February 1971
DJ-000-1-00

Marketing Publications
Building 204-2
Cherry Hill, New Jersey

Time Sharing Operating System

Programming System Information Manual

RCA

First Printing: February 1971 (DJ-000-1-00)

This manual supersedes the TSOS Software Description Information Manual (79-00-516) issued July 1968.

PREFACE

The TSOS Programming System Information Manual introduces the RCA Time Sharing Operating System (TSOS) and discusses, in general terms, its functional objectives, concepts and organization.

This publication is divided into five parts:

Part 1 consists of two sections: Section 1, a general description of TSOS and its evolution within RCA; and Section 2, a plan of system publications in support of its technology. The general description section discusses the evolution of TSOS and outlines the design objectives accomplished. The system publication plan section gives the purpose and scope of each TSOS publication. Audience level and type of user are outlined.

Part 2 consists of four sections: Section 1, TSOS program classes; Section 2, software organization and features supported; Section 3, description of the command language and its applicability to each user class; Part 2 concludes with Section 4, a general system flow outlining generation and usage of the system.

Part 3 consists of only one section which describes the unique portions of the RCA Spectra 70/46 hardware associated with TSOS.

Part 4 consists of eight sections: Section 1, the TSOS software system structure; Section 2, command language considerations; Section 3, data management concepts and facilities; Section 4, the diagnostic support system; Section 5, TSOS service (user) system and hardware checking concepts; Section 7, general system operations in batch and remote modes; and finally Section 8, a synopsis of the TSOS available application/user system.

Part 5 consists of two sections: Section 1, a general system summary; and Section 2, an illustrated "how-to-use-the-system" section.

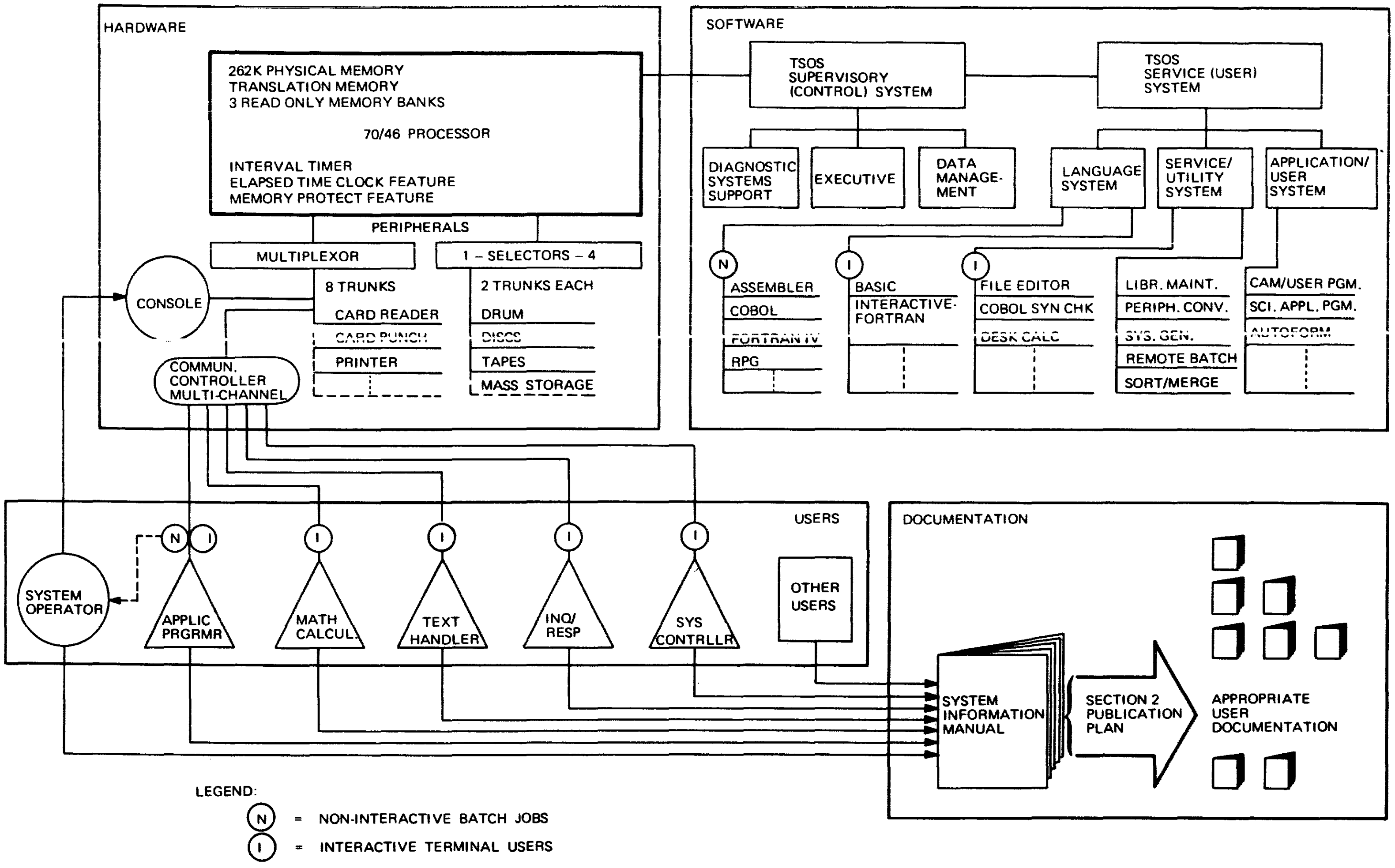
The Appendices contain a glossary of fequently used terms and a major subject publications cross-reference index.

No prerequisite publications are assumed for readers wishing a general overview of TSOS, as outlined in Part 1. Parts 2, 3, 4, and 5 assume that the reader is technically-oriented and has experience on third-generation, information-processing equipment and multiprogramming operating systems.

CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES



OVERVIEW OF TIME SHARING OPERATING SYSTEM (TSOS)

Part 1

Introduction to TSOS (Time Sharing Operating System)

Section 1: GENERAL DESCRIPTION

THE EVOLUTION OF RCA COMPUTING SYSTEMS

RCA computing technology has been in a continual evolution ever since its first computer was in operation. This section traces through the significant steps of this evolution for those who must evaluate the current "state of the art." The scope of RCA computing systems has expanded from serving the single user with a single problem to the servicing of many users demanding and receiving concurrent computer access. In order to handle this multi-user situation, there has been extensive development in software both to include programs which solve a large variety of specific problems and to exercise control over the entire computing facility with minimum human intervention.

This trend has grown chiefly from RCA's goal to achieve greater efficiency from its equipment and to make better use of human resources. The RCA Customer can neither afford to have computing equipment stand idle nor waste the time of skilled personnel by making them wait long periods of time for a problem solution, a report or, perhaps, nothing useful at all. These waiting periods arise in a large computing center whenever there is a long line-up of work requiring immediate service. Information processing managers, the persons who have been responsible for achieving higher efficiency from both mechanical and human resources, have been concerned with two traditional problems. First, the typical computing installation has, in addition to work demanding immediate attention, a number of jobs which can be scheduled in such a way that no one is actually penalized by waiting for them to be completed. Second, the procedure by which a problem is to be solved becomes so complex that only a very few experts will be able to understand it; the person with the problem is the best person to solve it, if he can do so without being put upon by the intricacies of a computer. What is needed, then, is a computing installation designed for many simultaneous users with various levels of expertise. Those users who require rapid response and who are not concerned with the complexities of the computer itself should be provided with simple terminals connected to the computer. Others users, who do not require such rapid response, should also be able to get the solutions or reports they require within a reasonable time frame.

Single-Program Execution

When RCA's first computing machines were put into operation, the method of performing tasks was, simply, one at a time. A program would be loaded into the computer, the "GO" button pressed and the results awaited. When that program had run its course, a second program would be loaded and run, and so on. Here was a single-program execution philosophy.

Improved Utilization

One of the chief disadvantages of the single-program execution philosophy is that the computer equipment is not fully utilized. Programs which make great demands on the internal data processing elements usually require the input/output equipment for only a small percentage of the time. Programs which require the input/output system much of the time usually permit the data processing elements of the computer to idle along at a fraction of their capacity. However, idle equipment is not the only drawback to single-program execution. Each time a user wishes to run a program he has to repeat the same long procedure. He is required to prepare his program in a sutiable form for the computer, describe his program requirements and expected behavior, send his program to the computer, wait his turn for machine time (usually the programs are processed in the order that they are submitted), allow time for solution and, finally, wait for the return of his results. The total elapsed time for all these steps to take place is the turnaround time.

Turnaround Time

Turnaround time under the single-program execution philosophy was usually a matter of many hours and often days. The computer operator, who had to read the operating instructions associated with each job and determine that the requirements of the program were met was one of the sources of lengthy turnaround time. The human operating speed is not a match for the processing speeds of the computer. Another source of delay, again involving the operator, was the problem of error definition. The error might be a machine failure, a failure on the part of the programmer who wrote the operating instructions or a mistake by the operator himself. In any case, the operator would not have adequate information on the problem to correct the error or he would require time to search for the solution. What was needed was some method for turning over to the computer some of the responsibility for managing and controlling its own facilities.

Job Streaming

To increase the utilization of the computing equipment, the computing centers began to require the programmers to write their programs and operating instructions in a standard format. There was also standardization at the computer. The programs to be executed were all available in the same way, the data for these programs was from a standard input device, and the results of these programs went to a standard output device. The computer under this standardization could now begin one program and go right on to a second and a third without interruption. There came into being a control program that monitored the transition from one job to the next. The effect of the standardization and the job-to-job transition monitor (usually called the job-stream monitor) was to reduce the dependence on the operator and to delegate some of his functions to the computer itself. Still further development in the efforts to increase equipment utilization led to more and more use of the computer itself to perform the clerical and error-recovery tasks that had been required of the computer operator. This control program that was devoted to the internal regulation and control over the total operation of the computing system became known as the operating system. More and more the operating system supplanted the computer operator and improved equipment utilization. The function of the computer operator was changed from being an essential intermediary to that of being an assistant to the computer.

Operating System

An operating system is a programmed method for making the decisions which in the early days of computing were left to a computer operator. In addition to providing standard responses to the standard error conditions (both in the programs and in the computing equipment), the operating system must allocate and control the resources of the computer. These resources include input/output devices, memory, time, and all of the other parts that comprise the modern computing installation. In addition, accounting records must be kept for each resource allocated to each program.

Multiprogramming

With the improvements in equipment utilization, the situation remained where a given program did not use the full facilities of the system all of the time. While one program in a job-stream would be loading the peripheral devices of the system, another would have been brought into the central data processor and be waiting for the peripheral devices to be free. Thus, one program would be dependent on another.

Under the control of an expanded operating system, several programs could be loaded at one time. One of the programs starts and the operating system surveys the other problems resident in the computer to see if any of them can utilize the remaining available equipment resources. The operating system continually examines each program in an attempt to keep all parts of the computing system occupied. Such control and manipulation of several different programs is called multiprogramming. The multiprogramming operating system was designed to make the most effective use of all equipment under any given situation. The multiprogramming operating system reduced the average turnaround time but it created some new problems of its own. One was that the equipment and the operating system became more complex in order to make the total system operation more efficient. This added complexity had many different effects. In the preparation of a program for execution, a user had to supply more information about his program than before in order to enable the operating system to control it. The solution of any problem that required deviation from the standard methods became increasingly difficult to handle. Unlike the earlier single-program execution, there was no way to predict the actual course and time of a given program through the system. Another problem was the increasing separation of programmer and computer. Although turnaround time was reduced, there was still no direct communication between the computer system and the programmer or the non-programmer such as the scientist or engineer with a problem to solve.

Communications

At the same time that multiprogramming operating systems were being developed, the capabilities of computing systems were being extended by the addition of communication equipment. There were three principal applications that found the communications devices necessary. One of these was data-gathering/data dissemination. Many different types of devices could be attached to a computer, allowing it to receive or send out data. Typically, these might be telephone lines with sending or receiving stations, telemetering equipment, or time card stations at the entrances to a large factory. Another form of communications application was message switching. One station can initiate a message and send it to one or more stations. Finally, there was the inquiry/response system, in which a collection of data files (data base) is probed for a piece of information or, conversely, the data base is altered by new information coming from some remote point. All three of these applications had in common the idea of placing the computer in one location and, through communications channels, sending information to or receiving information from other remote locations. Furthermore, the computer was used less for processing and more for handling data during the communication process.

Interactive Systems

The combination of multiprogramming and communications equipment lead into the next stage of operating system evolution: the interactive system. The interactive operating system differs functionally from the multiprogramming operating system in the manner of control of multiple user tasks within a given time frame. Multiprogramming allows two or more users access to a computer but discriminates among the users through the use of a priority scheme. In an interactive system, each individual terminal user receives his "fair share" of the system. The inquiry/response system is a rudimentary type of interactive system. A query is entered at a remote terminal, transmitted to the computer and, finally, elicts a response from the computer. However, the inquiry/response system does not permit the inquirer to do anything more than to request information from a single collection of data or to send new information for entry into such a data base.

In a truly interactive computing system, the terminal user is no longer restricted to utilizing a single data base but can create new data bases or programs or can alter existing programs. In fact, the user of an interactive system has available at his terminal all of the capabilities of the complete computing system.

Unlike a typewriter, a keypunch, or any other non-interactive device, the interactive terminal can be programmed to "talk back." If the user makes a mistake, he is advised immediately and may be prompted to provide corrections. He may ask for explanations of the various features that he may want.

Finally, and most importantly, he is able to get informative answers or reports in minutes instead of hours. This type of operation is completely feasible with present day equipment because the speeds of operation within the computer are sufficiently fast so that each user at his own terminal does not realize that any other terminal is in use. The ordinary "think time" of the human being is so much longer than the computer's reaction time that the computer appears to respond immediately. Each user believes that he has the computer all to himself.

A reasonable number of users can be accommodated at terminals, each feeling that he has the computer all to himself and the computer will still have the capability to run background programs. Background programs do not interact with any terminal, although they may be initiated from a terminal. They are exactly the same kind of programs that would have been run under the multiprogramming or job-stream systems or, in the early days, without any operating system at all.

There are many advantages to be gained from the interactive philosophy. One of the problems inherent in all previous systems was that of turnaround time. With an interactive system, the turnaround time is reduced to that required for the response. If the user's request can be answered quickly, it will be answered quickly; if the request begins a long computation, the answer will come back when the computation is completed. Another benefit of the interactive system is that the user is no longer separated from the computer. The user operates his own terminal, does his own programming and watches his own results as they appear.

TSOS PHILOSOPHY

RCA has developed an interactive, multiprogrammed operating system called TSOS. This operating system is designed for the 70/46 Spectra computer, a modified form of the 70/45. Together, the computer and the operating system offer a package that has application in many computing environments. TSOS combines its interactive capabilities with a complete facility to handle a full range of production-type tasks. Resources that are not occupied by interactive requests are used for carrying out all the typical production-type processes that can now be performed on an equivalent non-interactive multiprogramming computing system. Typical production tasks are those that can be given to the computer one day but whose solutions are not required until the next day. Often, these tasks require a relatively long time for solution and have no need for any ineractive operation. Such programs as payroll, inventory, billing and sales analysis runs are examples.

TSOS and the 70/46 provide greater production processing capabilities than any other equipment in the present Time Sharing Market. They provide interactive service for numerous simultaneous users. TSOS is, moreover, compatible with the Spectra 70/45 Tape Operating System (TOS) and existing Spectra programs can be handled without laborious conversion. Under interactive operation, all terminal users are treated with equality. The emphasis given when executing background programs, as compared with terminal processing, is determined by the particular computing installation.

Hence, a particular installation can control the allocation of resources between terminals and background programs; it is even possible to exclude one or the other. TSOS maintains accounting data for each program, showing the utilization of each of the system resources.

Resource Sharing

The RCA Time Sharing Operating System (TSOS) offers to each of its users an enhanced approach to the utilization of its computing resources. Advanced hardware and software technology, coupled with a comprehensive command/control language, yield a function-rich, total system design. This total system design effectively combines computer characteristics (computer memory, processor time and on-line storage resources) with dynamic human requirements and minimizes the need for the user to compete for these resources in order to accomplish their jobs on the system. (See Figure 1.)

The large demands on computer utilization are taken care of by batch jobs in the "background" mode. The individuals with small or occasional problems to solve may use the computer without waiting for these batch jobs to be completed. The Virtual Memory, time-slicing, on-line storage and program sharing techniques of TSOS permit a variety of concurrent demands to be handled with a minimum of scheduling conflicts and a maximum of system efficiency and throughput.

Virtual Memory

TSOS, through a combination of an enhanced addressing technique within the processor (translation memory), an on-line rapid paging drum and memory management control programming, has expanded the physical memory capability of the system to a virtual memory capability of over one million bytes for each user. (See Figure 2.) The advanced concept of memory allocation, program storage and memory management is referred to as paging.

The paging drum acts as an extension of physical memory. Active programs are divided into logical units called pages and reside on this drum until a particular page is required (demand paging).

The use of virtual addresses permits the program to refer to data and instructions as though they resided in a physical memory that is many times the computer's actual physical memory configuration. (See Figure 3.)

The user can concentrate on logical problem definition and need not concern himself with segmenting or virtual memory management, as this is handled by the system.

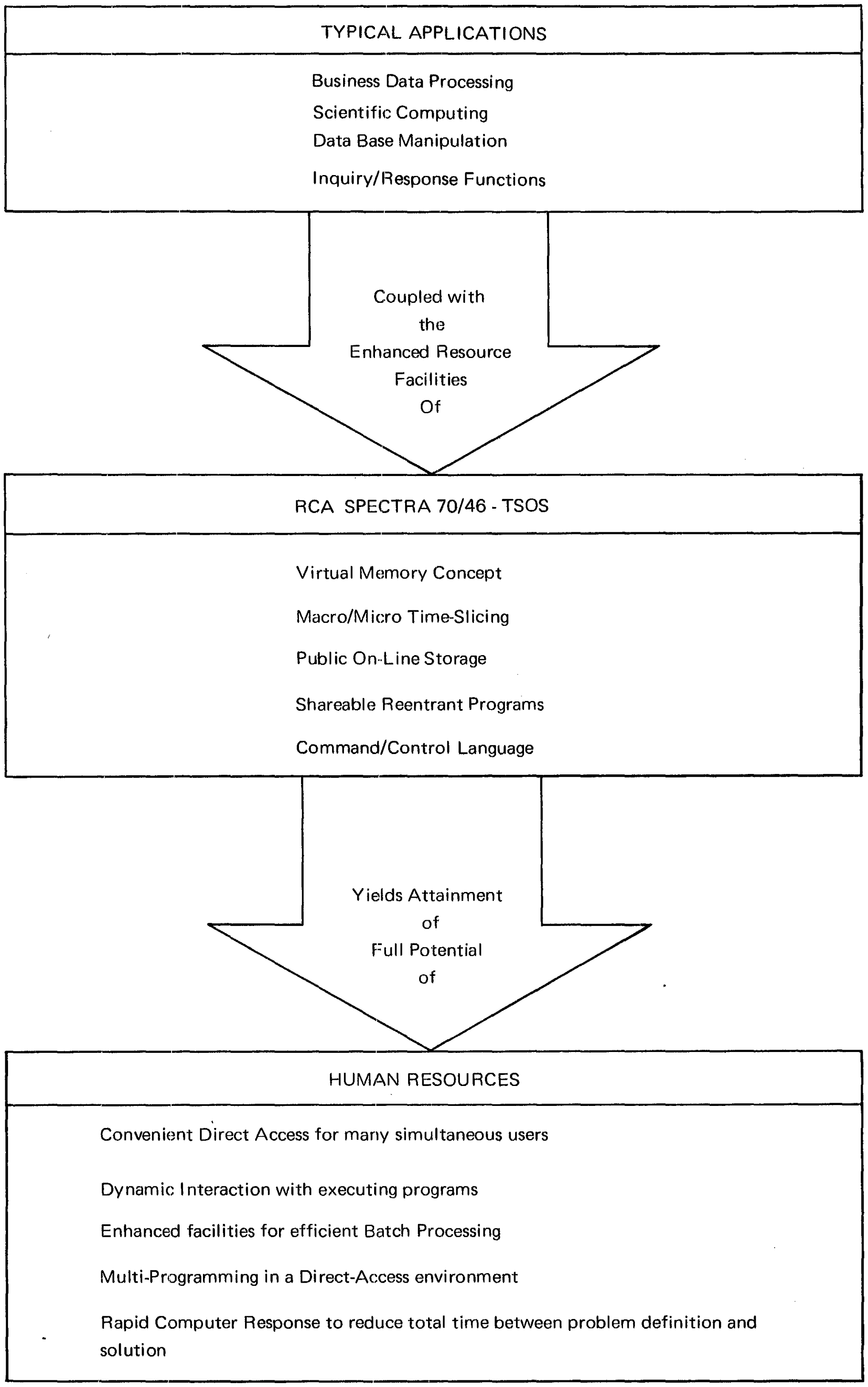

TYPICAL APPLICATIONS
Business Data Processing
Scientific Computing
Data Base Manipulation
Inquiry/Response Functions
Coupled with the Enhanced Resource Facilities Of
RCA SPECTRA 70/46 - TSOS
Virtual Memory Concept
Macro/Micro Time-Slicing
Public On-Line Storage
Shareable Reentrant Programs
Command/Control Language
Yields Attainment of Full Potential of
HUMAN RESOURCES
Convenient Direct Access for many simultaneous users
Dynamic Interaction with executing programs
Enhanced facilities for efficient Batch Processing
Multi-Programming in a Direct-Access environment
Rapid Computer Response to reduce total time between problem definition and solution

FIGURE 1. 70/46 TIME SHARING SYSTEM

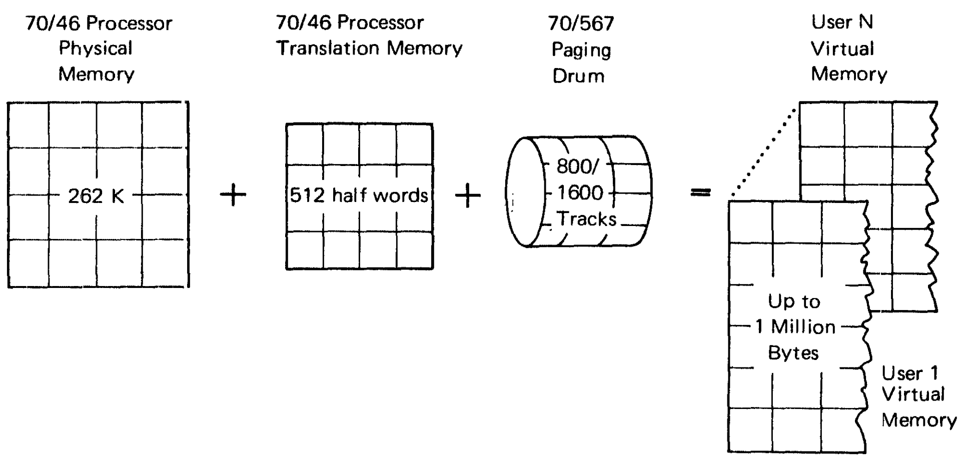


FIGURE 2. TSOS USER VIRTUAL MEMORY

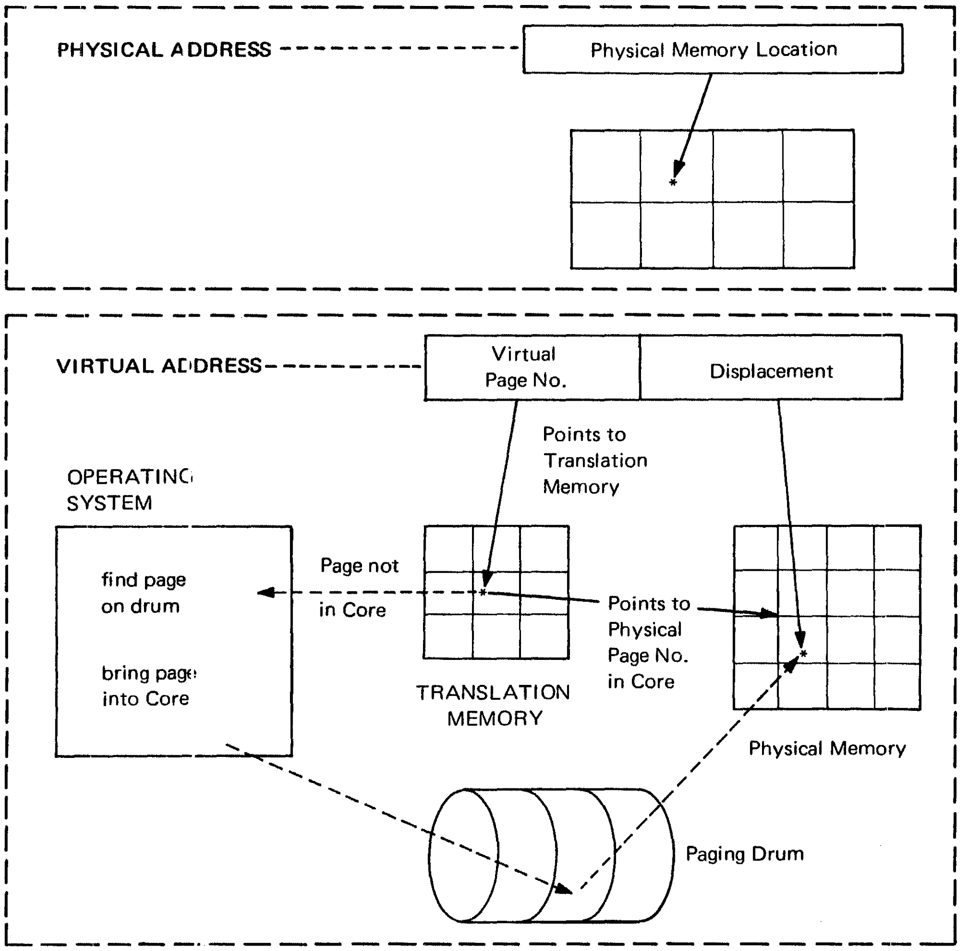


FIGURE 3. PHYSICAL vs. VIRTUAL ADDRESS

Time Slicing

RCA Spectra 70 third-generation multiprogramming systems (TOS, TDOS, etc.) allow up to six user programs to run concurrently. (DOS allows up to fourteen.) Each program's prioritv, relative to the other, is established when the program is loaded. Various classes of interrupts (I/O termination, etc.) trigger the execution of each level of priority. As a program waits for an external event to occur, the Supervisory Software scans the priority list and the next program that is waiting for processor time is entered.

The allotment of processor time among competing tasks within TSOS is accomplished by a combination of the TSOS Supervisory Software, a hardware Interval Timer and a system table of Task Queues. When a program reaches the top of the Ready Active Queue, the Interval Timer is set and the program is given control of the processor. (See Figure 4.)

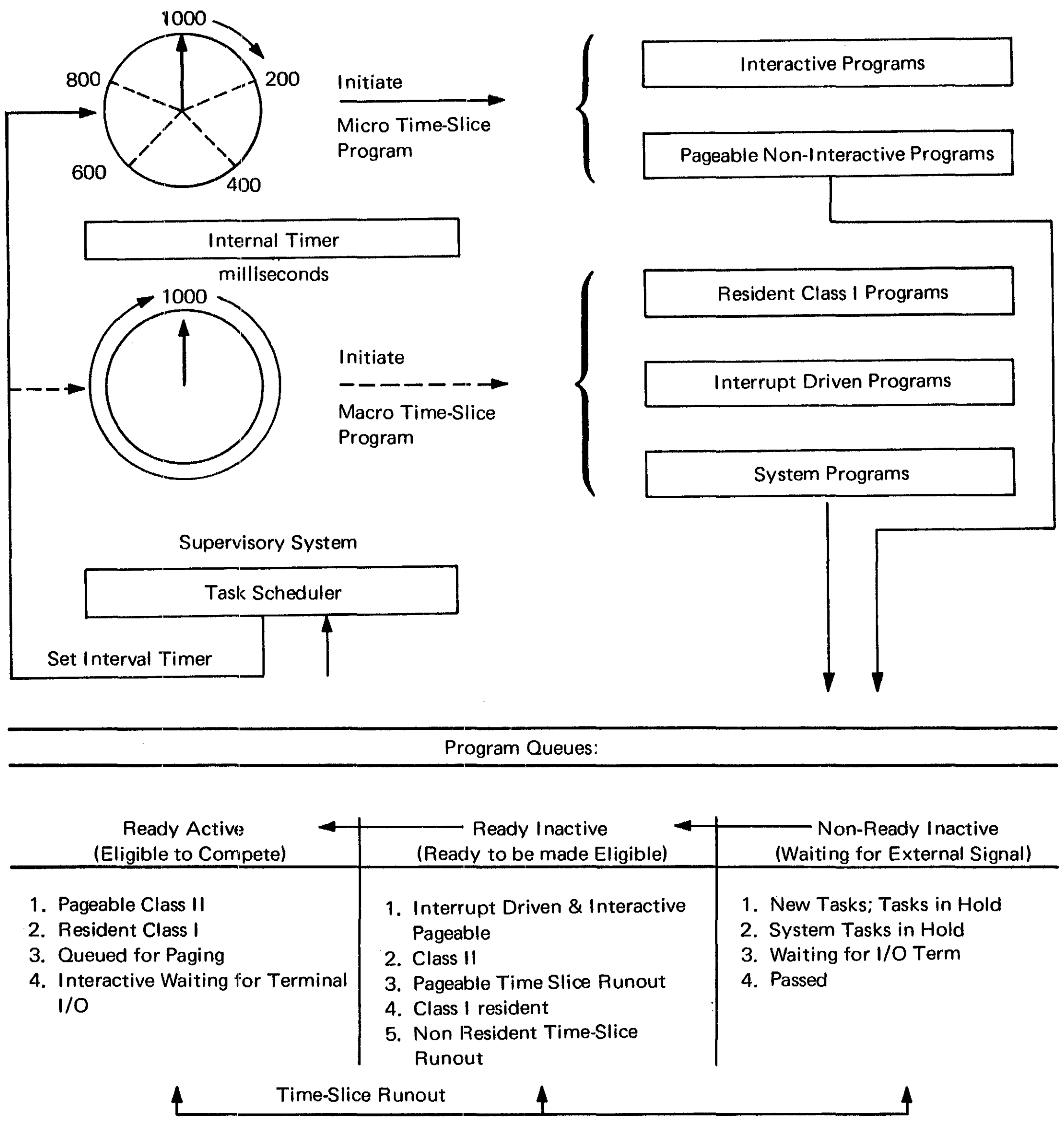

1000
200
400
600
800
Initiate
Micro Time-Slice Program
Internal Timer
milliseconds
1000
Initiate
Macro Time-Slice Program
Interactive Programs
Pageable Non-Interactive Programs
Resident Class I Programs
Interrupt Driven Programs
System Programs
Supervisory System
Task Scheduler
Set Interval Timer
Program Queues:
Ready Active (Eligible to Compete)
Ready Inactive (Ready to be made Eligible)
Non-Ready Inactive (Waiting for External Signal)
1. Pageable Class II
2. Resident Class I
3. Queued for Paging
4. Interactive Waiting for Terminal I/O
1. Interrupt Driven & Interactive Pageable
2. Class II
3. Pageable Time Slice Runout
4. Class I resident
5. Non Resident Time-Slice Runout
1. New Tasks; Tasks in Hold
2. System Tasks in Hold
3. Waiting for I/O Term
4. Passed
Time-Slice Runout

FIGURE 4. TSOS TIME SLICING

Each resident (nonpageable), interrupt-driven task and system task is given a one-second (1000 milliseconds) macro time-slice upon gaining control of the processor. When this time-slice runs out or an interrupt is generated, the task is placed on another queue. The queue to which an initiated task is transferred depends on the condition of the task at the time the transfer is made. The timer is stored or reset and the next Ready Active task is initiated.

Interactive (conversational) and pageable tasks are given five quanta (macro time-slices) of 200 milliseconds each. Every micro time-slice runout lowers a task's priority in the queues and when all five have run out, the task is labelled "macro compute-bound".

The system does not favor compute-bound tasks and controls the behaviour of pageable tasks by favoring those that use short bursts or processor time. The effect is to give the interactive user a higher priority when competing for the resources of the system than the batch user who does not require as rapid a response

On-Line Storage Sharing

In addition to the Paging Drum, where many active programs may reside, TSOS introduces the concept of Public Volumes as an extension of available storage space.

Under TSOS operation, a volume may be a removable 70/564 Disc Pack, a 70/590 Direct Access Storage System, a reel of mangetic tape, or a magazine of the 70/568 Mass Storage Unit. Magnetic tapes and magazines of the 70/568 are always considered to be private volumes. A public volume is a direct-access volume (other than a 70/568 magazine) that is mounted on-line throughout the entire period of system operation. (See Figure 5).

TSOS assumes that a user wishes a file to be stored on a public volume unless the user specifically asks for storage on a private volume. Each user, when joined to the system, is allocated a specified amount of public volume storage space and is notified when this maximum is approached. The space is not actually allocated until it is required and is referred to as the user's permanent public space allotment. This space is not used for work files created for the user by system programs (i.e., the TSOS Assembler, COBOL Compiler, etc.) although these files, too, are stored on public volumes. Control of files within the system is maintained by a system cataloging function.

