TCDMS SYSTEM SUMMARY

January 1976

Department of Housing and Urban Development
Office of the Assistant Secretary for
Policy, Development & Research

HUD Contract H-2073-R

INTER-REGIONAL INFORMATION SYSTEM

Regional Information Systems Department
Lane County Courthouse
Eugene, Oregon 97401

and

Data Processing Authority
4747 East Burnside
Portland, Oregon 97215

FILE COPY

BIBLIOGRAPHIC DATA SHEET	1. Report No. USACLCG20003	2.	3. Recipient's Accession No
4. Title and Subtitle TCDMS System Summary			5. Report Date January 30, 1976 6.
7. Author(s)			8. Performing Organization Report No.
9. Performing Organization Name and Address Lane County Government Lane County Courthouse Eugene, Oregon 97401			10. Project/Task/Work Unit No. 11. Contract/Grant No. H-2073-R
12. Sponsoring Organization Name and Address U.S. Dept. of Housing & Urban Development Office of Policy, Development & Research 451 7th St., S.W. Washington, D.C. 20410			13. Type of Report & Period Covered Special Technical Report 14.
15. Supplementary Notes			

16. Abstracts

This manual is from a USAC series produced by the Regional Information Systems Department of Lane County. It contains a conceptual introduction to the Telecommunications/Data Management System (TCDMS). It contains an analysis of the manner in which TCDMS processes transactions through its network of terminals, the relationship between the logical and physical file structures of its data base, a brief description of the system generation process, and a list of the hardware and software requirements for installation.

17. Key Words and Document Analysis

17a. Descriptors

Information System
Local Government
Computer System Programs
Data Retrieval

17b. Identifiers/Open-Ended Terms

Urban Information Systems Inter-Agency Committee
Municipal Information System
Lane County
Data Management System

17c. COSATI Field/Group 5B

Released for distribution by NTIS 18. Availability Statement	Unclassified 19. Security Class (This Report)	74 21. No. of Pages
	Unclassified 20. Security Class (This Page)	22. Price

FORM NTIS-35 (REV. 3-72)

USCOMM-D.C. 14952-P72

THE CITATION OF TRADE NAMES OR MANUFACTURERS IN THIS REPORT DOES NOT CONSTITUTE AN OFFICIAL ENDORSEMENT OR APPROVAL OF THE USE OF SUCH EQUIPMENT OR SOFTWARE.

0.1 PREFACE

In 1972, the Data Processing Authority (representing the city of Portland and Multnomah County, Oregon) and the Regional Information Systems Department of the Lane County government (representing the cities of Eugene, Springfield, Albany, Cottage Grove, and Florence; and Lane, Linn and Benton Counties, Oregon) formed an organization called the Inter-Regional Information System (IRIS). Its purpose was manifold:

> to solve some of the complex problems of public information handling through cooperative planning and development of hardware and software environments;
>
> to minimize the duplication of effort involved in writing application systems;
>
> to reduce the cost of governmental data processing; and
>
> to increase the quality of service to the taxpayer.

Since its inception, the IRIS organization has grown to represent over one hundred different city, county, state, and federal agencies serving over 70% of Oregon's population. Current projects include the Fleet Management System, the Assessment and Taxation System, and the Telecommunications Data Management System. Future involvement is anticipated in the areas of criminal justice, management analysis, human resources, geo-coding, and financial systems.

Much of the inter-regional success enjoyed by the IRIS organization has been facilitated by a cost-reimbursement contract with the Urban Information Systems Inter-Agency Committee (USAC). USAC is a consortium of ten federal agencies formed in 1968 to work together

with local governments across the United States in an effort to improve urban governance through more effective use of computer-based processing systems. USAC is sponsoring several research and development projects which will result in transferable, computerized information systems available to local governments throughout the United States.

With the support of USAC, IRIS is developing the system software foundation for the application programs which control these computerized systems. This foundation is called TeleCommunications/ Data Management System (TCDMS). This system contains two components which bring together the state-of-the-art features in both telecommunications and data base/data management systems.

The telecommunications component of TCDMS extends the power of the modern computer to the desk of each user. Its facilities include such features as terminal independent I/O functions, user-specified security, multiprogramming, priority scheduling, message switching, print-out spooling, on-line debugging, and remote job entry.

The data base/data management component of TCDMS optimizes the efficiency of data file construction and minimizes data redundancy by combining all files in the system into an integrated data base. Its facilities include data access flexibility, file and data element security, and application program independence from the physical file structure.

Perhaps the most important feature of TCDMS is the transferability of application systems it allows. Application programs running under TCDMS control are isolated from changes in the hardware or software configuration of the installation. This means that TCDMS-controlled application systems can be transferred between IRIS installations without the costly conversion efforts usually necessitated by such

exchanges.

TCDMS may be implemented on any IBM System 360/370 computer having 252K bytes or more of storage capacity. It will support IBM System 360/370 BAL, FORTRAN, COBOL, and DL/1 user languages. TCDMS will run in real core under the control of IBM OS or VS operating systems. The modular construction of TCDMS makes it hardware independent; it can operate with any IBM terminal hardware configuration.

The joint software development and maintenance by means of the regional and interregional cooperation of IRIS and the integrated data base/data communications approach of TCDMS are becoming an increasingly popular solution to the problems of information handling in the public domain. For further information and for documentation concerning IRIS, contact the Regional Information Systems Department, Lane County Courthouse, Eugene, Oregon 97401.

TABLE OF CONTENTS

0.2 INTRODUCTION

This book is about the Tele-Communications/Data Management System (TCDMS). It was written to provide TCDMS users with the information necessary to understand and evaluate the facilities of the system.

The Tele-Communications/Data Management System is a data communications system which acts as an interface between application programs and the resident IBM System 360/370 OS or VS operating system. It consists of two major components; a set of telecommunications modules which controls the processing of all transactions through the TCDMS network of terminals, and a set of data management modules which controls the placement, manipulation, and retrieval of information in the TCDMS integrated data base.

Either of the two components of TCDMS may be implemented without the other. The telecommunications component (TCS) will act as a terminal network controller in non-data base environments, or it will support the file structure of other available data management systems. The data management component (DMS) will function in a similarly independent fashion. It will interface to other teleprocessing systems or operate in a standard batch environment.

This basic independence of the two components of TCDMS is further enhanced by the modularity of their design. The functional units of both components are isolated into separate modules which are configured at system generation to provide a system tailored to meet the specific needs of each installation. Thus, each TCDMS installation operates with its own unique version of TCDMS, which can be modified to support evolutionary changes in the hardware or software environment with minimal impact on user application systems.

TCDMS is oriented toward the convenience of the application programmers and terminal operators using the system. The TCDMS application programmer utilizes the facilities of the system in a manner consistent with the programming conventions of the host language. TCDMS functions are invoked at the CALL or macro level of the host language; Assembler Language, COBOL, PL/1, or Fortran. Terminal operators may access TCDMS

application programs and utilities by simply entering an asterisk (*), and then the name of the desired program.

Since this publication is addressed to groups of people with differing technical backgrounds, it has been divided into several sections which may be read separately;

SECTION 1 - TELECOMMUNICATIONS CONCEPTS This section examines the logical structure of the telecommunications component of TCDMS with regard to how it accomplishes the processing of transactions through the TCDMS network of terminals

SECTION 2 - DATA MANAGEMENT CONCEPTS This section examines the structure of the TCDMS data base and the means by which physical and logical relationships are established between data elements.

SECTION 3 - USING TCS This section describes the function and use of each TCS programming function and user utility program.

SECTION 4 - THE TCDMS ENVIRONMENT This section examines the relationship between TCDMS and its hardware and software environment. System hardware and software specifications are provided. Consideration is also given to TCDMS system generation and its impact on the user installation.

All TCDMS users are assisted by a complete set of readable documentation, which includes the following publications:

TCDMS APPLICATION PROGRAMMER'S MANUAL - A guide to the efficient use of the TCDMS functions. Published separately for BAL, COBOL, PL/1, and Fortran programmers.

TCDMS USER UTILITIES MANUAL - A guide to the use of the TCDMS user utility programs.

TCDMS SYSTEM UTILITIES MANUAL - A guide to the use of the TCDMS privileged utility programs for system maintenance.

TCDMS SYSTEM PROGRAMMER'S MANUAL - A reference manual to assist system programmers in generation and maintenance of TCDMS.

TCDMS OPERATIONS MANUAL - A reference guide to assist TCDMS computer operators in maintaining a smooth system throughput.

TCDMS MESSAGES AND CODES MANUAL - An explanation of each TCDMS system message and return code.

TCDMS CONTROL BLOCKS MANUAL - A reference guide describing the TCDMS control blocks and their linkages.

TCDMS PROGRAM LOGIC MANUAL - A description of each module in the TCDMS system.

1.0 TELEPROCESSING CONCEPTS

Teleprocessing can be defined as a method of computing in which the source of data entry is physically distant from the computer. This lack of proximity with the computer is one fundamental difference between teleprocessing and batch (punch card) processing, where the input of data originates in the computer room. There are, of course, many other differences between teleprocessing and batch processing. But it is the ability of teleprocessing systems to make the computer available when and where the need for information arises that makes teleprocessing so important for the modern data processing installation.

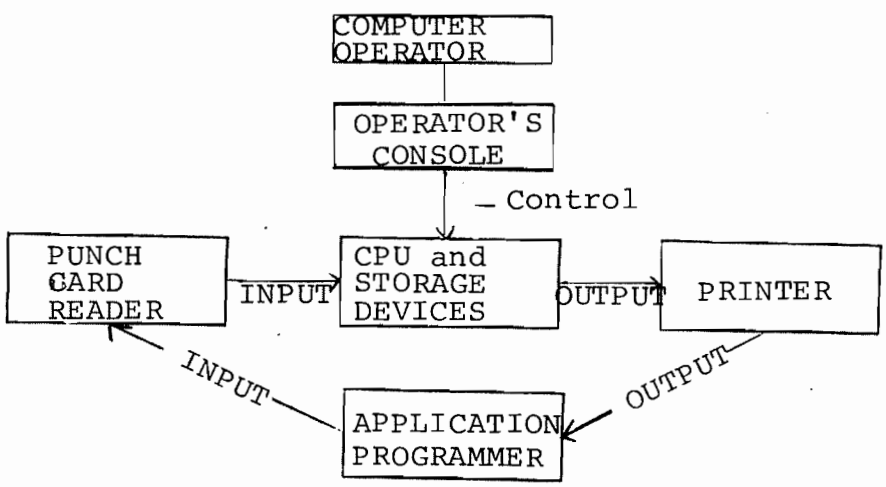
This section explains some of the basis conceptual differences between teleprocessing and batch processing and gives a conceptual overview of the manner in which TCDMS provides teleprocessing services for its users.

BATCH PROCESSING

Batch processing is the form of computing usually associated with the ubiguitous "computer punch card". The hardware environment of a typical batch processing computer consists of:

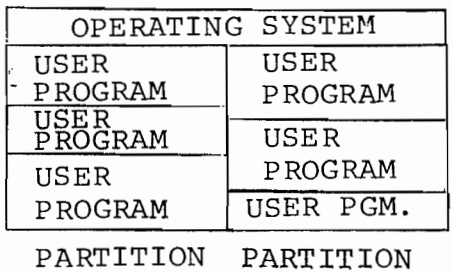
THE CPU AND ITS STORAGE DEVICES
THE PUNCH CARD READER
THE PRINTER
THE COMPUTER OPERATOR'S CONSOLE

These components are almost always located together in the same room. With the addition of two more elements; the computer operator and the application programmer, we can depict the relationship of these components graphically as follows:



The user of our batch system (i.e., the application programmer) submits information to the computer in the form of punched cards. The computer operator places this input into the card reader and enters commands on the operator's console. These commands cause the cards to be read and their input processed by the CPU and its storage devices. The results of this processing are written on the line printer and returned to the application programmer for analysis. The time consumed by this procedure is called "turnaround time" by computer personnel. It can vary from several minutes to several hours.

