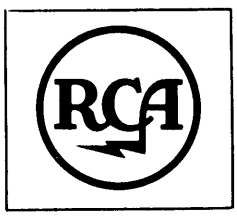
SPECTRA 70

RADIO CORPORATION OF AMERICA • ELECTRONIC DATA PROCESSING

SPECTRA 70
TIME SHARING
OPERATING SYSTEM
INFORMATION MANUAL



TIME SHARING
OPERATING SYSTEM

INFORMATION MANUAL

RADIO CORPORATION OF AMERICA

79-00-513
March 1968

First Printing: March 1968

CONTENTS

PREFACE

The RCA Time Sharing Operating System (TSOS) is designed for the Spectra 70/46 Time Sharing Processor. It provides access to the 70/46 computer for up to 48 simultaneous system subscribers. The terminal devices used by system subscribers may be connected to the 70/46 computer center by direct lines, leased lines, and/or message network.

TSOS provides a full range of source languages, user facilities, and debugging aids to permit any individual, novice or advanced, to engage in successful problem solving activities and program preparation and testing from remote locations. Included are source languages for both conversational programming and nonconversational batch programming, a desk calculator program, and file editing commands.

This document covers general information on time sharing's development and important role in data processing and specific information on the RCA Spectra 70/46 TSOS.

THE EVOLUTION OF COMPUTER OPERATING SYSTEMS

Computing technology has been in a continual evolution ever since the first electronic data processor was set into operation. This paper traces through the significant and pertinent steps of computer evolution for the sake of those who must now judge and evaluate the current scene. Briefly, the scope of computing systems has expanded from serving the single user with a single problem to the servicing of many users demanding and receiving computer access. In order to handle the multi-user situation, there has been extensive development in software including both programs which solve specific problems and those that control the entire computing facility. In addition to solving problems for the users, the computational ability of the present-day computers is also used to control and regulate the flow of problems through a computer installation with minimum human intervention.

This trend has grown chiefly from the need to achieve greater efficiency from the equipment and to make better use of human resources. We can neither afford to have computing equipment stand idle nor can we waste the time of skilled personnel by making them wait interminable periods of time to get back a solution, or 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 problems requiring immediate service. On the other hand, the persons who have been concerned with achieving higher efficiency from both mechanical and human resources have been concerned with two other problems. First, the typical computing installation has, in addition to problems demanding immediate attention, a number of problems 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 must not be made 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. 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. Other 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.

HISTORICAL EVOLUTION

When the 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.

IMPROVING UTILIZATION

One of the chief disadvantages of the single program execution philosophy is that the equipment is rarely fully utilized. Programs which make great demands on the internal data processing elements usually require the input/output equipment only a small percentage of the time. On the other hand, programs which required the input/output system 100% of the time usually permit the data processing elements of the computer

to idle along at a fraction of its capacity. But idle equipment was not the only drawback to single program execution. Each time a person wished to run a program he had to go through the same long procedure. He was required to prepare his program in a suitable 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 were processed in the order that they were 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 called turnaround time.

TURNAROUND TIME

Turnaround time under the single program execution philosophy became a matter of many hours and often days. One of the sources of lengthy turnaround time was the human operator who had to read the operating instructions associated with each job and determine that the requirements of the program were met. But the human operating speed is not a match for the processing speeds of the computer. Another source of slow-down, again involving the operator, was the error problem. The error might be a machine failure, a failure on the part of the programmer who wrote the operating instructions, or even a blunder 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 hunt down and 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 STREAM

In order 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. At the computer there was also standardization. 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 was sent to a standard output device. The computer under this standardization could now begin one problem and go right on to a second and a third program without interruption. There came into being a 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 function 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 human operator. This 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 human operator and improved equipment utilization. The function of the human operator changed from being an essential intermediary to being an assistant to the computer.

OPERATING SYSTEM

An operating system then, is a programmed method for making the decisions that, in the early days of computing, were left to a human operator. In addition to providing standard responses to standard error conditions (both in the programs and in the computing equipment itself), the operating system must also 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

In spite of the improvements in equipment utilization made so far, there still remained the situation in which a given program did not fully occupy all of the equipment all of the time: one program loaded the input/output system fully while another was limited by the internal data processing unit. Was there any way of fully loading all the equipment? In a large data center there are usually many different types of programs which require varied resources. Under the control of an expanded operating system, several programs could be loaded at one time. One of the programs is started; the operating system then 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 available 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 most effective use of all equipment under any given situation. The multiprogramming operating system reduced the average turnaround time somewhat, but it created some new problems of its own. One problem 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 might have to supply more information about his program than previously in order to enable the operating system to control it. The solution of any problem that required deviation from the standard methods become 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 slightly, there was still no direct communication between the computer system and the programmer -- or the non-programmer such as the scientist or engineer who had a problem to solve.

COMMUNICATIONS

At the same time that multi-programming operating systems are being developed, the capabilities of computing systems were being extended by having communication equipment attached to them. There are three principal applications that found the communications devices necessary. One of these was data gathering. Many different types of devices could be attached to a computer through which it could receive or send out data.

Typically, these might be telephone wires, telemetering equipment, or time card stations at the entrances to a large factory. Another form of communications application is message switching. One station can initiate a message and send it to any one or more other stations. Finally, there is 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 have 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 is used less for processing and more for handling data during the communication process.

INTERACTIVE SYSTEMS

The combination of multiprogramming and communications equipment leads quite naturally into the next stage of operating system evolution: the interactive system. The interactive operating system differs functionally from the multi-programming operating system in the control of multiple user tasks within a given time frame. Multiprogramming allows two or more users access to a computer but discriminates between the users through the use of a priority scheme; on the other hand, 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 can be entered at a remote terminal, transmitted to the computer, and, finally, elicit a response from the computer. Moreover, the inquiry/response system does not permit the inquirer to do anything more than request information from a single collection of data or to send new information for incorporation 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 alter existing programs. In fact, the user of an interactive system has available to him at his terminal all of the capabilities of the complete computing system.

There is, however, a difference. 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 will be 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 important, he may be able to get information answers, or reports in minutes instead of hours. This type of operation is completely feasible with presentday equipment because the speeds of operation within the computer are sufficiently fast 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 seems 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 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 multi-programming or job-stream systems, or even, in the early days, without any operating system at all.

BENEFITS OF INTERACTIVE SYSTEM

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 request by the user 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.

RCA TSOS PHILOSOPHY

RCA has developed an interactive, multiprogrammed operating system call TSOS. This operating system is designed for the 70/46 Spectra computer, a modified form of the 70/45. Together, the computer and operating system offer a package that has application in many computing environments. TSOS combines with its interactive capabilities complete facility to handle a full range of production-type tasks. Whatever resources are not occupied by interactive requests are given to 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, but have no need for any interactive operation. Such programs as payroll, inventory, billing and sales analysis runs are examples.