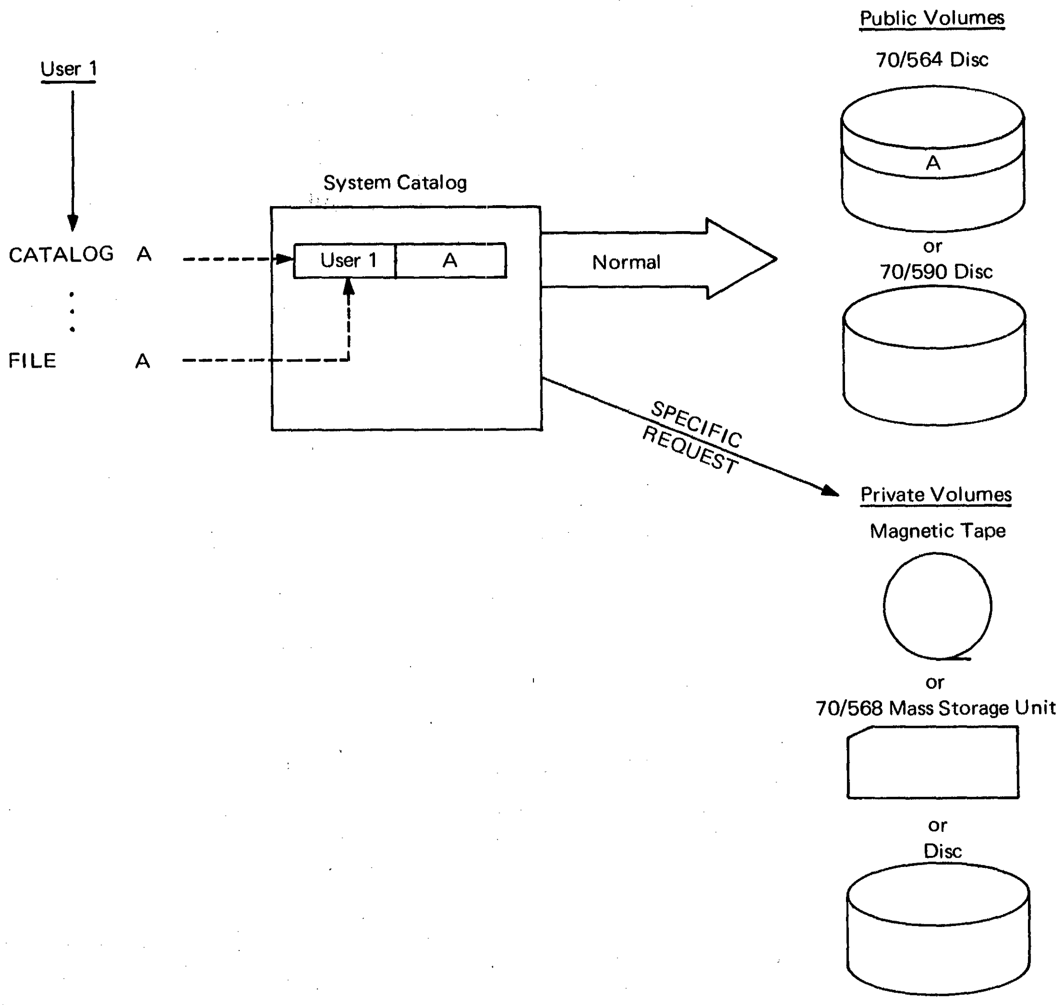


FIGURE 5. TSOS PUBLIC-PRIVATE VOLUMES

Public volumes are given attributes which tend to make them appear as extensions of physical memory storage rather than as conventional input/output devices. The attendant list summarizes their important attributes.

1. The public volume is the normal (or default) type of volume.

2. A given file may be contained on any number of public volumes without the user's knowledge.

3. Full file security and integrity are provided, yet the volume may be used concurrently by any number of tasks.

Program Sharing

Certain programs in an interactive environment are in demand by many concurrent users. To obtain greater efficiency, TSOS components such as BASIC (Beginner All-purpose Symbolic Instruction Code), IFOR (Interactive FORTRAN IV), and File Editor are reentrant-coded, sharable programs that require only one active copy in the system to serve many users. The ability to have sharable programs conserves total system memory and avoids the redundant loading of a unique copy of the program for each user. TSOS users themselves may also create reentrant-coded, sharable programs that may be introduced into the program library. (See Figure 6.)

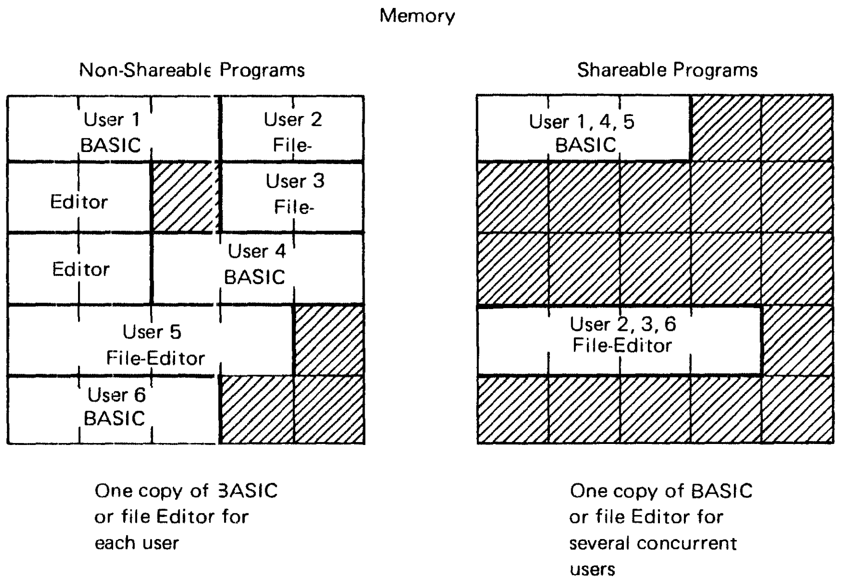


FIGURE 6. SHAREABLE vs NON-SHAREABLE PROGRAMS

TYPICAL APPLICATION

A manufacturing company of medium size is a typical application area for TSOS. A company of this sort generally uses a computer in conjunction with the following five activities (perhaps differently named):

accounting,
quality control,
inventory control,
staff, and
research and development.

Application Programmer

The accounting function generally produces the largest number of batch programs. Billings, accounts receivable, accounts payable, and payroll (together with associated preprocessing) are typical jobs. These are large production type processes requiring long runs, with the end result being printed reports and updated records on some storage medium.

The quality control group requires large statistical runs which present the same kind of demand to the system as the accounting group except that greater computational ability is usually required. The end result of these runs is usually printed reports. The quality control group may also use the interactive operation, for example, to analyze data taken from samples and entered through remote terminals.

Computational Non-Programmer

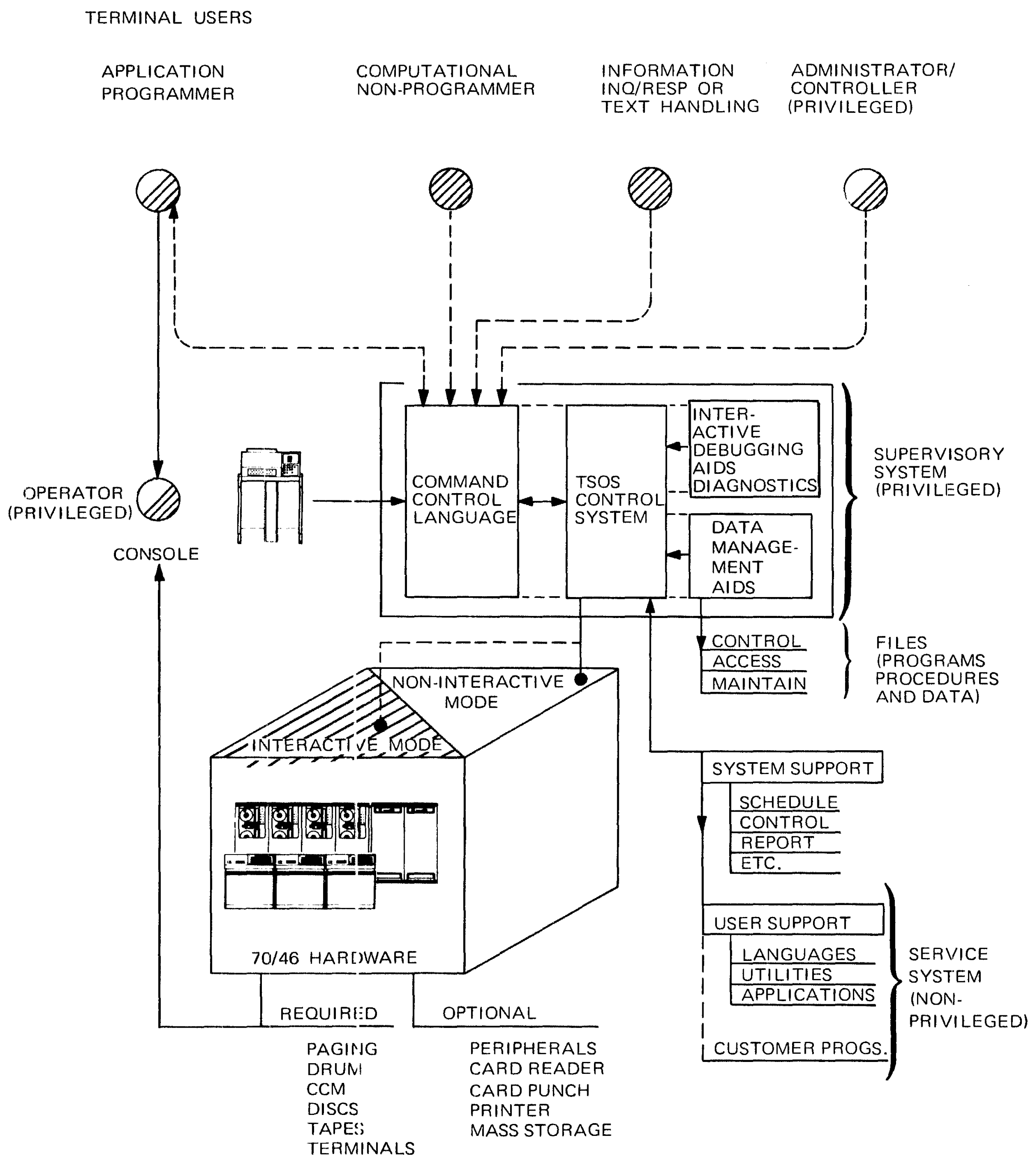
The inventory control group is concerned with production scheduling, inventory control, and possibly-process control. They will create a heavy demand for reports on an exception basis. There are many requests each day for small amounts of information. The average computer use time, however, is short. Large data runs are uncommon. The interactive capability of TSOS will suffice for the majority of the computer use by this group.

The research and development group typically has large and often complex problems to solve. Although this group frequently has its own computer, it must often rent time on a large, outside computer. The chances are that this research and development group has already begun to use interactive operation and possibly has at least one terminal connected to an interactive system.

Management Information – Inquiry/Response

The staff group is composed of individuals who are concerned with information management. Their aim is to improve the information flow in the organization, and their use of computers may range from large-scale models of the organizational information flow to small interactive information retrieval systems for management use.

With TSOS and the 70/46, all of these groups will be able to get their work done on a single computer system and with minimum scheduling problems. Program preparation will, in many cases, be reduced. Rental will, in general, be less because of a reduction in the total amount of equipment needed to satisfy the various application requirements. (See Figure 7.)



LEGEND:

▨ - - -	USER IN INTERACTIVE MODE WITH FAIR-SHARE TIME SLICE.
○——	NON-INTERACTIVE MODE USING BACKGROUND SCHEDULING ALGORITHM AND MULTI-PROGRAMMING.

FIGURE 7. TYPICAL 70/46 TSOS ENVIRONMENT

GENERAL

The complete publications library for the RCA Time Sharing Operating System is shown in Figure 8. Ordering numbers are not shown. Refer to the Publications Catalog for the latest ordering information.

Each publication is described briefly in this section. Abstracts of each hardware and software product comprising the RCA 70/46 Time Sharing Operating System are contained in the Spectra 70 Systems Information Manual publication. A complete listing of all hardware and software documentation can be found in the Publications Catalog.

Publications are grouped according to the audience to which they are directed:

1. All systems personnel (general reader).

2. System operational personnel (management, administrators, operators, programmers, and application users of either background or interactive operations).

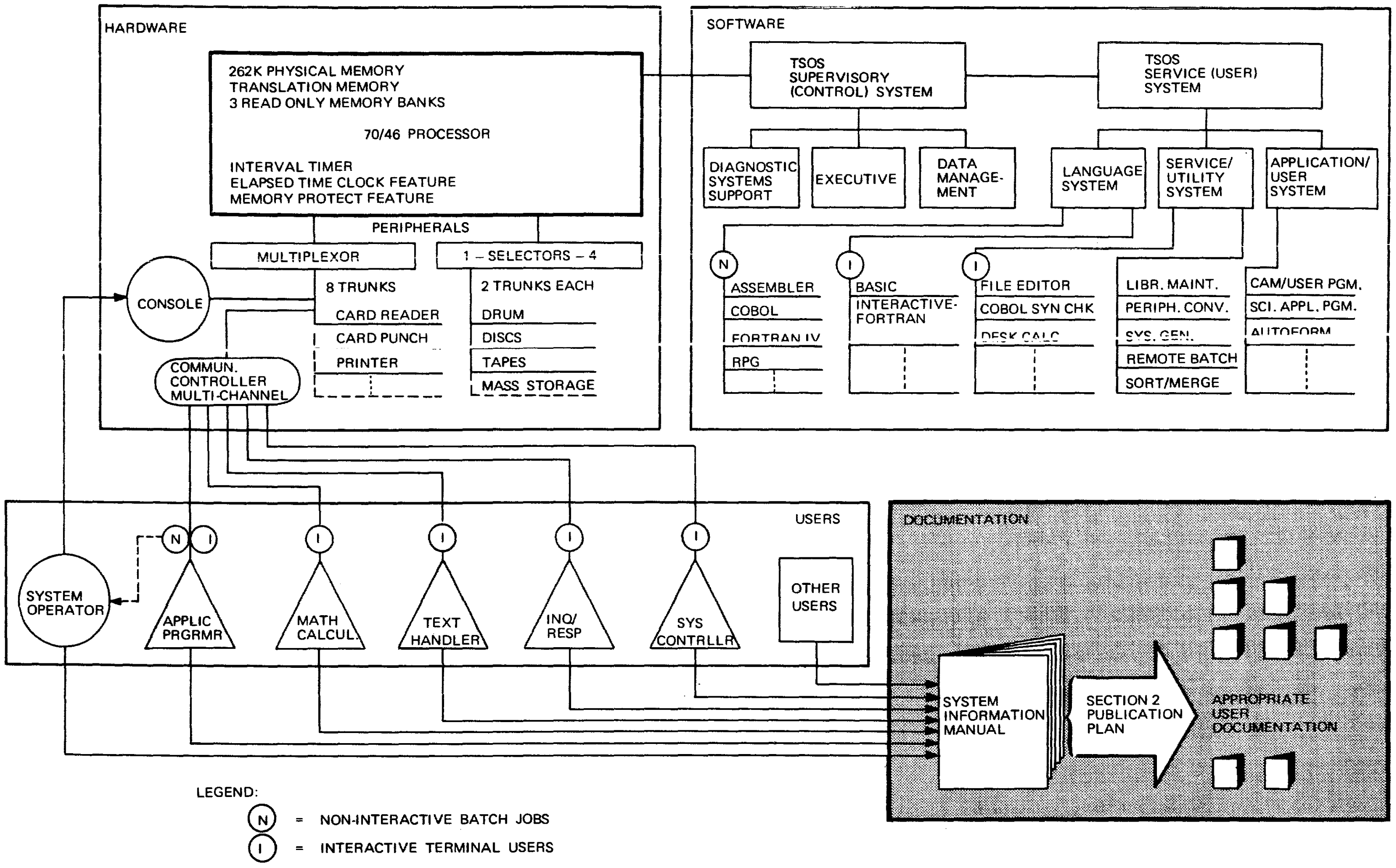
Two documents in the library are common to all systems personnel utilizing the RCA Time Sharing Operating System, i.e., the Executive Command Language Reference Manual and the Data Management System Reference Manual.

Following is a list of the TSOS publications with a brief abstract of each.

TSOS PUBLICATIONS

The TSOS PROGRAMMING SYSTEM INFORMATION MANUAL is the document that you are now reading. Its purpose is to give an overview of the system. It also tells you where you can find other published information about TSOS.

The OPERATOR'S GUIDE is a working document for the Computer Operator to aid him in manning the equipment at the computer installation. It describes the system facilities available to him. It contains a list of all of the messages and their meaning that the system can type on the operator's console typewriter. It also contains actual examples of operator action; for example, the system load procedure.



OVERVIEW OF TSOS — DOCUMENTATION

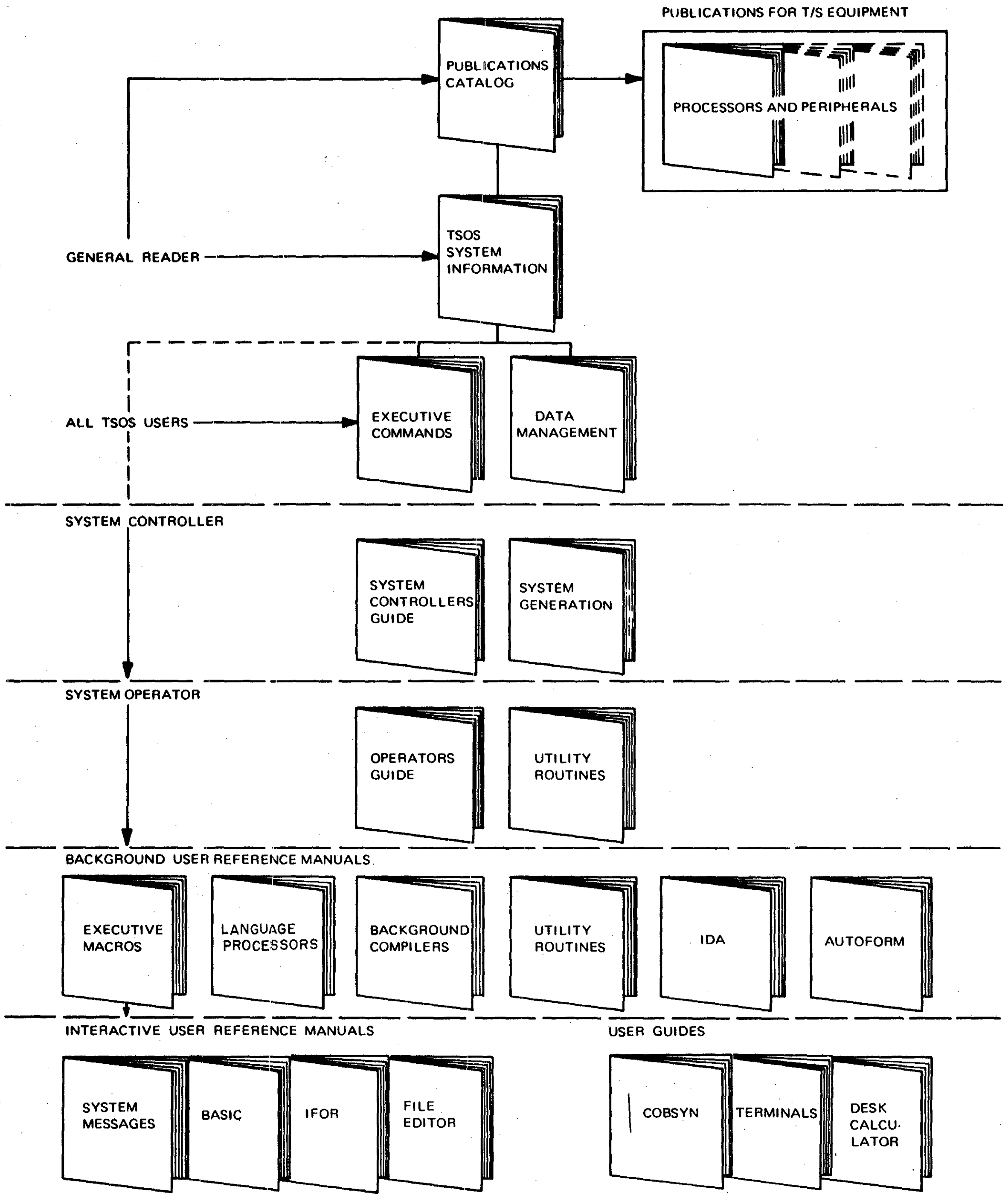


FIGURE 8. TSOS PUBLICATION PLAN

The TERMINAL USER MESSAGE MANUAL contains all the system messages that can occur at a remote terminal. System messages are those output by TSOS that are not initiated by one of the interactive components (whose messages are contained in their own reference document). The user is informed of the meaning of all system messages and advised of the action he should then take.

The SYSTEM CONTROLLER'S GUIDE aids the System Controller in managing the computer installation. It describes the commands that he can use and gives him practical tips on how to use them. It also contains actual examples of System Controller operations.

The SYSTEM GENERATION (SYSGEN) REFERENCE MANUAL explains the procedure for performing the system generation. (System generation is the initial software activity at the 70/46 site when new software arrives. It is the process of tailoring the TSOS system to fit the user's equipment configuration and software requirements.) In addition to the rules, this manual contains a console printout of a complete actual SYSGEN session.

The EXECUTIVE COMMANDS REFERENCE MANUAL describes the use of the Executive system commands. These commands, which are available for all classes of user, are the means by which the user communicates with the system. This document explains all of these commands, their operands, their use and contains illustrative examples.

The EXECUTIVE MACROS REFERENCE MANUAL describes the use of the Executive macro-instructions to the Background User. These are instructions that the user inserts into his Assembly language program to request that certain Executive functions be performed for him by the system when the program is executed. This document explains all of these macros, their operands, their use and contains illustrative examples.

The DATA MANAGEMENT SYSTEM (DMS) REFERENCE MANUAL is a complete reference document for all data management commands and macros. It provides the System Controller, the Programmers, and the Interactive User with all operands and usage rules for the DMS commands and macros.

The BACKGROUND COMPILERS REFERENCE MANUAL tells the Background User how to operate the FORTRAN and COBOL compilers and the Assembler. The rules and syntax for these languages are described in other documents. This manual provides complete instructions for the use of these two language processors.

The IFOR USER GUIDE is a highly example-oriented manual intended to assist the user of the TSOS interactive FORTRAN facility in operating IFOR. It contains many examples of actual IFOR terminal sessions.

The FILE EDITOR REFERENCE MANUAL gives a complete description of the rules of the TSOS File Editor. All of the statements are explained, their operands are described and rules are given for their use.

The FILE EDITOR USER GUIDE is a highly example-oriented manual intended to assist the user of the TSOS File Editor in his terminal operations. It contains many examples of actual File Editor terminal sessions.

The LANGUAGE PROCESSORS REFERENCE Manuals give the syntax of the programming languages supported by TSOS and the rules for writing programs in these languages. These are the Assembler, COBOL, FORTRAN IV, and the Report Program Generator (RPG).

The IDA REFERENCE MANUAL gives complete information for the use of all IDA commands and macros to guide both the conversational and background user in debugging his program.

The DESK CALCULATOR TERMINAL USER AID describes all operators and parameters of the Desk Calculator routine for the user wishing to simulate a desk calculator at a terminal.

The BASIC INFORMATION MANUAL is a stand-alone reference manual and user guide for use of the Beginners All-Purpose Symbolic Instruction Code Language under TSOS. It also contains an introduction to time sharing concepts and an introduction to programming, using BASIC. The interactive user needs no other document (except the Terminal User Aid-Terminals) to gain access to TSOS and to write programs in BASIC.

The COBSYN USER GUIDE is a stand-alone reference manual and user guide for use of the COBOL Syntax Checker under TSOS. The user needs no other document (except the Terminal User Aid - Terminals) to use COBSYN.

The TERMINAL USER AID – TERMINALS contains a description of the terminals that TSOS supports, and operating instructions for their use. With this document the interactive user can access the system, LOGON, and operate the equipment.

The TSOS UTILITY ROUTINES REFERENCE MANUAL instructs the background user in the operation of such batch processes as the volume initializer, loaders, and library maintenance routines.

The AUTOFORM REFERENCE MANUAL describes all commands and control operations of the Automatic Text Formatting system. It tells the user how to use Autoform and contains examples to guide the user to proper operation.

The IFOR REFERENCE MANUAL gives a complete description of the rules of the TSOS FORTRAN interactive facility. All of the statements are explained, all operands are described, and rules are given for their use.

Part 2

Overview of TSOS

Section 1: INTRODUCTION

SCOPE

The basic concepts of TSOS presented here provide the reader with a rapid understanding of how the system works from the user's point of view.

More detailed descriptions of the various TSOS features can be found in the appropriate software sections of this document. Hardware features can be found in the associated 70/46 processor and peripheral manuals.

PROGRAM CLASSES

The Time Sharing Operating System (TSOS) supports two types of programs which are referred to as Class I and Class II.

All Class I programs must be resident in physical memory and remain resident throughout their processing period.

Class I programs are restricted to the physical memory addressing capacity of the 70/46 Processor and thus do not undergo address translation. In addition to these restrictions, all Class I programs require contiguous physical memory locations and may not be pageable or utilize conversational features. (See Figure 9.)

Class II programs are supported by TSOS run in the 70/46 mode and utilize the full pageable/virtual memory features of TSOS. Class II programs can be divided into logical blocks of 4096 bytes (called pages) and be physically scattered throughout memory during their execution. Only the specific page or pages required at that instant of processing need be in memory while the remaining portion of the program resides in the paging drum awaiting requests for loading by TSOS.

Class II programs may be interactive (conversational) or non-interactive as dictated by the programmer.

CLASS I PROGRAMS	CLASS II PROGRAMS
70/45 Mode	70/46 Mode
Physical Memory Resident	Virtual Memory Resident
Physical Addressing	Virtual Addressing
Contiguous Memory Locations	Scattered Pages
Non-Pageable	Pageable
Non-Conversational	Can Be Conversational

FIGURE 9. TSOS PROGRAM CLASSES

Interactive (Conversational) Programs

Interactive programs engage in a dialogue with the user at a remote terminal. These programs are always Class II pageable programs and employ the resources of the 70/46 TSOS environment.

Non-Interactive (Background Batch) Programs

Non-Interactive programs do not directly interact with the user. Input instructions are specified in advance. Output and diagnostics are produced at the processor site. These programs may be initiated from a remote terminal but their execution is in the background. They may be Class I or Class II programs.

SYSTEM TUNING

The Time Sharing Operating System (TSOS) is "tunable" between background and interactive processing. This "tuning" is controlled by the System Operator in four distinct ways:

1. by specifying the allocation of available physical memory, in any desirable proportion, between the two classes of programs;

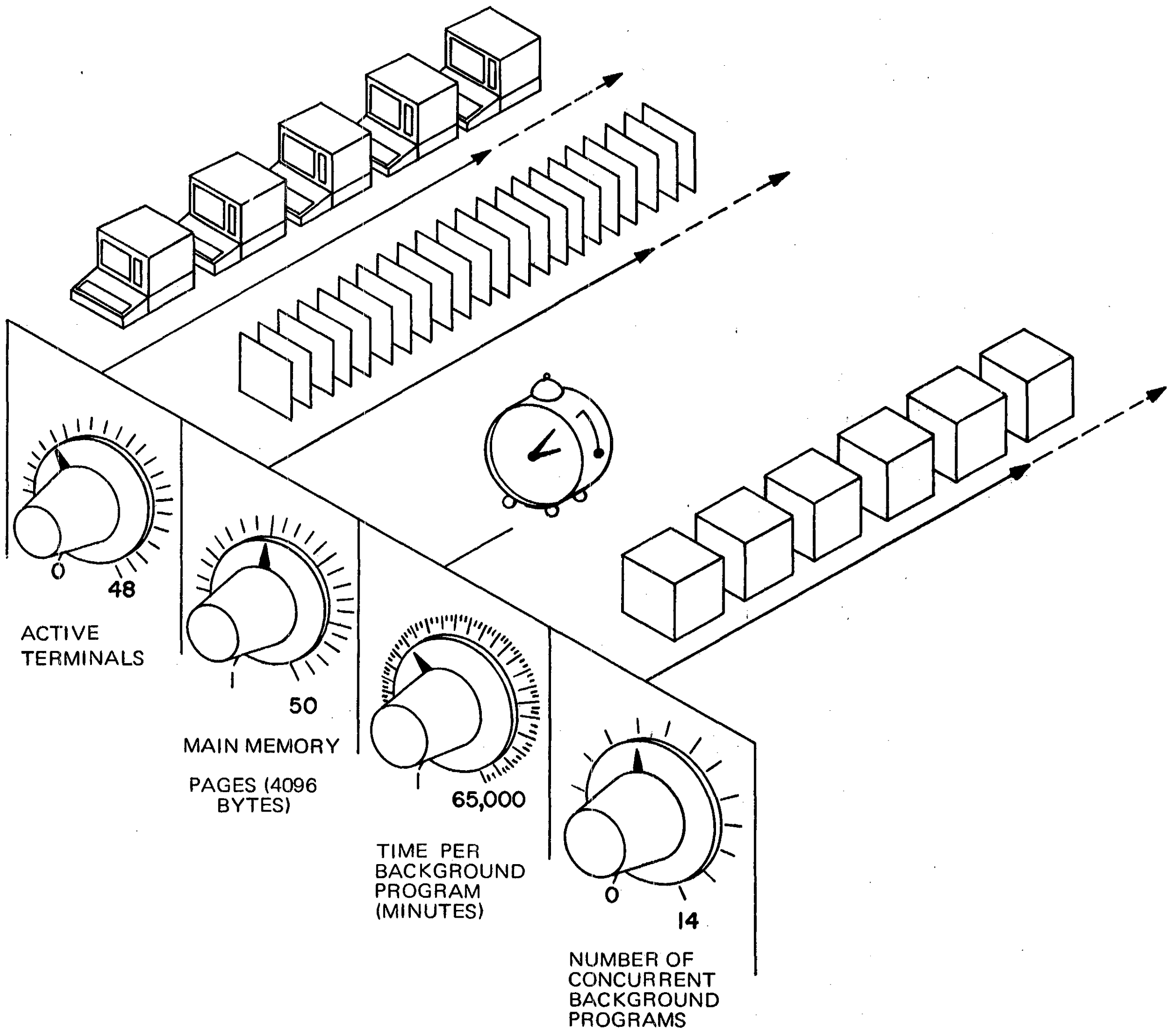
2. by varying the number of active terminals allowed into the system;

3. by controlling the total running times of background jobs; and

4. by controlling the number of concurrent background programs.

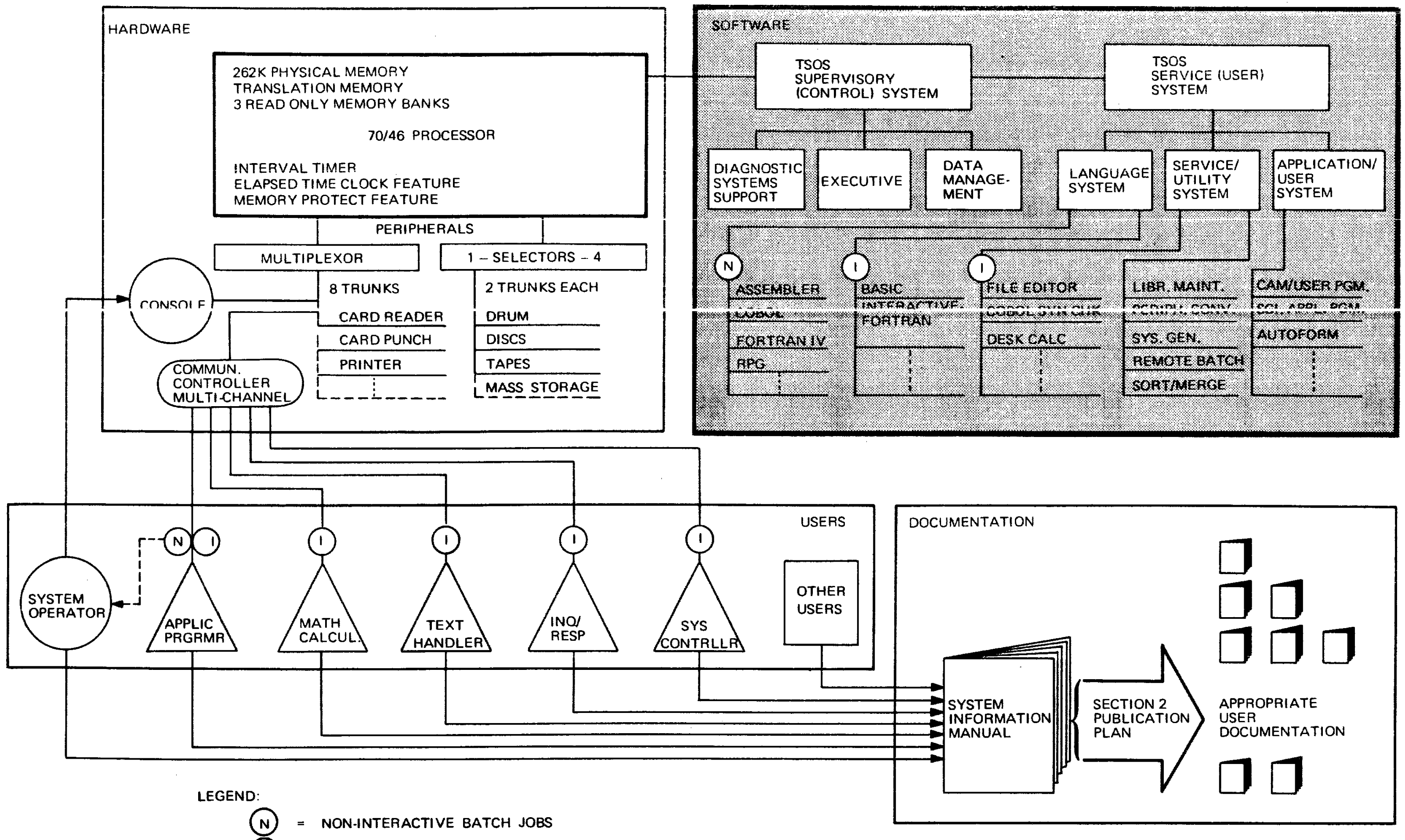
The size of physical memory is 262K bytes; however, approximately 40-60K bytes are always occupied by the resident portion of TSOS and by buffer areas for active terminals, leaving approximately 200K bytes available for user programs.