The software environment of our batch processing model can be illustrated graphically as follows:



User programs are placed in separate partitions of the computer's memory, where they run under the control of the resident operating system. The operating system performs the logical functions which control the resources of the computer; it allocates control of the CPU to the partitions according to a pre-set priority scheme, and performs the I/O operations necessary to read the program's punch card input and write the program's printed output.

WHY TELEPROCESSING?

It should be noted that, in our model batch system, the word USER is synonymous with the words APPLICATION PROGRAMMER. Since, in production environments, the application programmer is rarely the person for whom the computer ultimately provides services, it is apparent that one of the characteristics of batch processing systems is the separation of the user from the tool; i.e., the computer. This separation occurs in several ways;

Batch system require that the person in need of the computer's services bring the information to the computer to be processed. This is often not convenient.

Batch systems do not enable the user to make un-planned inquiries of data stored in the computer. A batch user cannot conveniently "brouse" through data.

Batch systems require the interface of trained data processing personnel between the computer and its users. This requirement raises the cost of data processing and degrades the security, accuracy, and convenience of the entire process.

Batch systems make relatively inefficient use of the time of both the computer and its users. Since batch programs are processed serially, the CPU often spends some of its time idle while waiting for non-computational operations to be performed. Batch system users similarly spend a great deal of time waiting for information to be processed. This precludes the use of computers to support over-the counter transactions.

These and other shortcomings of batch systems impose severe limitations upon the uses to which computers may be put. To overcome these limitations and to enable more people to become beneficiaries of the power of modern computers, teleprocessing systems were developed.

TELEPROCESSING

Teleprocessing systems differ from batch processing systems in two important ways; the number and location of the computer's sources of input, and the number of activities which may occur within the computer at a given moment.

In a teleprocessing system, the card reader and printer of our batch model are replaced by individual devices called terminals. A terminal is a device which is capable of interacting with the computer. Teleprocessing systems have networks of terminals ranging in size from a few to thousands of devices, each of which is capable of interacting with the computer on an individual basis.

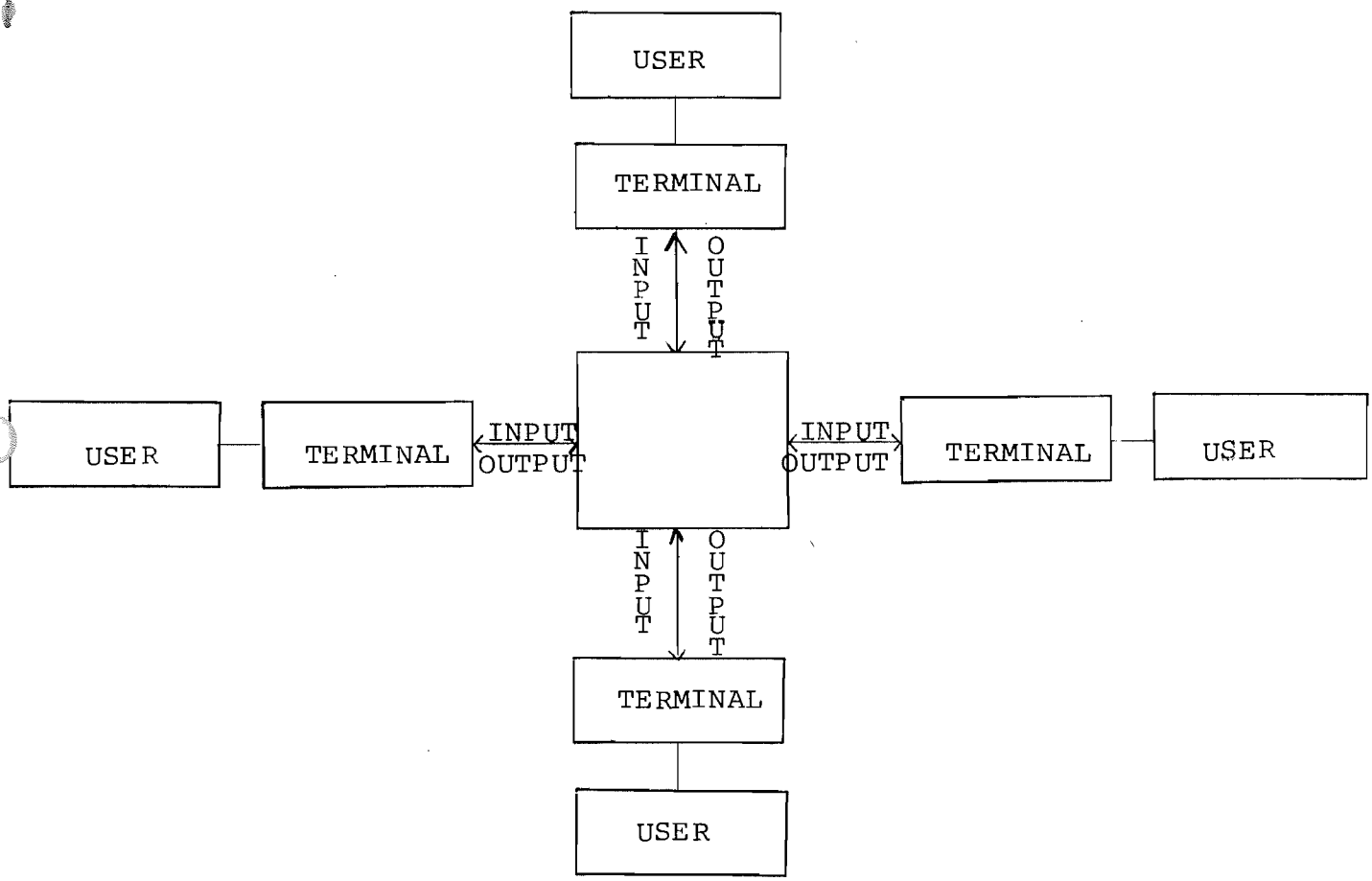
Because of this increase in the number of sources of input, teleprocessing systems are often capable of executing several user programs asyncronously.

The hardware environment of a typical teleprocessing computer consists of:

THE CPU AND ITS STORAGE DEVICES

THE TERMINALS

These components are rarely located together in the same room, and are aften separated form the computer by distances ranging from a few feet to thousands of miles. Each terminal can function as both an input device for sending information to the computer, and an output device for receiving information from the computer. With the addition of one more element; the user, we can depict the relationship of these components as follows:

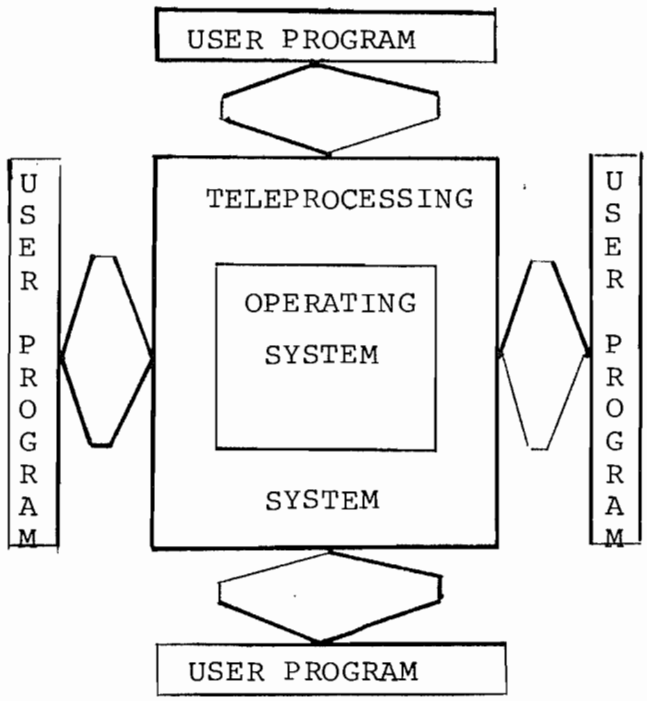


Each user in our model can initiate the flow of information to or from the computer when the need for such information arises. In a typical teleprocessing transaction, the user makes an inquiry of the data stored in the CPU and its storage devices, and the information is displayed at the user's terminal once the search criteria established by the user have been fulfilled. The time consumed by this process is called

"response time". It varies from thousandths of a second to several minutes.

Since the likelihood that more than one user will require access to the resources of the computer at any given moment is quite high, methods have been devised to control the flow of these resources to insure that all the users of the system obtain the services they need. This is accomplished by teleprocessing software.

The software environment of our teleprocessing model can be shown like this:



Each user in our teleprocessing model can initiate transactions with an application program by calling the program from a terminal. The teleprocessing system intercepts the call and passes control of the CPU to the appropriate program and teleprocessing or operating system routines, which perform the services requested by the user. The user program then causes the results of the computer's operations to be written to the user's terminal by invoking the appropriate system routines.

Our teleprocessing model achieves several significant advances over batch processing systems:

Teleprocessing systems extend the facilities of the computer to the physical location of the user.

Since the response time of teleprocessing systems is considerably less than the turnaround time of batch systems, the teleprocessing user can make several unplanned inquires of stored data in rapid succession. The computer thus becomes a tool in the decision-making process, rather than a repository for information about past events.

Teleprocessing systems do not require the intervention of trained personnel to support user transactions. Because of the direct means of information transfer between the computer and its users, the integrity of the transaction process is more easily insured.

Because a teleprocessing computer handles several inquires asyncronously, the user spends less time waiting for a response, and the CPU spends less time idle than in batch environments.

The teleprocessing component (TCS) of TCDMS provides its user with these and other advantages over conventional batch systems.

TELEPROCESSING WITH TCDMS

TCS provides all the facilities necessary for an installation to conduct its data processing in an on-line environment. It consists of a series of inter-related modules which can be grouped into the following functional areas:

TERMINAL I/O - These modules handle the communication between the CPU and the terminals in the TCS terminal network.

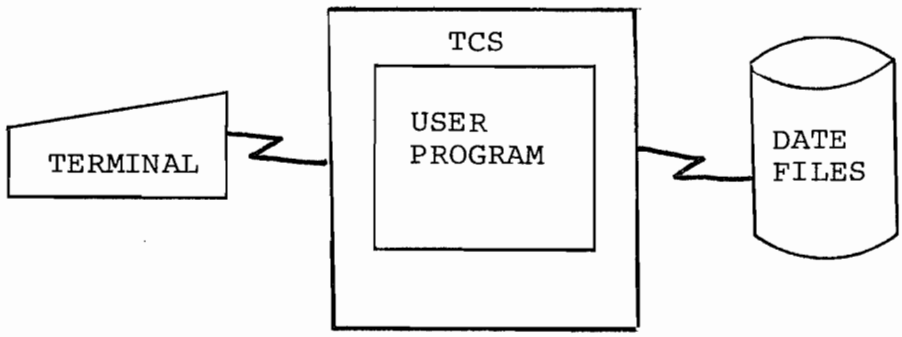
SYSTEM SERVICE ROUTINES - These modules manage the allocation of the computer's resources to the application programs and system tasks requiring them.