TSOS and the 70/46 can provide greater production processing capabilities than any other equipment in the present Spectra 70 series. It can provide interactive service for up to 48 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 interactive 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 resources.

LANGUAGES

The 70/46 TSOS offers the languages of BASIC, COBOL, RPG, FORTRAN, Interactive FORTRAN, and the Macro-Assembler. BASIC and interactive FORTRAN are especially designed for the creation and building of programs by conversational techniques at a remote terminal. COBOL, FORTRAN, RPG, and the Macro-Assembler are compatible with their TOS equivalents, and can be used both at terminals and for writing batch programs.

COMMONLY USED TIME SHARING TERMS

The following terms and definitions were purposely not included as a glossary because an understanding of their usage in this document is a prerequisite for the reader. Many individuals have been exposed to these terms, however, it is felt that a mutual understanding of them prior to reading the balance of this manual will assist in the comprehension of the following sections:

Batch Processing - In TSOS, the noninteractive processing of tasks controlled by a sequence of commands (either terminal issued or prestored); once control is given to the system scheduler, no information is transmitted from the system to the remote terminal.

Catalog - (N.) An indexed file consisting of file names and their location indexes. (V.) To include the name and location index of a file in an indexed file.

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.

Conversational - Describes the user/system dialogue in which a user at a terminal device and the time-sharing system communicate with each other using languages developed especially for this purpose, each interrogating the other and responding to the other's requests.

Data Management - The functions in TSOS of management of files and control of all problem program input/output.

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

File - A named collection of related records.

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

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 controlled by an executive routine that allows interleaved operation of two or more independent programs residing in one processor.

Non-conversational - Refers to the type of operation of a time sharing system in which there is no dynamic communication between user and system. Input and instructions are all specified in advance and output and diagnostics are produced at the central installation.

Non-privileged - The commands and facilities available to any terminal subscriber.

On-Line - A reference to any device directly connected to and controlled by a computer.

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.

Page - A set of 4096 consecutive bytes. Main memory in the 70/46 is divided into 64 pages, each beginning on a multiple of 4,096 bytes. Virtual memory contains 512 pages.

Paging - The software controlled process of interchanging pages in main memory and pages from the paging drum.

Privileged - A term used to describe any part of the system (commands or facilities) that can not be accessed by the terminal user.

Process Control - A control system used for monitoring and controlling the functions of a particular process.

Program Management - The function in TSOS of control of all user programs, facilitated by the separation of programs into classes.

Read Only - An attribute assigned to a file that is not permitted to be modified by a user during execution.

Real Time - The measurable time required for a given physical even to occur.

Real Time System - A computer system controlling a physical process in such a way that the system is able to respond properly to the process in a predictable length of time.

Reentrant - Describes a program that can be entered at any time by more than one user with no loss of information to any of the users.

Reusable - Describes a self-initializing program that can subsequently be entered by another user without reloading the program.

Segment - An area of contiguous virtual or permanent memory consisting of 64 pages.

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

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

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.

Task - All work undertaken by the system in response to commands issued by the user between 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.

Time Sharing - A system that allows multiple users to simultaneously access a computer from remote terminals; system schedule is based on time slices.

Time Slice - A quantum of time assigned to tasks.

Virtual Address - A location in extended main memory subject to segment translation when transferring information from virtual storage to main memory. (Extended main memory in the case of TSOS refers to the systems high speed drum which is a privileged device.)

Virtual Storage - Address space contained on the high speed drum that is available to users.

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.

DESCRIPTION OF THE 70/46 HARDWARE

The RCA 70/46 Time Sharing Operating System (TSOS) is designed to serve the batch processing and interactive user by making 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.

TSOS is a total systems design in that it represents an integration of time sharing hardware and software. A standard hardware configuration as shown in figure 1 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 micro-logic functions

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.

The RCA 70/46 Processor contains five memory components:

Main Memory

Non-Addressable Main Memory

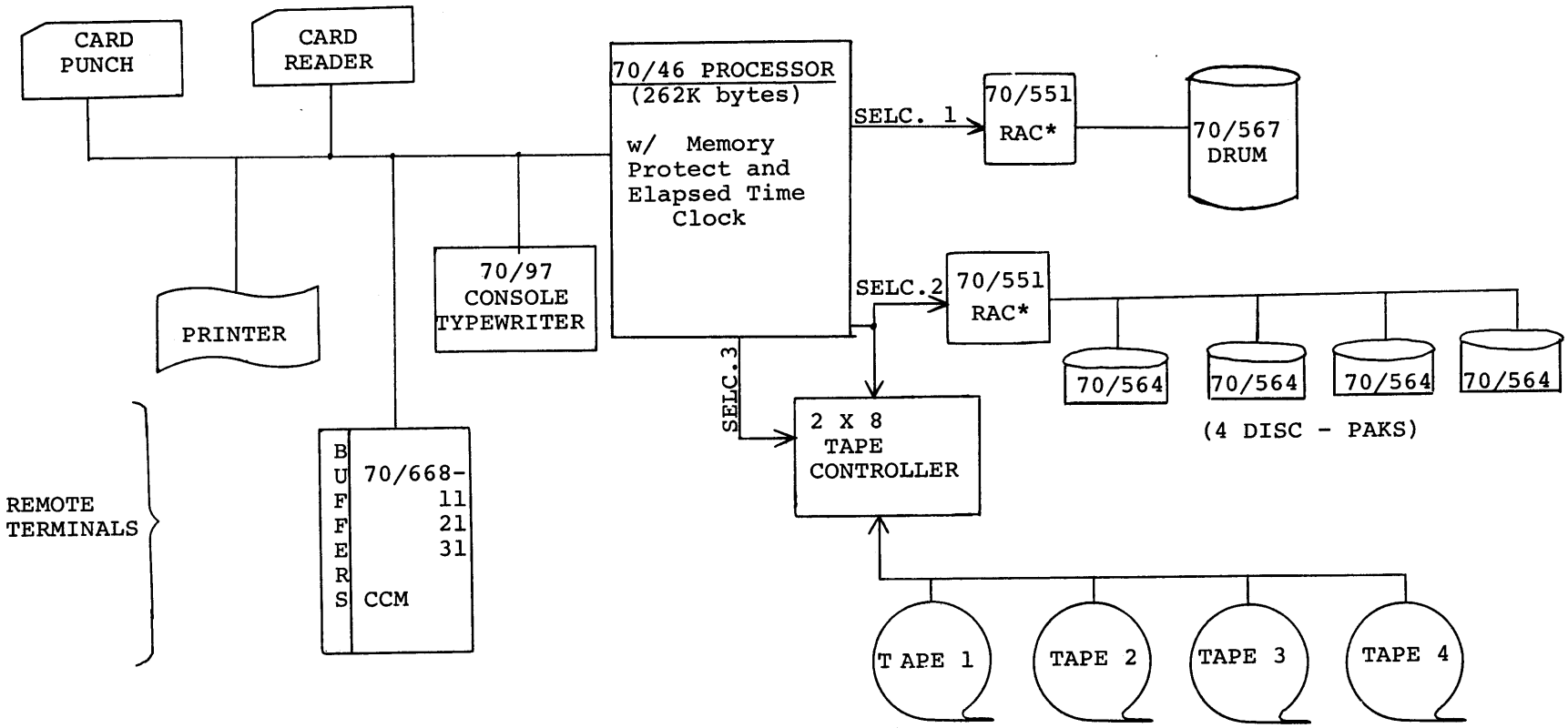
Scratch-Pad Memory

Translation Memory

Read-Only Memory

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

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.