Depending upon the processing requirments of an installation at any given time, the 200K bytes of available physical memory can be divided between background processing and interactive processing with both types being done concurrently. This tuning is accomplished by the system operator's BIAS Command. (See Figure 10.)


0
48
ACTIVE TERMINALS
1
50
MAIN MEMORY
PAGES (4096 BYTES)
1
65,000
TIME PER BACKGROUND PROGRAM (MINUTES)
0
14
NUMBER OF CONCURRENT BACKGROUND PROGRAMS

```
/BIAS TERLMT=n, DRUMBIAS=n, TIME=n, BATCHLMT=n
```

FIGURE 10. TUNING THE SYSTEM — THE BIAS COMMAND



LEGEND:

Ⓝ = NON-INTERACTIVE BATCH JOBS

Ⓘ = INTERACTIVE TERMINAL USERS

OVERVIEW OF TSOS — SOFTWARE

INTRODUCTION

The TSOS Software System consists of a Supervisory (Control) group of programs and a Service (user) group. (See Figure 11.)

The Supervisory programs operate in the privileged modes of the processor and control the resources of the system. These programs are identified as follows:

1. The Executive Group,
2. The Data Management System Group, and
3. The Diagnostic System Support Group.

The Service (User) programs operate in the non-privileged mode and support the user in performing the tasks he wishes done in the system. These programs include:

1. Language Processors,
2. Service/Utility Programs, and
3. Application/User Programs.

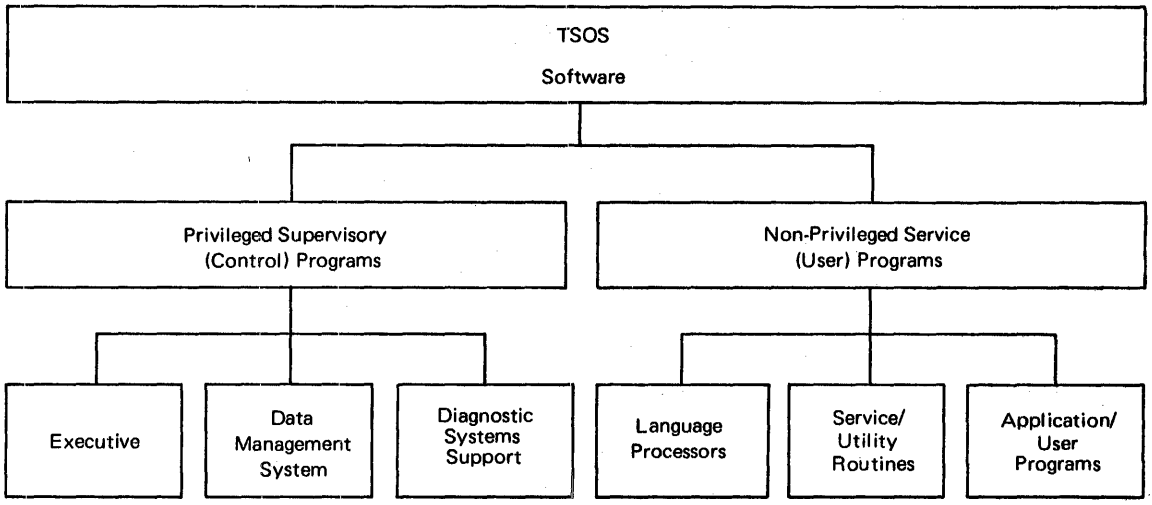

TSOS
Software
Privileged Supervisory (Control) Programs
Non-Privileged Service (User) Programs
Executive
Data Management System
Diagnostic Systems Support
Language Processors
Service/ Utility Routines
Application/ User Programs

FIGURE 11. TSOS SOFTWARE STRUCTURE

SYSTEM LOGICAL FILES

TSOS supports the following system files:

SYSIN (SYSCMD, SYSDTA)

SYSIPT

SYSLST

SYSOPT

SYSOUT

SYSIN

The commands and data issued by the user to direct the execution of his task are contained in the SYSIN file. SYSIN is not actually a logical system file; rather it consists of two files, SYSCMD and SYSDTA. All commands issued by the user to direct the system in the execution of his task must be contained in the SYSCMD file. The problem program cannot read this file; it is accessed only through privileged macros available to the control program.

SYSIN can also include data for input to the problem program such as source language statements to be processed by a compiler or input to the File Editor. Such data, which constitutes the SYSDTA file, is read by the problem program via the RDATA macro instruction.

To clarify the roles of these files, a task can be viewed as being in the command mode or in the problem program mode. When the task is in the command mode, the SYSCMD file is active; when the task is in the problem program mode, the SYSDTA file is active. One or the other, but not both, is always active during task execution.

SYSIN is defined as two logical files so that commands and data can be obtained from independent sources. For example, during the user's first session, he may wish to input both commands and FORTRAN source statements from a terminal. During the second session, the user may wish to input corrections to the file, using the File Editor, and type this input at the terminal. In this case, he prefers that SYSDTA be pointed to the terminal. After the correction process, he wishes to recompile his program, which is on disc as a cataloged file. In this case, he prefers that commands still be input from the terminal; however, he requires that SYSDTA be directed to his disc file.

For conversational tasks, the system assumes that the source for SYSCMD and SYSDTA is the terminal. For nonconversational tasks, the source for both files is the card reader (or tape, both of which are spooled), or an ENTERed file on a random access device. These initial sources are called the primary SYSCMD and the primary SYSDTA.

SYSIPT

This file is similar to the SYSDTA file; it is supported for TOS compatibility and functions similarly to the TOS Monitor's SYSIPT file. It is accessed by the problem program's issuance of the RDCRD macro. This file can assume all of the sources allowed for SYSDTA except the terminal. System performance would be adversely affected if Class I programs (which are by definition physically resident) could obtain input in an interactive mode.

Consequently, when a conversational task is initiated, no SYSIPT exists. When a nonconversational task is initiated, the system assumes that SYSIPT is the same file as the primary SYSCMD file. It should be noted that a task (even conversational) can define a SYSIPT file using the SYSFILE command. Like SYSDTA, if the file is defined in a procedure, its definition is reset when the ENDP command is processed to the same definition which it had when the last command in the primary SYSCMD file was processed.

Disc Format for System Input Files

As discussed previously, the system input files SYSCMD, SYSDTA and SYSIPT can be cataloged files on random access devices. These files can have Sequential (SAM) or Indexed Sequential (ISAM) file organization containing format V (variable) records.

SYSLST

The SYSLST file consists of information to be printed at the central computer facility. The file serves the same function in both conversational and nonconversational modes. Typically, the problem program, operating in conversational mode, directs diagnostic messages or response messages to SYSOUT (the terminal) but directs substantial output (complete listing, core dump) to the SYSLST file. The information comprising SYSLST is stored as a temporary file on a random access device; the file is spooled to the printer when the task is completed and is then erased. Output is sent to this file by the WRLST macro and the PROUT macro. The PROUT macro is supported for TOS compatibility. This file is automatically created for the user and cannot be redefined by the SYSFILE command.

SYSOPT

The SYSOPT file is primarily provided to support TOS compatibility. Output is sent to the file by the WRTOT macro-instruction. The system creates a temporary SYSOPT file which is automatically spooled to the card punch at task termination. SYSOPT cannot be redefined by the SYSFILE command.

SYSOUT

The SYSOUT file typically consists of system messages, the responses to command execution, and limited problem program outputs that are developed during the execution of a task. In conversational mode, the information comprising SYSOUT is printed at the user's terminal. In nonconversational mode, the information comprising SYSOUT is stored as a temporary file on a random access device. The file is spooled to the printer when the task is completed and is then erased. Output is sent to this file by the WROUT macro-instruction. This file is automatically created for the user and cannot be redefined by the SYSFILE command.

Summary

Table 1 summarizes the characteristics of the system logical files.

TABLE 1. SYSTEM LOGICAL FILES

File		Macros to Access the File	Source of Disposition of Data	Typical Usage
SYSIN	SYSCMD	Privileged macros - not available to user	1) Terminal (conversational task only) 2) Cataloged file on random access device 3) System card reader	Commands to direct task.
SYSIN	SYSDTA	RDATA	1) Terminal (conversational task only) 2) Cataloged file on random access device 3) System card reader	Source program input; data for File Editor.
SYSIPT		RDCRD	1) Cataloged file on random access device 2) System card reader	Source input for TOS language processors.
SYSLST		PROUT WRLST	System cataloged file which is automatically spooled to the printer and then erased.	Listable output from language processors.
SYSOPT		WRTOT	Written to a temporary system file on a random access device.	Object Module output for TOS language processors.
SYSOUT		WROUT	1) Terminal (conversational task only). 2) System cataloged file which is automatically spooled to the printer and then erased.	Messages from control program to user.

Redirecting SYSCMD and SYSDTA

When the system reads an ENTER or DO filename from SYSCMD, the specified file then becomes the SYSCMD file; such a file is called a procedure. A procedure can call another procedure; however, procedures can not be nested. When the ENDP command is processed in a procedure, the system redirects the SYSCMD file back to its primary source. Note that SYSDTA is in no way affected. Thus, the conversational user who calls a procedure still can have the terminal as the source for his data. Table 2 summarizes the Procedure File characteristics.

The SYSFILE command can be used to direct the SYSDTA file to a cataloged file on a random access device. The SYSDTA file can be redefined to again point to the primary SYSDTA file by the command.

SYSFILE SYSDTA=(PRIMARY).

SYSDTA is automatically redefined when a ENDP command is processed. At this point, SYSDTA is reset to the same definition which it had when the last command in the primary SYSCMD file was processed.

SPOOLING

TSOS is a direct-access oriented system. Whenever a program requires input from a low-speed device such as a card reader, the system will read (spool-in) this input and place it in temporary disc storage prior to delivering it to the program. A similar action (spool-out) is taken for output to a printer or card punch. The data is not actually spooled-out until the program's termination. Spooling allows several concurrent programs to "share" low-speed peripherals by simulating them on a disc.

This minimizes device competition among programs and helps to overcome the time differential between low speed peripherals and high speed processing. (See Figure 12.)

FILE CONCEPTS

In TSOS, all data entering, leaving and residing in the system is a file of some kind. For example, all of the following are files: the conventional input/output files used by data processing programs; source programs; object programs and subroutines; textual information to be organized and processed by the File Editor; and commands to be entered into the system.

Files can reside on private or public volumes. Files can also be marked as shareable or non-shareable. If a file is marked shareable, it can have "Read-Only" access or both "Read and Write" access. Control of files is maintained in the System Catalog and the user can specify, by means of entries in his Catalog command, how a file is to be accessed.

The system can also handle "Foreign" files. Foreign files are files that were not created by the current Data Management System with which the user is working. (See Figure 13.)

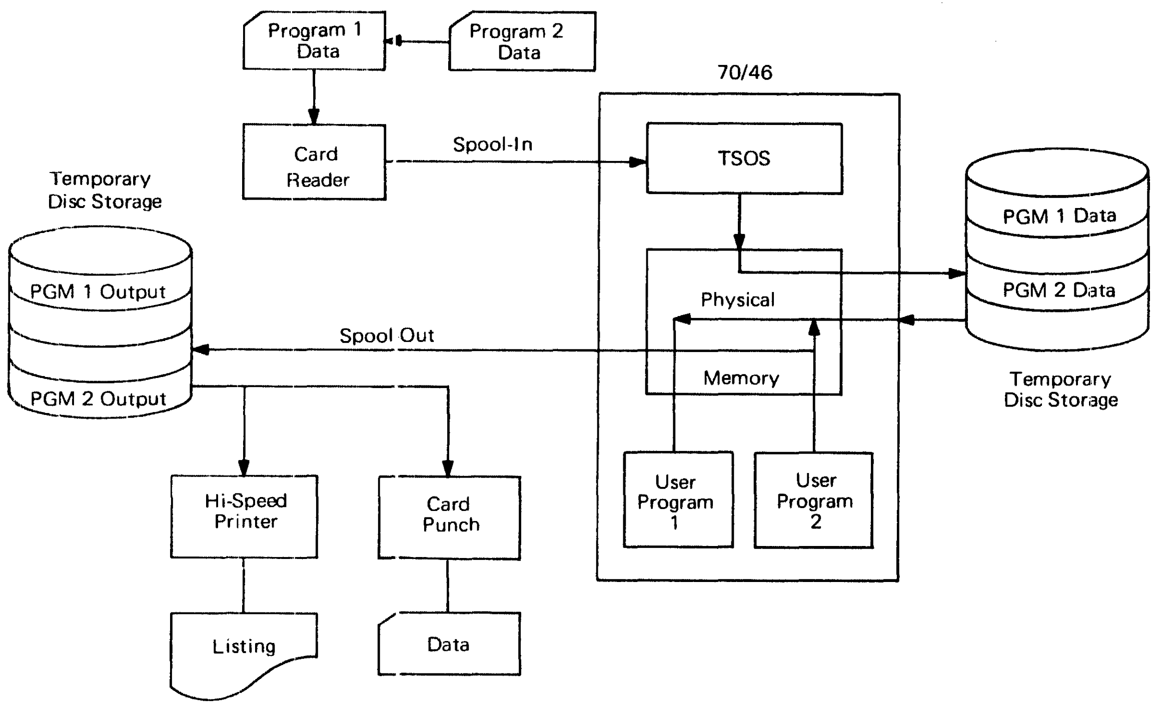


FIGURE 12. TSOS INPUT/OUTPUT SPOOLING

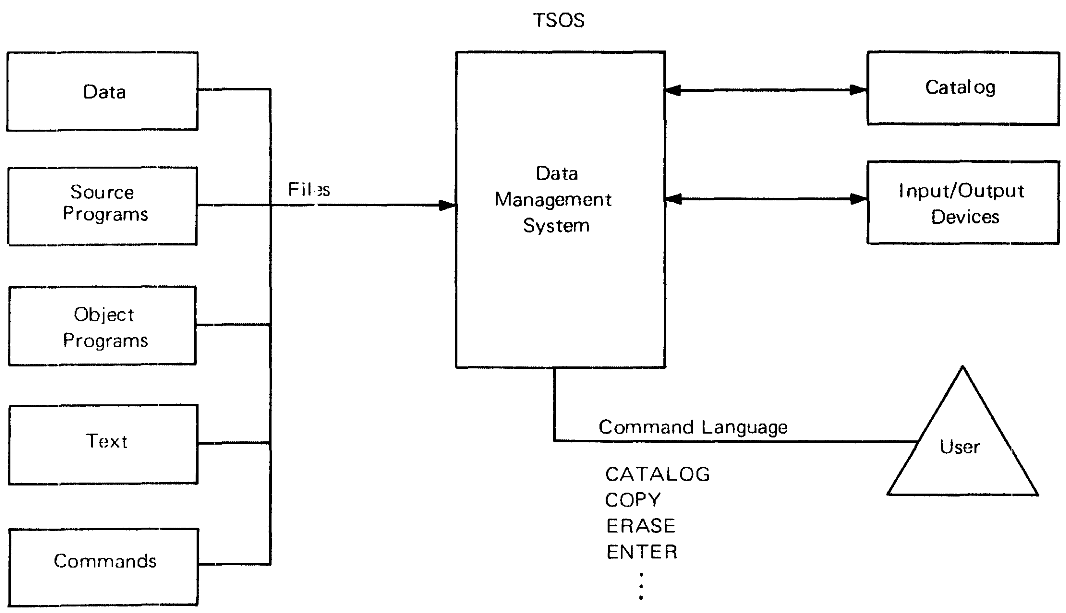


FIGURE 13. TSOS FILES

PROCEDURE FILES

In TSOS, a Procedure File is defined as a cataloged file containing a set of predefined operations (Commands). Data can also be stored in these files so that a command to execute a given program can be followed by the input for that program. At the user's option, the data for the program can be stored in another file and that file referenced in the set of predefined operations.

TSOS defines two types of Procedure Files: The ENTER File and the DO File. The major differences are in the File and Task delimiters, the File Activation Command, and how each is treated as a task. Table 2 summarizes the characteristics of each type of Procedure File.

TABLE 2. PROCEDURE FILE CHARACTERISTICS

	PROCEDURE FILES	
Characteristic	ENTER File	DO File
File Activation	The ENTER (filename) command.	The DO (filename) command.
Procedure file treated as separate task?	Yes.	No.
File delimiters	LOGON command, LOGOFF command.	PROC command, ENDP command.
Task delimiters	LOGON, LOGOFF commands in the Procedure file.	LOGON, LOGOFF commands in the initiating task.
Can be initiated from an Enter file?	Yes.	Yes.
Can be initiated from a terminal?	Yes.	Yes.
Can be initiated from a DO file?	Yes, but when the LOGOFF command is encountered control returns to the originally initiating task.	Yes, but when the ENDP command is encountered, control returns to the originally initiating task.
If the procedure is initiated from a terminal, does the terminal remain connected to the procedure?	No.	Yes.
Action that occurs due to terminal initiation.	The system types the task sequence number of the background task and returns a slash to the terminal. The user can then logoff if he desires, thus saving telephone charges, while the task processes in the background.	The initiated procedure is part of the initiating task. The system returns a slash when the Procedure is completed.

INTRODUCTION

The Command/Control language is the principal medium of communication between the user and the Time Sharing Operating System. The facilities of the Command language allow the user to construct, execute and debug his programs; to catalog, retrieve and manipulate his files; and to create his own procedures consisting of several commands and, optionally, data that can be named and called upon as required.

The user can use the Command language in the conversational (interactive) mode or the non-conversational (background) mode. In the conversational mode, the user maintains a dialogue with the system while it is executing operations for him. In the non-conversational mode, the Command language serves as a form of Job Control language since the user's requests are submitted for execution in prepared form and are not monitored by him.

Most of the commands have comparable macro-instructions that may be included in Assembly language programs to communicate with the system from the user's problem program.

USER CLASSES

TSOS identifies three distinct classes of user:

1. System Controller,

2. System Operator, and

3. Application User.

System Controller

The System Controller is a privileged user of the system and functions from a terminal usually located at the processor site. He is provided with special commands to aid him in initiating and controlling the other users of the system. In addition, he is a co-owner of all the files in the system. In conjunction with the System Operator, he is responsible for generating the specific system for a location and identifying the sharable portion of the system library. Besides his own special commands, the System Controller may use all of the other commands in the system. (See Figures 14 and 15.)

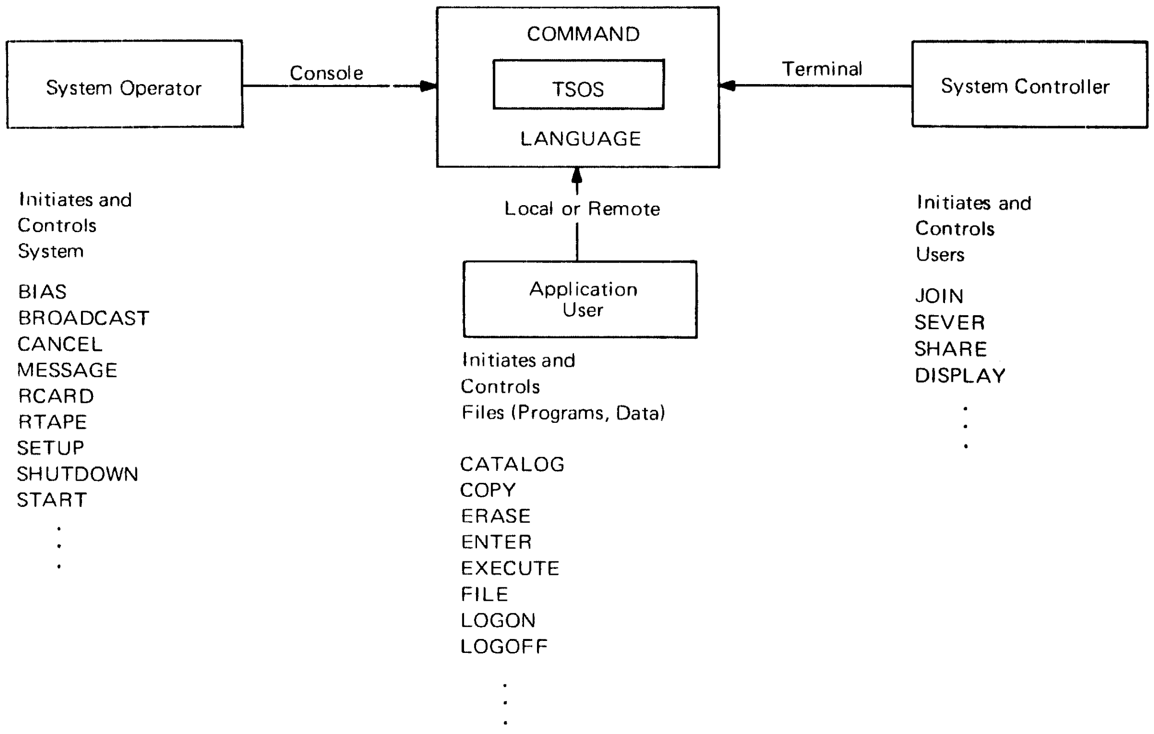


FIGURE 14. USER CLASSES AND COMMAND LANGUAGE

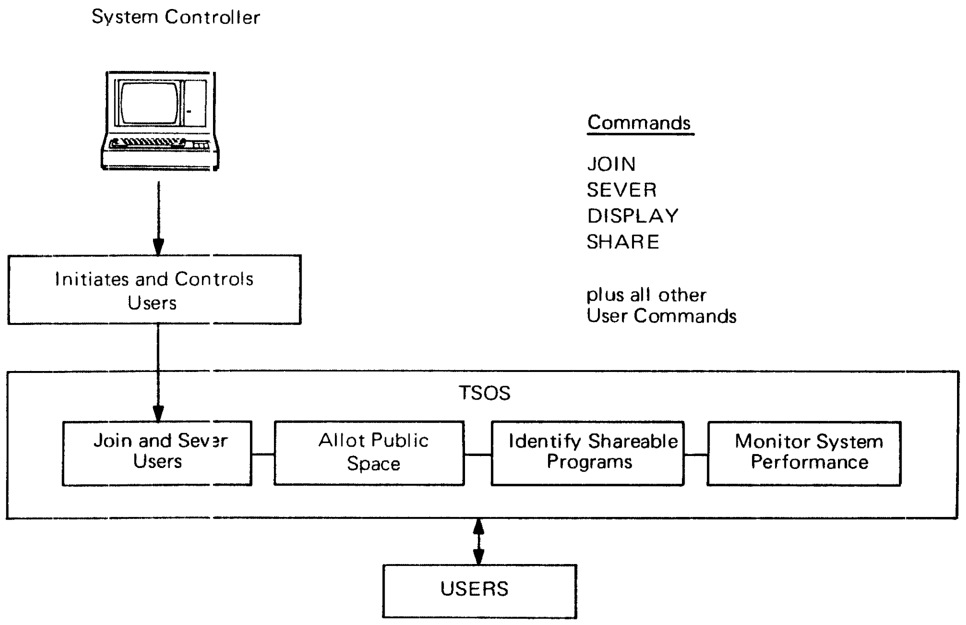

System Controller
Commands
JOIN
SEVER
DISPLAY
SHARE
plus all other User Commands
Initiates and Controls Users
TSOS
Join and Sever Users
Allot Public Space
Identify Shareable Programs
Monitor System Performance
USERS

FIGURE 15. TSOS SYSTEM CONTROLLER

System Operator

The System Operator is a privileged user of the system. He functions from the system console with a special set of commands that aid him in initiating and controlling the system resources. He can communicate with any or all interactive users to notify them of significant events regarding the system. (See Figure 16.)

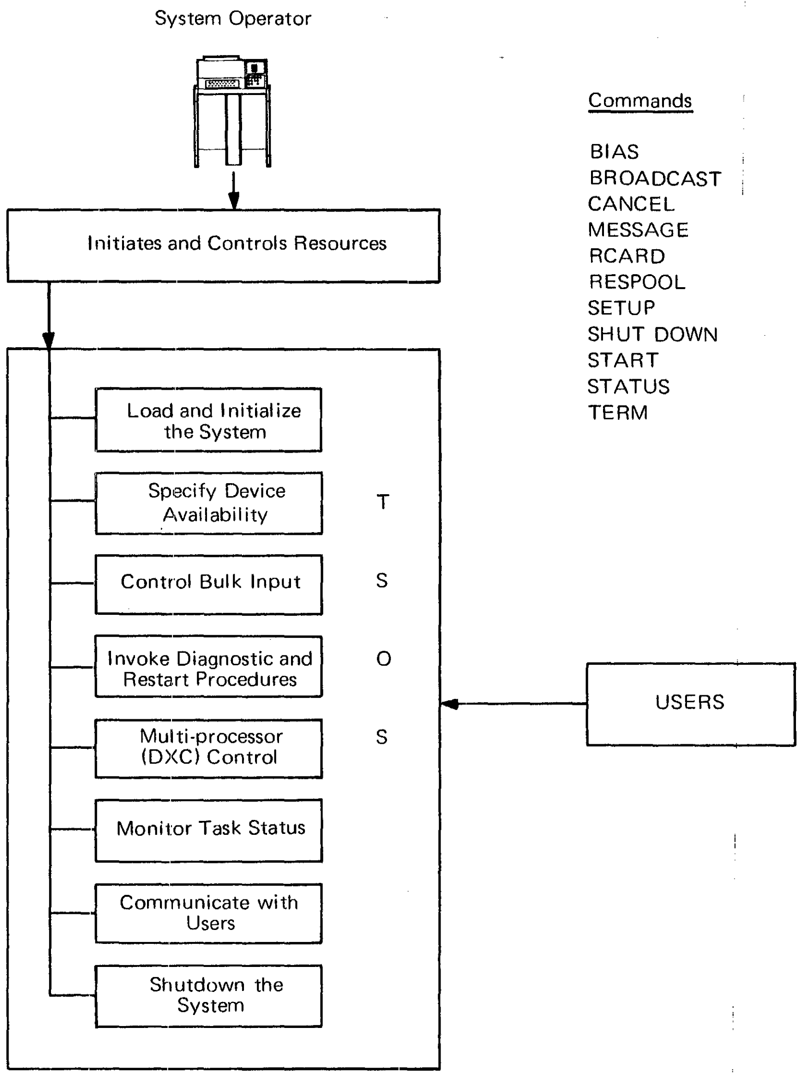


FIGURE 16. TSOS SYSTEM OPERATOR

Application User

The Application User has all of the TSOS commands available to him except those specifically reserved for the System Controller and the System Operator. He may enter these commands from a terminal, from a stored procedure file or from devices at the local installation.

The Application User may be in either of two categories:

1. Interactive (conversational) Terminal User or

2. Background (non-conversational) on-site user.

In addition, he may initiate batch jobs from a remote terminal and use a data terminal for remote batch entry.

Interactive User

Typical jobs for interactive users (see Figure 17) are

1. High level language (COBOL, FORTRAN) program syntax checking,

2. Interactive debugging,

3. Program correction, cataloging and execution,

4. Mathematical calculations,

5. Text handling,

6 Data Base Inquiry response information, or

7. Background job initiation from a remote terminal.

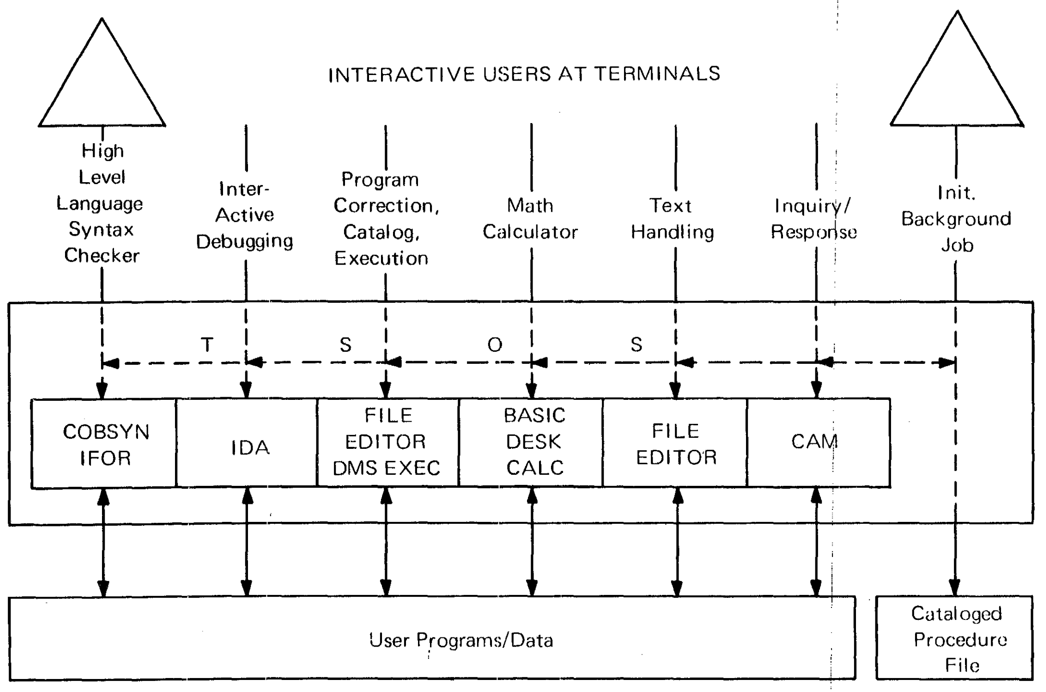
Background users (See Figure 18) are generally professional programmers who use the Command language in conjunction with the system Macro language to

1. Prepare programs in any of the several source language systems of TSOS,

2. Execute the resulting object programs,

3. Execute the service and utility routines such as the Linkage Editor, Sort/Merge and Library Maintenenace Programs and

4. Construct procedure files to act as Control Language job stream directors of multi-job processes.



SUPERVISOR COMMANDS			SUB-SET COMMANDS FOR:
Executive	Data Management	Interactive Debugging Aids	FILE EDITOR
LOGON	CATALOG	RESUME	BASIC
LOGOFF	FILE	AT	LIBR. MAINT.
ENTER	COPY	IF	LINKING LOADER
SECURE	ERASE	DUMP	CAM
⋮	CHANGE	DISPLAY	
	⋮	⋮	

FIGURE 17. TSOS INTERACTIVE USER

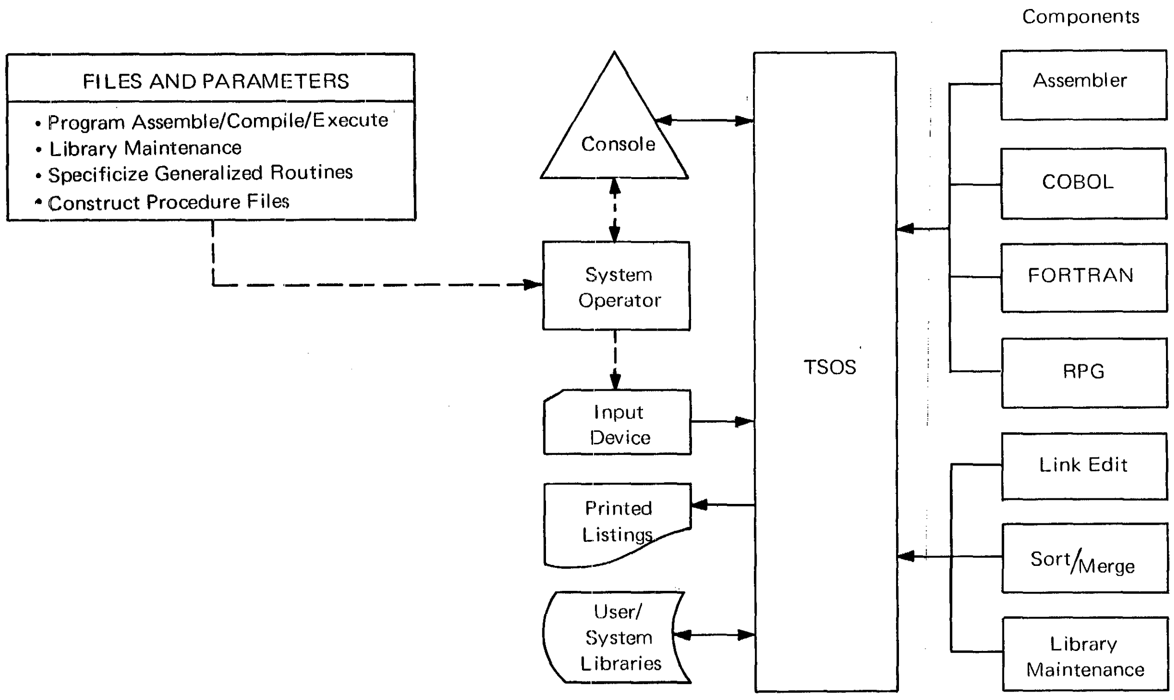


FIGURE 18. TSOS BACKGROUND USER

SYSTEM GENERATION

Certain steps must be accomplished before processing can begin; i.e., a system must initially be generated. Generation is the process of using the TSOS Master Tape (MASTAP) furnished by RCA and going through the steps of Basic System preparation and Control Program Generation. (See Figure 19.)