HIGH LEVEL LANGUAGE INTERFACES - These modules interpret high level language CALL statements to provide appropriate TCS services.

UTILITIES - These modules perform utility functions for TCS users.

These four groups of modules function together to control on-line and batch application programs which contain TCS functions, and terminals which request the services of a TCS utility. They provide the linkages

between the application program and its data files. The following diagram illustrates this relationship:



TERMINAL I/O

The terminal I/O modules of TCS control the flow of information between the CPU and the devices which comprise the TCS terminal network. They provide terminal control logic which is independent of any proprietary terminal access method. The terminal I/O modules may be divided into two groups:

DEVICE-DEPENDENT MODULES - These modules control all hardware - dependent I/O functions. They construct the channel programs, conduct the line protocol, and format data in accordance with the unique requirements of each device-type in the TCS terminal network. They isolate the application programmer from I/O formatting considerations.

DEVICE-INDEPENDENT MODULES - These are the general terminal I/O modules which operate independent of the hardware configuration to control the scheduling of terminal I/O operation, assign and manage queues and buffers, conduct error recovery, and transfer CPU control to the operating system or to the TCS system service routines upon detection of an SVC interrupt.

THE DEVICE-DEPENDENT MODULES

Each terminal-type in the TCS network has unique line control and data formatting characteristics. In order to insulate the application programmer and the rest of the TCS system from these device-dependent requirements, the designers of TCS isolated those portions of the system which relate to the idiosyncracies of each individual device into

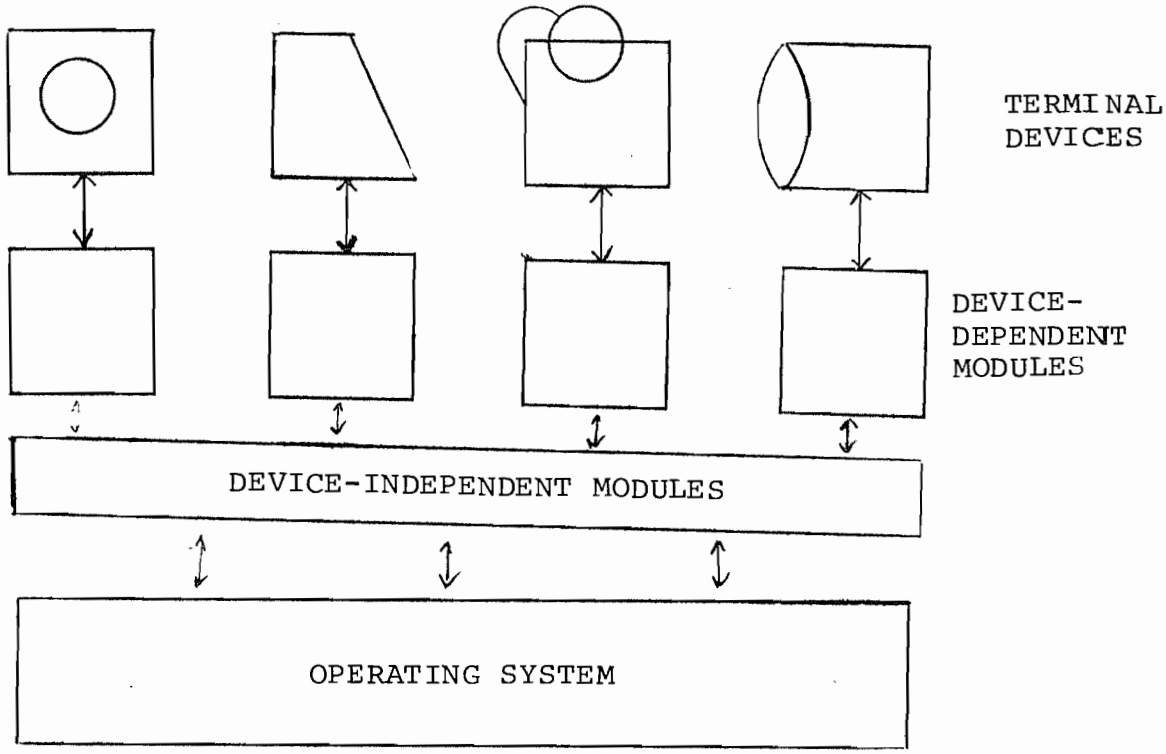
separate units called device modules and data modules.

Device modules build the strings of channel command words called channel programs which control the transfer of data between the CPU and each device in the terminal network. For attention interrupt devices, the device modules also conduct the line protocol necessary to insure that the terminal and the CPU do not attempt to send data to each other at the same time. There is one device module for each type of terminal in the TCS network.

Data modules conduct data handling for terminal I/O operations. They prepare and reformat device buffers and either insert or remove control characters from the data stream, depending upon the type of terminal I/O operation requested by the user. The terminal-independent functions enable the programmers to read or write data to a terminal without consideration to the control characters normally required in the data stream. The data module inserts these control characters for the program in accordance with the requirements of each terminal with which the program interacts. The terminal dependent functions enable the programmer to by-pass these data handling services and insert all terminal-dependent control characters directly into the data stream. There is one data module for each device-type in the TCS network.

Since the device-dependent modules are separate from the rest of the TCS terminal I/O logic, the TCS installation can add or delete terminal hardware without affecting TCS as a whole. Each time a new component is added to the terminal network, new device and data modules are link-edited into TCS. This design feature eliminates expensive conversion of existing application programs when the terminal configuration of the installation is modified.

The relationship of the terminal I/O modules is shown in the following diagram:



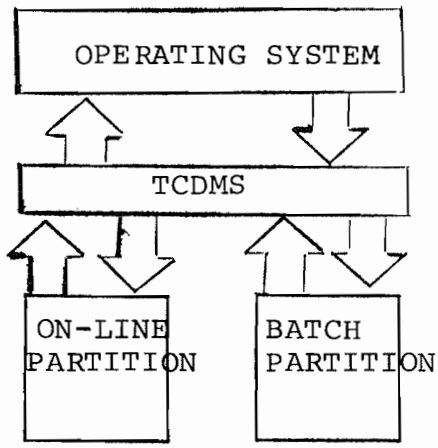
The terminal I/O portion of TCDMS consists of device dependent modules which are appended to the TCS terminal I/O logic to provide independence from changes in the hardware configuration of the installation.

THE DEVICE-INDEPENDENT MODULES

The device-independent terminal I/O modules of TCS perform terminal I/O functions which are not affected by changes in the hardware environment of the installation. These functions include interfacing with the resident operating system, queue management, buffer management, and error recovery.

OPERATING SYSTEM INTERFACE

TCS serves as an interface between the terminal operator or on-line program and the resident OS/VS operating system. It does not alter or duplicate any operating system functions, but merely allocates the services of the resident operating system to programs which contain TCDMS functions.



TCMDS controls all programs containing TCDMS functions. The resident operating system (OS or VS) controls TCDMS. The TCS operating system interface modules intercept certain interrupts which are generated by user programs containing TCS functions. These interrupts are examined by TCS to determine the nature of the services requested, and control is passed to the appropriate OS/VS routine or TCS module for further processing

The TCS operating system interface intercepts three types of interrupts:

ATTENTION INTERRUPTS - These interrupts are generated when an entry is made at a terminal. TCS determines the identity of the interrupting device. If it is not assigned to TCS, control is passed to the resident operating system for further processing. If the interrupting device belongs to the TCS terminal network, control is passed to one of the TCDMS terminal I/O modules discussed above for I/O processing.

SVC INTERRUPTS - Each TCS function generates a unique SVC interrupt which is intercepted by the TCS operating system interface to determine the nature of the services requested. Control is then passed to the appropriate TCS module or operating system routine for further processing.

PROGRAM INTERRUPTS - When an exception occurs during execution of a TCS program, the TCS operating system interface performs abnormal-end-of-job processing. This allows the resident operating system interface modules save certain information for a dump, so that the reason for the abnormal termination can be determined.

QUEUE MANAGEMENT

The queue management modules maintain a system of queues into which tasks awaiting further processing are placed. These queues can be classified as follows:

INPUT QUEUE - The input queue contains a list of terminal identification (TID) numbers of terminals which are waiting to send information to the computer.

READY-TO-RUN-QUEUE - The ready-to-run queue contains TID numbers which have sent information to be processed by an application programmer or a TCS utility.

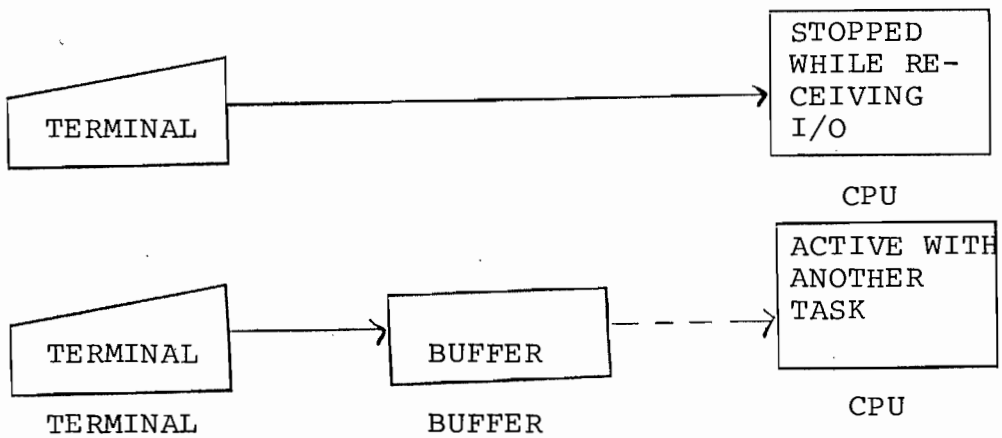
OUTPUT QUEUE - The output queue contains TID numbers of terminals to which application programs are waiting to send information.

COMPLETION QUEUE - The completion queue contains the TID numbers of terminals which have successfully completed an I/O operation.

TCS uses these queues to monitor and manage the flow of work through the system.

BUFFER MANAGEMENT

TCS maintains a system of buffers for temporary storage of information awaiting further processing. The number and size of these buffers varies with the number and types of devices in the TCS terminal network.



Since I/O operations are relatively slower than the processing speed of the CPU, TCDMS provides a system of buffers into which data from terminals is stored. The program then retrieves the information from the buffer at a much higher rate of transfer, thus utilizing the time of the CPU more effectively.

These buffers perform two important tasks. First, they compensate for the vast difference in speed between the rate at which the CPU can process information and the rate at which a terminal can send it. When a terminal I/O operation is performed, the data is placed into a buffer. Since terminal I/O operations are relatively slow in comparision to the processing speed of the CPU, the computer is free to perform other operations while this transfer of information is taking place.

Secondly, these I/O buffers provide a means of storing information during the period of time between when it is read from a terminal and processed by the computer or written by a program and actually sent to a terminal

ERROR RECOVERY

When an I/O error occurs while TCS is in the process of reading or writing to a terminal, information about the nature of the error is recorded on the TCS system log, and the I/O operation is retired. If the I/O operation is unsuccessfully completed after four tries, the operation is cancelled, the computer operator is notified to delete the terminal from the network until the error condition can be corrected, and a dump is produced to aid the system programmer in diagnosis of the problem.

The manner in which the TCS terminal I/O modules work together to provide complete terminal I/O support for TCS users may be demonstrated by following a transaction through the system:

STEP 1 - When an entry is made at a TCDMS terminal, the resulting attention interrupt is intercepted by an operating system interface module, which places the TID of the interrupting device into the input queue.