*Neither filescan nor record overflow is required.

FIGURE - 1 RCA SPECTRA 70/46 - STANDARD CONFIGURATION

Scratch-Pad 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 scratch-pad memory is uniquely addressable. Scratch-pad memory provides general address registers, program counters, and the registers required for both interrupt analysis and input/output servicing.

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.

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.

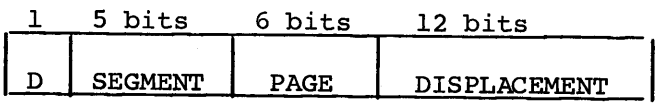
PAGING AND SEGMENTATION

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. At the discretion of the software, a 2,048 byte page size can also be used. This shorter page length makes it possible to pack physical main memory more tightly as well as reducing the transfer time between auxiliary storage and main memory for short routines that do not require a full page of storage space. Use of the 2,048 byte page, however, reduces the available virtual main memory space by half since the virtual memory addressing scheme provides for only 512 pages regardless of whether they are 4,096 bytes or 2,048 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 provides a total of two million bytes of virtual storage.

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



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.

TSOS SOFTWARE DESIGN

The RCA 70/46 Time Sharing Operating System (TS)S), is designed to control a medium-size, time sharing and multiprogramming batch system that will support up to 48 conversational and nonconversational user tasks simultaneously. Early design efforts were expended to determine the effects of the following two design goals on the scheduling and paging functions of the system:

A medium-size system intended primarily as a multiprogramming system that can also accommodate 48 conversational users.

System should have a reasonable response time, especially for conversational tasks, under maximum load conditions.

Results of simulation runs pointed the way toward the task scheduling and paging algorithms which would produce the most effective use of the processor. The basic goals of the scheduling and paging function were expanded to:

Provide conversational users with as fast a response time as possible.

Have at least one resident task in core to utilize processor time that would otherwise be idle when the paging mechanism is taxed.

Ensure that each task obtains an equitable share of processor time.

Ensure that the number of simultaneously active processing tasks will not oversaturate virtual memory.

Efficiently utilize the system's resources by delaying time-consuming operations consistent with system response times.

The RCA TSOS design offers the user an efficient medium-size batch processing and time sharing environment.

TSOS SOFTWARE DESCRIPTION

On the software side, the 70/46 TSOS creates an appropriate system environment to satisfy the following user requirements:

Convenient, direct access to the computer for many simultaneous users with facilities for dynamic interaction with an executing program.

Enhanced facilities for efficient batch processing in a multiprogramming random access environment.

Rapid response from the computer to reduce the overall time between problem definition and solution.

The system has the important characteristic of compatibility with the RCA Tape Operating System (TOS). This permits the user to progress easily and naturally from TOS to a more sophisticated random access oriented operating system (TSOS).

The system's multiprogramming facilities are more elaborate than those provided by TOS, and employ the following techniques:

Virtual Storage Management.

Use of a drum to retain frequently used portions of the supervisor, control programs, and system tables.

Spooling (buffering) of conventional card input files and printer destined output files on random access devices (or tape).

Construction of a single task queue from multiple input sources, and concurrent execution of as many of these tasks as the resources of the system allow. Nonconversational tasks may be entered into the task queue from remote terminals.

The considerably enhanced file management facilities in TSOS provide for:

Cataloging of all files in the system.

Specification by the user of file names and characteristics at execute time, or if he chooses, at assembly or compile time.

Sharing of files when specified by the user and system protection of the user's private files through passwords.

Indexed sequential access for files retained on random access devices.

Transcribing infrequently used or inactive files to or from the Mass Storage Unit.

Because convenient program preparation and testing is an important goal, the system includes facilities for:

On-line preparation and editing of files including symbolic programs and data by means of a powerful file editor.

Interactive, line-at-a-time compiling and interpretive execution of programs in a FORTRAN IV related language.

System direction through a flexible and easy-to-use command language, which includes provisions for prestored command procedures.

Program checkout with symbolic debugging commands which permit the user to request data from his program, to modify variables, and to specify locations in the program and conditions under which deferred commands are to be performed.

The command language permits individuals using the system to identify themselves, to specify work for the system, and to monitor that work. It is the principal means of communication between TSOS and individuals who use the system. These may be:

Users

System Administrators

System Operators

When granted access to the system, an individual is assigned one or more command privilege classes which determine the commands he is authorized to issue to the system.

System Administrators use the facilities of the command language to authorize user access to the system and to maintain and retrieve records of system use for accounting and administrative purposes.

Users have access to a large set of commands. Many will use only a small subset of those available; others, such as programmers, may employ them all at one time or another.

Operators use the command language to adjust the configuration of the system, to issue messages to users, and to perform various housekeeping chores on I/O devices in response to system messages.

DEFINITION OF A TASK

A task is defined as the unit of work specified by an individual to the system between the receipt of the LOGON and LOGOFF commands. Tasks may be run in either the conversational or non-conversational mode.

SYSTEM SUMMARY

TSOS is composed of three types of programs, each of which contributes to the functions of the system:

Supervisor

Privileged system service routines

Problem programs (some RCA-supplied, such as language processors; others user-written)

The classification of programs in the system is summarized in the following chart:

SYSTEM SUMMARY CHART

Privileged Programs	Non-Privileged Problem Programs
Supervisor Memory management Device handling Interrupt analysis Dispatching of control Statistics gathering System Service Routines Command language processing IDA (Interactive Debugging Aid) Resource allocation File catalog maintenance Bulk I/O facilities Problem program I/O facilities Terminal I/O facilities	Language Processors FORTRAN Macro Assembler Report Program Generator COBOL Interactive Basic Interactive FORTRAN Service Programs File Editor Assembly Diagnostic COBOL Syntax-Checker Deck Calculator Linkage Editor Sort/Merge Utility programs User Written Problem Programs

SUPERVISOR

The supervisor controls the execution of tasks and controls the hardware environment in which they operate. Although few of its operations are apparent to the typical user of the system, the supervisor is of interest because it manages the system environment.

The supervisor's basic function is to receive interrupts, sort them as to type and function, and initiate the appropriate routines to respond to each. By means of time slicing, it provides rapid and complete service to a number of users concurrently.