Basic System Generation initializes a disc designated as the system residence (SYSRES) volume to contain sufficient TSOS software to create the running control program and supporting software.

Control program generation runs under TSOS and reflects the local system hardware configuration as well as tailors the software to contain the program modules necessary for a particular installation.

Subsequent maintenance of the system is performed by a System Update program that runs under TSOS and is used to add or replace files on the discs from the new master tape.

JOINING USERS

Following system generation, the System Controller must JOIN the users to the system. He does this under TSOS by means of the JOIN Command. Only the System Controller may use this command. He obtains information from the various users desiring access to the system and enters a JOIN command for each.

The parameters of the JOIN command contain the following:

User-id - a one-to eight-character alphanumeric field which identifies the user to the system whenever be issues a LOGON command.

Password - used when the user desires an identification in addition to his user-id.

Account-No(s) - these are entries to be used for accounting purposes. Up to seven account numbers can be given for a single user.

Priority - a one-digit number specifying the highest priority permitted for any task submitted by this user; 1 is the highest, 9 the lowest.

Public-Space - indicates the maximum number of permanent space (tracks) that the user is allowed to allocate a public volumes.

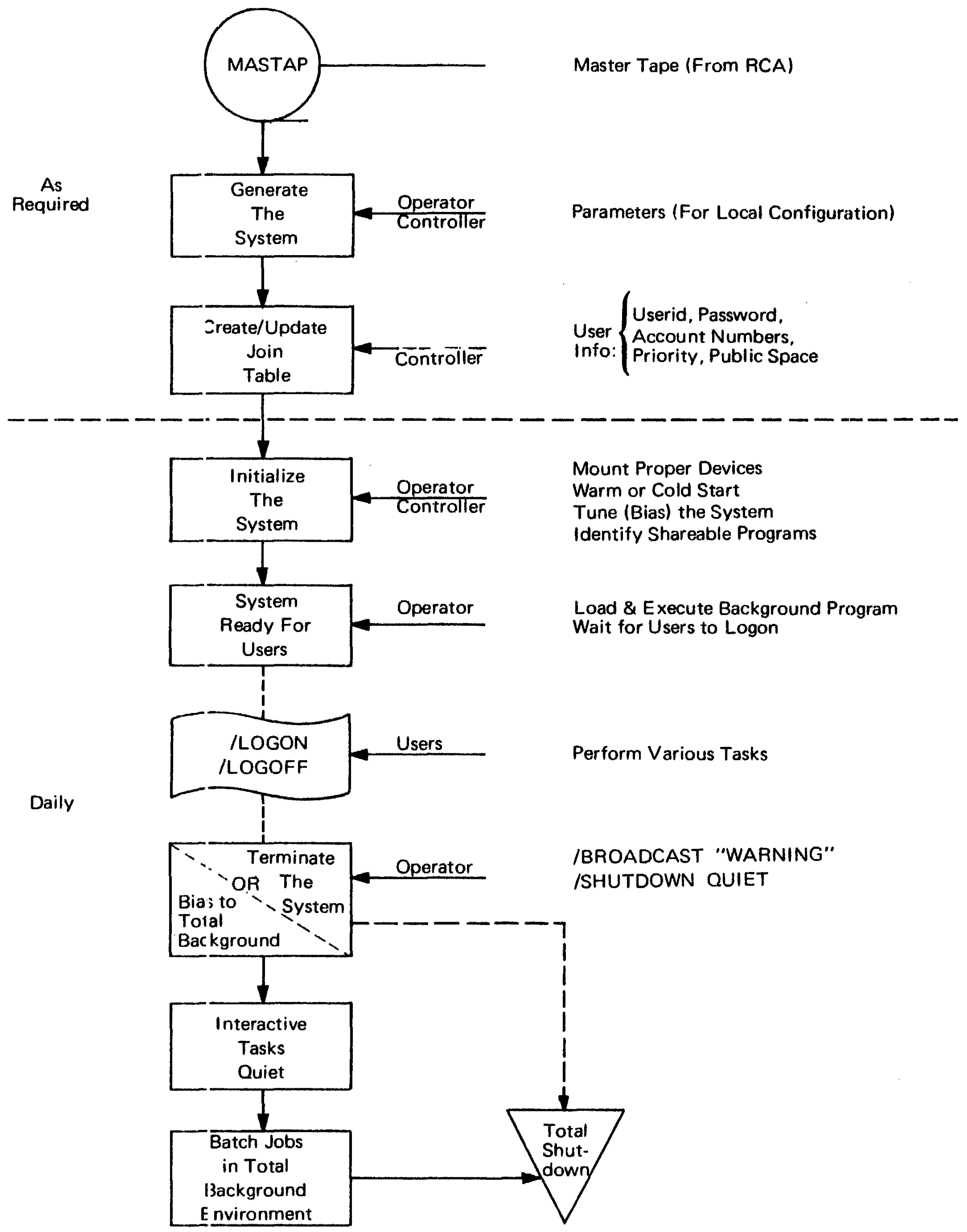


FIGURE 19. TSOS SYSTEM FLOW

Text - consists of a maximum of 32 characters which will be used to identify a mailing address for printer or punch output.

Any or all of the items associated with a user-id can be updated by the System Controller.

INITIALIZING THE SYSTEM

Once a system has been generated and the JOIN table entries created, it is ready for utilization by its many users. The operator mounts the proper volumes and establishes his BIAS requirements. The system asks him whether he wants a warm or cold start.

A warm start implies that he is picking up from where the system left off previously and tasks in the jobs-to-be-processed queue can now be completed.

A cold start assumes no tasks waiting in the jobs-to-be-processed queue and is equal to the initial session of the system.

After the system is started, the System Controller (or the System Operator) will ENTER a procedure file that will establish which modules (or libraries) in the system are to be made SHAREable. These programs have been reentrant coded and one copy will suffice for all users. A SHARE table is constructed from this input and is used throughout the time-sharing session.

Once the system is completely initialized, the operator will load at least one non-interactive job to be run while the interactive mode of the system is not busy. As each user dials up the system, he is asked to PLEASE LOGON and when his user-id, password, etc, have been verified, he is given a Task Sequence Number (TSN) and the system waits for directions (commands) from the user.

SUMMARY

The RCA Time Sharing Operating System is a virtual memory, time-sliced, interrupt-driven, remote-access computing system that offers many concurrent users the shared resources of a third generation data processing environment.

The users of the system employ Command language statements to direct the system in the performance of tasks. The system allocates a sequence number to each task and accounts for each users portion of the resources that he uses.

A supervisory group of Software programs manages the system resources as well as the Service, Utility and Application programs necessary for a total data processing function.

The system recognizes three general classes of users: the System Controller, the System Operator, and the Application user. Both the System Controller and System Operator are privileged. The Application user can function interactively from a remote terminal or on site in the local mode.

The many facilities available to each class of user are discussed in the following sections of this manual.

Figure 20 is a generalized schematic of the Time Sharing System. The external environment is represented by (1) the users and (9) the input/output devices of the system. The users communicate with a command/control language (2) which is interpreted and given to central control (3) for action. Depending on the command and the current circumstances (4) central control invokes the proper subcomponent (5), (6) or one of the service programs.

The various classes of interrupts (7) are analyzed and the task queues updated (8) depending on the priorities and availability of resources.

The entire function represents an integrated, advanced approach to attaining total system resource efficiency.

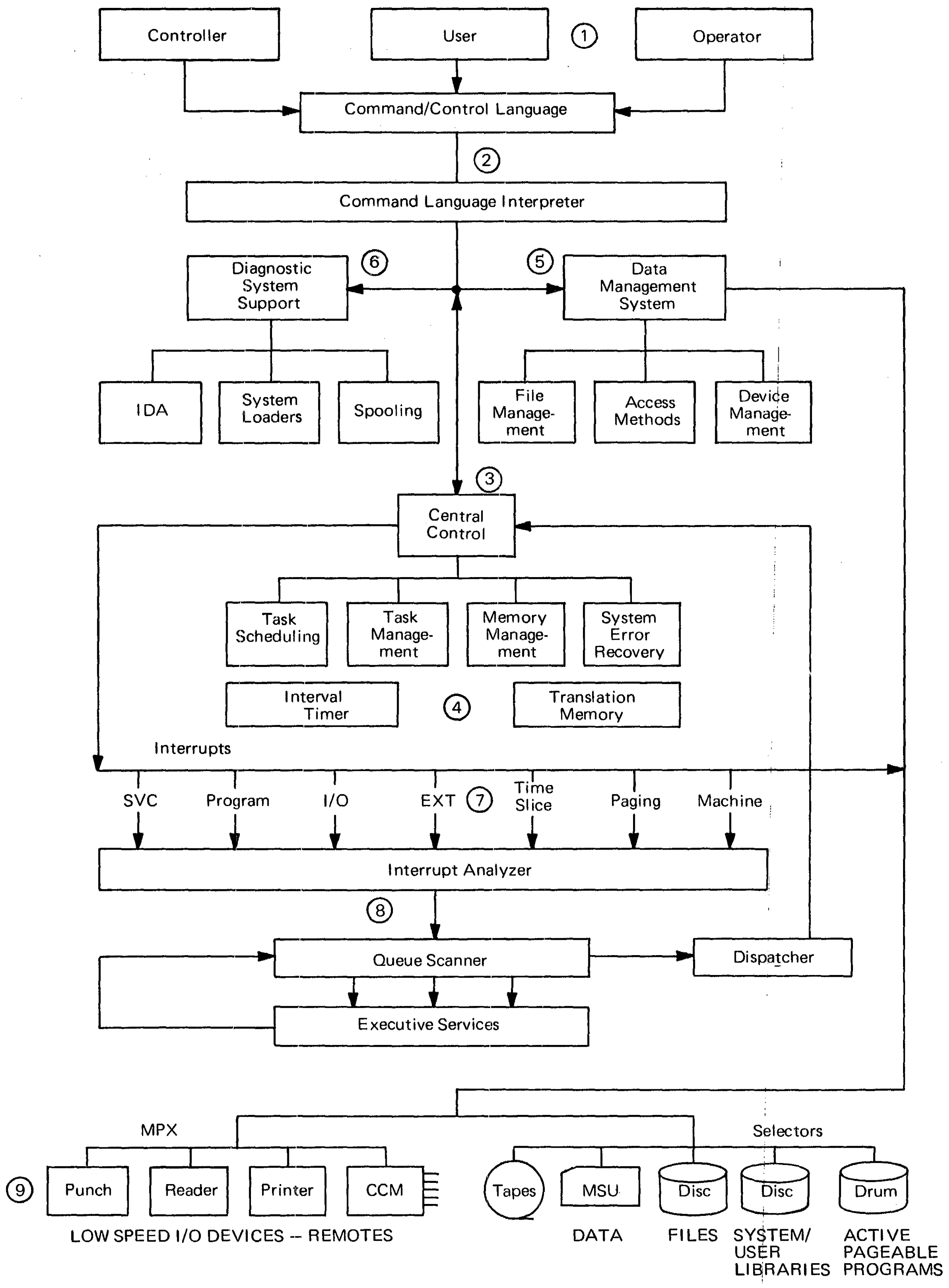


FIGURE 20. SCHEMATIC OF TSOS

Part 3

TSOS Equipment

Section 1: INTRODUCTION

GENERAL

The RCA 70/46 Time Sharing Operating System (TSOS) is designed to serve the batch processing and interactive user by utilizing the advanced features of the 70/46. These include hardware paging, special elementary operations, enhanced input/output capabilities, and all the interrupt-driven system capabilities of advanced third generation hardware.

DESCRIPTION OF TSOS HARDWARE

TSOS is a total systems design in that it represents an integration of time sharing hardware and software. A standard hardware configuration, as shown schematically in Figure 21, contains these time sharing hardware features:

A high-speed 70/567 drum memory unit (333KB data transfer rate),

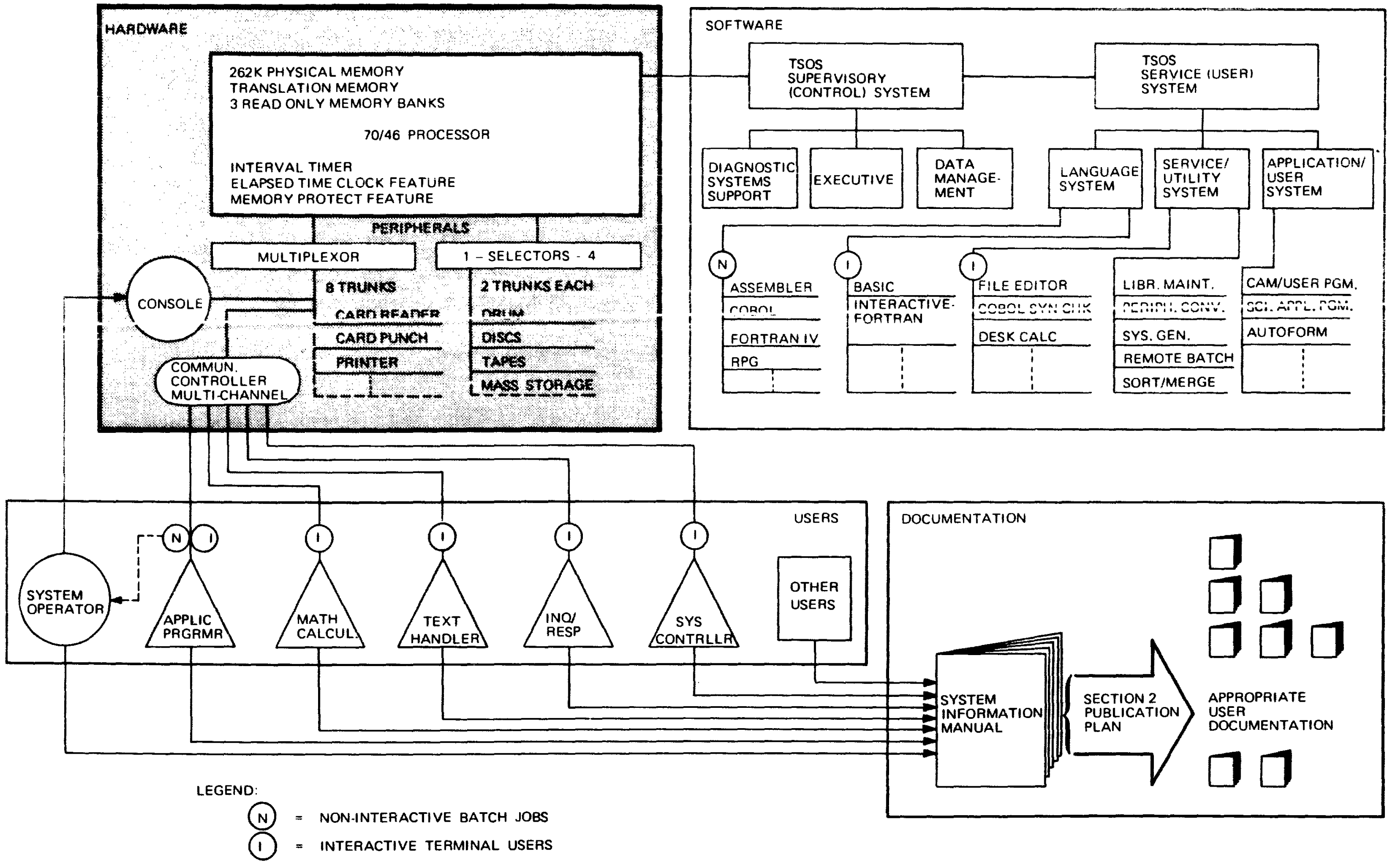
Fast translation memory (300 nanoseconds per halfword access),

Interval timer (100 microsecond interval),

Three banks of Read-Only Memory that provide special micrologic functions, and

System data rate that exceeds 1000 KB per second.

All instructions, character codes, interrupt facilities, formats, and programming features are functionally the same as corresponding features for the 70/35, 45, 55 Processors. In general, programs may be interchanged between the 70/46 Processor and the aforementioned Spectra Processors; provided system features are equivalent (including operating software) and the programs are independent of timing. Such interchanged programs are run on the 70/46 utilizing the 70/45 mode of processing.



OVERVIEW OF TSOS — HARDWARE

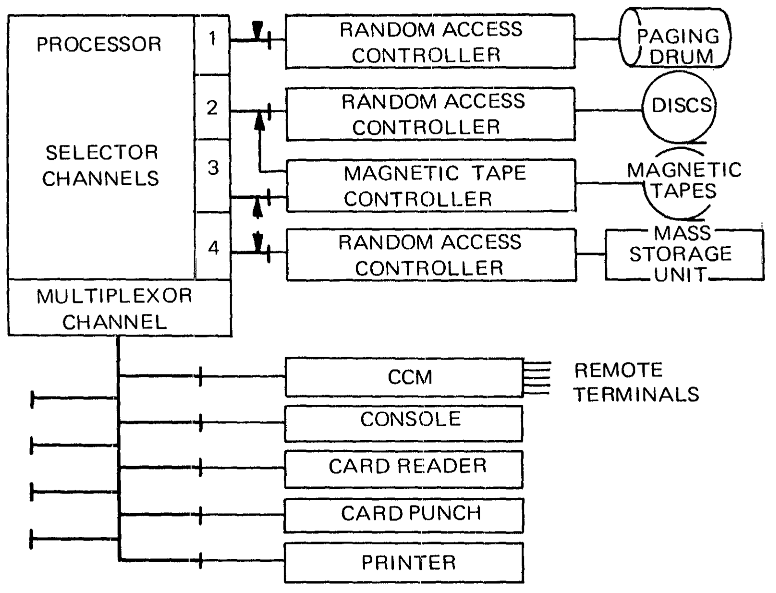


FIGURE 21. SCHEMATIC REPRESENTATION OF 70/46 CONFIGURATION

The RCA 70/46 Processor contains five memory components:

Main Memory

Non-Addressable Main Memory

Scratch-Pad Memory

Translation Memory

Read-Only Memory

Main Memory

Main Memory is the central storage (262K bytes) for both data to be processed and the controlling instructions.

Non-Addressable Main Memory

Non-Addressable Main Memory cannot be accessed by programming. It contains the subchannel registers that control the operation of input/output devices on the multiplexor channel.

Scratchpad Memory

Scratchpad Memory is a micromagnetic storage device consisting of 128 four-byte words and has an access time per word of 300 nanoseconds. Each word in scratchpad memory is uniquely addressable. Scratchpad memory provides general address registers, program counters, and the registers required for both interrupt analysis and input/output servicing.

Read-Only Memory

Three banks of read-only memory (ROM) are provided on the Model 70/46 Processor. Each ROM bank contains 2,048 54-bit words. The first ROM bank controls the elementary operations when in either 70/46 or 70/45 mode. The additional ROM banks provide specially wired micro-logic functions to assist and improve the efficiency of the TSOS executive control program.

Translation Memory

The translation memory is a magnetic storage device consisting of 512 halfwords (1,024 bytes) with an access time of 300 nanoseconds per halfword. Each halfword is uniquely addressed and contains a translation table element used in translating virtual addresses (high speed drum memory) to main memory addresses. The translation table is accessed by special micro instructions (elementary operations). Address translation does not require additional instruction time from that required by basic 70/45 timing. Translation memory is the controlling device for the TSOS paging and segmentation facilities.

PAGING AND SEGMENTATION

Paging provides the ability for a program to directly address more main memory space than is actually, physically available in the processor. The Spectra 70/46 uses special hardware and software to provide this capability.

The 70/46 main memory is divided into many blocks of equal size called pages. A 70/46 program can consist of many of these pages but, during any one execution stage, only those pages required for that execution stage need reside in main memory. The non-required pages are maintained in subsidiary storage. The 70/46 software relocates program pages dynamically within main memory such that programs are executable in different physical main memory pages. The 70/46 basic page size is 4,096 bytes.

A breakdown of virtual memory space is provided through segmentation. Segments are logical entities composed of groups of pages. There are potentially 32 virtual memory segments; each consisting of 64 virtual pages. However, only 8 virtual segments are implemented and currently provide a total of two million bytes of virtual storage.

A virtual address generated for paged programs has the following 24-bit format:

1 bit	5 bits	6 bits	12 bits
D	SEGMENT	PAGE	DISPLACEMENT

When in 70/46 mode, each memory address, except I/O execution and servicing, is translated if the D-bit within the address is zero. The translation process is performed by means of a "hardware table look-up" to obtain main memory addresses. Essentially, the Page and Displacement fields compose the 18-bit main memory address.

SPECTRA 70/46 PROCESSOR	PHYSICAL MEMORY 262 K BYTES 1.44 u SEC/2 BYTES NON-ADDRESSABLE PHYSICAL MEMORY SCRATCH PAD MEMORY 128 WORDS INTERVAL TIMER ELAPSED TIME CLOCK PROGRAM CONTROL & ARITHMETIC UNIT READ ONLY MEMORY BANKS 2048 56-BIT WORDS EACH 1 2 3 SELECTOR CHANNELS 1 2 3 4 MULTIPLEXOR CHANNEL 1 2 3 4 5 6 7 8 TRANSLATION MEMORY 512 HALF WORDS

FIGURE 22. 70/46 PROCESSOR COMPONENTS

SUMMARY

The 70/46 hardware environment represents a modular, expandable system the user can configure to suit the needs of his application.

A sample configuration, with typical installation mnemonics associated with each device, is shown in Figure 23.

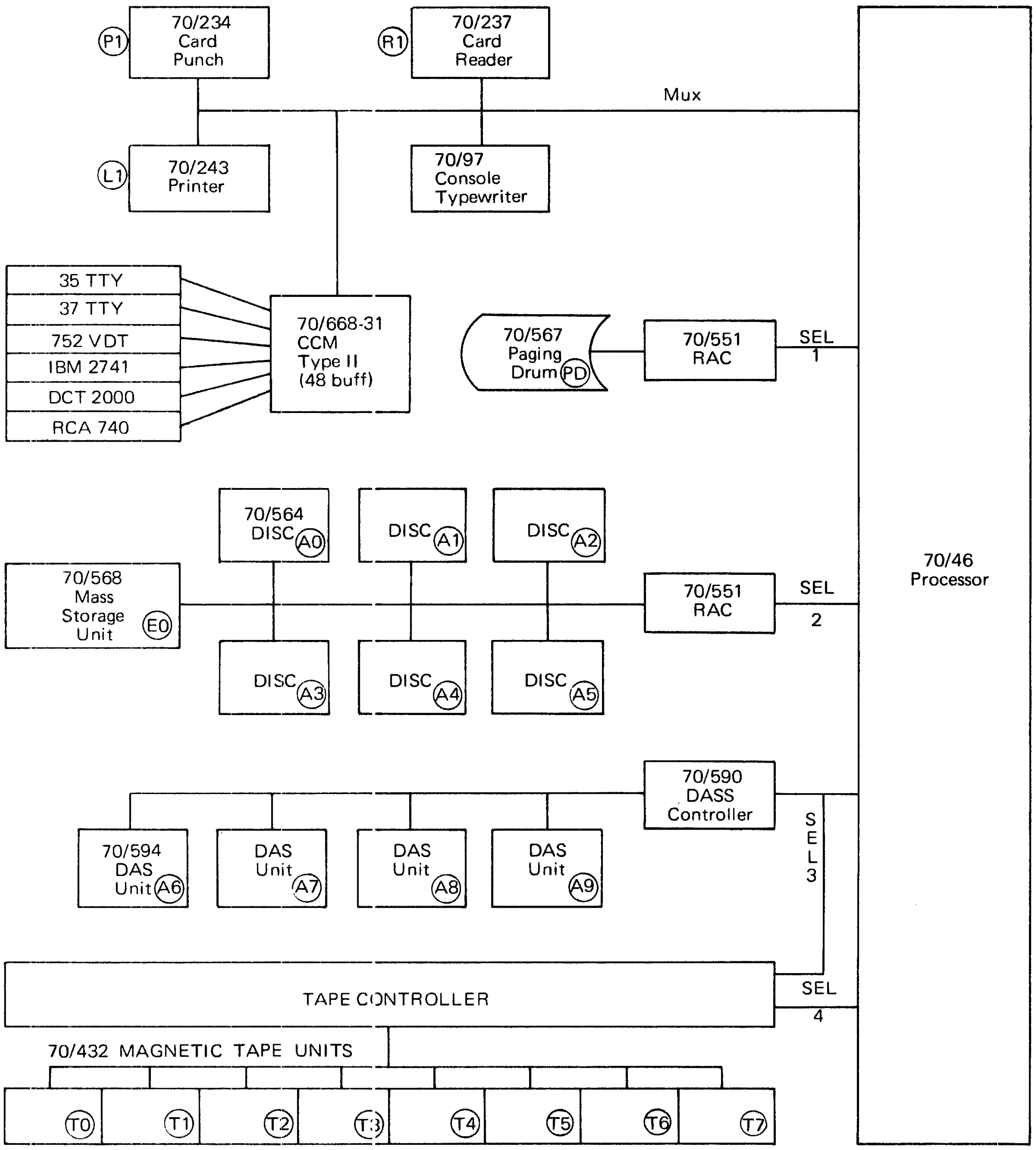


FIGURE 23. SAMPLE 70/46 CONFIGURATION

Part 4

Time Sharing Services Under TSOS

Section 1: INTRODUCTION

GENERAL

The Time Sharing Operating System (TSOS) software system provides direct-access facilities for interactive time sharing and remote batch processing for many simultaneous users at remote devices while concurrently supporting multiprogrammed local batch processing, multiple language translation, and many routine service functions.

TSOS supports this simultaneous task execution with two modes. In the conversational mode, the user communicates interactively with the system from terminals during program preparation and execution.

In the non-conversational mode, the user completely defines his task before commencing task execution and there is no dialogue between user and system during the acutal program execution.

However, a non-conversational task can still be optionally initiated through the conversational mode from a remote terminal. Regardless of mode, a single command language which includes job control information is used to interface all users with the system.

Up to 64 concurrent tasks may be executed in this environment, limited only by such factors as the desired response time, the nature of the program mix, and the availability of installation peripheral devices and remote terminals.

Of the 64 tasks, up to 48 can be interactive terminal programs and up to 16 can be background batch-processing type programs. Resources for this many jobs may not always be available because of the program mix. However, the system has been structured to schedule and execute as many tasks as the available resources will allow.

TSOS completely manages all activities associated with the virtual memory, eliminating from programmer concern all functions related to the paging algorithm. The system also creates temporary disc files for conventional card I/O and printer-destined output; these files are automatically spooled to and from the I/O devices when the tasks are initiated or have been completed.

By simulating the presence of several card and printer devices, the system allows many programs to do pseudo I/O on the same device concurrently. This eliminates the system degradation associated with both securing I/O devices and in using on-line I/O peripherals.

The entire operating system resides on one on-line disc. To start a session, the physical memory resident portion of TSOS (approximately 40KB) is boot-strapped to physical memory where it gains control and places the drum resident software onto the drum.

The non-pageable TSOS software will always remain on disc and be called directly from disc only when required (e.g., Class I language processors).

TSOS SOFTWARE SYSTEM

TSOS Software is structured (see Figure 24) into two general groups of programs:

1. Supervisory (Control) System Group
2. Service (User) System Group

The Supervisory (Control) Group runs in the privileged modes of the processor and cannot be directly accessed by the problem programmer.

The Service (User) Group runs in the non-privileged mode of the processor and is directed by the Supervisory Group to interface with the various users of the system.

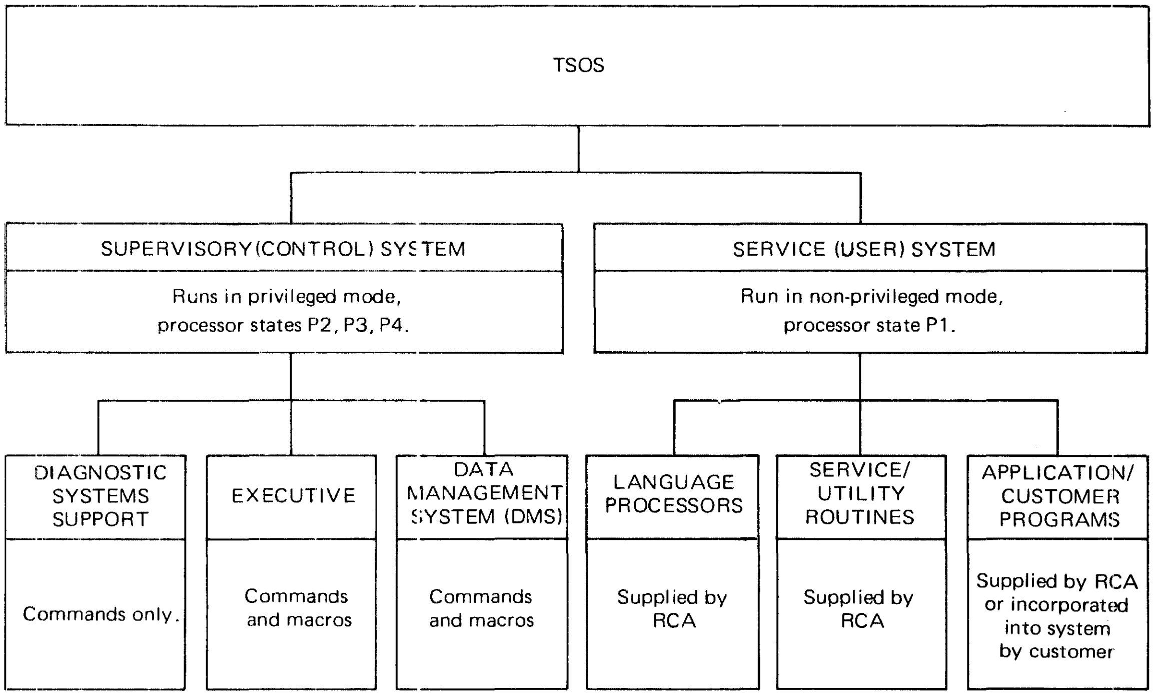
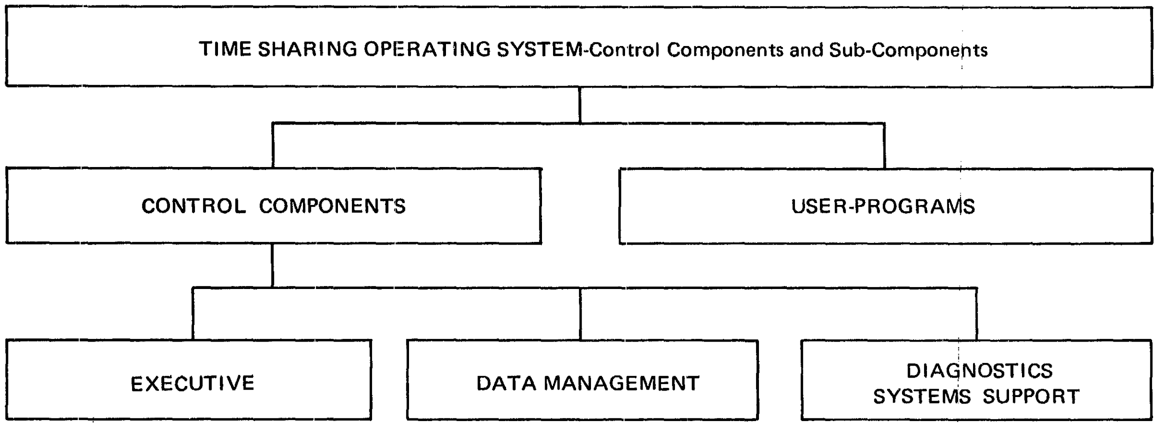

TSOS
SUPERVISORY(CONTROL) SYSTEM
Runs in privileged mode, processor states P2, P3, P4.
SERVICE (USER) SYSTEM
Run in non-privileged mode, processor state P1.
DIAGNOSTIC SYSTEMS SUPPORT
Commands only.
EXECUTIVE
Commands and macros
DATA MANAGEMENT SYSTEM (DMS)
Commands and macros
LANGUAGE PROCESSORS
Supplied by RCA
SERVICE/ UTILITY ROUTINES
Supplied by RCA
APPLICATION/ CUSTOMER PROGRAMS
Supplied by RCA or incorporated into system by customer

FIGURE 24. TSOS SOFTWARE STRUCTURE

Supervisory Programs

The TSOS Supervisory Group (see Figure 25) of programs consists of the Executive, the Data Management System and Diagnostic Systems Support.