STEP 2 - Terminal I/O module scans the input queue, determines the device-type of the interrupting terminal, and passes control to the appropriate TCDMS device module.

STEP 3 - The device module obtains a buffer of the required size, constructs the channel program, and issues the EXCP instruction which causes the residing operating system to read the data into an input buffer.

STEP 4 - When the read operation is complete, control is passed to another terminal I/O module which performs error checks. If an I/O error has occured, a counter is incremented and control is passed back to the device module to re-read the device. If no errors occured, or if error recovery was unsuccessful after four retries, the TID of the interrupting terminal is placed in the completion queue.

STEP 5 - When the TID is recovered from the completion queue, the control characters are removed by a device module, and control is passed to another module which scans the first few characters of the entry to determine its nature. The data is then placed in the appropriate queue; the message queue, if the data was a message to another terminal, the page queue if the entry was a paging request, or the ready-to-run queue, if the entry was an initial program call.

SYSTEM SERVICE MODULES

The TCS system service module consist of a loosely inter-related group of modules which control the allocation of system resources to the application program and system tasks requiring them. These modules may be broken down into the following functional groups:

SYSTEM GENERATION AND INITIALIZATION
RESOURCE MANAGEMENT
SYSTEM MONITORING

With the exception of the system generation and initialization routines, the TCS system serivce modules are invoked by the terminal I/O and operating system interface modules previously described.

SYSTEM GENERATION AND INITIALIZATION

By using the TCS system generation and initialization modules, the unique version of TCS which best fulfills the requirements of the installation can be defined. This definition process has two stages:

SYSTEM GENERATION - The process by which the TCS hardware and software environment is defined. The TCS system generation process describes the device configuration of the installation, the size and number of threads, file size, operating system configuration and other environmental attributes. There are over one hundred thirty parameters which may be specified during the system generation process. The modules comprising TCS are chosen and assembled, and them combined in a link-edit procedure.

SYSTEM INITIALIZATION - The initialization process is the means by which the computer operator can dynamically alter TCS during a production shift, or cold start TCS after a system failure. During initialization, the TCS operating system interface is set,

files and control blocks are opened, and threads and buffer pools are restored.

The specific parameters of the TCS system generation macros and the procedures for system generation and initialization are described in greater detail in the section of this publication entitles the TCDMS ENVIRONNENT.

RESOURCE MANAGEMENT

The TCS resource management modules provide control of the resources of the computer for other TCS modules. These resources include both main storage and the peripheral storage devices attached to the computer. In addition, the resource management modules provide algorithms used by other TCDMS modules for the manipulation of data as it is passed from module to module. The resource management modules may be broken into functional groups for the purposes of description. These functional groups are discussed below:

TASK MANAGEMENT

In order to enable several terminals to gain access to the resources of the computer at the same time, teleprocessing systems utilize a technique of resource sharing called multiprogramming. Multiprogramming systems allow several programs to execute asyncronously. TCS employs a technique of asyncronous processing called ROLLIN/ROLLOUT to allocate the resources of the computer to the programs and tasks requiring them.

The TCS task management routines divide the teleprocessing partition of the computer's memory into separate sub-partitions called threads. Each thread holds one executing task at a time, and each task executes asyncronously with the others. The number of threads in a TCS installation, and thus the number of tasks which may execute asyncronously, is dependent upon the size of the partition allocated to on-line programs and the size of the installation's programs.

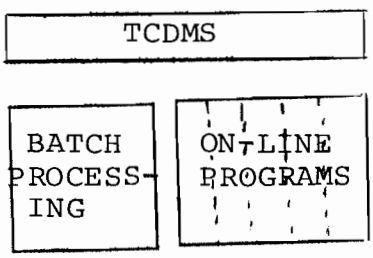
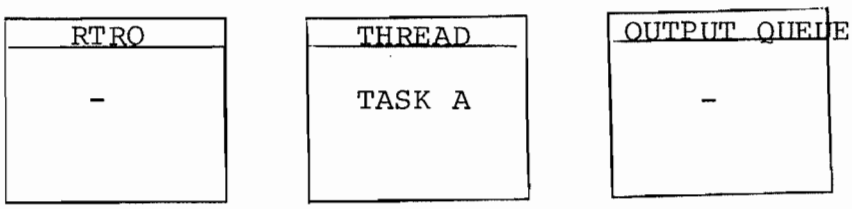


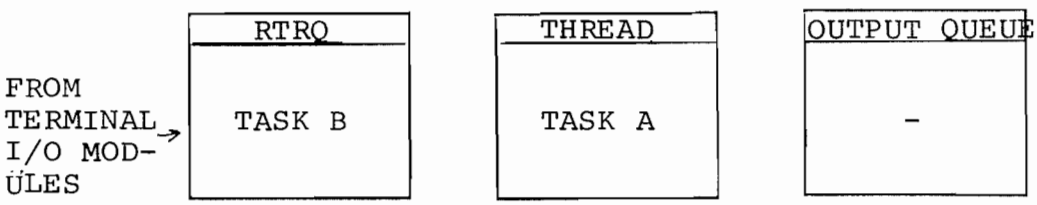
Figure 3 - Each thread occupies a separate area of the computers memory. The remainer of the computer is available to batch programs. The programs and tasks occupying TCDMS threads execute independent of one another. One program may not degrade the area of another program's thread.

Since terminal I/O operations are relatively a great deal slower in total execution time than other computer operations, the TCS task management modules make maximum use of the computer's time by removing an executing task from its thread each time it issues a terminal I/O instruction, and replacing it with another task which has completed its I/O. This process, which is called ROLLIN/ROLLOUT, is the means by which TCS thread areas are allocated to the various on-line programs and system tasks awaiting execution. The following example shows this rollin/rollout process in action:

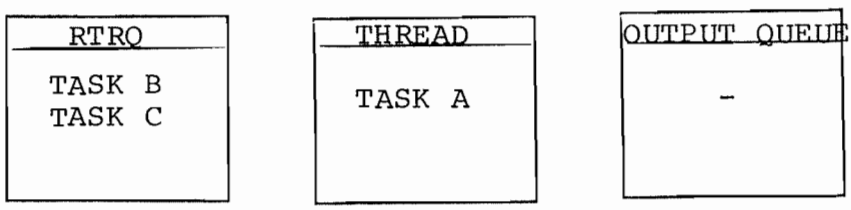
TASK A is executing in the thread. Both the ready-to-run queue and the output queue are empty:



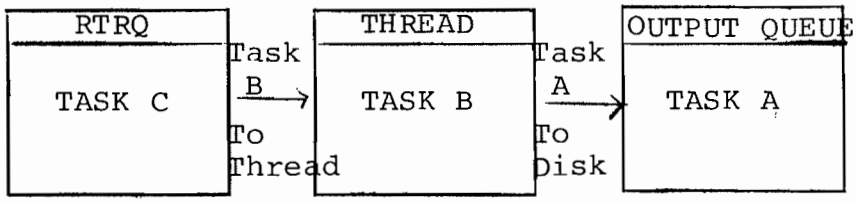
TASK B is placed into the ready-to-run queue by the TCDMS terminal I/O modules. Task A is still executing in the thread. The output queue is still empty:



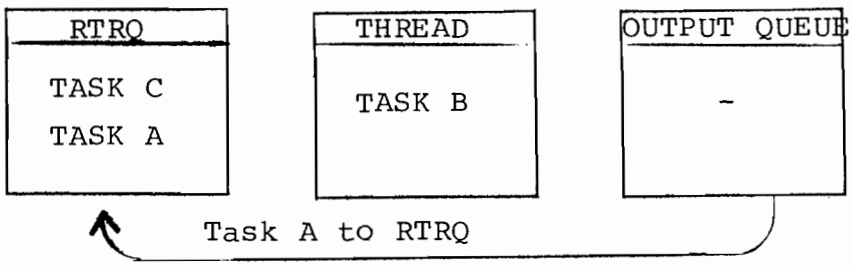
TASK C is placed into the ready-to-run queue by the terminal I/O modules. Task A is still executing in the thread. The output queue is still empty:



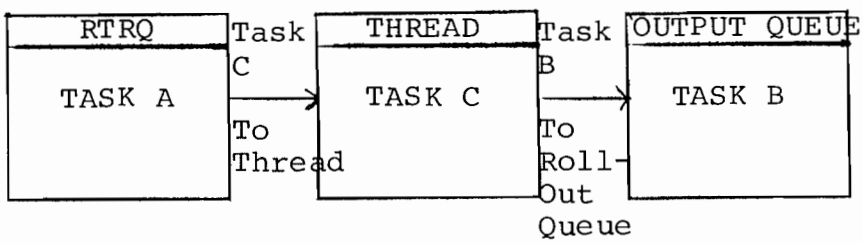
TASK A issues a terminal I/O command. The TCS task management modules place Task A into the output queue and load Task B into the thread to begin execution. Task C remains in the RTRQ:



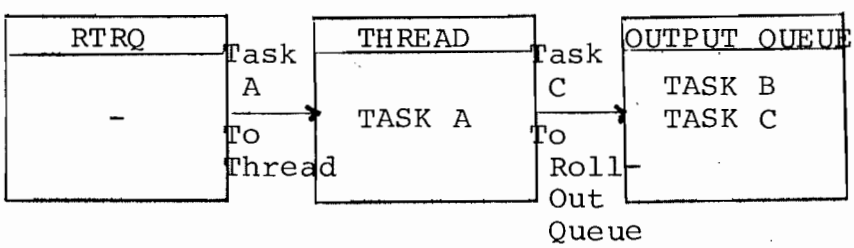
TASK A completes its I/O operation. It is removed from the output queue and placed into the ready-to-run queue by the TCDMS task management modules:



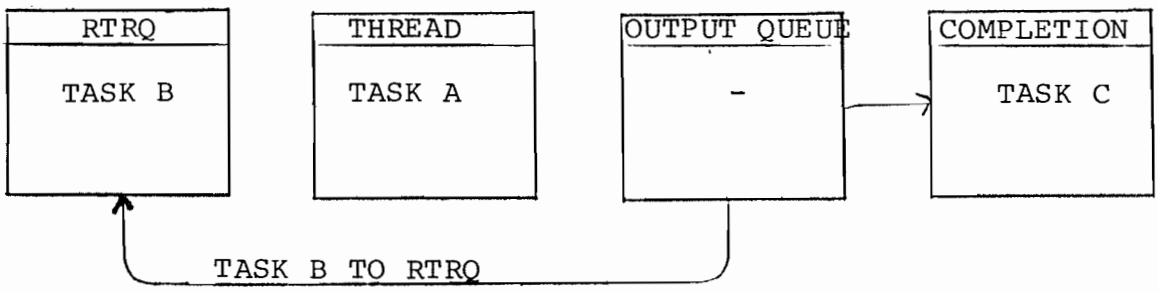
TASK B issues a terminal I/O command and is placed in the output queue. Task C replaces it in the thread. Task A remains in the RTRQ:



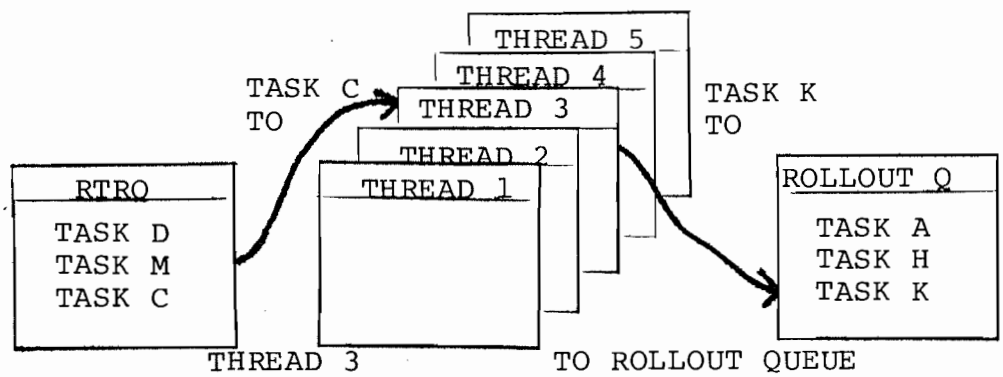
TASK C issues a terminal I/O command and is placed in the output queue behind Task B. Task A is rolled back into the thread:



TASK B completes its I/O operation and is placed in the RTRQ. Upon successful completion of its I/O operation, Task C completes its processing. The TCDMS task management modules pass task C to the completion Q, from which resident operation system retrieves it for normal end-of-job processing. Task A will remain in the thread executing until it issues another terminal I/O command, at which time the rollin/rollout process will continue.