SYSTEM SERVICE ROUTINES

Each task in the system requires auxiliary service routines for its execution.

A set of privileged service routines provides task management services such as:

Recognition and interpretation of system commands in the task's input stream, and initiation of the action required for each command.

Communication with terminals and the system operator.

Allocation of memory space required by a task.

Allocation and release of I/O devices.

Loading of problem programs.

Other privileged service routines provide the following file management facilities:

Cataloging of files.

File processing such as printing and punching.

Controlling data access -- checking for authorized access and controlling the sharing of data between users.

Loading and activation of required data access routines in response to problem program requests for data location or transfer.

Both the supervisor and the system serivce routines are privileged programs.

PROBLEM PROGRAMS

The problem programs in TSOS are classified as non-privileged; this is true for both RCA supplied programs as well as the user programs. The following list contains the RCA-supplied problem programs:

Language Processors

Nonconversational

FORTRAN IV
COBOL

TSOS Assembler
Report Program Generator

Conversational

FORTRAN IV
BASIC
Desk Calculator

Service Programs that may be used either conversationally or nonconversationally:

IDA (Interactive Debugging Aid)
COBOL Syntax Checker
File Editor
Linkage Editor

Other service programs include:

Sort/Merge

Utility Programs

All of these language processors and service programs are described further in a subsequent section entitled Program Management.

TSOS COMPONENTS

TSOS is a complex operating system that can be described from many standpoints. To the system user (at a remote terminal) the terminal/computer interaction is of prime consideration. Requirements such as programming facilities, data storage, system debugging aids . . . must be convenient to use and easily understood; and more, the terminal functions must be broad based enough to allow the novice to progress easily to more sophisticated programming techniques and data file structures. In order to acquaint you with the TSOS programming and data facilities, the system will be subdivided into the following categories:

TASK Management

DATA Management

PROGRAM Management

SYSTEM Management

The first three; TASK, DATA, and PROGRAM Management will be discussed from the standpoint of the terminal user. The last, SYSTEM Management, will provide the reader with the insight into some of the administrative functions of TSOS and clarify the need for system attributes such as security, file classification, accounting structure and so forth.

TASK MANAGEMENT

Because many tasks will be operated in both conversational and nonconversational modes, the system has been designed to minimize the differences between them from the viewpoints both to the problem program running under the system and of the user submitting the task to the system. Their intrinsic characteristic - that of either being interactive or noninteractive - requires some minor differences in operation.

A user can specify whether a program is to be run in the conversational or nonconversational mode. A program written to take advantage of this fact could, for example, prompt conversational users for input data, detect erroneous input and write diagnostic messages in the conversational mode, but abort the program when run in the nonconversational mode.

SYSTEM INPUT FILE (SYSIN)

The commands issued by the user to direct the system in the execution of his task are contained in the SYSIN file. In conversational mode, these commands are usually entered through the user's terminal. In nonconversational mode, the command procedure that directs the execution of a task is stored in a file. The SYSIN file may also include such data as source language statements to be processed by a compiler, programs in object module format to be processed by the linkage editor, commands to the file editor, and problem program input data.

PROCEDURE FILES

Control can be transferred from the primary SYSIN to a temporary SYSIN file by means of a PROCEDURE command. This command names a temporary file which is to serve as SYSIN until a LOGOFF or an ENDP command is encountered in that file. A LOGON command in the procedure file is ignored.

SYSTEM OUTPUT FILES (SYSOUT)

The system issues two types of messages to the conversational user:

Response messages generally tell the user of actions the system has taken in executing a command, or they request additional information.

Diagnostic messages inform the user of errors he has made in entering a command or its operands, and asks the user for corrections.

The SYSOUT file for a user's task consists of the system messages and the responses to command execution. If the task is in conversational mode, information placed in SYSOUT is printed immediately at the user's terminal. SYSOUT for a conversational task is not recorded by the system in any form other than the printed listing at the terminal. SYSIN and SYSOUT are thus interspersed on the terminal listing. Of course, any communications between the problem program and the user also appear on the terminal listing.

For a task executed in nonconversational mode, the information comprising SYSOUT must be stored by the system in that task's SYSOUT file. This file is printed automatically by the system when the task is terminated, and the resultant listing is obtainable at the central computer installation.

CONVERSATIONAL MODE

In conversational mode, the user communicates with the system through a terminal. The system communicates with the user by printing messages and data on the terminal printer. The system can, therefore, turn to the user to resolve any problems that arise during conversational task execution. Similarly, the user is free to change the activities requested of the system (including the execution of his problem programs) during task execution. Facilities are provided to the user for correcting individual characters in a line and also for deleting an entire line in the event that a typing error is made.

As every command is entered by the user, it is analyzed to determine if it is valid and if the user is privileged to issue it. If so, all actions requested by the command are performed before the user is permitted to enter the next command. If the command is not complete or is not valid as entered, the system issues a message to request additional user-supplied information before it is executed.

In the nonconversational mode, there can be no dynamic communication between the user and the system. The user must completely define what is to be done in the nonconversational task before task initiation. Any system messages resulting from execution of the task are placed in SYSOUT and will be received by the user in the listing from the central computer installation.

The user may request that the system add a nonconversational task to the task queue by issuing any of these commands: PRINT, WRITE TAPE, PUNCH, ENTER, or FILENAME.

CONDITIONAL EXECUTION OF COMMANDS

Task switches are provided which can be set, reset, examined, and tested either by command language statements or by macros within the user's program. Thirty-two switches are provided. All are reset to zero when a task is initiated. Switches 16 through 31 are reset to zero each time a STEP command is encountered.

The SKIP command is particularly useful when executing nonconversational tasks. Its first parameter specifies which task switches are to be tested. The second parameter indicates the next command to be executed if the task switches are set as specified.

The STEP command separates a task into steps. When one is encountered, the system resets the monitor switches (provided for TOS compatibility) and task switches 16 through 31. The STEP will be the next command executed if a preceding SKIP command so indicates, or if a program in the preceding step terminates abnormally. If abnormal termination occurs and no STEP command is present, the task is terminated.

DATA MANAGEMENT

In TSOS, a file is a named collection of related records. 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.

TSOS provides comprehensive facilities for systematic and convenient management of files. These facilities fall naturally into two categories:

File Cataloging and Management

Problem Program Input/Output

File management facilities provide the means for identifying files, for storing and retrieving them within the system, for sharing them with other users, for copying, modifying and erasing them and for defining their existence and use in the system. Problem program input/output facilities provide for the actual transfer of data to and from programs which are in execution.

FILE NAMES

A file name is a sequence of one or more simple names to identify a file. A simple name consists of from one to eight alphanumeric characters. The first character must always be alphabetic. If multiple simple names are employed to form a file name, a period must be used to separate them.