TIME SHARING OPERATING SYSTEM-Control Components and Sub-Components
CONTROL COMPONENTS
USER-PROGRAMS
EXECUTIVE
DATA MANAGEMENT
DIAGNOSTICS SYSTEMS SUPPORT

EXECUTIVE

- Remote Terminal Input/Output (RTIO)
  - Normal
  - Communications Access Method (CAM)
  - Remote Batch Processing (RBP)
- DXC
- Central Control
  - System Error Recovery
  - Task Scheduler
  - Task Management
  - Memory Management
- Remote Batch Processing – Spoolin
- Program Control
  - Program Management
  - TOS/TSOS Macros
- Job Control
  - Accounting
  - System File Management
  - Job Controller
- Input/Output Control
  - Console
  - Devices
- Command Language Processing
  - Terminal Controller
  - Command Processing

DATA MANAGEMENT

- File Management
  - Catalog Management
  - Command Processing
  - Allocation
  - Macro Processing
- General Service Macros
  - File Central Block (FCB)
  - JDFCB
  - OPEN
  - CLOSE
- Miscellaneous Macros
- Access Methods
  - SAM
  - ISAM
  - PAM
  - BTAM
  - EAM
- Device Management

DIAGNOSTICS SYSTEMS SUPPORT

- Interactive Debugging Aids (IDA)
- Spoolout
- System Loaders
- Desk Calculator and EO's
- Interpretive Scanner & Processor (ISP)
- Message Processor

FIGURE 25. TSOS CONTROL COMPONENTS

TSOS Executive

The TSOS Executive (see Figure 26) manages the total operating system environment. It serves as the interface among all users of the system, the computer hardware, and all remaining system software. As far as the user is concerned, the Executive appears to be an integral part of the computer.

The Executive's most frequently-used components are permanently resident in physical memory; other routines are quickly available from the paging drum. The Executive's locations are not addressable by other programs, it executes in the privileged mode and is not itself generally time-sliced.

The Executive receives interrupts, interprets them, sorts them as to type and function, and initiates the appropriate routines to respond to each. It spools all tasks introduced to the 70/46 to a common task queue.

By means of time slicing, it provides rapid and complete service for all these tasks, up to the maximum system resource level. The Executive routines control command language processing, system error analysis and recovery functions, job control and system file management, task and memory management, CPU time, the console and the paging drum, and peripheral device error recovery.

To facilitate scheduling in the system and to insure optimum processor utilization, a task schedule queue structure has been established in TSOS consisting of thirteen separate queues.

Tasks are placed in appropriate queues according to their needs and conditions and moved among the queues according to the algorithm of the task scheduler.

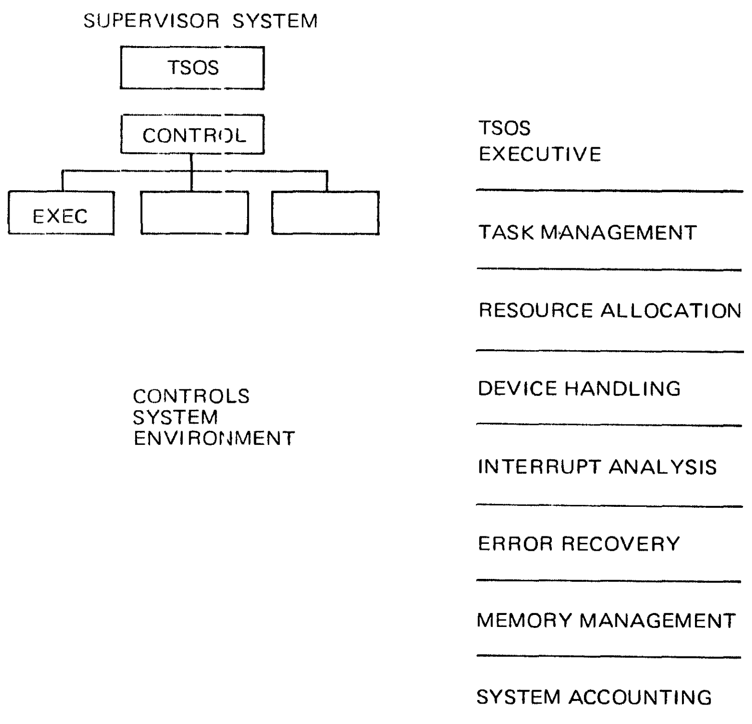


FIGURE 26. TSOS EXECUTIVE

Ready Active

1. Processor

   Interrupt driven tasks

   Terminal and non-terminal attached tasks

2. Processor

   Active ready resident (Class I) tasks

3. Paging routine

4. Active I/O waited tasks

Ready Inactive

5. Tasks with terminal responses just received

6. Tasks with I/O completed

7. Full-time slice tasks (computerbound)

8. Inactive ready resident (Class I) tasks

9. Time-slice runout tasks – Related to 7. above but separated to insure that a terminal attached computerbound task does not monopolize the processor.

Non-Ready Inactive

10. New tasks not yet fully set up for processing

11. System tasks

12. Inactive I/O waited tasks

13. System or interrupt-driven tasks temporarily passed.

To insure prompt response to terminal users, the Executive gives Class II interactive tasks top priority and schedules them on a first-in, first-out basis among the various queues.

Tasks are entered into the system set up for execution and, according to preestablished priorities assigned to each user by the System Controller, task processing is begun.

A Class II (Non-Resident) task normally enters the paging queue and after the required page has been placed in main memory, the task enters the processor queue. The Executive normally allows a task, which has gained control of the processor, to proceed with execution until it waits itself due to I/O requests, paging needs or a time slice runout. When the Class II processor queue is empty, control of the processor is given to active-ready resident (Class I) tasks.

Processor utilization is considerably increased by always having at least one active-ready resident task available in the system. Such a task will automatically use processor time that is not usable by Class II programs due to paging or I/O waits.

The Executive controls the allocation and utilization of virtual memory, physical memory, and the paging drum backing memory. Although most of this memory management is invisible to the user, it provides the facilities which allow each Class II task in the system to operate as if it had at its disposal up to 256 pages (1,048,576 bytes) of core memory. These pages are contiguously addressed so that the Virtual Memory of each user contains address 0 to 1,048,575.

The upper limit of this virtual memory is dynamically adjusted by the Executive during each task's execution depending on the memory requirements of the task. The information contained in each user's virtual memory is automatically placed into and removed from core storage as needed by the Executive paging algorithm without user knowledge or intervention.

The Executive collects over 20 categories of accounting statistics for each task during its execution. This accounting file is maintained on an on-line disc which may be interrogated by all users of the system.

The key to this file is a unique Task Sequence Number (TSN), which is assigned to each task as it enters the system. The TSN for each job is immediately communicated to the user as part of the Log-on dialogue which identifies the user to the system and permits use of the system by each user.

For each task, the accounting file contains (in addition to the TSN) the user identification code, charge account number, log-on and log-off times, spooling times, type of task, accumulated processor time, amount of storage used by type of storage, device usage data by device type, and task completion data.

A user can obtain the current status of his task from the accounting file. Using his TSN he may inquire if his job is still in queue, in execution, has been completed, or was abnormally terminated.

An additional facility, the Hard Copy Option, allows a user, on a video display terminal, to retain a copy of all of his terminal input and output information when he LOGSOFF. Exercising the options causes information read and written to be spooled out to the SYSLST File when the user's task is finished.

The System Controller can use this file for system control and scheduling. He can use the resources usage data to determine total system use by type of user, to anticipate problem areas, such as proper equipment configuration, and to make loading and resource allocation decisions.

Accounting operations can use the statistics file for cost analysis and cost center accounting.

The System Controller has an additional facility, the BULLETIN file, to automatically disseminate installation information to people using the system. Individual bulletins (normally 80-bytes) are created by the controller for this file and automatically sent to the user after his LOGON is accepted. If longer bulletins were created, the user may answer Y to a Continue (Y,N) printout and receive continuations. The non-conversational user only receives the first 80-bytes on his SYSOUT listing.

INTRODUCTION

Users communicate with the system through the Executive Command Language Processor routine.

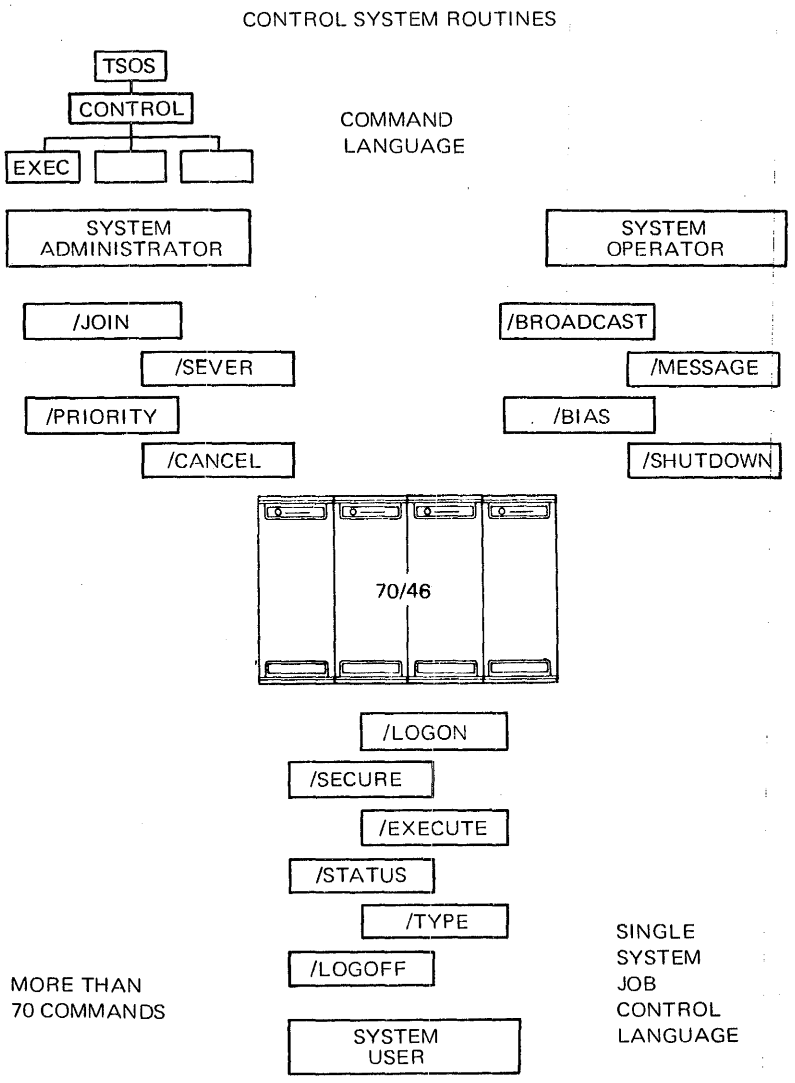


FIGURE 27. CONTROL SYSTEM ROUTINES — COMMAND LANGUAGE

There are over 70 commands which can be issued directly from remote terminals or central computer installation peripherals or indirectly from named and stored procedure files.

TSOS essentially has only one Command language including all job control information for servicing both the batch and the interactive modes. A single command task stream is accumulated from all types of input sources for all types of tasks: the Executive schedules this queue of commands by calling out the respective Control System Service Routines.

This repertoire of commands is divided into three classes to service the specific needs of the three classes of users who need to communicate with TSOS.

One class serves the needs of a System Controller who monitors and controls system use and resources.

A second set of commands is for the System Operator – the computer installation housekeeper.

A third set of commands is, of course, for the interactive and background users of the system.

When granted access to the system, each individual is assigned to one or more of these classes which in turn determines the commands he is authorized to issue. Certain controller and operator commands are privileged and cannot be issued by other users.

SYSTEM CONTROLLER COMMANDS

The System Controller's commands enable him to control the use of the system. He uses a JOIN command to authorize user access to the system and a SEVER command to remove a user from the system. The Controller uses commands to alter the PRIORITY of tasks in the system or to CANCEL tasks waiting for task scheduling or to terminate tasks already in execution.

SYSTEM OPERATOR COMMANDS

The System Operator's commands enable him to operate the system to best serve the needs of all users. He uses a BIAS command to dynamically "tune" the resources of the system to favor either Class I or Class II task execution.

He has a BROADCAST command which allows him to send messages to all terminal users currently active in the system, and a MESSAGE command to communicate with one specific user's terminal. The operator has commands which perform the various housekeeping chores required to configure the system to satisfy current operating demands, including SHUTDOWN of the system.

USER COMMANDS

The users of the system use their commands to gain simple and convenient access to the system, to manage their tasks, and to manipulate their data fields. They identify themselves to the system with a LOGON command, request EXECUTION of their problem programs and/or the RCA-supplied service programs, and terminate their tasks with a LOGOFF command.

The users can use the STATUS command to obtain information on the status of their tasks; the SECURE command to reserve the resources that their task execution will require; and the TYPE command to print a message on the system operator's console typewriter.

Conversational

Commands for conversational tasks are typically entered from a terminal keyboard. Commands for non-conversational tasks can be contained in a card deck or a magnetic tape file supplied to the central computer installation and initiated by the System Operator by using his RCARD or RTAPE commands to spool-in the tasks for scheduling and execution.

Non-Conversational

Non-conversational tasks can also be scheduled for execution by the ENTER command; in this case this command specifies that entry must be made to a cataloged file which contains the commands for a complete task.

When the ENTER command is used, it places a non-conversational task into the job stream, thus effecting a resequencing of the tasks in the job stream. When resources become available, the new task is executed.

The user must have previously completely defined the task in a cataloged file (including LOGON and LOGOFF commands). After having issued an ENTER command, the user is free to continue with other work since no transfer of control takes place.

Furthermore, both conversational and non-conversational tasks may obtain commands from a procedure file by issuing a SYSFILE filename command to identify a filename which contains a series of commands.

The ENTER filename command is the means by which the user directs the system to begin accepting commands from other than the current command input source. When the system receives the ENTER filename command, it opens the cataloged file which has this name.

The first record in this file is a PROCEDURE command which serves as the starting point for this temporary command file. An ENDP command is used in the procedure file to return control to the primary command input source. The procedure file technique can be used as a time saving device in those cases where a constant series of commands are frequently needed to describe work for the system. The command language thus offers the user considerable flexibility in its numerous functional uses and in its many operational operating environments.

The language is designed for introducing work to the system in both the conversational mode and non-conversational mode for all types of system users.

Table 3 summarizes the TSOS Command Language and control components.

TABLE 3. TSOS COMMAND LANGUAGE and CONTROL COMPONENTS

User	Command	Control Component
Controller	CANCEL	EXEC
	JOIN	EXEC
	PRIORITY	EXEC
	SEVER	EXEC
	STATUS	EXEC
	SHARE	DSS
	(Plus all other commands)	
Operator	BIAS	EXEC
	BROADCAST	EXEC
	CANCEL	EXEC
	MESSAGE	EXEC
	PRIORITY	EXEC
	RCARD	EXEC
	RESPOOL	EXEC
	SETUP	DMS
	SHUTDOWN	EXEC
	START	EXEC (DXC)
	STATUS	EXEC
	TERM	EXEC (DXC)
General User	AT	IDA
	BREAK	EXEC
	CALC	EXEC
	CATALOG	DMS

(Continued)

TABLE 3. TSOS COMMAND LANGUAGE AND CONTROL COMPONENTS (Continued)

User	Command	Control Component
General User (Cont'd)	CHANGE	DMS
	CONNECT	EXEC (CAM)
	COPY	DMS
	DATA	EXEC
	DISPLAY	IDA
	DO	EXEC
	DROP	DMS
	DUMP	IDA
	END	EXEC
	ENDP	EXEC
	ENTER	EXEC
	EOF	EXEC
	ERASE	DMS
	EXECUTE	EXEC
	FILE	DMS
	FSTATUS	DMS
	HOLD	DMS
	IF	IDA
	INTR	EXEC
	LOAD	EXEC
	LOGOFF	EXEC
	LOGON	EXEC
	MOVE	IDA
	PARAMETER	EXEC
	PASSWORD	DMS
	PAUSE	EXEC
	PRINT	EXEC
	PROCEDURE	EXEC

(Continued)

TABLE 3. TSOS COMMAND LANGUAGE AND CONTROL COMPONENTS (Continued)

User	Command	Control Component
General User (Cont'd)	PROPAGATE	IDA
	PUNCH	EXEC
	QUALIFY	IDA
	RELEASE	DMS
	REMARK	EXEC
	REMOVE	IDA
	RESUME	IDA
	RJOB	EXEC (RBP)
	RLOGOFF	EXEC (RBP)
	RLOGON	EXEC (RBP)
	RMSG	EXEC (RBP)
	ROUT	EXEC (RBP)
	RSTART	EXEC (RBP)
	RSTATUS	EXEC (RBP)
	SECURE	EXEC
	SET	IDA
	SETSW	EXEC
	SKIP	EXEC
	STATUS	EXEC
	STEP	EXEC
	STOP	IDA
	SYSFILE	EXEC
	TYPE	EXEC

Component abbreviations:

EXEC = Executive

DSS = Diagnostic Systems Support

DXC = Data Exchange Control

DMS = Data Management System

IDA = Interactive Debugging Aids

RBP = Remote Batch Processing

INTRODUCTION

The Data Management System (DMS) is the major element of the TSOS software that the Executive calls on to catalog, organize, and process all files in the system. (See Figure 28.) The DMS facilities fall into two major categories:

1. File Management, and
2. Problem Program I/O.

File management facilities provide for identifying files, for storing and retrieving, for copying, modifying and erasing them, and for defining their existence in the system.

Problem program I/O facilities provide for the actual transfer of data to and from user programs which are being executed in the system.

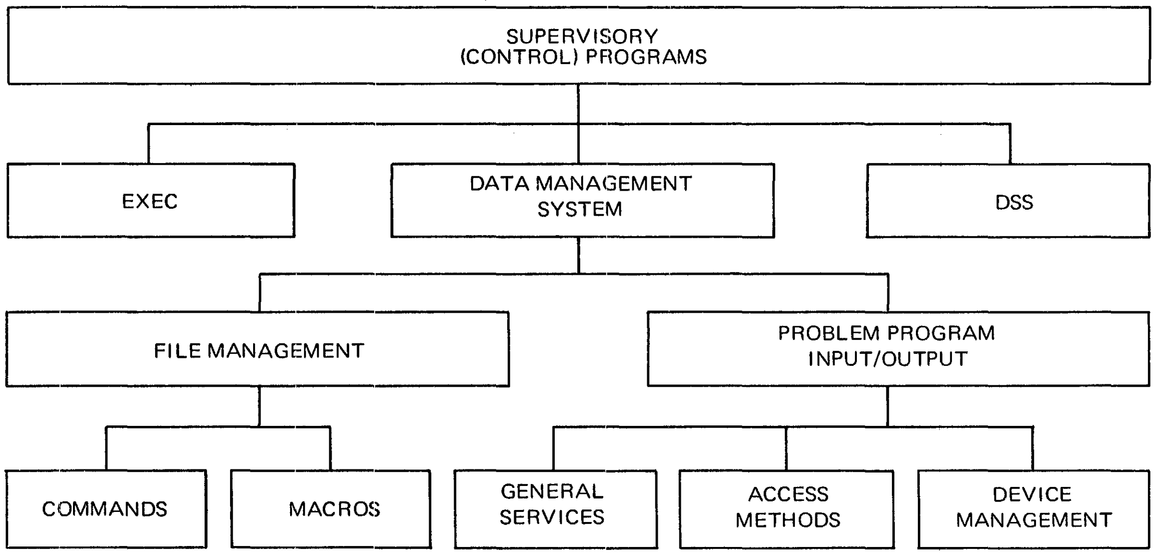


FIGURE 28. DATA MANAGEMENT SYSTEM COMPONENTS

FILE MANAGEMENT

In TSOS, a file is a named collection of related records. A file can consist of conventional I/O data, such as an array, inventory records, customer accounts, etc., source program code, object program code, subroutines, or textual information. These files, though meaningful to the creator, are only meaningful to TSOS when the user defines to the system how he wants each specific file cataloged and processed.

Each user assigns one or more simple names to identify each of his files to the system. When multiple names are used, the names, starting from the left, name a category which includes all names to the right as subcategories. This structure facilitates the storage and retrieval of file segments based only on the filename.

A fully qualified name identifies an individual file and includes all of the simple names. A partially qualified file name identifies a file segment by omitting one or more of the rightmost simple names.

Files in the system reside on physical storage devices referred to as volumes. A volume may be a removable disc pack, a reel of magnetic tape, or a mass storage magazine.

A file is said to be stored in the system if it resides on one or more of these direct-access or magnetic tape volumes, and if the identification of these volumes (volume serial number) is available in the system catalog.

VOLUME MANAGEMENT

Volumes are classified in TSOS as either public or private. A public volume is a direct-access device which may be used by many tasks concurrently and must be on-line during the entire period of System operation.

The use of a private volume is restricted to one task at a time and, therefore, need only be mounted when that task refers to it. Direct access volumes may be private or public, while magnetic tape volumes are always classified as private.

The System Controller specifies the maximum amount of space on public volumes which each user may acquire. This space is not actually allocated to the user until it is required, and is referred to as the user's public space allotment. System work files created on behalf of a user are stored on public volumes, but the space occupied by these system work files is not charged to the user's public space allotment.

If, in the process of inserting or adding records to a file, the space allotment is exhausted, the system transfers control to an address specified by the user's problem program.

His program can then take appropriate action; e.g., make space available by deleting an old file or move an inactive file to the mass storage unit or a private volume.

TSOS supports the 70/568 Mass Storage Unit as a private volume and allows the user to transcribe files between a mass storage unit and disc volumes.

The on-line storage available in the system can thus be greatly increased by use of the much cheaper cost-per-bit Mass Storage Unit.

The COPY command transcribes a file from a Mass Storage Unit to a disc volume or from a disc volume to a Mass Storage Unit.

This command provides the user with a flexible means for managing his file allocation on public volumes. When his disc file storage approaches exhaustion, a user can issue the COPY command to transcribe a file to a Mass Storage Unit.

He can then issue an ERASE command to release the space occupied by that file and thereby reclaim it for other purposes. These commands provide a convenient means for using the mass storage device as a secondary storage medium for inactive or infrequently used files.

The system has thus provided a means of effectively increasing the on-line public space allotment available to each user. His data files while in actual use are available on the fast access disc; when inactive, the files are stored in the Mass Storage Unit, yet are still immediately available to the user through the COPY routine.

TSOS has given public disc volumes attributes which tend to make them appear as extensions of core storage rather than as conventional I/O devices. A given file may be contained on any number of public volumes without the user's knowledge.

Though a public volume may be used concurrently by any number of tasks, full file security and integrity are provided. Public volumes are always available for allocation to a user's task, within the limits of the public space allocation established for him by his System Controller.

When it is necessary to retain files in the system, users will make the most effective use of TSOS by storing their files on public volumes.

If a user requires private volumes, he may incur a wait for peripherals on which to mount his volumes. Each time a request is made for a device on which to mount a private volume, the system must determine whether or not it can honor the request, based on the current requirements throughout the system for those devices.

The user may specify that his task be placed in a task queue until the devices required can be allocated by the system.

Files used within a task need not be cataloged if their use is temporary (i.e., confined to the task). However, at the end of a task, uncataloged files must be placed on private volumes, or they will be erased. The system assumes that a user desires storage on a public volume unless he specifically asks for a private volume.

SYSTEM CATALOG

The system catalog is a special system file which resides on a system residence direct access volume. This catalog is used to retain the file descriptions which must be stored within the system so that, once a file is created, it can subsequently be located by its filename.

When a user is originally JOINED to the system, he is assigned a unique user identification code. This identification is appended to the filename of every file cataloged by a user in order to distinguish his files from those of other users.

FILE SECURITY

The catalog is organized into user groups to facilitate file retrieval. Files to be cataloged may be stored on either public or private volumes.

A user must have a valid user number to gain initial access to the system. Additionally, the DMS provides a significant "Multilevel" file security environment for the many different types of user files that reside on the system's volumes.

Private volume file protection is provided since the system restricts them to one user and only the creator of such files can gain access to them.

Password Protection

When a user creates a file on a public volume, he is recognized as its owner. No other user of the system may obtain access to the file, unless the owner specifies that it is a common file, or permits access by associating a password with it.

A file, with which a password has been associated, will be accessible to any user who can provide the correct password, the user identification code of the owner, and the filename. Only the owner of the file may assign or remove the password.

To prevent several users from updating the same file at the same time, the system maintains interlocks on files while in use. A task will not be granted read access to a file if another task is updating it. Similarly, write access to a file is not granted if any task is reading it.

Additional protection may be specified for a file by designating it to be read only at the time it is cataloged. Once created, such a file may not be updated by any user except the owner and then only after he has changed the file's Read Only status.

DATA MANAGEMENT ACCESS METHODS

The Data Management System supports five problem program I/O access methods:

1. Sequential Access Method (SAM),

2. Indexed Sequential Access Method (ISAM),

3. Basic Tape Access Method (BTAM),

4. Primitive Access Method (PAM), and

5. Evanescent Access Method (EAM).

Primitive Access Method (PAM)

The Primitive Access Method (PAM) is used by the various access methods (as well as by privileged users) as shown in Figure 29.

The most significant point concerning PAM is that all direct access volumes are assumed to be pre-formatted into physical blocks of 2048 bytes and that each file contains physical blocks of 2048 bytes.

Note that BTAM never uses PAM and that SAM (tape) need not use it.

The Access Method facilities provide for the actual transfer of data to and from problem programs which are in execution. They allow for considerable flexibility in file organization. A file organization defines the overall relationships of the component records into which the file is subdivided.

The component records are called logical records because each is a logical entity containing information for the problem program that is to process the file.

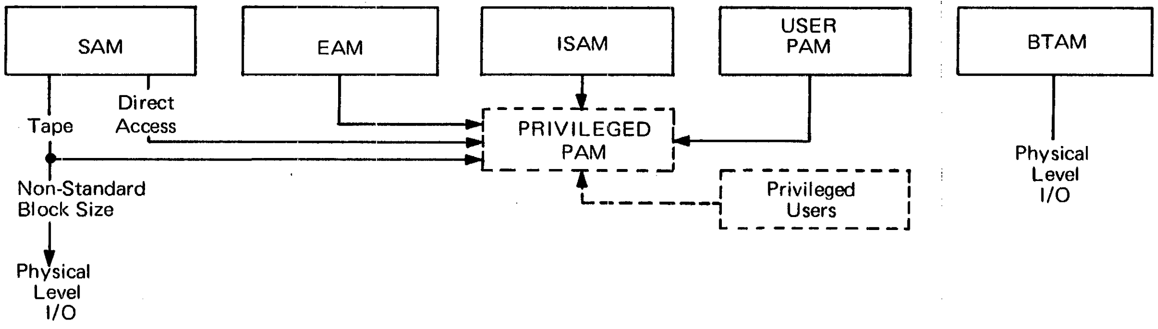


FIGURE 29. ACCESS METHOD RELATIONSHIPS

Sequential/Index Sequential Access Methods (SAM/ISAM)

SAM and ISAM are provided for logical level I/O processing. Using these methods, the logical records are retrieved and designated for output by use of GET and PUT macros. Blocking and buffer scheduling are automatically handled by the system.

SAM is device independent and allows for sequential processing of fixed-length, variable-length, and undefined record formats on both magnetic tape and random access devices.

ISAM provides for processing fixed-length or variable-length records on direct-access devices in either a sequential or non-sequential manner.

When programs have special I/O requirements that make logical level processing undesirable, the user may create his own file organization on private volumes.

Table 4 illustrates the access methods, record formats, device types, and allowable files from other access methods.

TABLE 4. ACCESS METHODS, RECORD FORMATS, DEVICE TYPES AND ALLOWABLE FILES FROM OTHER METHODS

Access Method	Record Formats	Device Types	Other Access Method Files Allowed As Input	Comment
PAM	Fixed	Direct Access Tape (single reel, std., blks)	SAM ISAM	Assumes 2048-byte block
SAM	Fixed Variable Undefined	Direct Access Tape	PAM	
ISAM	Fixed Variable	Direct Access	PAM	
BTAM	Fixed Undefined	Tape	PAM SAM	Assumes one record/ block
EAM	Fixed	Direct Access		Assumes 2048 byte block

Evanescent Access Method (EAM)

EAM is a specialized access method designed primarily to process temporary files in an optimum manner. Thus, EAM may suggest the Efficiency Access Method or the Evanescent (temporary) Access Method.

Efficiency is achieved in the following ways:

1. EAM files are not cataloged.

2. VTOC operations relating to space assignment and label processing are not performed.

3. OPENing of an EAM file requires zero disc pulls.

4. FILE card processing, FCB completion, and device assignment operations are not performed.

5. The counterparts to the above operations are not performed at CLOSE or ERASE time.

6. Fetching of data, with the action macros, utilizes less CPU time.

7. Processing areas (for example, FCB, logical routines) required for file operations are considerably smaller than those for the standard access methods (ISAM, SAM, etc.).

This efficiency is achieved by specializing and limiting the functions which EAM will support. Illustrative restrictions demonstrate this:

1. An EAM file is temporary.

2. It cannot be shared. In particular, the file cannot be multiply opened (i.e., if a particular EAM file is open, another program (routine) cannot open it). Note that a task may concurrently process different EAM files.

3. No user storage allocation, device assignment, or label processing is permitted.

4. Users can only transfer 2048 byte blocks of data.

5. An EAM file always resides on a public volume(s).

6. An EAM file is not supported by the checkpoint/restart components of the system.

INTRODUCTION

The third major component of the TSOS Supervisory System is Diagnostic Systems Support. This component contains the system message processing logic, the system loaders, special EO and Desk Calculator logic as well as the Interactive Debugging Aids component.

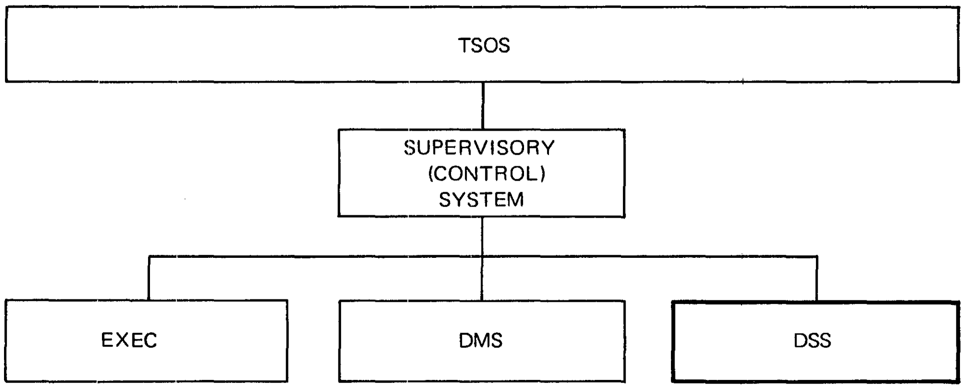


FIGURE 30. DIAGNOSTIC SUPPORT SYSTEMS

INTERACTIVE DEBUGGING AIDS (IDA)

The Interactive Debugging Aids (IDA) is a program checkout language whose commands are a subset of the command/control language system.

IDA may be used conversationally via a remote terminal which is on-line during the problem program execution or in a nonconversational mode in which a terminal is not used except to initiate optionally the IDA session. In the nonconversational mode, the debug session is prepared by building a procedure file of IDA commands.

To use IDA, the problem program is loaded and then the IDA software is called on to display and manipulate variables, data fields, instruction locations, and registers during the program execution.

The locations of all these parameters can be specified either symbolically or hexadecimally. Symbolic debugging is made possible by requesting that the language-processor generated Internal Symbol Dictionary be saved at program assembly or compilation time.

IDA allows for complete program control with the use of conditional and deferred execution commands. IDA commands are not embedded in user programs and, therefore, no recompilation is required.

IDA eliminates the dual frustration of debugging at the machine language level and the problems associated with adding and removing special coding to facilitate debugging and program maintenance.