The narrative above illustrates how the TCS task management modules control the rollin/rollout process for one thread. To further increase the number of terminals which can use the computer at a given moment, most TCS installations use multiple threads. In this instance our rollin/rollout illustration would look like this:



When multiple threads are run in a TCDMS installation, tasks waiting in the RTRQ may be placed into any of the threads which are left empty by the rollout of another task.

Because of the great speed with which modern computer function, rollin/rollout operations are performed in thousands of a second, and thus are transparent to the terminal operator. The TCDMS task management modules give the illusion that each terminal is the only one operating in the system.

The priority of the threads in the TCS partition may be set at system generation time, and may be dynamically altered by the computer operator to improve system throughput. In addition, certain application programs and/or certain files may be "locked" to a particular thread. This insures the integrity and response time of high priority applications.

LIBRARY OF APPLICATION PROGRAMS

TCS stores its application programs in special disk areas called libraries. These libraries are maintained by TCS. Programs are stored in the TCS libraries in both load module and source formats.

Each entry in the load module libraries consists of a load module preceeded by a prefix which contains the program name and descriptive attributes about the program. These attributes include:

* Program Size
* Thread Number (if the program must execute in a particular thread)
* Security Attributes
* Priority Status
* Transactional or Conversational Status

In addition, TCS maintains a library directory which contains the name of each program and its location in the library. Programs stored in the library may be added, altered, or deleted by on-line utilities provided by TCS.

When a terminal operator attempts to use a TCS program, the library management modules find the program in the TCS library, scan the descri-

ptive information about the program, and place the load module in the ready-to-run queue for rollin to a thread by the task management routines.

The library management modules also provide the security which protects TCS programs from unauthorized use or alteration. When an application programmer catalogs a program into the TCS library, the attributes associated with the entry define the persons and/or terminals having access to the program. These security measures enable the individual programmer to easily and effectively control the uses to which programs may be put.

STORAGE PROTECTION

The TCDMS storage protection module provides each TCDMS application program with protection from interference by any other programs running in the system. Each time a TCDMS program begins execution, its thread area is assigned a unique storage protection key. This enables the program to alter the contents of storage in its own thread, but prevents it from altering the area of any other thread in the system. Thus, a TCDMS application program may not degrade the execution of any other program in the system by inadvertantly changing the contents of another program's storage.

SYSTEM MONITORING

TCS contains several modules which monitor the flow of work through the system. They provide protection against program loops, trap statistics about significant events as they occur, and communicate with the computer operator and/or the user whenever significant conditions arise.

LOOP DETECTION

TCS provides a timing algorithm which terminates application programs after they have executed for a specific period of time without performing a terminal I/O operation. This prevents the tie-up of system resources which can occur when an executing program encounters an instruction loop. The period of time which must elapse before a looping program is terminated may be changed by the computer operator.

STATISTICS AND CAPTURE ROUTINES

The TCS statistics and capture module enable the user installation to record significant events as they occur. Some of the events which may

be recorded on the TCS capture tape include the following:

OP CODE USE
SD FILE USE
PRIVILEGED PROGRAM CALL
PROGRAM CALL
SECONDARY PROGRAM CALL
APPLICATION PROGRAM GIVEN CONTROL
APPLICATION PROGRAM CANCELLED BY OPERATOR
OF CHARACTERS RECEIVED FROM TERMINAL
OF CHARACTERS SENT TO TERMINAL
MESSAGE SENT
MESSAGE RECEIVED
PAGING REQUEST
DATA BASE CALL
ROLLIN COMPLETED

The TCS statistics and capture modules record several levels of statistics. These levels are specified at system generation time to indicate the extent of detail required for the installation's reporting needs. In addition, individual statistics may be recorded dynamically by the computer operator to satisfy short-term diagnostic requirements.

OPERATOR COMMUNICATIONS

TCDMS provides the computer operator with a set of priveliged functions which may be used form the system console to control the flow of work through the TCS terminal network. These functions are involved by enterning commands in response to the operating system outstanding reply for TCDMS. These commands are as follows:

ADD	Add either a local terminal or a group of remote terminals to the TCS network
CANCEL	Cancel a TCS application program
DELETE	Delete either a local terminal or a group of remote terminals from the TCS network.
DISPLAY	Display the status of a TCS thread or task
EOJ	Terminate TCS
IGNORE	Ignore the input from a terminal or group of terminals in the TCS network
RESTART	Reactivate IGNORED terminals

CTL	Designate certain terminals as privileged for the purpose of system maintenance
SWAP	Change thread priorities

MESSAGES AND CODES

TCS provides a comprehensive set of error messages and return codes which are displayed at the users terminal, the operator's console, or written to the system log or capture tape when a significant event occurs. The routing of thse messages may be changed to facilitate the installation's error reporting procedures.

When a message indicating an application program error appears at a TCS terminal, it is automatically followed by a dump of the program's thread area. The dump may be referenced for diagnostic purposes at the user terminal by using a TCS on-line utility.

HIGH LEVEL LANGUAGE INTERFACES

The TCS high level language interfaces interpret user program calls for TCS services and invoke the appropriate TCS modules. The interface modules are link-edited to all BAL, COBOL, FORTRAN or PL/1 programs running under TCS control.

TCS user programs obtain system services by use of the TCS functions. The TCS functions are involved at the CALL or MACRO level in the user program in a manner consistant with the coding conventions as the host language. Each TCS function is described indetail in the portion of this publication entitled ON-LINE PROGRAMMING WITH TCS.

UTILITIES

TCS utilities are general purpose routines which perform services for all TCS users. There are two types of TCS utilities:

USER UTILITIES - These are utilitiy programs for application programmers and terminal operators. They perform message sending, librarian services, program debugging and editing, remote job execution, and other general services. The TCS user utilities are involved at the terminal by entering:

* NAME

when NAME is the name of the utility program desired.

SYSTEM UTILITIES - These are utility programs for system pro-

grammers and computer operators. They perform tasks such as file maintenance, library compression, disk formatting, and other services useful in maintaining system throughput. The TCS system utilities are invoked either from TCS control terminals (privileged terminals for use by system maintenance personnel) or by their placement in privileged TCS programs.

2.0 USING TCS

For the purpose of analysis, the manner in which most computer programs process information can be reduced to the following steps:

1) Input of Information
2) Manipulation of Information
3) Output of Information

Recalling our batch model of the previous section, the INPUT OF INFORMATION step in this process consists of reading the information contained on punch cards into the computer's memory by use of a card reader. The MANIPULATION OF INFORMATION step of the process consists of operations performed with or upon this information by the computer. The OUTPUT OF INFORMATION step of the process consists of writing the results of the computer's manipulation to a printer.

Teleprocessing programs utilize these same three steps. In fact, on a conceptual level there is no basic difference between the logic of teleprocessing programs and that of batch programs. Then both receive information from a source external to the computer, process the information, and send the results of the processing to another point external to the computer.

The differences which exist between batch and on-line programs can be attributed to two things:

The flexibility resulting from the sophistication of terminal hardware.

The real-time nature of teleprocessing environments.

Teleprocessing hardware presents a mind-boggling array of possibilities to the programmer. A typical teleprocessing environment might include both hardcopy (printed paper) and softcopy (CRT tube) terminals, graphics devices capable of producing digitized input from a map location, intelligent terminals capable of processing information, audio response terminals capable of producing output in the form of

human speech, or devices capable of recording printed output on microfilm.

This array of equipment vastly increases the number of ways in which input and output can be formatted and displayed by the on-line program. Thus, while the I/O portions of batch programs are usually similar from program to program, the I/O portions of on-line programs often vary considerably with the nature of the hardware environment in which they operate. It should be noted that the device independent I/O functions of TCS enable the programmer to minimize this variation by handling all I/O formatting tasks.

While on-line programs may utilize the same basic processing steps as batch programs, the speed and frequency with which they go through these steps is vastly increased. On-line programs typically process hundreds or thousands of input-manipulation-output cycles each working day. These cycles are called transactions. Since these transactions usually support over-the-counter inquires, the speed with which they are completed is critical. Hence, on-line programs tend to be small and efficient in their execution time.

Despite the device-orientation and smallness of on-line programs, the programming conventions of the batch and TCS environments are the same. Batch programs may be converted to run on-line under TCS control by replacing their I/O portions with TCS terminal I/O functions.

All the TCS functions are invoked at the CALL or MACRO level in the user program in a manner consistent with the programming conventions of the host language. TCS supports application programs written in IBM Basic Assembler Language, COBOL, and FORTRAN. Sample TCS function usages for each of these languages are shown below.

For Basic Assembler Language
to read a terminal

```
MCALL   READ,AREA=INPUT,LEN=100
```

For COBOL
to read a terminal

```
CALL 'READ' USING INAREA,LENGTH
```

For FORTRAN

to read a terminal

```
CALL READ (Return Code,Area,Length);
```

The functions shown on the following pages comprise all the TCS functions currently available. They are shown in a generalized pattern using the conventional COBOL language CALL statement, with brief descriptions in place of the actual argument names. The use of square brackets indicates that the item enclosed is optional, and need not be coded. In all cases, a return code is provided to indicate the completion status of the operation. This return code is tested by the programmer using the normal technique for testing return codes in the host language.

THE ABEND FUNCTION

```
CALL 'ABEND' USING [USER-SUPPLIED ABEND CODE]
```

The ABEND function initiates TCS abnormal termination processing for the user program. A hexadecimal dump of the user program's thread is produced. The optional abend code is displayed at the terminal (for on-line programs) or in the printout (for batch programs).

The ABEND function is useful in tracing the execution of a program. Several ABEND functions, each with a different abend code, may be imbedded at critical locations in a program to provide a trace of its execution.

CALL 'CAPTUR' USING	Location of the data to be captured, length of the data to be captured, [user-supplied identification number] .

The CAPTUR function causes data to be written from any area in the user program to a special capture file on tape or disk.

TCS writes a prefix on each record captured by the user program. This header includes the following information:

NAME OF PROGRAM
TERMINAL IDENTIFICATION (TID) NUMBER
TYPE OF CAPTURE (i.e., on-line, batch, priveleged)
DATE
TIME OF DAY

The CAPTUR function is useful in monitoring the execution of a program to provide a record of certain transactions.

```
CALL 'DATE' [J]
```

The DATE function returns the current date to the user program. If the J option is used, the Julian date is returned. If no option is used, the date is returned as a packed, unsigned number.