Thus INVENT, ENGINE.PAYR.MASTER, A.B.C.D. are file names composed of one, three and four simple names, respectively. A file name may not contain more than 22 simple names and may not exceed 44 characters, including the separators.

When more than one simple name is employed in a file name, the names starting from the left are considered to name the category within which the next simple name is a subcategory. This structure for file names facilitates the storage and retrieval of files based only on their names.

File names may be either fully or partially qualified. A fully qualified file name identifies an individual file and includes all the simple names of the file.

A partially qualified file name identifies a group of files by omitting one or more of the rightmost simple names of the files. The group of files referenced includes all those files having qualifiers identical to those in the partially qualified name.

VOLUME CONCEPT

A file is said to be stored in the system if it resides on one or more direct access or magnetic tape devices (volumes) and if the identification of these volumes is available in the system catalog. In TSOS, a volume may be a 70/564 removable disc pack or a reel of magnetic tape.

At system initiation time, the system operator identifies each direct access device and its associated volume as either public or private. A public volume is a direct access volume and must be mounted during the entire period of system operation.

A public volume may be used by many users concurrently.

A private volume need not be mounted during the entire period of system operation. Its use is restricted to one used and it need only be mounted when that user refers to it. A direct access volume may be a private volume, while magnetic tape volumes are always classified as private volumes.

DEVICE MANAGEMENT

Files to be cataloged can be stored on either public or private volumes. 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 may be erased. The system assumes that a user desires storage on a public volume unless he specifically asks for a private volume.

Public volumes are always mounted and available for allocation to the user's task, within the limits of public allocation established for him by his installation. Users generally make the most effective use of TSOS by storing their files on public volumes and cataloging them, when it is necessary to retain files in the system.

FILE PROTECTION AND SHARING

When a user creates a file 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 common file can be accessed by any user. 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 file name. 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 the file while it is being used. A task will not be granted read access to a file if another task is updating it. Similarly write access to a file will not be granted if any task is reading it. This interlock is maintained from the time the file is OPENED until it is CLOSED or until the task is terminated.

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 used for output or updating by any user, including the owner.

SYSTEM CATALOG (SYSCAT)

The system catalog is a special file which resides on one or more direct access devices. It is used to retain the file descriptions which must be stored within the system so that, once a file is cataloged and created, it can subsequently be located by its file name. When a user is originally JOINED to the system, he is assigned a unique user identification code. This identification is appended to the file name of every file cataloged by a user in order to distinguish his files from those of other users.

The catalog is organized into a hierarchy of indices to facilitate file retrieval. Each index corresponds to a qualification level of the file name; the number of levels is determined by the user.

Although the system catalog usually resides on the system residence volume, the user may direct that portions of his catalog be stored as separate files on other direct access volumes, either public or private. When such extension is selected, the lowest level index within the system catalog points to the volume which contains the secondary catalog. Similarly, indices in a secondary catalog can point to other secondary catalogs.

FILE ORGANIZATIONS

A file's organization defines the overall relationships of the component records into which the file is logically 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.

In TSOS, the system provides facilities for manipulating two types of files. However, the user may create his own file organization and use the basic access methods described below. The two organizations supported by the system are:

Sequential Files

Indexed-Sequential Files

SEQUENTIAL ACCESS METHOD (SAM)

The sequential access method processes logical records according to this physical sequence in the file. Logical records are retrieved by use of the GET or GETR macro instructions, which supply a logical record to the program. The access method anticipates the need for records based on their sequential order, and normally will have the desired record in storage, ready for use. Logical records are designated for output by use of the PUT macro. The program can normally continue as if the data record were written immediately, although the access method's routines may perform blocking with other logical records, and delay the actual writing until the output buffer has been filled. Buffers are automatically scheduled by the system.

The sequential access method is for the most part, device independent and allows files on both magnetic tape and random access devices to be processed.

INDEXED-SEQUENTIAL ACCESS METHOD (ISAM)

The Indexed-Sequential Access Method processes indexed-sequential files. It may be used to:

Create an indexed-sequential file in a sequential or nonsequential manner

Update records in a sequential or nonsequential manner

Retrieve the logical records of the file in a sequential or nonsequential manner

Insert new records in their proper logical sequence within the file

Delete selected records from the file

Records in an indexed-sequential file may be of fixed or variable length format. The maximum record length may not exceed the track capacity of the direct access device upon which the file is to be stored.

BASIC DIRECT ACCESS METHOD (BDAM)

BDAM is the most primitive, but most flexible of the access methods provided for direct access devices. It allows the user to create and maintain files organized in any manner he chooses.

Variations of the READ and WRITE macros allow the user to add, retrieve, and update records. In spite of this flexibility, complete file security is provided whether the file resides on a public or private volume.

The unit of data exchange between the system and the user program is the block. Only records described as fixed length unblocked or undefined will be accepted by BDAM. Any blocking or deblocking routines must, therefore, be supplied by the user. When a file is being created, the user may supply a key with each record to facilitate subsequent retrieval of the record. This key will be recorded as the hardware key portion of the record on the direct access device. Note, however, that when a file contains keys, each record must have a key and all keys must be the same length.

BASIC ACCESS METHOD FOR TAPE (BTAM)

BTAM is used to process files on magnetic tape. The block is the unit of data exchange with the problem program. If blocked records are to be processed, the user must supply his own blocking and deblocking routines. The user is also responsible for doing

his own buffer scheduling. Data may be transferred from a tape directly to a specific area in storage or from a storage area to a tape without first having to move it to a buffer.

PROGRAM MANAGEMENT

A fundamental objective of TSOS is to increase system effectiveness by treating separately two program classes. Problem programs are divided into classes for execution, depending on the services they require. If certain programming conventions are followed, however, a given program may be executed as a member of either class.

The primary advantages of the program class concept are:

Compatibility is obtained. The system will execute almost any TOS object program with minor changes. The full repertory of TOS software provided by RCA is immediately available to the TSOS user.

System throughput and efficiency is increased by keeping Class I programs resident in physical core. The total paging load on the system is reduced since programs to be run in a batch-processing environment are not paged. Normally, conversational programs which are in a constant dialogue with the user terminal are placed in Class II. Paging normally occurs in these programs, however, the disparity between terminal and processor speed tends to keep this paging overhead relatively small.

Class I programs provide the system with peripheral device growth potential. The addition of new or special purpose input/output devices can be accommodated without requiring modification of existing input/output routines (e.g., TSOS Data Management System) because full, device oriented control of input/output functions is provided through the TOS compatible physical I/O routines. In this manner Class I provides an open ended environment for input/output devices.

Class I programs are executed in 70/45 mode. Class I programs are not only kept resident in physical core but memory references within these programs are "direct" and do not employ the 70/46 address translation technique.