INTRODUCTION

The TSOS Service (User) System is the user-program, non-privileged portion of the total Time Sharing Operating System. It consists of those components that are controlled by the supervisory system. The major components are logically divided into three groups:

1. Language Processors,

2. Service/Utility programs, and

3. Application/User programs.

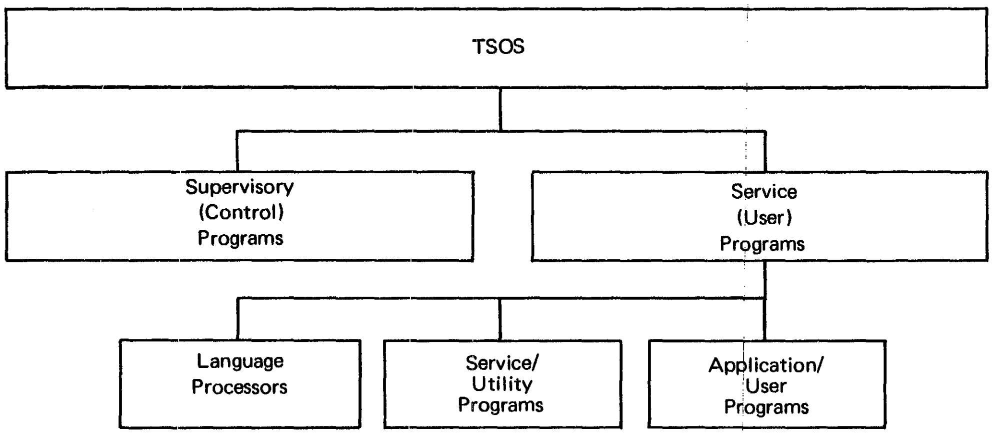
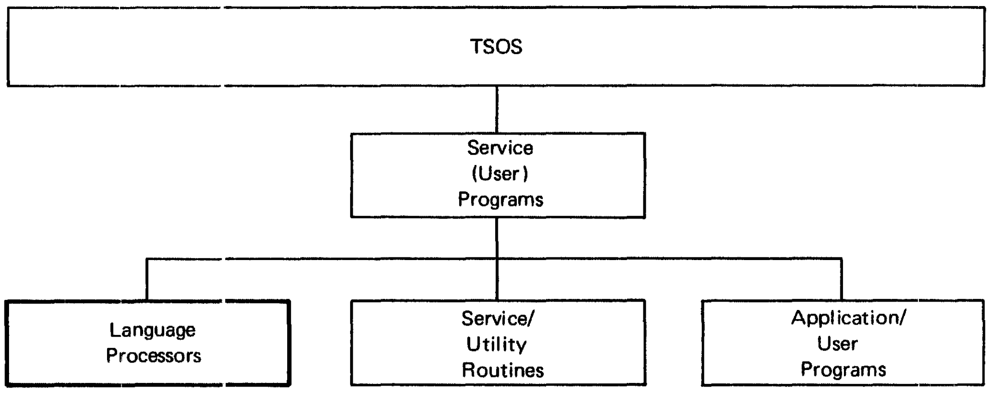

TSOS
Supervisory (Control) Programs
Service (User) Programs
Language Processors
Service/ Utility Programs
Application/ User Programs

FIGURE 31. TSOS SERVICE (USER) SYSTEM



CONVERSATIONAL

BASIC

Interactive FORTRAN (IFOR)

Fast FORTRAN (Conversational)

NON-CONVERSATIONAL

Class I Processors producing Class I object code

RPG

Fast FORTRAN

Class II Processors producing Class I or Class II object code:

ASA Fortran IV (Version 3)

COBOL (Version 3)

Macro Assembler

FIGURE 32. TSOS LANGUAGE PROCESSORS

LANGUAGE PROCESSORS

The TSOS Language Processors provided with the system fall into two categories.

1. Conversational Processors:

   a. Extended BASIC

   b. Interactive FORTRAN

   c. Fast FORTRAN

2. Non-Conversational Processors:

   a. Macro-Assembler

   b. COBOL (2 versions)

   c. FORTRAN IV (2 versions)

   d. RPG

   e. Fast FORTRAN

CONVERSATIONAL PROCESSORS

Extended BASIC

For the mathematical calculator, TSOS provides the BASIC Language. BASIC is a Class II program which generates Class II object code only at execution time from the user's source code file. This language, which has become a time sharing standard, was originally developed as part of a Dartmouth College system.

This simple but versatile language is very easy to learn and use; has automatic program input editing; has few restrictions and ambiguities; and is specifically designed for convenient terminal use.

TSOS offers Extended BASIC which is a form of this language which has simplified and increased the power of the arithmetic and control statements, the I/O statements, and the matrix commands.

Interactive FORTRAN (IFOR)

For the more experienced user, TSOS has an interactive FORTRAN system which provides time sharing in a language designed for computer professionals. This system consists of an interpreter which accepts the full ASA FORTRAN IV language, the same language accepted by the TSOS background Class I FORTRAN compilers.

This interactive interpreter provides the user with complete syntax checking, text editing, powerful debugging commands, a desk calculator facility, and immediate execution of his program from a remote terminal.

This language processor interpretively executes single statements one at a time without generating object code. This facility expedites the development of problem solution since the user can experiment with alternate algorithms throughout the program preparation phase.

Using these versatile facilities, the typical user will construct his FORTRAN program, syntax check it, modify, test, and debug it in this immediate execution environment and then subsequently compile it using one of the TSOS background FORTRAN compilers to generate clean, efficient object code if the program is to be retained in the system.

Fast FORTRAN

The Class II Batch Fast FORTRAN compiler allows the user to execute Fast FORTRAN (see Fast FORTRAN under non-conversational processors) from a remote terminal and to receive and enter data from a remote terminal.

NON-CONVERSATIONAL PROCESSORS

Macro Assembler

For sophisticated conventional batch type program preparation, TSOS provides a flexible Assembler. This assembler is a Class II shareable program which accepts assembly statements and macro definitions from remote terminals or cataloged disc files. Output from the assembler includes

1. A listing file which contains the source program, object code, and diagnostic information;

2. An object file ready for link editor processing – Class I or Class II code is specified at link edit time;

3. A diagnostic file which may be interrogated from remote terminals or spooled to the computer center printer; and

4. An Internal Symbol Dictionary for use with the IDA debugging system.

FORTRAN IV/COBOL

TSOS supports a full ASA FORTRAN IV compiler and an ANSI COBOL Compiler. These compilers operate as Class II type programs which generate object modules that can be executed as Class I programs or Class II programs. These TSOS FORTRAN and COBOL compilers provide facilities for

1. Retrieving source input from cataloged data files on disc or from a card reader,

2. Generating object modules to disc,

3. Writing listings to disc for automatic spooling to printer at task termination,

4. Generating an Internal Symbol Dictionary for subsequent use with the IDA system when debugging at program execution time, and

5. Preparing a cataloged diagnostic file on disc for subsequent query through a post-compilation conversational mode Class II program or listing at the computer center.

The object code generated by the compilers (pageable Class II programs) interfaces with the TSOS Data Management System for I/O.

Report Program Generator (RPG)

The Report Program Generator is a simplified programming language that produces a printed report without requiring a detailed knowledge of computer programming. The source program specifications are prepared to describe the input data, the output forms and the calculations to be performed.

The principal function of the RPG is program generation. A machine language program is generated in accordance with the specifications furnished. RPG automatically allocates the necessary storage locations, provides linkage to input/output operations and includes constants and other designated information. During the data processing state, the generated machine language program processes the user's input data files and produces the output files and/or printed reports.

Fast FORTRAN

Fast FORTRAN is a batch-oriented (non-conversational) compile and go FORTRAN. Many small jobs can be compiled and executed within a single execution of fast FORTRAN. Code is compiled directly into memory and executed after compilation. Object module decks are not produced. Fast FORTRAN system trades-off some of the full FORTRAN IV options for increased compilation speed.

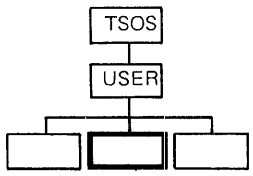
Fast FORTRAN was derived from the WATFOR compiler, originally authored by the University of Waterloo, Canada.

INTRODUCTION

The first routine of the system is an extremely flexible package designated the File Editor. The File Editor is a shareable re-entrant Class II program which enables the user to create, modify, and display cataloged files.

FILE EDITOR

The File Editor is designed to be used conversationally from a terminal so the user can control the processing; one command at a time. The Editor can, however, receive its commands non-conversationally from a system file.



SERVICE/UTILITY
PROGRAMS

FILE EDITOR

COBOL SYNTAX CHECKER

DESK CALCULATOR

LINKAGE EDITOR

ASSEMBLY DIAGNOSTIC

SORT/MERGE

UTILITIES

PERIPHERAL
DIAGNOSTIC PROGRAMS

REMOTE JOB ENTRY

SYSTEM BACKUP AND
RECOVERY

DATA EXCHANGE CONTROL

COMM's ACCESS METHOD

FIGURE 33. TSOS SERVICE/UTILITY ROUTINES

File Editor Session

To begin a File Editor session, the user opens an existing ISAM file or catalogs a temporary ISAM file for the session. This file is designated the principal file. During the session, the File Editor can sequentially access previously cataloged ISAM or SAM files to write data into the principal file or to read data from the principal file. These I/O files are called secondary files.

File Editor Commands

The File Editor command language, which is comprised of twenty verbs, is the medium by which the user specifies the editing functions to be performed on his files.

The File Editor has control commands (OPEN, HALT, SET, RESET, VERIFY) that enable the user to open and close files, to set and reset automatic features like symbolic pointers and session switches, and to terminate the File Editor session.

Input/output commands (TEXT, PRINT, MOVE, GET, INPUT, WRITE) provide for creating new lines in the principal file as well as copying lines existing in secondary files to and from the principal file.

Line content commands (FIND, DELETE, ALTER, UPDATE, CHANGE) are provided for scanning portions of the principal file and/or modifying the contents of the file at the string or character level.

Special commands are used to extend the facilities of the previously mentioned File Editor commands. A JUMP command is provided for changing the sequence of execution of commands if a specified criteria is met. The File Editor LOOP command defines a series of commands which will be executed successively a specific number of times.

The File Editor has a procedure call facility that enables the user to call a predefined file of File Editor commands. The procedure call is a single statement that allows a user to execute a complicated file editing function by issuing a single command.

The file of the File Editor commands is called a procedure definition. Each procedure definition has a header statement, a series of edit statements, and a trailer statement. The procedure call command can pass positional parameters to a procedure definition file allowing many users to vary the use of a single File Editor procedure file from session to session.

The File Editor can be used for problem program preparation and maintenance for the 70/46 or for other systems, general file editing and maintenance, and data base inquiry and response. The power and flexibility of the File Editor are limited only by the user's imagination.

COBOL SYNTAX CHECKER

The COBOL Syntax Checker is a re-entrant routine that aids the terminal user in the construction and maintenance of COBOL source programs.

The COBOL source program may be entered from a remote terminal or retrieved from a cataloged indexed sequential file on disc.

The user communicates with the Checker by using a set of ten (10) command verbs. All commands must be entered from a terminal - even though the source COBOL program itself may come from a disc file.

Syntax Command Language

The Syntax Checker Command language has session control verbs (FILE, ENTER, CHECK, STOP) for identifying the program file and starting and stopping the checker session; information verbs (WHY, HELP) for requesting more data about the current error or the checker commands; and correction verbs (CORRECT, MODIFY, IGNORE) which facilitate error correction.

A complete program or only a single COBOL division may be processed. The COBOL Syntax Checker recognizes over 300 errors. When an error is detected in a COBOL statement, the user is immediately given control for making corrections.

The program displays on a user's terminal the line number in which the error occurred and designators which point to the error.

If the user is unable to make an immediate correction, he can issue the IGNORE command and the error will be included in the source program file for subsequent correction.

The COBOL Syntax Checker includes, as a subroutine, the File Editor program so that more extensive editing functions may be performed during a Syntax Checker session.

A programmer, typist, or keypunch operator can enter source statements at a terminal and receive statement-by-statement syntax validation. A typist or keypunch operator could exercise the IGNORE option and hand the terminal error listing to the programmer for revision.

The program source statements can also be introduced to the system as a file created from punched card or magnetic tape input. The Syntax Checker can then be invoked from a terminal and the user notified at the terminal of any errors.

The Checker allows programmers to prepare "clean" source programs in minimum time, eliminating trial compilations to isolate syntax errors. This capability is very useful for program maintenance as well as initial program preparation.

After preparing a "clean" source program, the programmer can ask for a COBOL compilation from his terminal. Linkage and execution can also be scheduled from the terminal; and results can be printed at the computer installation, or stored as a file, for subsequent interrogation from the user's terminal.

DESK CALCULATOR

The TSOS Desk Calculator facility enables the user to operate his remote terminal as a desk calculator, without need for any knowledge of programming. The Desk Calculator program provides arithmetic, exponential, trigonometric, and logarithmic functions.

Technical, clerical, and accounting people can learn to use the Desk Calculator commands and put them into use at a terminal with minimal training.

The user types in commands and data through the keyboard of his terminal. The indicated calculations are performed by the program, and the results are immediately returned to the terminal user.

One general and ten special accumulators are provided. The general accumulator receives the results of the calculation. The special accumulators provide temporary storage areas which can be accessed or displayed at the terminal.

Numeric quantities are entered in fixed-point format with a maximum of 15 digits. Output to the terminal consists of error messages, the general accumulator value, and the contents of any of the special accumulators requested by the user.

The user has the option of displaying numeric values in a decimal short floating-point format, long floating-point format, or fixed point format.

The Desk Calculator is intended for the user (both commercial and scientific) who needs an economical, convenient, and easy-to-use calculating facility. The program, including user data, requires only one page of resident core and almost negligible computer time; yet offers a fast and easy-to-use wide range of powerful computational functions.

LINKAGE EDITOR

The TSOS Linkage Editor is used to produce program load modules from the object modules generated by the language processors for both Class I and Class II programs.

In a time sharing environment, several remotely located programmers may often work on the development of a single major program. The Linkage Editor can be invoked to link the object modules that have been compiled or assembled separately and produce a single loadable program.

The Linkage Editor also does linking for program overlays; obtains and links modules from the object module library; allocates record space for the Internal Symbol Dictionary; prepares program load maps; furnishes diagnostic messages and lists of unresolved references; and converts object modules produced by the TSOS Assembler into either Class I or Class II load modules.

DYNAMIC LINKING LOADER

The TSOS Dynamic Linking Loader provides an alternate "load and go" environment for Class II program modules. This Loader allows the loading of programs without requiring binding via the Linkage Editor.

This Loader is not intended to encompass all functions available in the Linkage Editor. It is designed to aid the user, who has relatively few modules, by not penalizing him with the core requirements and time needed by the larger user.

This Loader provides a method for conversational language processors to include subprograms that have been produced by other language processors. It also provides an Internal Symbol Dictionary File if requested.

The Dynamic Linking Loader is an integral part of the TSOS Control System. When the EXECUTE or LOAD commands are issued, the command language processor determines if the program module is in object or load format. If an object module is found, control is automatically passed to the Dynamic Linking Loader.

ASSEMBLY DIAGNOSTICS

The TSOS Assembler collects diagnostics in a disc file during program assembly. A TSOS Class II shareable program allows the user to interrogate this file.

This Assembly Diagnostic Program is used conversationally; however, the user has the option of examining his diagnostic file in two different modes.

The first is a standard conversational mode, and is called the TERMINAL mode. The user's commands are accepted from the terminal and responses are immediately returned to the terminal.

The second mode of operation is called the PRINT mode. In this mode, the user's commands are still accepted from the terminal or a procedure file. However, the responses are not constructed until the end of the session and the results are spooled to an installation high-speed printer.

The PRINT mode permits the user to extract and construct selected data to produce tailored diagnostic assembly listings.

SORT/MERGE

TSOS includes conventional Sort/Merge Generators for tape and disc. The Sort/Merge, like all TSOS software, can be called and scheduled for execution from remote terminals if input data requirements have been satisfied.

Sort/Merge control statements may be entered and modified on-line by means of a conversational pre-processor.

The Sort/Merge Generators create tailored Sort/Merge programs based on user-supplied parameters and allow for inclusion of own-code routines.

Sort/Merge Operation may be optionally initiated to run as a background or a foreground task. In addition, the user may specify a secondary work area (that is, tapes can be used to hold information when available disc area is exhausted.)

FORTRAN IV/COBOL DIAGNOSTICS

The FORTRAN and COBOL compilers catalog and generate a Diagnostic File if requested at compile-time. The file contains English-language messages describing each source error detected and a summary of the severity of errors.

The conversational user can invoke the interactive Diagnostic (BDIAG) routine to access the Diagnostic File and receive the error information at his remote terminal.

The compilers use the TSOS Data Mangagment System to catalog, classify and place the Diagnostic File on a public volume in the user's public space. The file name given combines the user's identification code, the source program name and the simple name ERRFILE, i.e., $USERID. SOURCE-NAME ERRFILE.

Two Diagnostic commands allow the user to receive the results of his compilation. The STATUS command gives him a summary record of the total count of error messages plus the four severity types. The PRINT command types out the error messages specifically requested. If the file does not contain any error messages, the routine will send a NO ERRORS message to the user.

UTILITIES

Library Maintenance Routines

A Library Maintenance routine is supplied for use in creating and maintaining object module libraries. This routine also maintains the directory that is included in the library to speed searches made by the Linkage Editor and the Dynamic Linkage Loader. Additional routines maintain the Macro Libraries (MLU) and the COBOL Source Libraries (CLU).

Peripheral Conversion Routines

TSOS supports a full line of Peripheral Conversion routines to enable programs to convert data from one input/output medium to another with a minimum of effort.

Edit Routines

The Utilities package also includes a Self-loading Device Edit, a Self-loading Memory Edit, a TSOS System Generator/Initiator, and a Volume Initializer.

The Device Edit gives the user an easy method of examining information read from, or written to, a random access device or a magnetic tape; by providing the ability to edit and print all, or selected portions, of the information contained on these devices.

The Memory Edit is intended for use by the operator as an emergency measure. This routine is completely self-contained, including routines to perform I/O at the hardware level.

Thus, the Memory Edit does not require services from the Executive or Data Management System, and runs independently of all other software. It dumps all or selected portions of memory including scratchpad memory and translation memory.

A spool-out function to support remote terminal devices (RCA 70/740 and IBM 1050) includes validation of Spool-out requests for remote terminals as well as scheduling of punch requests from remote Spool-out.

System Generator/Initiator Routines

The TSOS Systems Generator/Initiator is used to adapt or tailor TSOS to the specific user's equipment configuration. It selects those programming components that are required for the user's particular processing requirements by generating a Control Program that is maintained on the system resident disk.

Volume Initializer Routine

The Random Access Volume Initializer (VOLIN) is a routine that prepares random access volumes for use with the TSOS.

A surface analysis is made of each volume for any defective tracks. Home Address records and Track Descriptor records are written on each track, and a Volume Table of Contents is established for later use by the Data Management System.

HARDWARE MAINTENANCE ROUTINES

TSOS supports a group of on-line hardware maintenance routines that can be called out by the operator to check out installation peripherals without resorting to system shutdown.

This type of software takes on added significance in the time sharing environment. With many human beings simultaneously involved directly with the system, availability of the hardware takes on new dimensions.

Disc Hardware Check Routines

The TSOS Disc Hardware Check routine (DISC HCR) is part of a family of TSOS on-line peripheral test routines. These programs reside with the TSOS software and run, on a demand basis, as Class I user programs.

This multipurposed routine is intended for use by hardware maintenance personnel. It is to be used to test the 70/564 Disc Storage Unit without resorting to system shutdown. In addition to this, the DISC HCR provides troubleshooting aids and confidence tests for the disc and closely related equipment.

Basic Processor Unit Exerciser (BPUEXR)

The primary function of the BPUEXR is to exercise the internal logics associated with those instructions performing an arithmetic function on the 70/46 Basic Processor, concurrent with other nonconversational and conversational programs.

The BPUEXR operates on-line under the TSOS Executive as a Class II Permanent Task and nonconversational program. This program is not reentrant and is activated to run at intervals indicated by the operator command, BPURUN.

The objective of the BPUEXR is to detect arithmetic instruction malfunctions under multiprogramming, time-shared conditions.

Card Punch Program Routine

The TSOS Card Punch Program routine (PCHTST) determines the operability of the Model 70/234 or 70/236 80-column, row-oriented, Card Punch and that its control electronics are functioning properly.

The PCHTST routine operates under the control of the TSOS Executive System, as a Class I (TOS and TDOS compatible) user's program utilizing TOS physical level FCP.

This routine can be used to test all device functions of the Model 70/237 Card Reader. The routine includes special hardware testing functions.

Printer Device Segment Routine

The Printer Device Segment routine tests the hardware functions within the 70/242, 243-10, and 243-20 Printers. In the event of a malfunction, this program will diagnose the error condition and report (by way of the console typewriter) information necessary to correct this condition.

This routine also provides the capability for on-line troubleshooting and preventive maintenance.

Seven-Track Tape Hardware Check Routine

This routine is used to test seven-track 70/432-442, and -445 Magnetic Tape Units when these units are connected to a Model 70/473 Tape Controller. This routine is used by hardware maintcnance personnel; system shutdown is not required.

The routine is run as a Class I user program. It uses physical level FCP and creates I/O situations, both normal and abnormal, and interprets the results form the I/O operation.

Nine-Track Tape Hardware Check Routine

This routine is used to test nine-level Model 70/432, -442, and -445 Magnetic Tape Units. This routine is used by hardware maintenance personnel; system shutdown is not required.

The nine-level tape routine is run as Class I user program. It uses physical level FCP and creates I/O situations, both normal and abnormal, and interprets the results from the I/O operation.

REMOTE BATCH PROCESSING

One of the normal processing modes of the 70/46 is remote batch processing (also called Remote Job Entry).

This method of processing permits the entry of non-conversational tasks into the TSOS JOB STREAM from remote terminals for scheduling, execution, and subsequent return of output to the remote terminal.

Remote Batch Processing support is provided for the RCA 70/740 -70/741 Data Terminal and the Univac DCT 2000.

These Terminals provide a nominal 200-300 card-per-minute input and a nominal 250-300 line-per-minute printer output over leased or dial-up voice-grade lines.

Users of these remote terminals normally dial up the system and, after connection is established, initiate the card reading function. The central computer system will spool-in the card deck which will include all necessary commands, including Job Control information.

The task is then placed in the system job queue and treated like all other tasks in the system. It is scheduled according to priority, the necessary system resources are allocated, and the task execution commences.

The terminal may log-off (if specified in the job control information) without waiting for task completion. Output files for the job will be created on disc.

If the user has specified that the output is to be returned to the remote terminal, the system will dial out to the specified remote terminal and, after connection, spool the file to the remote printer.

SYSTEM BACKUP AND RECOVERY

TSOS provides facilities to preserve and safeguard user programs and information in the system. These backup and recovery functions include routines for Task Suspension, Checkpoint and Restart, File Reconstruction, and System Self-Protection.

Task Suspension is used to suspend execution of a task, and resume execution later. Both the user or TSOS can suspend a task. The task can be resumed from the point at which it was suspended without any loss of information.

Task Suspension is provided for a conversational user who wants to leave his terminal for an extended period before finishing his task. In addition, conversational and non-conversational tasks can be suspended prior to system shut-down and resumed when the system is brought up again.

In suspending a task, TSOS completes all outstanding input/output requests, closes all files opened for the task, and, in general, brings the task to an orderly stop.

The contents of the user's virtual memory, the status of all files, device repositioning parameters, and sufficient other information to enable the task environment to be reconstructed are preserved.

Checkpoint and Restart is similar to suspension except that the task is automatically continued after all the information needed to restart the task is stored.

The principle reason for checkpointing is to permit restart in the event that processing, subsequent to the checkpoint, is believed to be invalid because of a hardware malfunction.

File Reconstruction routines are provided to facilitate reconstruction of files stored on direct-access volumes. A copy of the entire file as of a certain previous date is saved.

A second file is created with a copy of each record written to the file subsequent to the creation of the original master copy. Reconstruction routines are used to rebuild the file from these back-up files in the event of a system failure.

System self-protection facilities are provided to facilitate reviving the system in the event of power failure, hardware failure, system software failure, or unscheduled shutdown.

The system is in effect required to perform a system dump. The dump information includes the contents of all virtual memory and all data stored on public volumes. It can be used by the System Operator to reestablish the status of the system to that at the time of the dump.

DATA EXCHANGE CONTROL

TSOS is capable of receiving terminal command language from another Spectra processor. The control program receives messages which inform TSOS of terminals dialing in, disconnecting, logging on the system, etc. Other messages are exchanged in order to inform the other processor what action is being taken from a given terminal or that no traffic is present. These items are conveyed by means of message types.

TSOS exchanges messages with the other processor via the Data Exchange Controller (DXC) on an inquiry/response basis; i.e., the TSOS control program reads and the other computer writes. Upon completion of this transmission, the receiver acknowledges the message and the process is reversed.

The exchange of information continues until the data is exhausted or the session is terminated.

TSOS is capable of terminating the session by means of a console message or by means of a termination command from the auxiliary computer. The auxiliary computer may be any processor that can communicate with TSOS via a DXC device.

COMMUNICATIONS ACCESS METHOD

TSOS supports a Communications Access Method (CAM) which facilitates the implementation of inquiry/response type programs as part of the normal TSOS environment.

CAM is a set of Class II subroutines which allows a user-prepared TSOS problem program to conveniently communicate with one or more remote terminals.

The terminals supported under CAM are:

70/752 Video Data Terminal
AT&T 33, 35 and 37 Teletypewriters
IBM 2741

Terminals which can be polled are:

70/752 Video Data Terminal
70/750 Video Data System with 70/751 Terminals

Programs using CAM reside within the normal TSOS environment as either conversational or nonconversational tasks, and have available to them all TSOS facilities including the use of system files.

CAM subroutines are reentrant, permitting concurrent execution of all programs using the CAM software.

The Command language repertoire includes but a single command for user communications with programs running under CAM.

The CONNECT command allows the user to attach and detach his terminal to a program utilizing CAM by correlating a name specified in the CONNECT operand to a name which has been specified by a program executing under CAM.

This CONNECT command may be issued instead of a LOGON command to establish an inquiry/response environment.

Once his terminal has been "connected," the user may disconnect (hang up); and by redialing the same line number, be automatically "reconnected" to the same communications program.

A terminal is not actually connected to the CAM program, but rather it is the telephone line number which is connected. Consequently, if the last user disconnects by hanging up, anyone subsequently dialing the same number will automatically be attached to the CAM program instead of the normal TSOS mode. The second user can disconnect from the CAM program by issuing the CONNECT command with "TSOS" as its operand.

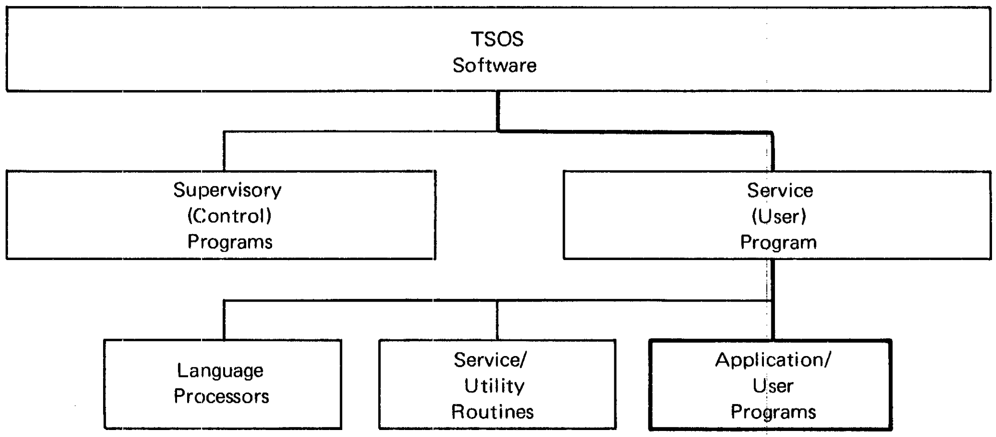


FIGURE 34. TSOS APPLICATION USER PROGRAMS

GENERAL

Although the user will typically create his own application programs (that is; payroll, personnel, stock-control, etc.), RCA supplies packaged application programs that the user may incorporate into his system.

Beside a wide range of Scientific and Industry-Oriented Application programs, RCA furnishes the TSOS user an Automatic Text Formatting System (Autoform) which facilitates the production of typewriter or printed documentation through the use of a computer.

AUTOFORM

The RCA AUTOFORM system was developed primarily to eliminate many of the clerical tasks which a professional employs when writing, editing, and formatting technical documentation.

The system is applicable to all industries which use documentation as a basic support tool.

AUTOFORM captures keypunched text and maintains it on direct access storage devices.

Data is maintained on the 70/564 or 70/590 Disc Units and can be modified; using the AUTOFORM commands and overlays whose calling sequence is variable depending on the tasks to be performed.

Text Processing Features

Typical text processing features supported by AUTOFORM are:

1. Pagination Control,

2. Paragraph Control,

3. Constant Line Control,

4. Word Index,

5. Footnotes,

6. Cross Reference,

7. Table of Contents, and

8. Word Hyphenation.

The minimum equipment configuration for the system is

1. Input device(s)

    a. FRIDEN 7102 Flexowriter Remote Terminals,

    b. Cards/Magnetic Tape, and

    c. 70/752 Video Data Terminal.

Note: It is economically important that the remote terminal accomodate paper tape to allow off-line data preparation.

2. Output device(s)

    a. Upper/Lower case printer,

    b. Magnetic Tape, and

    c. Punched paper tape.

3. Six work files (on disc or magnetic tape drives) and a master text file.

System Flow

Figure 35 simulates a document path through the AUTOFORM system.

1. Typist produces a hard copy plus paper tape which is input to the system.

2. AUTOFORM creates a galley listing which is reviewed by the author and corrections are retyped by the typist.

3. Corrected input is resubmitted to AUTOFORM and the cycle is repeated until the final galley is proofread.

4. Output is in the form of magnetic tape and if the author wishes, it can be entered into the RCA VideoComp system.

5. The final document is produced and can now be printed and distributed.

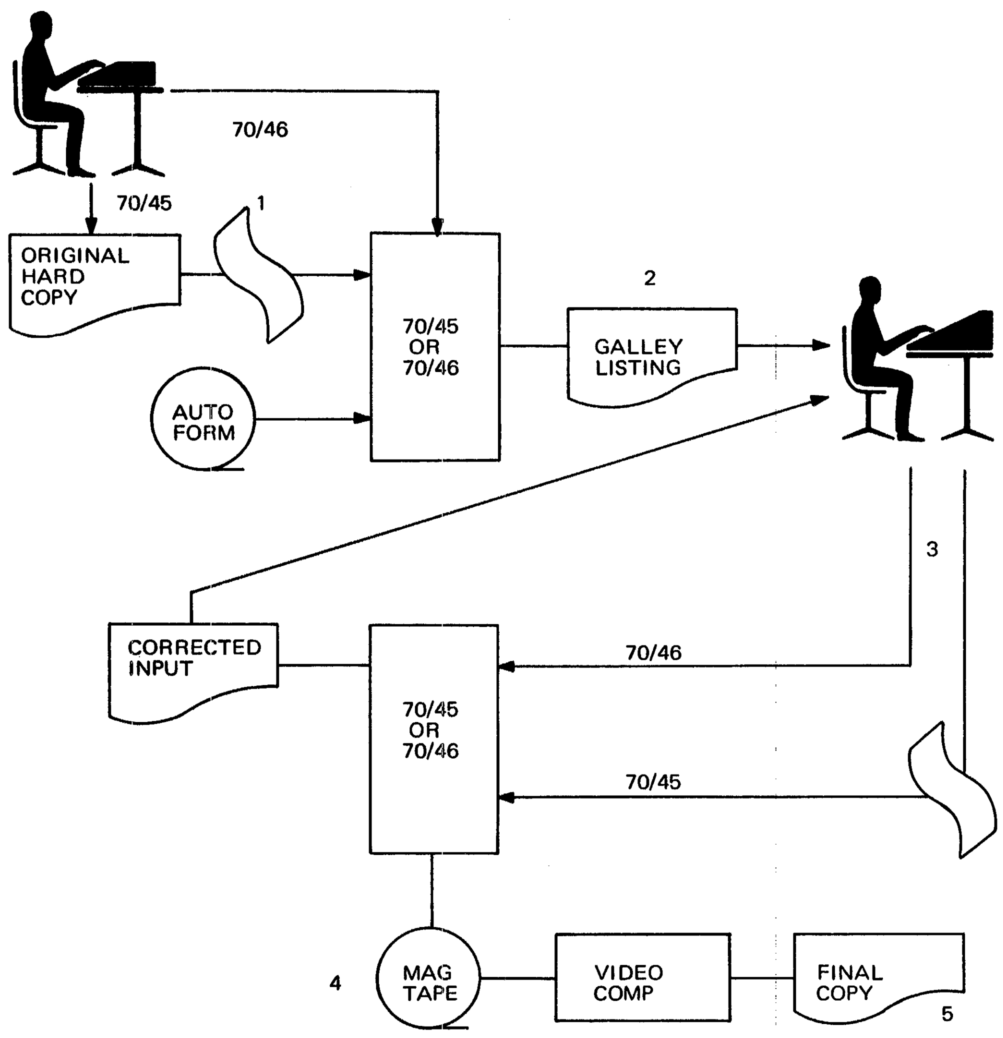


FIGURE 35. AUTOFORM SYSTEM FLOW

TIME-SHARING SCIENTIFIC APPLICATIONS PROGRAMS

The scientific applications programs are written in FORTRAN IV and currently operate on a 70/45 System using the FORTRAN Compiler. These programs can be recompiled to produce object decks that operate efficiently on the 70/46 System as Class I programs.

