    SP.TRA6,0
406  396          ADVANCE   TIME(V$JOB.IVAL)
407  397          WRITE,B
408  398          ASSIGN    JOB.NAME,14
409  399          ASSIGN    SP.FA1,1
410  400          ASSIGN    SP.FS1,6
411  401          ASSIGN    SP.CI1,0
412  402          ASSIGN    SP.PI1,0
413  403          ASSIGN    SP.TRA1,150
414  404          ASSIGN    SP.FA2,2
415  405          ASSIGN    SP.FS2,6
416  406          ASSIGN    SP.CI2,0
417  407          ASSIGN    SP.PI2,0
418  408          ASSIGN    SP.TRA2,150
419  409          ASSIGN    SP.FA3,3
420  410          ASSIGN    SP.FS3,6
421  411          ASSIGN    SP.CI3,0
422  412          ASSIGN    SP.PI3,0
423  413          ASSIGN    SP.TRA3,150
424  414          ASSIGN    SP.FA4,4
425  415          ASSIGN    SP.FS4,6
426  416          ASSIGN    SP.CI4,0
427  417          ASSIGN    SP.PI4,1
428  418          ASSIGN    SP.TRA4,150
429  419          ASSIGN    SP.FA5,0
430  420          ASSIGN    SP.FS5,0
431  421          ASSIGN    SP.CI5,0
432  422          ASSIGN    SP.PI5,0
433  423          ASSIGN    SP.TRA5,0
434  424          ASSIGN    SP.FA6,0
435  425          ASSIGN    SP.FS6,0
436  426          ASSIGN    SP.CI6,0
437  427          ASSIGN    SP.PI6,0
438  428          ASSIGN    SP.TRA6,0
439  429          ADVANCE   TIME(V$JOB.IVAL)
440  430          WRITE,B
441  431          ASSIGN    JOB.NAME,15
442  432          ASSIGN    SP.FA1,2
443  433          ASSIGN    SP.FS1,2
444  434          ASSIGN    SP.CI1,0
445  435          ASSIGN    SP.PI1,0
446  436          ASSIGN    SP.TRA1,10
447  437          ASSIGN    SP.FA2,2
448  438          ASSIGN    SP.FS2,3
449  439          ASSIGN    SP.CI2,0
450  440          ASSIGN    SP.PI2,0
451  441          ASSIGN    SP.TRA2,3
452  442          ASSIGN    SP.FA3,4
453  443          ASSIGN    SP.FS3,2
```

```
454  444          ASSIGN       SP.CI3,0
455  445          ASSIGN       SP.PI3,1
456  446          ASSIGN       SP.TRA|3,3
457  447          ASSIGN       SP.FA4,0
458  448          ASSIGN       SP.FS4,0
459  449          ASSIGN       SP.CI4,0
460  450          ASSIGN       SP.PI4,0
461  451          ASSIGN       SP.TRA|4,0
462  452          ASSIGN       SP.FA5,0
463  453          ASSIGN       SP.FS5,0
464  454          ASSIGN       SP.CI5,0
465  455          ASSIGN       SP.PI5,0
466  456          ASSIGN       SP.TRA|5,0
467  457          ASSIGN       SP.FA6,0
468  458          ASSIGN       SP.FS6,0
469  459          ASSIGN       SP.CI6,0
470  460          ASSIGN       SP.PI6,0
471  461          ASSIGN       SP.TRA|6,0
472  462          ADVANCE      TIME(V|JOB.IVAL)
473  463          WRITE,3
474  464          TERMINATE
475  465          GENERATE     900000,0,1
476  466          WRITE,8
477  467          TERMINATE,R
478               START        1


PARAMETERS:
JOB.NAME     SP.FA1     SP.FS1     SP.CI1     SP.PI1
SP.TRAN1     SP.FA2     SP.FS2     SP.CI2     SP.PI2
SP.TRAN2     SP.FA3     SP.FS3     SP.CI3     SP.PI3
SP.TRAN3     SP.FA4     SP.FS4     SP.CI4     SP.PI4
SP.TRAN4     SP.FA5     SP.FS5     SP.CI5     SP.PI5
SP.TRAN5     SP.FA6     SP.FS6     SP.CI6     SP.PI6
SP.TRAN6     SP.PI0


VARIABLES:
JOB.IVAL


NUMBER OF TRANSACTIONS ALLOWED:   264
```