A user may request that all or any part of his Class II program be kept resident during its execution. Thus he may take advantage of such Class II features as increased effective memory size, dynamic storage allocation, and module sharing without paying the additional overhead of paging. The system will honor such a

request only if adequate resources are available.

The scheduling algorithm can be parameterized so that better servicing is provided to either Class I or Class II programs, as the installation deems appropriate. This is independent of the servicing specified by priorities for programs within a class.

It is important to note that programs in either class may be executed as conversational or nonconversational tasks. The consequences of running a Class I program conversationally (which normally implies time consuming dialogue with the terminal user) coupled with the program's residence characteristic, must be carefully considered by the installation. Additionally, programs in either class may have their execution deferred if additional resources (e.g., private I/O devices, additional memory) specified in a SECURE command are not available.

The class of a program may be specified at assembly or compile time, at link edit time, or at load time.

PROBLEM PROGRAM PREPARATION

The problem program preparation facilities provided for TSOS users include:

File Editor

Syntax checkers

IDA (Interactive Debugging Aid)

Language processors

Linkage Editor

Assembler Diagnostics Language

In problem program preparation, the user typically employs the File Editor and syntax checkers to prepare a source program which is then translated by a language processor. The nonconversational language processors include:

FORTRAN IV

TSOS Assembler

Report Program Generator

COBOL

These language processors convert the source program to object module form. Typically one source program is translated into one object module. Object programs produced by TSOS language processors must be processed by the TSOS Linkage Editor before they can be loaded and executed.

A TSOS user who wishes to use the system in conversational mode has access to facilities which include:

Conversational FORTRAN IV. This is an incremental compiler and interpreter which allows the user, at his terminal, to combine program preparation and execution by interacting with the conversational FORTRAN compiler. The program is compiled one statement at a time and is executed interpretively.

Conversational BASIC. An advanced BASIC interactive compiler which offers program preparation facilities.

Desk Calculator. A conversational program which simulates a commercial desk calculator using a remote terminal. This facility is described further in a subsequent section.

File Editor. In conversational mode, the user interacts with the execution of the File Editor and he can furnish input to the File Editor and inquire about the contents of the data file being processed. Typically the data file is a source program destined for processing by one of the nonconversational language processors. The File Editor is described in detail in a subsequent section.

OBJECT MODULE FORMAT

Each of the TSOS language processors produces object modules as described below. An object module consists of a series of logical records. There are five types of records and each contains a type identifier.

The five record types are:

Identifier	Title
ESD	External Symbol Dictionary Record
TXT	Text Record
RLD	Relocation Record
ISD	Internal Symbol Dictionary Record
END	End Record

A given object module will contain:

One or more ESD records

One or more TXT records

Zero or more RLD records

Zero or more ISD records

One END record (always the last record in a module)

CONTROL SECTIONS

An object module consists of one or more control sections (CSECTS). A CSECT is a sequence of instructions or data and is the smallest separately relocatable unit of an object module.

Each CSECT in an object module produced by a language processor is assigned certain attributes (or traits) by the language processor. These traits can be changed by the user through the use of the Linkage Editor TRAITS control statement.

A CSECT may have one or more of the following attributes:

Read Only

Public

Privileged

Common

SHARED LOAD MODULES

A shared load module is a routine which can be concurrently executed by several users running as Class II programs. This means that only one copy of the routine need exist physically in main memory or on the paging drum. Each user is able to execute the shared routine as though a copy of it were loaded into his virtual memory exclusively for his use. In fact, the shared routine does have space allocated for it in each user's virtual memory so that the shared routine appears to be simultaneously part of each user's program.

Several requirements must be met by all object modules that are to be incorporated into a shared routine. Each such object module must satisfy the following conditions:

Every CSECT in the Module must be Read Only and Public.

The object module must not contain relocatable address constants.

The user must not use a Linkage Editor OVERLAY statement to change the name of the load module in which the object module appears.

The shared routine must have been written using the standard linkage conventions for called programs.

LINKAGE EDITOR

The Linkage Editor is one of the TSOS service programs. It converts object modules to load module format, links object modules that have been compiled or assembled separately, and produces a loadable program consisting of one or more load modules.

Object modules produced by language processors must be processed by the Linkage Editor before they can be loaded for execution. This applies to both Class I and Class II programs.

LINKAGE EDITOR FUNCTIONS

The Linkage Editor can perform the following functions:

Translate one or more object modules into load module format.

Link two or more modules to form one load module.

Link two or more object modules in an overlay structure consisting of two or more load modules.

Obtain object modules from secondary inputs.

Search an object module library file to obtain modules which are then translated and linked into the load modules being produced. (Explicit search)

Rename entries and external references within an object module.

Delete entry names from an object module.

Define a New standard entry point for an object module.

Collect control sections with COMMON attribute and reserve space for them in the load modules. Both named and "blank" COMMON control sections are supported.

Satisfy those unresolved external references, which are not explicitly excluded, by an automatic search of object module libraries, including the system call library (SYSLIB).

Replace, delete or rename control sections within object modules. Automatic replacement of CSECT's within a module takes place when more than one CSECT has the same name.

Change the attributes of control sections.

Produce, at the user's option, a load module map and/or a cross reference listing of external definitions and external references.

List all outstanding, unresolved, external references at the completion of linkage processing.

Furnish diagnostic messages relating to errors and inconsistencies in the user's input.

IDA (INTERACTIVE DEBUGGING AID)

IDA is a program checkout language whose statements are a subset of the control language system. IDA may be used conversationally or nonconversationally, and its facilities are available to both Class I and Class I programs.

IDA STATEMENTS

The IDA commands described below can be combined into statements which either request immediate execution of one or more commands, or request deferred execution.

Deferred execution is achieved by beginning the statement with an AT phrase, making it dynamic; its execution is dependent upon arrival of control at the location in the object program specified in the AT command. An AT phrase may also specify execution of the statement at program termination time (normal or abnormal).

Execution of a IDA statement (whether immediate or dynamic) can be made conditional by the inclusion of an IF phrase. The IF phrase defines a logical expression that must be true to allow execution of the statement.

IDA COMMANDS

The IDA commands are:

CONVERT

DISPLAY

DUMP

FILL

MOVE

QUALIFY

REMOVE

RESUME

SET

STOP

FILE EDITOR

The TSOS File Editor provides for creating, modifying, and displaying files. The editor may be used in the nonconversational mode, but it is intended primarily for use from a terminal so that the user may control processing command by command. The editing commands may come from SYSIN or a program. Responses or messages from the editor will be returned to SYSOUT or the program accordingly. The format for commands and allied messages is uniform and independent of the source.

FILE TYPES

The editor will manipulate either lined files or unlined files; its full power, however, is available only with lined files. An unlined file is a sequential file of logical records accessed serially, i.e., having processed one record, the user may only process the next. The user may generate such a file one record at a time, or may add records at the end. Provision is made for its transformation to a lined file.