Statistical System

The Spectra 70 Statistical System consists of a series of statistical programs and a monitor system. The monitor provides unified operating procedures, unified input/output, unified error recovery procedures, and automatic sequential processing of problems. Variables can be expressed as single- or double-precision variables. The following named program routines are presently part of the system. Because the system is open-ended, additional programs can be added.

Analylsis of Variance

Factor Analysis

Regression and Correlation

Nonlinear Regression

Function Minimizer

Description and Tabulation

Multivariate Analysis

Canonical Analysis

Regression Analysis

Variance Analysis

DTM

COGO

Traffic Signal Progression

Sales Forecasting and Control System

Special Complex and Real Matrix Subroutines

Complex Arithmetic Subroutines

Princeton Interactive FORTRAN (PIFOR)

Princeton Interactive FORTRAN (PIFOR) is an enhanced version of the BTSS FORTRAN PI compiler. The compiler is incremental; that is, each statement is compiled when entered. The object code generated is capable of immediate execution and need not be modified as other statements are entered into the program. In addition to the language components, the PIFOR language contains a comprehensive set of debugging facilities and system commands which give the user control over both program structure and file structure.

PIFOR geenerates saveable object code and source files compatible with File Editor. Interactive FORTRAN (IFOR), fast FORTRAN and the Background FORTRAN IV Compiler (BGFOR) are capable of accepting the PIFOR source files as input provided that the source files are maintained in standard FORTRAN formats.

Part 5

Summary of RCA Time Sharing Operating System

Section 1: GENERAL DESCRIPTION

TSOS includes advanced interactive facilities for fast and convenient on-line access for up to 48 concurrent terminal users. Operating simultaneously with this interactive system is an enhanced batch processing facility capable of concurrently running many multiprogrammed background jobs.

A single powerful command language facilitates dynamic user/system interface in both operating modes. Priority service is provided for the remote terminal users; and they have access to all system software through their terminals.

Work for the batch processing mode can be initiated from remote terminals as well as from conventional I/O devices.

PROBLEM SOLVER	PROFESSIONAL PROGRAMMER	INFORMATION SEEKER
COMMAND LANGUAGE	COMMAND LANGUAGE	COMMAND LANGUAGE
CONV. BASIC	I.D.A.	D.M.S.
CONV. FORTRAN SYSTEM	CONV. FORTRAN SYSTEM	FILE EDITOR
D.M.S.	ASA FORTRAN IV SYSTEM	CAM
FILE EDITOR	ASA FORTRAN IV COBOL	
DESK CALCULATOR	COBOL SYNTAX CHECKER	
	ASSEMBLER	
	D.M.S.	
	FILE EDITOR	
	DESK CALCULATOR	

FIGURE 36. TSOS SOFTWARE SUMMARY

The multifunction operating system funnels all tasks presented to it from all sources to a single task queue for scheduling and execution.

All these tasks are time sliced. Conversational tasks from terminal users receive system priority as a group since the normal increment of work is short. Batch tasks have assigned to them all remaining CPU time resouces.

Efficient spooling has been incorporated to facilitate I/O throughput and improve peripheral device utilization.

Included in the software is a complete range of batch and conversational language processors, powerful debugging and file editing systems, remote syntax checking, an extensive Data Management System, a Communications Access Method, hardware diagnostics routines, and a complete line of utilities.

In TSOS, the language processors and service programs are indistinguishable from user prepared programs. The System is "open-ended" to the point that new language processors can be added to the present repertoire without altering the TSOS Executive System.

The complex problem of computer task scheduling has been simplified. Turn-around time no longer requires a stipulated period of hours or days.

Tasks can be introduced to the system with a minimum of effort. Their format is checked as they are entered; if errors are found, they are reported immediately to the user.

Adjustments can be made and new output acquired all within a matter of minutes. The user can stay with a problem until he has carried it to a logical conclusion.

TSOS creates an environment ideally suited for the problem solver, the user concerned with formulating problem solutions conveniently and receiving results rapidly.

One of the greatest advantages of TSOS is its capability to reduce turnaround time. Problem solvers will benefit tremendously because they are often faced with problems which occur at unpredictable times and require immediate solution.

The problem solver at his remote terminal will now have more time and incentive to use his talents creatively than ever before.

Two conversational problem solving language processors are provided: Extended BASIC and Conversational FORTRAN.

Experienced programmers can employ the FORTRAN compiler. Less experienced users can use the BASIC language.

Problem solvers can use the versatile facilities of these compilers to design and check out their solution algorithms on-line.

The structure of the 70/46 in-house time sharing system is such that the user can now have available the total data base of his company for input directly to his problem programs.

The Data Management System offers a wide range of file organization techniques. The File Editor has all the facilities needed for preparing test data and manipulating program and data files. The Desk Calculator is available for simple explicit calculations.

The man-machine interaction, a prime consideration in the problem solving environment, is further enhanced by the easy-to-use TSOS command language.

For the professional programmer, the user concerned with the preparation of large application programs, TSOS provides facilities for preparing on-line COBOL, FORTRAN, ASSEMBLY LANGUAGE, and RPG programs.

The programmer has immediate access to all system software from his remote terminal. He has available all of the diagnostic resources provided in a batch environment.

In addition, he can employ conversational syntax checkers to isolate language syntax errors prior to compilation, and these errors can be immediately corrected on-line.

Using the unique IDA debugging facility, the programmer can use his terminal to monitor and control his object program execution interactively.

Large program dumps and/or voluminous program listings can be scheduled into the spooling queue for the high-speed printer from the remote terminal.

The programmer can use the interactive compilers to check program algorithms on-line. He can invoke the File Editor to prepare test data and construct test files. He can schedule his background program compilations from his terminal.

The Desk Calculator is available for determining the value of constants and checking intermediate program results.

The system is designed to serve as a program preparation utility that can be used to prepare programs for competitive systems and other Spectra systems as well as for the 70/46.

The TSOS language processors are compatible at the object level with other Spectra lines.

At the FORTRAN and COBOL level, the 70/46 can be considered as a program preparation tool for many competitive systems.

Since the 70/46 virtual memory enables a programmer to prepare object programs of up to one million bytes, it can be considered for preparation of programs to be run on large-scale machines.

After a program (that has been prepared using a TSOS language processor) has been debugged with IDA and tested in a simulated environment, the TSOS File Editor can be invoked to transform the program into one that is completely acceptable as input to the foreign batch production computer.

TSOS editing facilities can also be employed to append job control information to the program file so that it can be directly introduced to the other computer. These editing procedures can be preserved in the 70/46 System for repetitive use.

For the information seeker and text handler (users concerned with constructing, updating and accessing a data base), TSOS provides both the language and storage capabilities they require.

The Command language, File Editor, and Communications Access Method interface the information seeker at his remote terminal with the full power of TSOS.

Although not readily apparent to the information seeker, the Data Management System makes available a myriad of file organizations and data access methods required to support a good general information retrieval environment.

The File Editor has powerful facilities for creating and updating data bases and for interrogating these files via remote terminals.

File Editor command sequences can be prestored in the system and called to action with a single command.

The Communications Access Method can be used to establish an inquiry/response environment for the use of Information Seekers who have no knowledge of the system.

By placing the data base on public volumes, it can be made immediately available to the information seekers from all the terminal devices in the system.

Information seekers from their remote terminals can also request the use of private volumes or use the COPY command for the Mass Storage Unit, and thereby gain access to a data base of unlimited size.

For the information seeker who does not require on-line access, TSOS has a powerful batch processing facility for the production of operating reports and management information summaries.

TSOS also supports a variety of application programs supplied by RCA. Programs such as AUTOFORM, and Scientific Application routines, allow the user highly sophisticated programming applications while minimizing his own programming effort.

In summary, TSOS has been designed to improve the use of that most valuable commodity - human resources, as well as to provide an efficient multiprogramming environment for batch or production processing.

It provides an excellent problem solving capability, an elaborate program preparation facility, and easy-to-use inquiry/response software.

It provides an economic and flexible environment in which problem solvers and professional programmers can innovate.

TSOS can be used not only to construct program solutions but also to design them on-line.

This approach to problem solving and program preparation is feasible because the penalty for failure is small.

The users are employing only a very small portion of the system resources, and each user has the ability to get on and off the System at will without interferring with other system activities.

The following examples illustrate uses of the system in a daily environment. The tasks represented are typical of the many types of user applications that TSOS is designed to serve. Comments are associated with each example to familiarize the reader with some of the basic conventions of the system and user interface. For all terminal initiated tasks, the examples presuppose that the user has dialed into the system and established a connection. The system will then issue the prompting message to the user:

%PLEASE LOG ON.

All prompting messages from the Executive component of the system are preceded by a percent (%) sign. Following the prompting message, the system will then type a slash (/) on the next line and await the user's response. Other components within the system have their own conventions for user input.

The printouts in these examples are simulated. The Model 33 Teletypewriter (TTY) terminal prints the alphabetical character 0 with a slash through it and the numeric zero without a slash. Model 35 TTY prints the alphabetical character 0 without a slash and the numeric zero with a slash. Note also that all printed lines from the terminal have a non-printable END OF TRANSMISSION character (ETX) as the last character of the line.

Table 5 is an example of a conversational (interactive) task and shows the interaction between the user and the system in a simple BASIC session. It also illustrates how the user's terminal becomes various system logical files depending on the input or output of the system.

Line 1-3	After the user establishes connection with the system and is asked to LOGON, he enters his user-id and any other information required by the System Controller at Join time. The system indicates acceptance of the user's LOGON information, assigns a task sequence number (TSN) and types a slash (/) on the next line.
Line 4-5	The Basic program asks the user if this is a new or old program and types an asterisk (*) on the next line. The user replies NEW and BASIC then asks for the program name which the user supplies in the same line.

TABLE 5. EXAMPLE 1 — TOTALLY TERMINAL-CONTROLLED TASK

Line	Sender	SYS Logical File/Device	Input/Output
1	Exec(utive)	SYSOUT	%PLEASE LOGON
2	User	SYSCMD	/LOGON user1, etc.
3	Exec	SYSOUT	%LOGON ACCECPTED etc.
4	User	SYSCMD	/EXEC (BASIC)
5	Exec	SYSOUT	%LOO1 DYNAMIC LOADER INVOKED
6	BASIC	SYSOUT	NEW OR OLD
7	User	SYSDTA	*NEW
8	BASIC-User	SYSOUT-SYSDTA	NEW PROGRAM NAME--BAS1
9	BASIC	SYSOUT	READY
10	User	SYSDTA	*10 READ A, B, C
	.	.	.
	.	.	.
	.	.	.
	User	SYSDTA	*90 END
11	User	SYSDTA	*LIST
12	BASIC	SYSLST	(Source listing of program)
13	User	SYSDTA	*BYE
14	User	SYSCMD	/LOGOFF
15	Exec	SYSOUT	%B003 LOGOFF AT 1338 etc.

Line 9-12	BASIC types that it is ready and the user enters his program lines with a line number for each line. The last line contains an END statement. The user then asks BASIC to LIST his program and BASIC prints the program on the terminal, ending with an asterisk on the next line.
Lines 13-15	The user types BYE to BASIC and the system types a slash. The user then Logs off and the system notifies him of his LOGOFF time, date, TSN and gives him the total Central Process (CPU) time used.

TABLE 6. EXAMPLE 2 — LOCAL (NON-TERMINAL ATTACHED) TASK

Line	Sender	SYS Logical File/Device	Input/Output
1	Operator	SYSCMD	/RCARD
2	Card Reader	SYSCMD	/LOGON userid etc.
3			/EXEC (EDIT)
4		SYSDTA	*OPEN APROG. NEW
			*TEXT
			.
			.
			.
			*#END
5		SYSLST	*PRINT 1, $
6		SYSDTA	*HALT
7		SYSCMD	/LOGOFF

Table 6 is an example of a Local (non-terminal attached) task. The user's COMMANDS and data are punched on cards and placed in the card reader. The operator's console is SYSOUT and the printer is SYSLST.

Line 1	The operator enters the RCARD (Read Cards) Command from the console and the system "spools-in" the cards to a temporary disc storage space, then reads the card images one at a time.
Line 2	The LOGON contains all the user information as though it came from the terminal except that the user supplies the slash.
Line 3	The user invokes The File Editor program and the Dynamic Linking Loader.
Line 4-5	The user is now giving information to the File Editor program followed by the END (of Text) and a command to print the output of the session.
Line 6	File Editor will place the output on temporary disc storage and print (spool-out) this output on the system printer at the end of the task.
Line 7	The card terminates The File Editor session and returns to the system.
Line 8	This card terminates the TSOS session.

TABLE 7. EXAMPLE 3 — TERMINAL INITIATED LOCAL CARD INPUT TASK

Line	Sender	SYS Logical File/Device	Input/Output
1	User	SYSCMD	/LOGON etc.
2	User	SYSCMD	/SYSFILE SYSDTA=(CARD)
3	User	(See below)	(See below)
4	User	SYSCMD	/EXEC (EDIT)
5	User	SYSCMD	/LOGOFF etc.
6	Card Reader	SYSDTA	OPEN A,NEW
7	Card Reader	SYSDTA	TEXT 10, 15
	.	.	.
	.	.	.
	.	.	.
8	Card Reader	SYSDTA	#END
9	Card Reader	SYSDTA	HALT
10	Card Reader	SYSDTA	/EOF

Table 7 is an example where the user initiates a task from a terminal but the input for the task is on cards at the local computer site.

Line 1-2	After the user LOGS ON he redirects SYSDTA to the local card reader using the SYSFILE command.
Lines 3-5	He calls the operator to insure that his cards are in the card reader, types in the EXEC (EDIT) command and LOGS OFF.
Lines 6-10	These cards are input to the File Editor program. They are spooled in and then executed one at a time. The HALT ends the File Editor session and the EOF terminates the input.

TABLE 8. EXAMPLE 4 — TERMINAL INITIATED BATCH ASSEMBLE AND LINK EDIT

Line	Sender	SYS Logical File/Device	Input/Output	
1	User	SYSCMD	/LOGON USER4, etc.	
2	User	SYSCMD	/FILE LINK=A,SYMDEV=(SYSUTS,(,(T0))) . .	
3	User	SYSCMD	/FILE LINK=D, SYMDEV=(SYSLIB,(,(T3)))	
4	User	SYSCMD	/PARAM TAPE=YES	
5	User	SYSCMD	/SYSFILE SYSIPT=(CARD)	
6	User	SYSCMD	/EXEC ASMDOS	
7	User	SYSCMD	/EXEC LINKED	
8	User	SYSCMD	/LOGOFF	
9	Card Reader	SYSIPT	START	
10	Card Reader	SYSIPT	BALR Assembly . Source . Input .	
11	Card Reader	SYSIPT	END	
12	Card Reader	SYSIPT	/EOF	
13	Card Reader	SYSIPT	PROG	Linkage
14	Card Reader	SYSIPT	INCLUDE	Editor
15	Card Reader	SYSIPT	CLASS 2	Parameters
16	Card Reader	SYSIPT	/EOF	

Table 8 is an example of a terminal initiated batch assembly and link edit.

Line 1-3	After logging on and verifying that his card deck is in the computer site card reader, the user assigns devices for a batch assembly task using the File Command.
Line 4-6	He asks for his output from the assembly to go to tape and directs SYSIPT to the card reader prior to executing the Assembly program.
Line 7-8	The user directs the system to execute the Linkage Editor program after the Assembly is finished and LOGS OFF.
Line 9-12	These statements on cards represent the Assembly Source input. The /EOF card returns control to the system.
Line 13-16	These parameters are the input to the Linkage Editor program.

TABLE 9. EXAMPLE 5 — TERMINAL INITIATED TASK USING PROCEDURE FILE AND INPUT FROM PUBLIC VOLUMES (DISC)

Line	Sender	Device	Input/Output
1	User	Terminal (SYSCMD)	/LOGON etc.
2	User	Terminal (SYSCMD)	/ENTER RUNFILE
3	User	Terminal (SYSCMD)	/LOGOFF
4	Procedure File (Run File)	Pub. Vol. (DISC)	/LOGON etc.
5	Procedure File (Run File)	Pub. Vol. (DISC)	/FILE . .
6	Procedure File (Run File)	Pub. Vol. (DISC)	/PARAM
7	Procedure File (Run File)	Pub. Vol. (DISC)	/SYSFILE SYSDTA=SFILE
8	Procedure File (Run File)	Pub. Vol. (DISC)	/EXEC BGFOR2L
9	Procedure File (Run File)	Pub. Vol. (DISC)	/SYSFILE SYSIPT=LEFILE
10	Procedure File (Run File)	Pub. Vol. (DISC)	/EXEC LINKED
11	Procedure File (Run File)	Pub. Vol. (DISC)	/LOGOFF
12	FORTRAN Source Input (SFILE)	Pub. Vol. (DISC)	ΔPROGRAM NAME
13	FORTRAN Source Input (SFILE)	Pub. Vol. (DISC)	ΔCOMMON IDUM (1024) . .
14	FORTRAN Source Input (SFILE)	Pub. Vol. (DISC)	ΔEND
15	FORTRAN Source Input (SFILE)	Pub. Vol. (DISC)	EOF
16	Link Edit Params (LEFILE)	Pub. Vol. (DISC)	CLASS 2
17	Link Edit Params (LEFILE)	Pub. Vol. (DISC)	PAGE ILFDATAD
18	Link Edit Params (LEFILE)	Pub. Vol. (DISC)	/EOF

Table 9 is an example of a terminal initiated task using a procedure file and input from public volumes.

Line 1-3	The user logs on, enters a previously created procedure file and immediately logs off.
Lines 4-8	The Procedure file (named RUNFILE) contains a LOGON command, File commands to assign devices for a FORTRAN Compilation and directs the SYSDTA input to another file on public discs called SFILE.

Line 9-11	Termination of the Compilation input returns control to the procedure file where SYDTA is re-directed to LEFILE on public disc and the Linkage Editor is executed. Control is returned to the procedure file which terminates the session with a LOGOFF.
Line 12-15	This file (LEFILE) is the input to the Linkage Editor run. Note that line 16 asks for a Class II (pageable) program to be created. Line 17 directs the Linkage Editor to begin the program on a page boundary.

TABLE 10. EXAMPLE 6 — COBOL COMPILATION WITH IDA, PLACE OBJECT MODULE IN LIBRARY ON DISC

Line	SYS Logical File/Device	Input/Output
1	SYSCMD	/LOGON etc.
2	.	/SECURE . .
3	.	/SYSFILE SYSDTA=SFILE
4	.	/PARAM SYMDIC=YES
5	.	/EXEC BGCOB2
6	.	/CAT LIBNAM
7	.	/FILE LIBNAM,SPACE=(6,3)
8	.	/EXEC LMR
9	.	CONTROL OUTFILE=LIBNAM
10	SYSDTA	ADD SOURCE=*
11	.	END
12	SYSCMD	/LOGOFF
Execute compiled COBOL program with IDA		
13	SYSCMD	/LOGON
14	.	/FILE . .
15	.	/LOAD (PROGNAM, LIBNAM), IDA=YES . . /RESUME
16	.	/LOGOFF

Table 10 is an example of a COBOL Compilation with IDA, Place Object Module in Library on Disc.

Lines 1-5	The user logs on, SECUREs devices for his run, directs SYSDTA to a disc file called SFILE (which contains his source data for COBOL Compilation), specifies that he wants a SYMBOLIC Dictionary generated with his program, and directs the execution of the COBOL Version II Background Compiler.
Lines 6-7	Following compilation, he catalogs a Library entry and allocates (File Command) space of 6 half pages (2048 bytes each) for primary space and 3 of secondary space for this library.
Line 8-12	He then executes the Library Maintenance Routine (LMR) and adds the object program with its symbolic dictionary (both now in temporary disc space) to the library and logs off.
Line 13-16	When he wishes to execute his compiled program using IDA (Interactive Debugging Aids) he logs on, allocates space with the File Command, and LOADS his program and the Internal Symbol Dictionary (ISD) using the Dynamic Linking Loader. The LOAD Command does not cause immediate execution of the program and the user can now enter symbolic debugging commands prior to entering the RESUME to execute his program.

Remote Batch Processing (RBP)

User:

Dials in and ENTERS The /RSTART Command from his Remote Batch Terminal (Work Station) to attach the terminal to the Central System.

System:

Identifies the unique name of the work station and its hardware characteristics, logically attaches it to the system and sends message to the remote station advising that it is being monitored for input.

User:

Enters the /RLOGON Command to begin a Remote Batch Processing Session.

System:

Validates the user information in the LOGON Command and grants the user access to the system.

User:

Supplies input (that is, RJOB to identify JOB n, LOGON, input for JOB n, LOGOFF, another RJOB, etc.) until all has been accepted.

TABLE 11. REMOTE BATCH PROCESSING INTERACTION

Remote Status	Command	Comment	Control system
Inactive	.		Always in Processing state
	.		
Active	/RSTART	Work Station attached	
Processing	/RLOGON	Begin RBP session	
	/RJOB	Define Job 1	
	/LOGON	FILE A	
	.	.	
	.	.	
	.	.	
Work	/LOGOFF	.	Central
	/RJOB	Define JOB 2	
Station	/LOGON	FILE B	System
	/DATA	.	
	.	.	
	.	.	. Card Reader
	/END	.	. Printer
	/LOGOFF	.	. Console
	/RSTATUS	Determine Job Status	
	/ROUT	Process Job Output	
Active	/RLOGOFF	End RBP Session	
	/RSTOP	Detach work station	
Inactive			

System:

Notifies user of Jobs accepted and processes according to the input stream. After all input has been processed, the system switches to Write mode and begins transmission of message and job output queued to the work station.

User:

Issues /RLOGOFF to end RBP session.

System:

LOGS user off and waits for another user.

User:

Issues /RSTOP to disconnect terminal.

APPENDIX A

GLOSSARY OF TIME SHARING TERMS

Access Method

A conventional data management technique, available to the user, for indicating the organization of data and method of transfer between an input/output device and the memory storage of the system. TSOS supports five access methods:

PAM - Primitive Access Method

EAM - Evanescent Access Method

SAM - Sequential Access Method

ISAM - Indexed Sequential Access Method

BTAM - Basic Tape Access Method

Active (program)

A program that has been loaded and is using the resources of the system.

Alphanumeric

A generic term for ALPHA-betic characters, NUMERIC digits and certain special characters such as a hyphen.

Actual address

An address that refers to a specific location in physical memory (contrast with virtual address).

Allocate

To grant a resource to, or reserve it for, a task.

Argument

A variable upon whose value the value of a function depends. The arguments of a function are listed in parentheses after the function name, whenever that function is used. The computations specified by the function definition occur using the variables specified as arguments.

Arithmetic statement

A type of statement that specifies a numerical computation.

Asynchronous

Not synchronized with respect to any other event (usually in a timed sequence). An event whose occurrence is not synchronized with program execution, and, hence, can occur at any time.

Attribute

A characteristic used to further define an object; for example, attributes of data include record length, record format, data set name, associated device type and volume identifications, use, and creation date.

Auxiliary storage

A storage device that supplements the physical memory of a computer. Instructions and data cannot be accessed directly within the computer but must first be transferred into main memory (contrast with physical memory). (See virtual storage.)

Background program

A program, usually of the batch processing type, that can be executed whenever the facilities of a time-sharing system are not required by programs of higher priority, which are usually conversational programs. (Contrast with foreground program.)

Backing (memory)

(See auxilliary storage, virtual storage.)

Batch processing

A technique in which jobs are collected into groups to be processed serially in the nonconversational mode. (Also called: stacked job processing.)

BT (beginning tape)

A marker at the beginning of a reel (volume) of magnetic tape that is sensed by the tape station to indicate the beginning of a reel.

Block	(n.) A generic term referring to a sequential collection of related items (e.g., storage block, record block); (v.) to group records for the purpose of conserving storage space or increasing the efficiency of access or processing.
Bootstrap	To cause an automatic load and execution of an initializing program that sets up the processor for the remaining components of a controlling program.
Buffer	(n.) (1) A storage device used to compensate for a difference in the rate of data flow, or time of occurrence of events, when transmitting data from one device to another; (2) portion of main storage into which data is read, or from which it is written; (v.) the act of filling and emptying buffers.
Call	The transfer of control from a routine to a subroutine.
Catalog	(n.) System file consisting of user and filenames, their characteristics and their locations; (v.) to include the name and location of a file in the system catalog.
Cataloged File	A file that is represented in the system's catalog that provides the means for locating it.
CPU (central processing unit)	The 70/46 Processor and its integrated hardware components.
Class I (program)	Programs that are restricted to the physical memory addressing capacity of the system. These programs must reside in physical memory throughout their execution and require that physical memory be assigned to them contiguously.

Class 1 (memory)

That portion of virtual memory that is occupied by the core-resident modules of the Control Program. All Class I pages are marked privileged and non-pageable. No drum image exists and these pages are always in physical memory throughout the time-sharing session.

Class II (program)

Programs that do not require contiguous main memory for their execution and may reside in the System's Virtual Memory. These programs are broken up into "pages" (4096 bytes), use the Virtual Address Mode of the system, and are normally "paged" between the system drum and main memory.

Class 5 (memory)

That part of the user's virtual memory occupied by dynamically allocated pageable areas acquired for a user task by the Control program. Used for DMS acquired I/O buffers, etc.

Command control block (CCB)

An area in a User's program in which data is passed between the user and the physical Input/output device accessed by the program. Used for physical level I/O programming.

Command language

The communication medium between the system and the individuals using it, allowing them to specify work for the system, and providing a means of monitoring that work.

Command procedure

A sequence of commands that has been placed in a cataloged file for the purpose of directing a supervisory or processing program in the execution of a task.

Compile

To translate a high level source language into computer language.

Constant

A quantity that does not change either from one execution of a program to another, or during execution of that program; a number that remains fixed.

Control system

That collection of programs in the operating system which aids in controlling the operations and managing the resources of a computer system. Example of TSOS control components are the Executive, Data Management, and Diagnostic System Support.

Conversational

Pertains to the type of system operation in which there is dynamic communication between the computer and its user. The computer program examines the input supplied by the user, usually from a remote terminal, and formulates questions or comments which are directed back to the user. (Contrast with nonconversational.)

Conversational (Task)

In TSOS, a conversational task is a task that is initiated and run from a remote terminal.

CSECT (control section)

A block of coding that can be relocated, independently of other coding, at load time without impairing or altering the operating logic of the program. CSECTS are the basic input to a language Translator program. (See object module.)

DSECT (dummy section)

A control section that is assembled but is not part of the object program. It is a convenient means of describing the layout of a storage area without actually reserving storage.

Data

That information which is processed by the program.

Data management

A general description of the collective functions of a programming system that provide access to files, enforce data storage conventions, and regulate the use of I/O devices.

Data organization

A reference to any one of the data management conventions for the arrangement of a file.

Debugging

Process of locating errors in a program and correcting them.

Device independence

The ability to request I/O operations, the nature of which is determined by the characteristics of the file rather than the particular device type upon which it is stored.

Direct-access

A type of storage medium which allows information to be accessed by positioning the medium or accessing mechanism directly to the information required, thus permitting direct addressing of data locations.

Direct addressing

The technique of accessing an instruction or data by referring to its actual location.

DTF (define the file)

A TOS mnemonic for the general class of TOS File Definition macros. (i.e., DTFDA means Define The File for Direct Access, etc.) TSOS DMS uses the Access Methods - BTAM, etc., for comparable purposes.

Dump

(1) To copy the contents of all or part of storage onto an output device so it can be examined; (2) data resulting from 1; (3) a routine that will accomplish 1.

Dynamic program loading

The process of loading a program module into virtual storage, based on an explicit or implicit reference to that program module by an executing program, rather than loading that program module independent of whether it will be referred to by an executing program.

Dynamic program relocation

The action of moving or relocating a program, before it has completed execution and without modification, to another part of storage, in a manner that permits subsequently resuming its execution.

Dynamic storage allocation

The allocation of storage space to a procedure based on the instantaneous or actual demand for storage space by that procedure, rather than allocating storage space to a procedure based on its anticipated or predicted demand.

ETW (end tape warning)

A marker at the end of a physical reel (volume) of tape that is sensed by the Tape Station to indicate the end of a reel.

ETX (end of transmission)

A character signifying the end of a transmission to or from a remote terminal. The actual bit configuration of this character varies with the remote terminal system.

EXCP (execute channel program)

A macro instruction at the physical I/O level that allows the programmer to communicate with an I/O device without using the standard DMS or TOS FCP logic.

Entry

A two-byte half-word in Translation Table Memory that references a virtual page address and includes its control bits.

Entry point

Any location in a program to which control can be passed by another program.

Event

An occurrence of significance to a task or procedure, typically the completion of some asynchronous activity.

Extent

The locations or range of locations on I/O devices, or volumes; occupied by or reserved for a particular file.

External reference

A reference to a symbol defined in another program module.

External storage

The part of secondary storage that is available to the users, for data storage (i.e., the part of secondary storage not used for auxiliary storage).

External symbol

A symbol whose value is of interest to a number of program modules, not only the program in which it is defined; a symbol contained in a program module dictionary.

Fetch

To cause the transmission of information from a location in main storage to a central processing unit.

FIFO (first in first out)

A queuing technique that processes information in the same sequence as it entered the queue.

File

In TSOS, a file is a named collection of related records. For example, all of the following are files: The conventional input/ output data used by data processing programs; source programs; object programs and subroutines; textual information to be organized and processed by the File Editor, etc.

File, NULL

A file that is logically empty. A file that is cataloged by the system and whose space remains allocated but contains no data.

File, shareable

A file that the user catalogs with the SHARE= YES parameter. Files marked as shareable can be accessed by any user who can provide the identification code of the owner, the filename and the appropriate password (if required).

Foreground program

A program which takes precedence over nonconversational programs in a time sharing system. (See Conversational program.) (Contrast with background program.)

Foreign (volume)

In DMS, a foreign volume is one that is not cataloged in the system (i.e., was not created by the particular DMS system currently in use by the user).

Hardware key field

Part of the track descriptor record (Record 0) or data record found in each track of a random access device. The Key Field contains control information concerning the record. It can vary in size from 1 to 255 bytes.

Indexed file

A file for which a location index has been created and is maintained.

Input line

A group of characters ending with a carriage return (RETURN), which are entered into the time sharing system by the user from his teletypewriter.

I/O, logical level

A type of input/output processing where the user functions at the record level, using the GET and PUT, OPEN and CLOSE logic of the system to perform his device handling, record blocking and deblocking, error recovery, etc.

I/O, physical level

A type of input/output processing where the user functions at the physical device level, constructing his CCB's and CCW's in conjunction with the EXCP macro to control the device himself. Record manipulation, error recovery, etc., are the responsibility of the user.

Installation

A general term for a particular physical plant or computing center; in context of the overall function it serves and the individuals who manage it, operate it, apply it to problems, service it, and use the results it produces.

Interlock

The "locking-out" mechanism that prevents more than one user from updating the same file at the same time.

Interrupt-driven

A central processor that responds to various external and internal events by interrupting its current processing to respond to the event.

Job

A unit of work for the computer system from the standpoint of installation accounting and operating system control. A job can consist of one or more job steps. In TSOS, a job is a step within a task.

Job step	A unit of work for the computer system from the standpoint of the user.
Job stream	All the processing to be performed in a particular time interval by a computer system under the control of an operating system.
Job library	A set of user-identified object program modules.
Language processor	A general term for any assembler, compiler or other routine that accepts statements in one language and produces equivalent statements in another language.
Library	In general, a collection of objects (for example, files or volumes) associated with a particular use, and the location of which is identified in some type of directory. A collection of user-written programs stored within the system, generally on disc.
Line error	An error in the transmission of data over telephone lines.
Line feed	Rotation of the teletypewriter platen up one line, accomplished either by striking the LINE FEED key, or by receipt of a "line feed" character from the time sharing system.
Line file	A virtual index sequential data set that is organized by line number with keyboard/printer-length records.
Line number	A number assigned by the user by which each statement can be identified. It is associated with a single statement and by which reference may be made to that statement.

Linkage

The means by which communication is effected between two programs.

Linkage editor

A program that produces an executable object program module by transforming other program modules, optionally combining separate control sections into a single cross-reference among them, replacing, deleting, and adding control sections on request.

LINK-NAME

This is the file-definition name that provides the connection between the general service File Control Block (FCB) macro and the DMS Job Control language.

Load

The process of reading the beginning of a program into storage and making necessary adjustments and/or modifications to the program so that it may have control transferred to it for the purpose of execution.

Load library

A collection of bound modules. (See load module.)

Load module

An object module converted (bound) by the Linkage Editor program into executable form.

Location-free procedure

Procedure which, for the purpose of execution, is independent of its location in storage; a procedure that does not contain any constants which must be changed as a result of its relocation in storage.

Locate-Mode

In the Indexed Sequential Access Method (ISAM), locate mode means that the user requests the location of the current record in the buffer area. The user is responsible for transferring data to and from the buffers. (Contrast with move mode.)

Location index

The table of addresses and keys used to indicate where the records of a file are stored.

Logical record

A record, from the standpoint of its content, function, and use rather than its physical attributes; that is, one that is defined in terms of the information it contains.

Loop

Repeated execution of a portion of a program.