The DATE function is useful whenever the user program must identify the current date.

THE EOJ FUNCTION

```
CALL 'EOJ'
```

The EOJ function initiates TCS end-of-job processing for the user program. A standard EOJ message is written to the terminal. No dump is produced.

```
CALL 'TFGET' USING
     'TFGETU'
```

File Name

Location in User Program where records are placed

[Search Argument]

```
CALL 'TFPUT' USING
     'TFPUTU'
```

File Name

Location in User Program where records are placed

[Relative Record Number]

```
CALL 'TFENQ' USING
     'TFDEQ'
```

File Name

The TCS file I/O functions enable the user on-line program to use data files organized for the BDAM and ISAM access methods. TCS batch programs can use standard ISAM, BDAM, USAM, or BPAM access method commands. These file I/O functions do not access files in the DMS data base.

TCS provides the following file I/O functions:

TFGET	retrieves one or more records from a file
TFGETU	retrieves on record from a file for update
TFPUT	adds one or more records to a file
TFPUTU	adds one updated record to a file
TFENQ	holds a file for exclusive access by one program
TFDEQ	releases a file for access by other programs.

The TCS file I/O functions enable the user program to process files either sequentially or directly. For direct processing, the user may specify a search argument.

THE GETCHR FUNCTION

```
CALL 'GETCHR' USING
```

Location in user program where information is placed

Amount of information requested

[Terminal identification (TID) number about which terminal is requested]

The GETCHR function provides information about the environment in which the user program is running. This information includes the following:

DEVICE-DEPENDENT INFORMATION	GENERAL INFORMATION
TID of specified device	Initial program called
Number of characters per line	Most recently fetched program
Number of lines per display	Most recently loaded program
Buffer size	Status of calling program
TCS device module number	Cataloged size of calling program
TCS data module number	Actual size of calling program
Status of terminal	Thread size
Physical address of terminal	Load address of calling program
Default TID (for screen to hardcopy utility)	Installation identifier
Relative address of terminal for control unit	Thread number

This information is useful in a wide variety of applications. For example, use of the GETCHR function enables the user program to dynamically ascertain the nature of its hardware environment and tailor its use of the terminal I/O functions accordingly.

```
CALL 'MESGSW' USING
```

Message control block for user program

Location of message text in user program

Length of message segment

[Destination codes to which message will be sent]

[Number at destination codes listed]

[Identity of terminal to which a reply will be sent]

The TCS message switching function, MESGSW, enables the user program to send messages or data to any terminal in the TCS terminal network. It should be noted that the MESGSW function controls the handling of messages which originate from an executing on-line program. Messages originating from a terminal are handled by the message switching utility.

Messages are formed in pieces called segments. When the user program completes the last segment of a message, TCS combines the segments and sends the message to its specified destination.

The user program may specify the recipient of a message either by individual terminal identification (TID) number, or by referencing groups of one or more TIDS defined by a destination code. Security restriction may be affixed to a message by assigning it one or more class codes. All messages are treated as text by TCS, using the maximum line length of the receiving device.

The TCS message switching modules maintain a system of queues for each terminal in the network. Messages are placed in these queues when they are sent. If a receiving terminal cannot display a message when it is sent, the message may be retrieved from its queue at a later time.

TCS message switching is useful in any circumstance where information is transferred from one terminal in the network to another.

```
CALL  LOAD    USING
      LOADT
      FETCH
```

Name of overlaying module

Location in user program where module is to be placed

Length of overlaying module

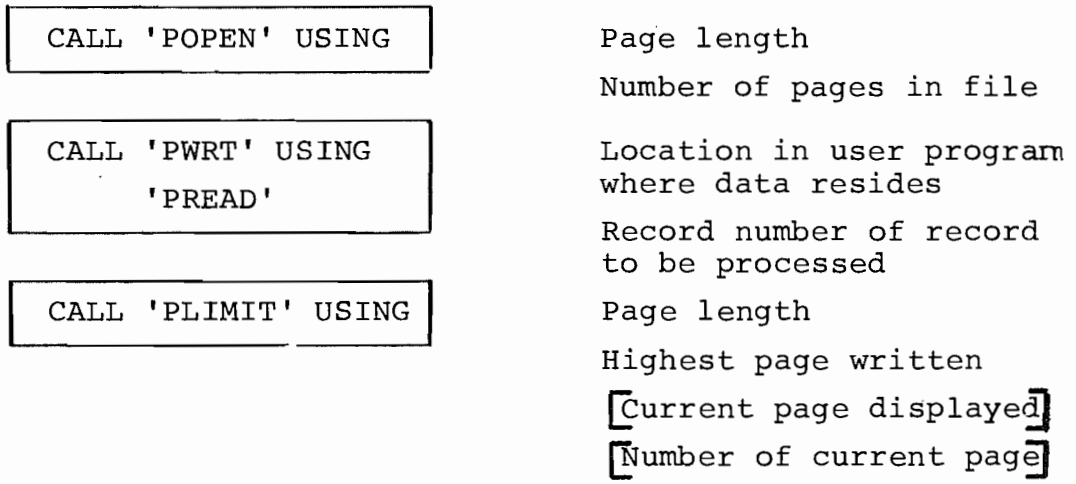
In addition to supporting OS or VS planned overlay, TCS provides three functions which may be used to overlay all or part of the user's thread.

The LOAD function loads a module into the user program's thread. The user program retains control.

The LOADT function loads a module into the user program's thread, and transfers control to the new module. The program attributes remain those of the user program.

The FETCH function loads a module into the user program's thread, transfers control to the new module, and changes the program attributes to those of the new module.

THE PAGING FUNCTIONS



The paging functions enable on-line programs which produce large amounts of output to arrange the data for convenient viewing by terminal operators. With the paging functions, the program's output data is arranged in groups called pages in much the same manner as information is placed in the pages of a book. The terminal operator may display the information one page at a time by using the paging utility.

The POPEN function creates and initializes a page file for later use by the program. Page files are stored on direct access devices.

The PRWT function writes data one page at a time to individual pages in the page file created with the POPEN function.

The PREAD function reads data one page at a time from the page file to an area in the user program.

The PLIMIT function enables the user program to determine the number of the last page displayed by the terminal operator, and to place an upper limit on the number of the highest page available for display.

THE PRINTOUT SPOOLING FUNCTIONS

CALL 'PSOPEN' USING	Printout control block name Destination code to which printout will be sent Number of destination codes specified
CALL 'PSPUT' USING	Printout control block name Location in user program of data to be written
CALL 'PSCLOS' USING	Printout control block name

The printout spooling functions enable on-line programs to create printout data sets which are generated one line at a time and spooled to a disk until the entire data set is complete. Spooled printouts may be written to any terminal or group of terminals in the TCS network.

The user program may specify the recipient of a spooled printout either by individual terminal identification (TID) numbers of by referencing groups of one or more TIDS defined by a destination code. Security restrictions may be assigned to a printout by assigning it one or more class codes.

The PSOPEN function creates and initializes a printout data set for later use by the program. The PSWRT function writes a line of output to the data set. The PSCLOS function closes the data set and sends it to the terminal(s) specified in the printout spooling control block.

The printout spooling functions enable several on-line programs to concurrently generate printouts for the same device without getting their outputs intermixed.

THE SD FILE FUNCTIONS

CALL 'SDOPEN' USING	File name
	Record length
	File size
	[Authorized TID]
	[Highest record written]
	[Password]
CALL 'SDWRT' USING	File name
'SDREAD'	Location of record in user program
	[Record number]
	[Authorized TID]
	[Record length]
CALL 'SDCLOS' USING	File name
'SDDEL' USING	[Authorized TID]

The SD file functions enable TCS programs to access and maintain disk files. SD files contain fixed length records, the size of which is specified by the user program. They may be accessed either sequentially or directly. The maximum SD file size, the number of SD files that may be open by one program at a given time, and the longevity of a program's files are TCS sysgen options.

The information stored in SD files can be protected from unauthorized access by the same security measures protecting the programs which create them. In addition, programs having several SD files may restrict their access by any combination of the following criteria:

APPLICATION PROGRAM NAME
TERMINAL IDENTIFICATION (TID) NUMBER
PASS WORD

The SDOPEN function creates and initializes all SD files for the user program. SD file security is specified in the SDOPEN function.

The SDWRT function writes a record to the file queues with the SDOPEN function. The SDREAD function reads the specified record from the file, and places it in the specified area of the user program.

The SDCLOS function closes an SD file which had previously been opened with the SDOPEN function. When it has reached the limit of maximum files open at a given time; a user program can use the SDCLOS function to temporarily remove a file to make space for another.

The SDDEL function permanently deletes an SD file from the SD file disk area.

The SD file functions are useful whenever a program requires disk space to temporarily or permanently store information. They isolate the application programmer from disk formatting considerations.

THE SETEID FUNCTION

CALL 'SETEID' USING	Bit mask table

The SETEID function allows the user program to disable functions which program attention (PA) or program function (PF) keys have been assigned by the installation. TCS recognizes three PA keys and twelve PF keys on the terminal devices it supports. When one of these keys is depressed and the SETEID function has been used, TCS does not perform the function assigned to the key, and the interrupt is passed to the user program.

The SETEND function enables the user program to ignore one or all of these interrupt keys. If the SETEID function is used which the program is in conversation with a terminal which neither has nor stimulates these keys, the function is ignored.

THE ROLOUT FUNCTION

```
CALL 'ROLOUT'
```

The ROLOUT function causes the user program to be rolled out of its thread.

The ROLOUT function is useful when the user program will be executing for a prolonged period of time. It increases overall system throughput by allowing other programs to execute in the thread. The program which issues a ROLOUT function is placed in the READY-TO-RUN queue, and is rolled back into the next available thread.

THE TIME FUNCTION

CALL 'TIME' USING	Format of time desired

The TIME function returns the time of day to the user program. The time is returned either as hours, minutes, and seconds elapsed since the previous midnight; the number of hundredths of seconds elapsed since midnight, the number of CPU timer units elapsed since midnight.

THE TERMINAL I/O FUNCTIONS

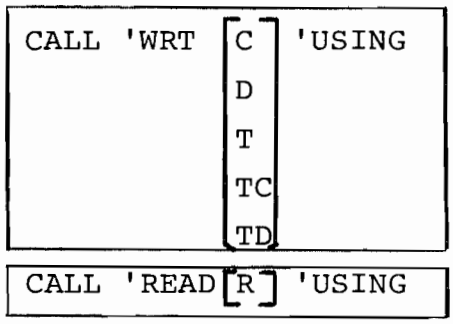
The TCS terminal I/O functions enable the user program to communicate with the terminals in the TCS terminal network. There are two kinds of terminal I/O functions provided: terminal-independent functions and terminal-dependent functions.

The terminal-independent I/O functions provide the terminal control characters required for the user program to read or write to terminal devices. When the program reads data from a terminal, TCS removes the control characters inserted in the data stream by the device before the data is passed to the program. When the program writes data to a terminal, TCS inserts the necessary control characters before actually sending the data. An optional new-line symbol may be imbedded into the data stream by the user program. If it is not used, the carriage or cursor is automatically returned where a line is filled.

The use of terminal-independent I/O function is encouraged in environments where a program may be called from more than one terminal type. It isolates the program from these device considerations.

The device-dependent I/O functions require that the program provide and manipulate all terminal and line control characters during both input and output operations. Although they offer the user programmer more flexibility in display formatting, they limit the applicability of the program to single-device environments.