A lined file is an indexed sequential file on a random access device and consists of logical records or lines, each of which has a unique eight-digit line number. The user may insert new lines or may modify or delete existing lines. Lines are addressed by their line numbers if they are known or they can be computed.

EDITOR COMMANDS

The file editor commands are of five general types: control commands, line moving commands, line content commands, mode setting commands, and conditional commands.

Control commands initialize the editor to process a file, terminate processing, or establish modes and default cases in the editor mechanism. The control commands are:

EDIT

HALT

QUALIFY

RESET

SYNTAX

VERIFY

Line commands insert, delete, and move lines or parts of lines within the file being edited.

DELETE

GET

INPUT

MOVE

OUTPUT

Line content commands enable the user to access and modify all those lines in his file which satisfy specified conditions. The parameters designating the lines may be character count parameters or string parameters. The former specify fields in every line by character position; the latter cause a search of the lines by contents.

ALTER

CHANGE

FIND

Mode setting commands cause the editor to enter one of the two special modes: text mode or procedure mode.

Three instructions are available to direct the flow of control through a sequence of editor commands. These conditional commands allow the user to jump or loop through several commands.

JUMP

LOOP

NAME

DESK CALCULATOR

The Desk Calculator is a conversational program, designed to simulate a commercial desk calculator by use of a remote terminal. Numeric quantities may be entered and displayed in either fixed or floating point. The functions provided are:

Add, Subtract, Multiply, Divide, Square Root, Powers, Exponential, Trigonometric and Hyperbolic functions, and Logrithms (Natural and Common).

Sixteen "accumulators" are provided to hold intermediate results.

SYSTEM MANAGEMENT

This section describes some system considerations of significant concern to the user. The following topics are discussed:

Access to the System

Acquisition of Task Resources

File Space on Public Volumes

Restoring Data in the System

Bulk Output Facilities

System Support for Mass Storage Files

System Accounting File

System Protection of Programs and Data

System Shutdown

ACCESS TO THE SYSTEM

Granting access to the system amounts to defining the individual to the system. The following information is supplied to the system by the administrator when he issues the JOIN command:

Identification Code: 4 digit decimal number. Each individual has a unique code.

Task Password: 1 to 8 characters. The installation may specify whether passwords are required at LOGON time. This is not related to the file protection facility provided through passwords.

Charge Numbers: The account numbers to which the user is authorized to accumulate charges made for use of the system.

Priority: A single digit code, 0 through 9, that specifies the highest priority the user may request for work he submits to the System's Task Scheduler. Highest priority is 9; lowest is 0.

File Space: This parameter specifies the amount of public file space that this user is to be permitted to use.

Privilege Class: One-letter codes, which indicate the command set available to the individual. Codes are currently assigned for:

System administrator

System operator

User

Individuals may be assigned more than one privilege class. If an individual conversing with the system issues a command which is forbidden to his privilege class, the system will respond with a diagnostic message, ignore the command, and await another command from the user.

ACQUISITION OF TASK RESOURCES

The system allows the user to specify his task's requirements for system resources in two ways:

Statically by means of the SECURE command.

Dynamically (during the execution of programs within a task) by means of macro instructions.

The resources which may be secured are private I/O devices (volumes) and additional memory.

The SECURE command must appear immediately after the LOGON command and only one SECURE command is permitted for each task. The task is not executed until all the specified resources are available.

Two options are provided in the macros for the user who wishes to request resources dynamically. The first indicates that the program will use the specified resource only if it is available. The system will advise the calling program whether the resource is available. The second specifies that the program must have the additional resource before it can continue. If the system can not satisfy the request, a message is sent to the operator notifying him of the task's identify and the resource requested. The operator must then decide whether to abort this task, cancel another task to free the required resource, or suspend, if possible, the task until the resource is available. If a nonconversational task is suspended, the system will periodically attempt to resume processing of the task. Additionally, messages will be written to SYSOUT to inform the user of the task's status.

FILE SPACE ON PUBLIC VOLUMES

When each user is joined to the system, the system administrator specifies the amount of space on public volumes which the user may acquire. The space is not actually allocated to the user until it is required.

PRESTORING DATA IN THE SYSTEM

The user can submit decks of punched cards to the system operator to be entered into the system. Two types of input are permitted:

Command Procedure File

Data Files

A command procedure file contains all the commands required to run a nonconversational task. It is placed in the task queue to be executed as soon as the required resources are available (consistent with the priority of the task). The command procedure may include data that is to be read by the user's object program at execution time. The operator may submit a batch containing any number of command procedure files to the system when he specifies the READ CARDS command; each file constitutes a separate nonconversational task which has no relation to any other. Data files contain any information the user wishes to catalog, including com-

mand or program data. Unlike a Command procedure file, it is not executed after being read.

The facilities described above are also available to the user who wishes to enter data from tape into the system.

BULK OUTPUT FACILITIES

The PRINT command may be used in either the conversational or nonconversational mode. Execution of this command results in the creation of an independent nonconversational task to list the specified file on the high-speed printer. The user may specify that the first byte of each record is a control character which specifies the machine code or the ASA code for printer carriage control.

The WRITE TAPE command is similar to the PRINT command, except that the specified file is written to tape for subsequent off-line printing.

The PUNCH command may be used in either the conversational or the nonconversational mode. It results in the creation of an independent nonconversational task which punches a file onto cards.

SYSTEM SUPPORT FOR MASS STORAGE FILES

Access to files stored on 70/568 Mass storage Units is provided through the ACTIVATE and DEACTIVATE commands. Class I programs can also access 70/568 files directly by using the TOS FCP facilities supported by TSOS.

The ACTIVATE command transcribes a file from a Mass Storage Unit to a public or private volume on a direct access unit.

Once a file has been transcribed, it can be accessed using the TSOS Data Management System. Direct access files created by any of the access methods (including indexed sequential) can be transcribed using this command. The Mass Storage file specified in the ACTIVATE command must be cataloged and must have been transcribed to the 70/568 using the DEACTIVATE command.

The DEACTIVATE command transcribes a file from a direct access unit to a Mass Storage Unit. The file to be transcribed must be cataloged, and a Mass Storage file must have been created and cataloged prior to issuing the DEACTIVATE command.

These commands provide the user with a flexible method for managing his file space on public as well as private volumes. For example, when his file storage capacity on a direct access unit is exhausted, a user can issue the DEACTIVATE 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 on the direct access device, and reclaim it for other purposes. Thus, these commands provide a means for using the Spectra 70/568 Mass Storage Device as a lower level storage medium for "inactive" or infrequently used files. The DEACTIVATE command will, at the user's option, create, on magnetic tape, a "backup" copy of the file being transcribed. In fact, the user may specify that tape creation is the only transcription desired. If a file can not be ACTIVATED from the Mass Storage Unit, a message is written to SYSOUT. The conversational user could then reissue the ACTIVATE command, specifying an input file which resides on magnetic tape (backup volume).