Macro-instruction

A general and collective description of a macro-instruction statement, the corresponding macro-instruction definition, the resulting assembler language statements, and the machine language instructions and other data produced from the assembler language statements; loosely, any one of these representations of a machine language instruction sequence.

Macro time-slice

An interval of time allocated to an active program. In TSOS, a macro time slice is used for resident, non-pageable and system tasks. Currently, the macro time-slice is 1 second (1000 milliseconds) (See micro time slice.)

Magnitude

The size of a quantity as distinct from its sign. Thus, +10 and -10 have the same magnitude.

Main storage

Storage which is directly addressed by the registers of a central processing unit; hence, storage from which program instructions may be fetched and executed. (Same as physical storage.)

Micro time-slice

An interval of time that represents a division of a macro time-slice. In TSOS, a micro time-slice is 200 milliseconds and is used to control pageable and interactive tasks. (See macro time slice.)

Module (Programming)

The input to, or output from, a single execution of an assembler, compiler, or linkage editor; a source or program module; hence, a program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. (See object module.)

Move Mode

In ISAM, move mode means that the user specifies the location of the record in his program and the system is responsible for moving it to or from the buffers.

Multiprocessing

The simultaneous use of two or more interconnected processors forming a configuration capable of running multiple independent programs with common use of hardware facilities.

Multiprogramming

A technique by which a computing system can interleave execution of two or more generally unrelated programs, parts of which are residing together in main storage.

Name

A string of characters that identifies a statement, file, or program; usually, the string is associated with the location of that which it identifies.

Nonconversational

Pertains to the type of operation in a system where there is no dynamic communication between the computer and its user. Data and instructions are all specified in advance. This type of operation generally corresponds to job stream processing. (Contrast with conversational.) (See batch processing.)

Non-privileged

The commands and facilities available to any terminal subscriber.

Object (program) module

The output in card image format of a single execution of an assembler, compiler, which constitutes input to the dynamic loader or the linkage editor; an object module consists of one or more sections in relocatable (though not executable) form and an associated program module dictionary.

Object module library (OML)

Contains a sequenced collection of object modules.

Off-line

A generic reference to a device which may not be directly controlled by a computing system with which it is being used.

On-line

A generic reference to a device which can be directly controlled by a computing system to which it is electrically connected.

On-line storage

The storage devices under direct control of the computer. In TSOS, on-line storage consists of three direct access levels, virtual memory (on the highspeed drum), disc storage volumes, and mass storage; and private volumes on magnetic tape.

Operating system

An organized collection of routines and procedures for operating a computer.

Page

A logical division of a program or data that has a fixed virtual address but can in fact reside in any region of the computer's physical storage.

A set of 4096 consecutive bytes. Physical memory in the 70/46 is divided into 64 pages, each beginning on a multiple of 4096 bytes. Virtual memory contains 512 pages, corresponding to the 512 Entries in Translation Table Memory.

Paging

The process of transmitting pages of information between main storage and auxiliary storage, especially when done for the purpose of assisting the allocation of a limited amount of main storage among a number of concurrently executing programs.

The technique of interchanging pages belonging to various programs between physical and virtual storage (paging drum).

Peripheral device

Any device that is on-line (attached to the processor) other than remote terminals. Tapes, Disc, Printer, etc., are considered to be peripheral devices.

Permanent public storage space

The space on public volumes allocated to each user at JOIN time.

Physical storage (memory)

The same as main storage. (Contrast with virtual storage.)

Privileged

A term used to describe any part of the system (commands or facilities) that cannot be accessed by the general user.

Privileged (mode, program)

That part of TSOS that functions in other than the P1 or user state. The four states of the Spectra 70/46 processor are the P1 (User) state that interprets and executes the user program; also called the problem-oriented state: the P2 (Interrupt Response) state that performs specific program tasks as directed by the interrupt control state; in TSOS, the P2 state is also interruptible: the P3 (Interrupt Control) state that analyzes the cause of the interrupt; and establishes its priority and links to the related interrupt response routine: and the P4 (Machine Condition) state that is entered whenever a machine check or power failure occurs.

Problem program

Any routine that performs processing of the type for which a computing system is intended; including routines that solve problems, monitor and control industrial processes, sort and merge records, perform computation, process transactions against stored records, etc.; generally interpreted to be any non-system program. (See RCA Program and User Program.)

Procedure

The step-by-step process for the solution of a problem, especially the machine instructions embodying the solution.

Procedure file	A set of pre-defined operations (commands) that specify a sequence to be performed. A procedure file may also contain data to be used as input for a program. In TSOS, the two types of Procedure Files are the ENTER file and the DO file. Either can be initiated from a remote terminal; however, when an ENTER file is initated from a terminal, the terminal does not remain connected to the procedure as it does when a DO file is initiated. The ENTER file becomes a background (non-conversational) task.
Program	(n.) A generic reference to collections of instructions and data produced for the solution of some well-defined problem; (v.) to create and/or produce these collections.
Program (RCA)	A problem program supplied by RCA as part of the operating system for a computer installation. Examples of RCA supplied problem programs are language processors such as the Assembly System, COBOL Compiler, Report Program Generator, as well as utility programs such as the Linkage Editor, COBOL Library Update, and Self-Loading Memory Print. (See user program.)
Read-only file	A collection of instructions and data that is never modified or written into during its execution.
Real time	The actual time during which a physical process transpires, especially if that processor is monitored or controlled by a computing system.
Record	A general term for any unit of data that is distinct from all others within a file.
Recursive	Repetitive on a cyclic basis; of, or pertaining to, a program that calls or transfers control to one of its own entry points.

Reentrant

An attribute of a program that allows the program to be interrupted during execution, entered by another user, and subsequently reentered (at the point of interruption) by the first, user, and produce the desired results of all users; a program with an intermediate state of execution that is totally restorable when it is reentered after an interruption.

Relocation

The movement of a program from one place in storage to another, including the modifications to the program required therefrom.

Resident (program)

In TSOS, a resident program is not pageable and must remain in contiguous physical memory throughout its execution.

Resource

Any facility of the computing system or programming system required by a task; the facility may be main storage, input/output devices, a central processing unit, files, and control and processing programs.

Response time

The average time that a terminal user must wait to receive a response from the time sharing system; a measure of the rate at which the dialog between man and machine takes place.

Return

The return of control from a procedure to an invoking procedure; for example, the returning of control from a subroutine to the routine which originated the call.

Reusable

Describes a self-initializing program that can subsequently be entered by another user without reloading the program. (Contrast with REENTRANT.)

Routine

A set of instructions arranged in a desired sequence to cause a computer to perform a desired operation or sequence of operations. The term routine is sometimes used to refer to the whole or part of a program that has some general or frequent use. (See subroutine.)

Run time parameter (RTP)

A TOS (Tape Operating System) term for parameters that supply necessary information to TOS user programs at the time the program are to be executed. They may be used to assign devices to TOS programs, position files, check standard tape labels, etc.

Scheduling algorithm

A series of rules and decisions used to determine how central processing unit time (as well as other system resources) is to be allocated among the tasks contending for it.

Secondary storage

The on-line storage of a computing system that is not directly addressed by the registers of of central processing unit.

Segment (TSOS)

An area of contiguous virtual memory consisting of 64 pages.

Sequential access storage

A type of storage device from which the data records are accessed in a serial or sequential manner.

Serially reusable

A reusable program which is not necessarily reenterable.

Service program

Any of the standard programs that assist in the use of a computing system and in the successful execution of problem programs, without contributing directly to control of the system or production of results; including utilities, simulators, test and debugging routines.

Session

The period of time during which the user engages in a dialog with the time sharing system.

Shared files

Files that are designated either as "common" or "shared" that can be executed or accessed by multiple users simultaneously.

Shared routine

A routine which can be simultaneously or concurrently executed by several users.

Spool-in

A term used to describe the transcription of input to temporary direct access files, prior to delivering it to the program.

Spool-out

A term used to describe the transcription of output for relatively slow speed devices such as printers card-punches, etc., to temporary direct access storage until program termination.

Subroutine

A routine that can be part of another routine, and may be used at more than one point in a program, or which is available for incluison in other programs or use by other programs. (The distinction between the terms routine and subroutine has become obscure and the term now perferred to describe this subunit is subroutine.) (See routine.)

Supervisor

The program modules, supplied by RCA as part of TSOS, that control and monitor the usage of the time sharing system.

Supervisory programs

A generic reference to the programs that have the primary function of scheduling, allocating, and controlling system resources, rather than processing data to produce results.

Symbolic device name (SDN)

A TOS name associated with each logical file in a TOS program. Used to assign the actual I/O device to this file. The name is generally six characters in length, such as SYS007, etc. This is not the same as the installation mnemonic for a device, which is a 2 character code such as A1, etc., assigned at System Generation time.

System management

The function of TSOS that allows the user to specify his tasks and requirements for the system's resources - statically, prior to execution, or dynamically, during the execution of programs within a task.

Synchronous

Occurring with a regular or predictable time relationship.

System catalog

A special random access resident file used to locate user and system files.

System files

These are files that are accessed directly by control program routines only. They provide data and facilities required by control program functions and are used in the implementation of these functions. The two general classes of System files are temporary and logical. Both Classes have standard names assigned to them.

Temporary system files are created and disposed of by routines that use the facilities of the Job Control Component of TSOS. Their logical names are:

SYSOUT - printer destined system messages that record the commands and their responses.

SYSLST - printer destined listings such as those produced by an assembly or Linkage Editor run.

SYSOPT - card punch destined files such as an object deck produced by an assembly.

These temporary system files are given temporary filenames when they opened. After Spoolout processes these files, they are deleted and their disc space is restored to the system.

Other system files managed by the system are strictly logical in nature. They never receive filenames nor do they reside on disc. They are SYSCMD, SYSDTA, and SYSIPT.

SYSCMD is the logical system file that comprises the commands issued by a user in his task. The Control program alone reads this file.

SYSDTA is the logical system file that comprises the input data for an object program. This file is accessed by the TSOS macro RDATA.

SYSIPT is the logical system file for TOS compatibility. It is the counterpart of the TSOS SYSDTA file and is accessed by the TOS macro RDCRD.

Task

A unit of work for the computer system from the standpoint of the Control program. Generally, all work undertaken by the system in response to commands issued by the user between his LOGON and LOGOFF Commands.

Task management

The function in TSOS of control of all user tasks, facilitated by the command language.

Task scheduling algorithm

The technique that balances the time and access intervals of system subscribers to make the most efficient use of the central processor and system resources consistent with the required response times.

TM (tape mark)

A single character on magnetic tape surrounded by gaps that is generally used as a point of demarcation for data groups. On 9-channel tape, a tape mark has a bit configuration of (13) hex.

On 7-channel tapes, the configuration depends on the system that generated it.

Temporary storage

System storage on public volumes used for holding files and/or data until needed. Automatically released after use.

Throughput

A measure of the rate at which work can be performed by a computing system.

Time sharing

A method of using a computing system whereby a number of users can concurrently execute programs with which the users may interact during execution, and be generally assured some minimum amount of program execution per unit time.

Time slice

The time allotted during which each task executes before it is placed in a queue of tasks that are contending for the system resources. (See macro and micro time slice.)

Translate addressing

The technique of accessing an instruction or data by converting a virtual address in an instruction into its actual address at the time the instruction is staticized for execution.

Translation memory

The magnetic storage device in the 70/46 Processor containing a table used in translate addressing. Contains 512 half-word locations.

Truncations

Shortening of a number by dropping digits without rounding. The result is that portion of the number preceding the decimal point.

Turnaround time

The elapsed time between submission of a task to a computing center and the return of results.

User

Anyone who uses the services of a computing system.

User files

User files include both temporary and permanent files entered into the Time Sharing system by the user.

User program

A problem program written by the user. Examples of user-written problem programs are Payroll, Inventory Control, and Demand Deposit Accounting. (See RCA program.)

Virtual address

An address generated by a program which references virtual storage and must, therefore, be translated into a main storage address when it is to be used.

The user of virtual addresses permits the program to refer to data and instructions as though they resided in a main memory that is many times the computer's actual main memory configuration.

Virtual storage

Storage inaccessible to the TSOS user.

Those portions of auxiliary storage that contain instructions and data in the form of pages and are addressed by use of virtual addresses. In TSOS, the virtual storage device is a magnetic drum. (Also called: extended physical memory.) (Contrast with physical storage.)

Volume

A logical assignment of system storage space; a public volume is an operator-defined volume of direct access storage; a private volume may be either a direct access device or a magnetic tape and is also operator defined.

Volume sequence number

A four-digit number that indicates the sequence of a volume in a multivolume file. The volume sequence number is included in the standard file header label upon creation of the file.

Volume serial number (VSN)

A six-digit number assigned to a volume at initialization time and included in the standard volume label for identification purposes.

Volume table of contents (VTOC)

A file on a direct access device that contains the file label and information regarding the addresses of user data in the file.

Wait condition

As applied to tasks, the condition of a task that is dependent on events to enter the ready condition.

APPENDIX B.

SYSTEM INDEX

The manuals that document the Time Sharing Operating System are described in Section 2, Part 1. A subject index is given in this Appendix to aid the reader in locating information about the system. The current Publications Catalog should be used to obtain the ordering numbers and revision levels of these documents.

TOPIC	DOCUMENT
Abnormal Program/Task Termination	System Controllers Guide
Access Methods	Data Management System Reference Manual
Accounting	System Controllers Guide
ACT Macro	Executive Macros Reference Manual
ADEXT Macro	Executive Macros Reference Manual
ADIAG	Assembler Reference Manual
ASCII Macro	Executive Macros Reference Manual
Assembly Language Characteristics	Assembler Reference Manual
ASSGN Macro	Data Management System Reference Manual
AT Command	IDA Reference Manual
AUTOFORM	Autoform Reference Manual
Background Compiler	Background Compilers Reference Manual
BASIC Language	BASIC User Manual
BDIAG	Background Compilers Reference Manual
BIAS Command	Operators Guide
BIAS Command	System Controllers Guide
BIAS Command	Executive Command Language Reference Manual
Binary Operators (Desk Calculator)	Terminal User Aid — Desk Calculator
BKPT Macro	IDA Reference Manual
BREAK Command	Executive Command Language Reference Manual

(Continued)

TOPIC	DOCUMENT
Breakpoint Macro	IDA Reference Manual
BROADCAST Command	Operators Guide
BROADCAST (BCST) Command	Executive Command Language Reference Manual
BROADCAST Command	System Controllers Guide
BTAM	Data Management System Reference Manual
CAMRD Macro	Executive Macros Reference Manual
CANCEL (CAN) Command	Executive Command Language Reference Manual
CANCEL Command	Operators Guide
CANCEL Command	System Controllers Guide
Card Loader	Operators Guide System Controllers Guide
CATAL Macro	Data Management System Reference Manual
CATALOG Command	Data Management System Reference Manual
Cataloged File	Data Management System Reference Manual System Controllers Guide
CCB Macro	Data Management System Reference Manual
CCM	SYSGEN Reference Manual System Controllers Guide
CHANGE Command	Data Management System Reference Manual
CHECK Macro	Data Management System Reference Manual
CHNGE Macro	Data Management System Reference Manual
Class I Program	Utilities Reference Manual System Controllers Guide
Class II Program	Utilities Reference Manual System Controllers Guide
CLOSE (file) Processing	Data Management System Reference Manual
CLU	Utilities Reference Manual
COBOL Compilations	Background Compilers Reference Manual COBOL Reference Manual
COBOL Language Characteristics	COBOL Reference Manual
COBOL Library Update Routine	Background Compilers Reference Manual Utilities Reference Manual
COBOL SORT Facility	COBOL Reference Manual
COBOL Version III Background Compiler	Background Compilers Reference Manual

(Continued)

TOPIC	DOCUMENT
COBSYN	COBSYN User Guide
CODE	CODE Reference Manual
COMTY Macro	Data Management System Reference Manual
CONNECT Macro	Executive Macros Reference Manual
Console Typeouts	Operators Guide
Control Program Generation	SYSGEN Reference Manual
COPY Command	Data Management System Reference Manual
COPY Macro	Data Management System Reference Manual
Core Saturation	System Controllers Guide
CPCI Macro	Data Management System Reference Manual
CSTAT Macro	Executive Macros Reference Manual
DATA Command	Executive Command Language Reference Manual
Data Management	Data Management System Reference Manual
DCT-2000, Description	Terminal User Aid — Terminals
DDEV Macro	Data Management System Reference Manual
DDRL TDOS Routine	Operators Guide
DEACT Macro	Executive Macros Reference Manual
Device Error Recovery	Data Management System Reference Manual
Desk Calculator	Terminal User Aid — Desk Calculator
Diagnostic Routine	Assembler Reference Manual Utilities Reference Manual
DISPLAY Command	IDA Reference Manual
DMODE Macro	Data Management System Reference Manual
DMS Commands	Data Management System Reference Manual
DMS-Supported TOS Run-Time Parameters	Data Management System Reference Manual
DO Command	Executive Command Language Reference Manual
DPAGE	Utilities Reference Manual
DROP Command	Data Management System Reference Manual
DTYPE Macro	Data Management System Reference Manual
DUMP Command	IDA Reference Manual

(Continued)

SYSTEM INDEX (Continued)

TOPIC	DOCUMENT
DXC	Operators Guide
Dynamic Linking Loader	Utilities Reference Manual
EAM	Data Management System Reference Manual
EBCD Macro	Executive Macros Reference Manual
EDT	EDT Reference Manual
END Command	Executive Command Language Reference Manual
ENDP Command	Executive Command Language Reference Manual
ENTER(E) Command	Executive Command Language Reference Manual
ENTER File	Executive Command Language Reference Manual
EOF Command	Executive Command Language Reference Manual
ERASE Command	Data Management System Reference Manual
ERASE Macro	Data Management System Reference Manual
ERFLG Macro	Executive Macros Reference Manual
EXCOM Macro	Executive Macros Reference Manual
EXCP Macro	Data Management System Reference Manual
EXCPW Macro	Data Management System Reference Manual
EXECM Macro	Executive Macros Reference Manual
EXECUTE (EXEC) Command	Executive Command Language Reference Manual
EXECUTIVE Command	Executive Command Language Reference Manual
EXECUTIVE Macros	Executive Macros Reference Manual
EXIT Macro	Executive Macros Reference Manual
EXITP Macro	Executive Macros Reference Manual
EXLST Macro	Data Management System Reference Manual
EXRTN Macro Instruction	Data Management System Reference Manual
FCB (File Control Block) Macro	Data Management System Reference Manual
FBUF Macro	Executive Macros Reference Manual
FETCH Macro	Executive Macros Reference Manual
File Access Methods	Data Management System Reference Manual
FILE Command	Data Management System Reference Manual
FILE Editor	File Editor Reference Manual

(Continued)

SYSTEM INDEX (Continued)

TOPIC	DOCUMENT
FILE Macro	Data Management System Reference Manual
File Management	Data Management System Reference Manual
FILES Command (RTP)	Data Management System Reference Manual
File Security	Data Management System Reference Manual System Controllers Guide
FLOAD Macro	Executive Macros Reference Manual
FORM Macro	Executive Macros Reference Manual
FORTRAN Compilations	Background Compilers Reference Manual
FORTRAN Compilations	FORTRAN Reference Manual
FORTRAN Language Characteristics	FORTRAN Reference Manual
FORTRAN Version III Compiler	Background Compilers Reference Manual
FSTAT Macro	Data Management System Reference Manual
FSTATUS Command	Data Management System Reference Manual
GBUF Macro	Executive Macros Reference Manual
GDATE Macro	Executive Macros Reference Manual
GEPRT Macro	Executive Macros Reference Manual
GET (file) Processing	Data Management System Reference Manual
GETOD Macro	Executive Macros Reference Manual
GETSW Macro	Executive Macros Reference Manual
GTMAP Macro	Executive Macros Reference Manual
HELP Command	Messages To Terminals
HOLD Command	Data Management System Reference Manual
IBM 2741, Description	Terminal User Aid — Terminals
IBM 2741, Operation	Terminal User Aid — Terminals
IDA	IDA Reference Manual
IF Command	IDA Reference Manual
IFOR Language Characteristics	IFOR Reference Manual
IHBPXR Command	Operators Guide
INALTRAN Program	SYSGEN Reference Manual
Interactive FORTRAN	IFOR Reference Manual

(Continued)

SYSTEM INDEX (Continued)

TOPIC	DOCUMENT
Internal Symbol Dictionary	Utilities Reference Manual
INTR Command	Executive Command Language Reference Manual
ISAM	Data Management System Reference Manual
JOIN COMMAND	EXECUTIVE Command Language Reference Manual System Controllers Guide
JOIN File	System Controllers Guide
LIBRARY Maintenance ROUTINE	Utilities Reference Manual
LINKAGE EDITOR	Utilities Reference Manual
LNTYP Macro	Executive Macros Reference Manual
LOAD Command	Executive Command Language Reference Manual
Load Library Formats	Utilities Reference Manual
Load Module Transcriber	Utilities Reference Manual
LOADM Macro	Executive Macros Reference Manual
LOADER UPDATE (LDRUPD)	Utilities Reference Manual
LOGOFF Command	Executive Command Language Reference Manual
LOGON Command	Executive Command LAnguge Reference Manual
LPOV Macro	Executive Macros Reference Manual
Macro Libraries	Utilities Reference Manual
MACRO LIBRARY UPDATE (MLU)	Utilities Reference Manual
MATRIX Operations	BASIC User Manual
MERGE OBJECT MODULES (MOM)	Utilities Reference Manual
MESSAGE Command	System Controllers Guide
MESSAGE (MES) Command	Executive Command Language Reference Manual
MESSAGE PROCESSING	Utilities Reference Manual
MESSAGES, TERMINAL	Messages To Terminals
MONTB Macro	Executive Macros Reference Manual
MOVE Command	IDA Reference Manual
NOLNS Macro	Executive Macros Reference Manual

(Continued)

TOPIC	DOCUMENT
Object Module Library Formats	Utilities Reference Manual
Object Program Formats	Utilities Reference Manual
OPEN (file) Processing	Data Management System Reference Manual
PAM	Utilities Reference Manual
PARAMETER (PARAM) Command	Executive Command Language Reference Manual Background Compilers Reference Manual
PASS Macro	Executive Macros Reference Manual
PASSWORD Command	Data Management System Reference Manual
Passwords, Data Files	Data Manangment System Reference Manual
Passwords, User-id	Executive Command Language Reference Manual
PAUSE Command	Executive Command Language Reference Manual
PDUMP	IDA Reference Manual
PDUMP Macro	IDA Reference Manual
PDUMP Procedure	Operators Guide
POST Compilation Routines	Background Compilers Reference Manual
PRIME Macro	Executive Macros Reference Manual
Primitive Access Method (PAM)	Data Manangement System Reference Manual
Principal File	File Editor Reference Manual
PRINT Command, Spoolout	Utilities Reference Manual
PRIORITY Command	Executive Command Language Reference Manual
PRIORITY Command	Operators Guide
PRIORITY Command	System Controllers Guide
PROCEDURE (PROC) Command	Executive Command Language Reference Manual
Procedure File	Executive Command Language Reference Manual File Editor Reference Manual
Program Debugging, On-Line	IDA Reference Manual
PROPAGATE Command	IDA Reference Manual
PROUT Macro	Executive Macros Reference Manual
Public Volume	Data Management System Reference Manual System Controllers Guide

(Continued)

TOPIC	DOCUMENT
PUNCH Command, Spoolout	Utilities Reference Manual
PUT (file) Processing	Data Management System Reference Manual
QUALIFY Command	IDA Reference Manual
Queue File	System Controllers Guide
QUIET Macro	Data Management System Reference Manual
Random Access Volume Initializer (VOLIN)	Utilities Reference Manual
Random Access Pam-Oriented Support (RAPOS)	Utilities Reference Manual
RCARD (RC) Command	Executive Command Language Reference Manual
RCARD Command	Operators Guide
RDATA Macro	Executive Macros Reference Manual
RDCRD Macro	Executive Macros Reference Manual
RELEASE Command	Data Management System Reference Manual
REL Macro	Data Management System Reference Manual
RELM Macro	Executive Macros Reference Manual
REMARK Command	Executive Command Language Reference Manual
Remote Batch Processing Commands	Executive Command Language Reference Manual
REMOVE Command	IDA Reference Manual
REP Cards	Utilities Reference Manual
REQM Macro	Executive Macros Reference Manual
RESUME Command	IDA Reference Manual
RETRN Macro	Executive Macros Reference Manual
RJOB Command	Executive Command Language Reference Manual
RLOGON Command	Executive Command Language Reference Manual
RLOGOFF Command	Executive Command Language Reference Manual
ROUT Command	Executive Command Language Reference Manual
RSTART Command	Executive Command Language Reference Manual
RSTATUS Command	Executive Command Language Reference Manual
RSTOP Command	Executive Command Language Reference Manual
RMSG Command	Executive Command Language Reference Manual

(Continued)

SYSTEM INDEX (Continued)

TOPIC	DOCUMENT
SAM	Data Management System Reference Manual
SAVE Macro	Executive Macros Reference Manual
SECURE (SEC) Command	Executive Command Language Reference Manual
SEND Macro	Executive Macros Reference Manual
SET Command	IDA Reference Manual
SETBF Macro	Executive Macros Reference Manual
SETIC Macro	Executive Macros Reference Manual
SETSW Command	Executive Command Language Reference Manual
SETSW Macro	Executive Macros Reference Manual
SETUP Command	Data Management System Reference Manual
SETUP Command	Operators Guide
SEVER Command	Executive Command Language Reference Manual
SEVER Command	System Controllers Guide
SHARE Command	System Controllers Guide
SHARE Command	Executive Command Language Reference Manual
SHIDER	Utilities Reference Manual
SHUTDOWN Command	System Controllers Guide
SHUTDOWN Command	Executive Command Language Reference Manual
SHUTDOWN Command	Operators Guide
SHUTM Macro	Executive Macros Reference Manual
SKIP Command	Executive Command Language Reference Manual
SLI Typeouts and Operator Responses	Operators Guide
SMODE Macro	Data Management System Reference Manual
SORT/MERGE, TSOS	Utilities Reference Manual
SORT Facility, COBOL	COBOL Reference Manual
SPEXT Macro	Executive Macros Reference Manual
Spool-out	Utilities Reference Manual
Spool-in	Utilities Reference Manual
SPRG Macro	Data Management System Reference Manual
SPSEQ Macro	Executive Macros Reference Manual

(Continued)

SYSTEM INDEX (Continued)

TOPIC	DOCUMENT
START Command	Operators Guide
Static Loader	Utilities Reference Manual
Statististical & Historical I/O Error Rates (SHIOER)	Utilities Reference Manual
STATUS Command	Executive Command Language Reference Manual
STATUS Command	Operators Guide
STATUS Command	System Controllers Guide
STEP Command	Executive Command Language Reference Manual
STOP Command	IDA Reference Manual
STUT1 Macro	Executive Macros Reference Manual
STUT2 Macro	Executive Macros Reference Manual
STXIT Macro	Executive Macros Reference Manual
SVC 83 Macro	Executive Macros Reference Manual
SYSDTA Command	Executive Command Language Reference Manual
SYSFILE Command	Executive Command Language Reference Manual
System Controller's DISPLAY Program	System Controllers Guide
SYSIPT	Executive Command Language Reference Manual
SYSLST	Executive Command Language Reference Manual
SYSOUT	Executive Command Language Reference Manual
SYSRES	Executive Command Language Reference Manual System Controllers Guide
System Catalog	SYSGEN Reference Manual
System Controller's Responsibilities	System Controllers Guide
System Generation	SYSGEN Reference Manual
SYSUPD, System Update Program	SYSGEN Reference Manual
TABLE Macro	Utilities Reference Manual
Tape Label Processing	Data Management System Reference Manual
Tape Loader	SYSGEN Reference Manual System Controllers Guide
Task Management	Executive Command Language Reference Manual
TASKLIB	Utilities Reference Manual
Teletype Operation	Terminal User Aid — Terminals

(Continued)

TOPIC	DOCUMENT
Temporary COBOL Library Transcriber and Update	Utilities Reference Manual
TERM Command	Operators Guide
TERMD Macro	Executive Macros Reference Manual
Terminal Messages	Messages To Terminals
Terminals, Use of	Terminal User Aid — Terminals
TERMJ Macro	Executive Macros Reference Manual
TMODE Macro	Executive Macros Reference Manual
TOCOM Macro	Executive Macros Reference Manual
TOS Load Library Formats	Utilities Reference Manual
TOS Load Module Transcriber	Utilities Reference Manual
TOS Macros	Data Management System Reference Manual
TSOS Accounting	System Controllers Guide
TSOS Loaders	Utilities Reference Manual
TSOSMT, MASTAP Update Routine	SYSGEN Reference Manual
TPLAB Command (RTP)	Data Management System Reference Manual
TYPE Command	Executive Command Language Reference Manual
TYPE Macro	Data Management System Reference Manual
TYPIO Macro	Data Management System Reference Manual
UNARY OPERATORS (Desk Calculator)	Terminal User Aid — Desk Calculator
Utility Routines	Utilities Reference Manual
VDC Command (RTP)	Data Management System Reference Manual
VDT Operation	Terminal User Aid — Terminals
VOL Command (RTP)	Data Management System Reference Manual
VOLIN	Utilities Reference Manual
Volume Concepts	Data Management System Reference Manual
VPASS Macro	Executive Macros Reference Manual
VTOC Formats	Data Management System Reference Manual
WAIT Macro	Data Management System Reference Manual
WAITM Macro	Executive Macros Reference Manual

(Continued)

SYSTEM INDEX (Continued)

TOPIC	DOCUMENT
WREND Macro	Executive Macros Reference Manual
WRLST Macro	Executive Macros Reference Manual
WROUT Macro	Executive Macros Reference Manual
WRTRD Macro	Executive Macros Reference Manual

Title	TSOS Programming Sysem Information Manual
Document No.	DJ-000-1-00
Date	February 1971

RCA Computer Systems

Your comments and suggestions will help us to furnish publications that are more useful to you.

Is this publication:

Complete in its coverage?

Logically organized?

Technically accurate?

Easy to understand?

Other Comments (Use additional page if necessary).

Name ______________________ Street or Box No. ______________________

Job Title ______________________ City ______________________

Company ______________________ State __________ Zip __________

Fold

FIRST CLASS
PERMIT NO. 16
CAMDEN, N. J.

BUSINESS REPLY MAIL – no postage necessary if mailed in the United States

Postage will be paid by addressee

RCA | COMPUTER SYSTEMS DIVISION
CHERRY HILL, 204-2
CAMDEN, N. J. 08101

ATTN: Publication Services

Fold

Publications Purchase Order

RCA|Computer Systems Division
Camden, N. J. 08101

RCA

Order Number

Item No.	Quantity Ordered	Ordering Number	Description or Title of Material	Complete if Publications Are To Be Purchased	
				Unit Price	Totals
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					

Ship To:

Bill To: (Complete if Publications Are To Be Purchased)

Customer P.O. #

Sub Total	
Indicate Applicable Sales Tax	
Total Cost	

Check Appropriate Block:

Bill My Company | |

Remittance Enclosed | |

If Publications are to be purchased forward this Form and any enclosures to:

RCA|Computer Systems Division
Reproduction Services
Camden, N. J. 08101

Publications purchased will be furnished subject to all terms and conditions stated on the reverse side of this Form.

Ship via

Date Required

Authorized Signature

Date

All other requests: Forward this form to the nearest RCA District Office.

27-41-002

9/70

Terms And Conditions
Applicable To The Sale of RCA Publications and Forms

RCA

PRICES

All prices are subject to change or withdrawal without notice and all shipments will be billed at prices in effect on date of shipment. Unless otherwise specified or required by law, all prices will be billed exclusive of state and local sales and similar taxes, and such taxes will appear as additional items on invoices.

TRANSPORTATION

All shipments will be f.o.b. destination.

On shipments where Purchaser requests transportation involving expenses beyond those involved on transportation normally selected by RCA, the Purchaser will be responsible for payment of such extra costs.

RCA reserves the right to ship from any location subject to the foregoing transportation terms.

DELIVERIES

It is the desire of RCA to meet requested delivery schedules. However, RCA shall not incur any liability due to any delay or failure to deliver for any reason. Any delivery indication furnished by RCA only represents the best estimate of the time required to make shipment. The delivery of part of any order shall not obligate RCA to make further deliveries, and RCA reserves the right to decline servicing any order in whole or in part.

Of necessity, inventories and current production must be allocated in such a manner as to comply with applicable Government regulations. In the absence of such regulations, RCA reserves the right to allocate inventories and current production when, in its opinion, such allocation is necessary.

TERMS OF PAYMENT

Invoices shall be rendered at time of shipment and shall be payable net 30 days from date of shipment.

Partial shipments will be invoiced as made, and payments therefor are subject to the above terms.

GENERAL

In no event shall RCA be liable for indirect, consequential or special damages.

Information furnished by RCA is believed to be accurate and reliable. However, no responsibility is assumed by RCA for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of RCA.

This Agreement shall be governed by the laws of the State of New York and constitutes the entire Agreement between the parties with respect to the subject matter hereof. It shall prevail regardless of any variation in the terms and conditions of any other submitted by the Purchaser.