With one exception; both types of I/O functions can be used in one program.



Location in user program from which data is written

Length of data

Logical line length of terminal

Location in user program where data is placed

Length of data

Number of bytes available to be read before READ operation was initiated

The terminal-independent write functions provide several processing options.

WRT - Writes data to terminal

WRT [C] - Writes data to a terminal and waits for response. TCS places the terminal operation response into a buffer for later retrieval by the program

WRT [D] - Writes data to a terminal, and then cancels the user program after the write operation is completed

WRT [T] - Writes text to a terminal. The T option prevents the splitting of words between lines.

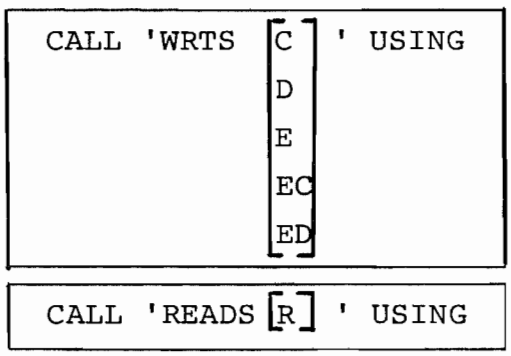
In addition, the T option can be used in combination with the D and C options.

The programmer may specify a line length for the write operation by use of the logical line length parameter. If this parameter is not specified, TCS uses the physical line length of the device being written to.

The terminal-independent read function provides two processing options:

READ - Reads a terminal operator's entry. An optional parameter returns the length of the operator's response to the program. Another optional parameter enables the user program to specify one or more delimiters. This parameter causes TCS to read the terminal operator's entry and stop when the first delimiter is encountered in the data stream.

READ [R] - Reads the terminal operator's entry and returns the TCS location counter to the point where it was before the read option was begun. A subsequent read function starts at the beginning of the data. An optional parameter returns the number of bytes read to the user program. A return code indicates the relationship between the length of data read and the length of the terminal operator entry.



Location in program from which data is rewritten

Length of data

Location in program where data is placed

Length of data

Number of bytes available to be read before READS operations was initiated

List of delimitation

The terminal-dependent write functions provide several processing options:

WRTS - Writes data to a terminal

WRTS [E] - Erases a CRT screen before writing data to a terminal.

WRTS [C] - Writes data to a terminal and waits for a response. TCS places the terminal operator's response into a buffer for later retrieval by the program.

WRTS [D] - Writes data to a terminal, and then cancels the program after the write operation is completed.

In addition, the E option may be used with the D and C options.

The terminal-dependent read functions provides two processing options:

READ - Reads a terminal operator's entry. An optional parameter returns the length of the operator's response to the program. Another optional parameter enables the user program to specify one or more delimiters. This parameter causes TCS to read the terminal operator's entry and stop when the first delimiter is encountered in the data stream.

READ [R] - Reads the terminal operator's entry and returns the TCS location counter to the point where it was before the read option was begun. A subsequent read function starts at the beginning of the data. An optional parameter returns the number of bytes read to the user program. A return code indicates the relationship between the length of data and the length of the terminal operator entry.

THE TCS UTILITIES

The TCS utilities are on-line programs which perform general purpose tasks for TCS programmers and terminal operators. These tasks include the following:

WRITE SOURCE PROGRAMS - The TCS utility UPDS enables the programmer to write programs from an on-line terminal in any of the four TCS user languages.

ASSEMBLE/COMPILE SOURCE PROGRAMS - The TCS utility UPDS also allows the programmer to submit a completed source program for remote job execution (RJE) in a batch partition.

TEST SOURCE PROGRAMS - The TCS utility UQ enables the programmer to monitor the execution of a job. For example, the programmer can use a UQ to examine the SYSOUT data set associated with a job in order to determine if there were any assembler/compile errors. The UPDS utility can then be used to correct any errors and the job can be resubmitted.

CATALOG LOAD MODULES - Once a program is tested and debugged. The programmer can place an operational version of the program into the TCS library and protect it with security restrictions using the ULIB utility.

DEBUG OBJECT PROGRAMS - If an error occurs during operational use of a program, the UDUM utility can be used to examine the dump of the program's thread and the error can be corrected in the object code with the ULIB utility and/or in the source code with the DPPDS utility.

SEND MESSAGES - The message switching utility can be used to send messages between terminals or display messages sent to terminals without the intervention of a user program.

DISPLAY PAGED OUTPUT - The user can display pages built by a program which used the paging function.

The TCS utilities are invoked by using the general command:

*UTILITYNAME [command(s)]

at a terminal. The following pages describe each of the above utilities in greater detail.

THE LIBRARY UTILITY - ULIB

ULIB is used to maintain programs in the TCS library of application programs. ULIB is invoked by the general command

*ULIB, command

The following commands and options are available:

*ULIB,CAT, program name, [descriptive parameter]

The CAT command causes a program to be cataloged to the TCS library. The descriptive parameters include program size, thread lock number, and security attributes.

*ULIB, ZAP, program name, location, data

The ZAP command causes data at the location to be either compared with the data supplied in the command or replaced with data supplied in the command, at the option of the terminal operator. The ZAP command is used to correct coding errors detected by the compiler or assembler.

*ULIB, DIS, program name(s)

The DIS command causes certain descriptive information about a program or group of programs to be displayed. These attributes are:

Program Name Actual Program Size
Cataloged Program Size Thread Number
Number of address constraints in program

*ULIB, DEL, program name

The DEL command causes a program to be deleted from the TCS library.

THE DUMP UTILITY - UDUMP

Whenever a TCS program terminates abnormally, an error message is sent to the terminal which was using the program when the error occurred, and a dump of the thread in which the program was executing is produced. UDUMP is used to reference these dumps from a terminal.

When a dump is produced, certain information is added to the dump as it is written to a disk. This information includes the program name, the date and time of failure, the TID of the terminal using the program when the error occurred, the number assigned to the dump, the error message which was displayed at the user terminal when the error occurred, the contents of general purpose registers Ø through 15 at the time of the error, the contents of the floating point registers at the time of the error, the thread address, the program load address, and the Program Stauts Word (PSW) of the program. This information is displayed on the first page of the dump. The subsequent pages of the dump are displayed by depressing the enter key.

The general commands for UDUMP are as follows:

*UDUMP ALL - Displays a list of dumps in the file.

*UDUMP PRINT NAME OR NUMBER - Prints the dump for the program name specified or the dump number specified.

In addition, during the use of UDUMP, the terminal operator may enter one of the following commands:

R=M - Causes machine addresses to be displayed. This option is in effect when UDUMP is invoked.

R=T - Causes the addresses displayed to be relative to the beginning of the thread.

R=P - Causes the addresses displayed to be relative to the beginning of the program.

R=* - Causes the subsequent addresses displayed to be relative to the current address displayed.

CRn - Causes the display to begin at the address in register n.

Address - Change display to the address specified.

Snnnnnnnn - Search for the first occurrence of the character string represented by nnnnnnnn.

± displacement - Move the display forward or backward by the displacement specified.

THE MESSAGE SWITCHING UTILITY

The message switching utility enables the terminal operator to send or receive messages to or from any terminal in the network. The commands for the message switching utility are as follows:

*M. dest code(s). text of message - Causes the message to be sent to the terminal(s) represented by dest code.

*M.dest code(s).class code(s).text of message - Causes the message to be sent to the terminal(s) represented by dest code. Also causes the message to be assigned the class codes specified.

*MDISABLE - Causes the terminal from which the command is entered to be disabled from receiving all but urgent messages.

*M.ENABLE - Caused the DISABLE status to be removed for the terminal from which the command is entered.

*M.D.[.TID] [.TIME] - Causes display of the message switching status either of the terminal from which the command is entered, or (when a TID number is entered) any non-priveleged terminal in the network. This status includes the following information:

- Sending and receiving class codes
- Disabled status
- TID of alternate terminal
- Message number, status, and other information for all messages sent to the terminal which have not been received.

The time parameter enables the terminal operator to retrieve the above information for a terminal during a specified time interval.

*M.n[.TID] - Causes the message with the number n to be displayed. If the TID parameter is used, messages queued to other terminals can be displayed.

*M.R - Causes the first page of the message currently being displayed to be re-written.

*M.DELETE.n - Causes the message with the number n to be displayed.

*M.PURGE - Causes the queue assigned to the terminal from which the command was entered to be purged of messages.

*M.ALT=tid - Causes the TID specified to be assigned as the alternate for the terminal from which the command was entered. This means that if the terminal becomes inoperative or disabled, any messages set to it are re-routed to its assigned alternate.

*M.ALT=REMOVE - Remove a terminal from alternate status.

The paging utility enables terminal operators to display the pages in a paging file. Each terminal has one paging file assigned to it. The pages in the file are numbered sequentially beginning with 1.

The commands for the paging utility are:

*?/C - Causes the most recently displayed page to be re-displayed

*P/n - Causes the page with the number n to be displayed

*P/N - Causes the next page to be displayed

*P/P - Causes the previous page to be displayed

*P/A - Causes the highest page in the file to be displayed

*P/D - Deletes the page file assigned to a terminal

THE TCS SYSTEM STATUS UTILITY - UQ

The UQ utility enables the terminal operator to monitor and control the execution of jobs in the batch partition, and to monitor the performance of TCS. The functions of UQ are divided into two catagories:

DISPLAY OPERATIONS

*UQ [display operation] [option]

Display names and status information for active system tasks, user tasks, and subtasks in the batch partitions.

Display all queued jobs by class or job name or display the status of the input, output, and hold queues.

Display the input JCL, system messages, SYSIN and SYSOUT data sets.

Display the status of all system tape drives.

CONTROL OPERATIONS

*UQ [,job name] [,control operation]

Place a job on hold. This operation causes a job to remain in its queue until it is released.

Release a held job. This operation causes a job to be placed into the normal job stream.

Cancel a job. This operation permanently removes a job from the job stream.

3.0 THE TCDMS ENVIRONMENT

The TCDMS environment has three elements; the hardware configuration of the installation, the resident software of the installation, and the unique version of TCDMS operating at the installation. This section describes the hardware and software which may comprise the TCDMS environment, and the methodology of planning, tailoring, and installing TCDMS.

THE TCDMS HARDWARE ENVIRONMENT

The TCDMS hardware environment consists of the central processing unit and its peripheral storage devices (disks and tapes) communications line and their controlling devices and terminals. The minimum hardware configuration required to support TCDMS includes the following:

1 IBM System 370/145 with 252K bytes storage
1 IBM System 370/145 compatable card reader/punch
1 IBM System 370/145 compatable printer
1 IBM System 370/145 campatable Tape Drive and Control Unit
1 IBM Model 3330 Disk Drive and Control Unit
1 Terminal and Control Unit from the list below

Minimum Storage Requirements:

TCS 100K bytes
DMS 40K bytes
TCDMS 140 K bytes

Other hardware devices supported by TCDMS include:

Central Processing Units
IBM System 370/145, 155, 158,165,168
Communications Controllers
IBM 3705
IBM 2914
Terminal Devices
IBM 2740 Models 1 and 2 (Basic and Station Control)
IBM 2260 Remote
IBM 3271 Models 1 and 2

IBM 3272 Models 1 and 2
IBM 3275 Models 1 and 2
IBM 3277 Models 1 and 2
IBM 3767
IBM 3284 Models 1, 2, and 3
IBM 3286 Models 1 and 2
TEKTRONICS 4014 Graphics Scope
Ultronics CRT
Ultronics Impact Printer
Ultronics Thermal Printer

THE TCDMS SOFTWARE ENVIRONMENT

The TCDMS software environment consists of the operating system and other system software which controls TCDMS, and the user programs controlled by TCDMS. The following operating software is required to operate TCDMS:

IBM System 370 OS/MFT with or without HASP
IBM System 370 OS/VS1 Release 3

The following high level languages and file access methods are supported by TCDMS:

IBM System 370 BASIC ASSEMBLER LANGUAGE
IBM ANS COBOL
IBM PL/1
IBM FORTRAN IV G
IBM Data Language/1 (DL/1)
Multiple Access Directory System (MADS)
Data File support
Basic Direct Access Method (BDAM)
Indexed Sequential Access Method (ISAM)

Generally, the hardware and software elements in the TCDMS environment are established before the installation of TCDMS. For this reason, TCDMS may be readily adapted to satisfy a wide variety of installation dependent requirements. This process of adaptation has two parts; system generation and system initialization.