SYSTEM ACCOUNTING FILE

The system maintains an accounting file on a public device which may be inspected and listed by the System Administrator. It is anticipated that many installations will also wish to write programs which employ the system utilization data collected in this file to bill their users.

The system accumulates the following utilization factors for each task:

Elapsed time during which private devices were allocated.

Accumulated central processor time.

Amount of storage used.

Actual LOGON and LOGOFF times.

The accumulated processor time is the time the task was actually in control of the processor. Time devoted to some I/O management (e.g., I/O interrupt servicing, I/O queuing, error recovery) is considered to be system overhead and is not included.

Each task entry in the accounting file also contains the following additional information:

User Identification Code.

User Charge Number.

Source of task (terminal number if conversational, blank if nonconversational).

Task termination information (e.g., normal termination, abnormal termination due to I/O device error).

The following data, although not part of the accounting file,

is also collected by the system for each user:

Amount of public file space permitted.

Amount of public file space actually used.

SYSTEM PROTECTION OF PROGRAMS AND DATA

TSOS provides the following facilities to preserve and safeguard user programs and information in the system:

Task Suspension.

Checkpoint and Restart.

File Reconstruction.

System Self-protection.

TASK SUSPENSION

It is sometimes desirable to suspend the execution of a task, and later to restore the task to active status. Either the user or the system can suspend a task. It is necessary that this be accomplished in such a way that the task can be resumed from the point of interruption without any loss of information. The system provides this task suspension facility for Class II programs.

Task suspension is provided for a conversational user who wishes to leave his terminal for an extended period without concluding his task. Conversational and nonconversational tasks can also be suspended whenever the system must be shut down.

Suspending a task requires that the system complete all outstanding input/output requests (including error recovery procedures), close all files opened for the task, and, in general, bring the task to an orderly stop.. The system preserves the contents of the user's virtual memory, the status of all files, device repositioning parameters for magnetic tape and direct access files, deferred IDA statements which have not been satisfied, and sufficient information regarding the task's interface with the system to enable the task environment to be reconstructed.

CHECKPOINT AND RESTART

Checkpointing is similar to suspension except that the task is automatically continued after all the information needed to restart the task is stored (e.g., user's virtual memory, repositioning parameters). The principal 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

Facilities for reconstruction of files stored on direct access devices are provided in the form of a set of utility routines. These routines can be employed by the user to reconstruct direct access files for which the necessary back-up files have been previously established. Typically, there are two back-up files:

A complete copy of the entire file as of a certain previous date.

A file which contains a copy of each record written to the file subsequent to the creation of (1) above. For example: a file of "after" images of all updated records.

SYSTEM SELF-PROTECTION

Facilities are provided at the system level 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. This information assists the system operator in reestablishing the status of the system environment to that at the time of the dump.

TYPICAL APPLICATION AND INSTALLATION

A manufacturing company of medium size is a typical application area for TSOS. A company of this sort usually has the following five activities (perhaps differently named) using a computer:

accounting,

quality control,

inventory control,

staff, and

research and development.

The accounting function has the least use for the interactive capability of TSOS; on the other hand, it produces the largest number of background 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.

The inventory control group is concerned with production scheduling, inventory control, and possible 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 may well suffice for the majority of the computer use by this group.

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.

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

With TSOS and 70/46, all of these groups will be able to get their work done on a single computer 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 amount of equipment.

TSOS SYSTEM HIGHLIGHTS

The following lists highlight the facilities provided by each of the major components of TSOS:

TASK MANAGEMENT

A user can issue commands interactively or from a pre-stored procedure file.

Unit record devices may be utilized by directing program output through the system's spooling operations.

Task scheduler will optimize paging and segmentation overhead by "look-ahead" paging based on module usage.

A user may change modes during task execution; e.g., compile conversationally and execute nonconversationally.

Portions of a task may be executed conditionally by using task switches in conjunction with SKIP and STEP commands.

DATA MANAGEMENT

A data storage file may be accessed at any level by using either fully qualified or partailly qualified file names.

Files may be stored on either public or private volumes and may be cataloged.

The system catalog entry for a file may be organized into a hierarchy of indices to facilitate file retrieval.

Provides sequential, indexed-sequential, and user defined files for direct access and two types of sequential file organizations for magnetic tape.

Protection of shared files is provided by passwords, a read-only attribute, and interlocks maintained by the system when a file is being used.

PROGRAM MANAGEMENT

TSOS will execute almost any TOS program either without change or with minor modifications.

System throughput and efficiency is increased by keeping Class I programs resident in core.

A user may request all or any part of a Class II program be kept resident during execution; thus, Class II features are available without paging overhead.

The system resources can be modified by parameters to provide better service depending on the Class I/Class II mix at a particular installation.

Programs in either class may be run conversationally or nonconversationally.

The class of a program may be specified at compile time, at link edit time, or at load time.

A program module may be shared by several users and only one copy of the module need be resident in main memory or on the paging drum.

The IDA debugging facility can be used with any object program.

Language processors include FORTRAN, COBOL, Macro Assembler, RPG, Interactive FORTRAN, and Interactive BASIC.

Service programs include File Editor, IDA, Assembly Diagnostic Package, COBOL Syntax Checker, Desk Calculator, Linkage Editor, Sort/Merge, and other utility routines.

SYSTEM MANAGEMENT

The DEACTIVATE and ACTIVATE commands allow the storing and retrieving of infrequently used files utilizing the 70/568 mass storage unit.

Suspended tasks may be restarted from the point of interruption without loss of information.

Facilities for file reconstruction for direct access devices are provided through special utility routines.

Facilities are provided at the system level to facilitate reviving the system in the case of unscheduled shutdown.

TSOS SUMMARY

The position of the 70/46 and TSOS within the total computer picture would seem to be most advantageous at this time. Its price is approximately $30,000 per month; that puts it in the medium-scale field. The cost-performance ratio puts 70/46 TSOS in a unique class. To equal this system would require an onsite production processor of the same capabilities and up to 48 interactive consoles served from or by another interactive system. The potential users of interactive system are legion. They include universities, manufacturing, transportation, utilities, insurance, banking, and other financial institutions, publishing, aerospace, and government: in fact, anyone who has production processing requirements and needs to service the types of users mentioned earlier can use remote access terminals such as offered by TSOS. Within this long list of potential users, there are two categories: the commercial and the scientific. The scientific problem solving applications using the interactive operations are probably obvious and easy to appreciate. But the commercial applications may seem less apparent, but are just as significant. Consider such examples of possible interactive usage as bond and portfolio analysis in the stock broker's office, cash flow calculations, model building, and statistical analysis. In general, the interactive system offers to anyone who already finds computing a necessity, an enhanced operation. The problem is only one of letting them find out with careful encouragement from the computer manufacturer what interactive programming can mean to such an organization.