SYSTEM GENERATION

TCDMS system generation is the process by which the individual instal-

lations version of TCDMS is defined and created. This process has three steps; planning, creating the TCDMS source tape, and generating TCDMS on site.

The planning step consists of analyzing the needs of the installation to determine the optimum system configuration. Consideration must be given to the quantity of main storage available to teleprocessing applications, the number and type of programs and terminals using the system, the maximum acceptable response time, and a host of other factors.

Once the needs of the installation have been established, the TCDMS source tape is created. This tape consists of the JCL to assemble and link-edit the modules of TCDMS, and an assembler language version of each module to be included in the system.

One of these modules contains the global symbols by which TCDMS is defined. There are over one hundred of these parameters to be specified in defining the exact nature of the system and its environment. In general, these parameters describe the hardware and software environmnet of the installation and the attributes of TCDMS. These attributes range in significant size from the size and number of threads in the system to the maximum number of new-line symbols which may be imbedded in the data stream by a user program during a write operation. When the values for these parameters have been specified and the appropriate modules have been placed on the source tapes, TCDMS can be generated on-site.

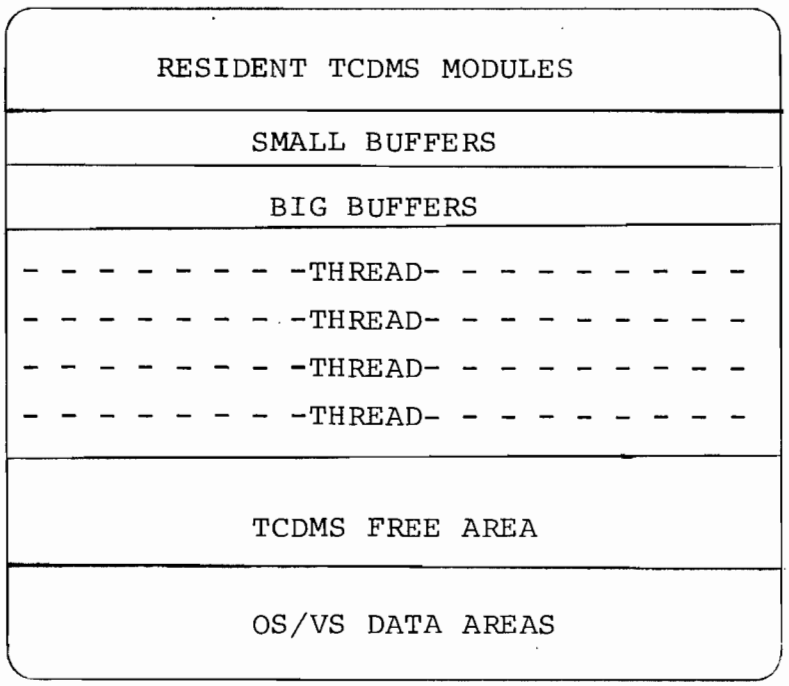
The on-site generation process consists of; 1) executing the JCL to assemble and link-edit the resident modules of TCDMS, creating the TCDMS libraries, assembling and then cataloging the transient modules and system utilities to these libraries, and executing TCDMS in the operating system job stream.

TCDMS INITIALIZATION

Once the system generation process is complete and TCDMS has been executed, the system initialization process can occur. Initialization is the process by which space is allocated in main storage for TCDMS control blocks, resident modules and resident directories. The initialization process formats the teleprocessing partition of main storage

in the manner shown below:

THE TELEPROCESSING PARTITION



The initialization process also allocates space on direct access storage devices for SD and Paging files and their directories, rollin/rollout directories, TCDMS libraries, the message file, and DMS files. When initialization is complete, TCDMS is ready to process user requests.

4.0 DATA MANAGEMENT CONCEPTS

Data may be defined as the representation of information. All computer programs handle data. In some programming environments, the individual programmer is responsible for the design and maintenance of his own data files. In other environments these tasks are handled by complex systems call Data Management Systems. The basic difference between these two approaches to data handling is the degree of separation between the application program and its data. It is measured by the amount of knowledge the application programmer must possess about his data in order to handle it.

The data management component (DMS) at TCDMS provides a complete system of data management services which isolate the user from concern with the physical attributes of the data stored in its data base. This section gives a conceptual overview of the manner in which TCDMS performs these data management services.

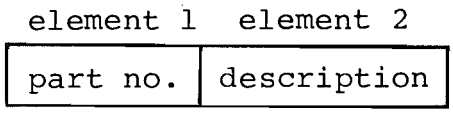
THE STRUCTURE OF THE DMS DATA BASE

THE DATA ELEMENT

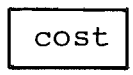
The unit of data available to the DMS application programmer is the data element. The data element is also the smallest unit of data in the DMS data base. Its length may vary according to the attributes of the element. Data elements are stored within the data base in compressed format. There are thirteen types of data compression available. When the data base is established at each installation, the compression type for each element is set. DMS converts data to and from these compressed formats as it transfers data to and from the data base. The user program normally handles only the external, or uncompressed form.

THE DATA SEGMENT

All data elements in the DMS data base are contained in segments. A segment can contain one or more elements. A segment is the smallest unit of data handled by DMS. When an application program requests a data element, DMS retrieves the entire segment which contains that element, and places it in a special area called the Segment Work Area. The element requested is then passed to the program from this segment work area.



a multi-element segment

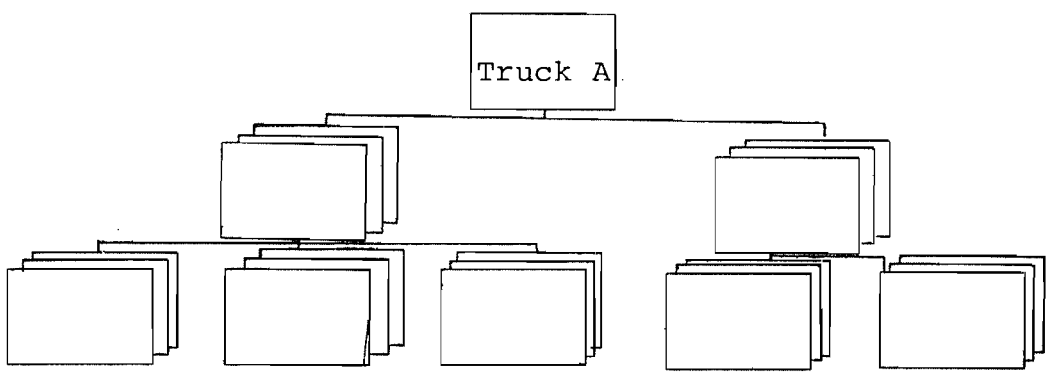


a single-element segment

DMS data elements are stored in segments, which may be comprised of one or more elements. Elements are usually combined in segments to facilitate rapid retrieval of information which is usally found in the same user request.

THE FAMILY

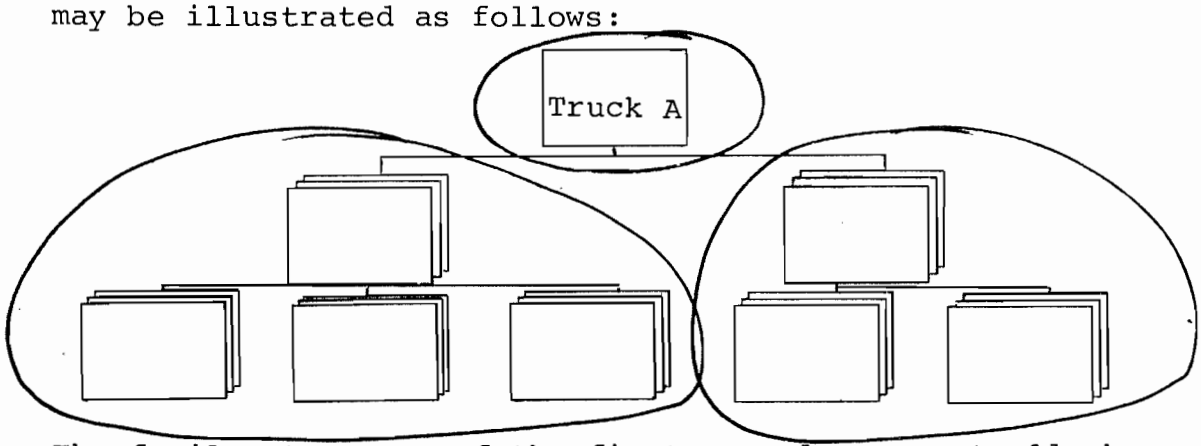
Each segment in the DMS data base must exist in relation to some other segment. When a user program combines one or more segments in a heirarchical relationship, the result is a family. Thus, a DMS family is a collection of segments which are logically dependent on one segment for their meaning. The segment which gives the family structure its meaning is called the Root Segment. The diagram below illustrates this concept.



This data structure is a family of data segments related to the root segment, TRUCK A. All the elements in the family are given meaning by the fact that they describe TRUCK A.

THE FILE

There are two kinds of DMS files; physical files and logical files. A physical file is a group of segments which is stored in one contiguous block of space on a direct access storage device. A logical file is a group of segments which are logically related by a user program. Using our previous example, this concept may be illustrated as follows:



The family structure of the first example may actually be composed of data which resides in three different physical files; the truck file, the components file, and the service file. When they are logically related to one another by a user program, they become that program's logical file.

Data segments are stored together in physical files on direct access storage devices in a manner which is defined when the data base is initialized. This physical grouping is irrelevant to the user program which is only concerned with the logical relationships between elements.

Each DMS logical file may contain up to 256 heirarchical levels of

data. The DMS data base may contain one or more files, which may be separately indexed, loaded, or reorganized. Data in one DMS file can be a pointer to data in another file, this providing extensive interrelationships between files.

A pointer is an element or segment in one file which defines the location of segments in another file. Rather than actually containing data, a pointer segment contains information which describes where the data is stored in another file. Any segment which is pointed to from another file contains back pointers that identify the segments which point to it. Pointers in one file always identify the root segments of families in another file.

There are two types of data in DMS files; multiple chain files and root chain files. Multiple chain files contain data that is used by several programs. Individual programs access this data with pointer segments within their own files. Root chain files contain data that is pertinent to an individual program. These files may contain actual data, or they may consist of pointers to multiple chain files. These two types of files provide the networking capabilities within the DMS data base.

The data elements available to a program, and the logical relationships between these elements are defined in a special control block called a Data Base Control Block (DBCB). Once this definition has occurred, the user program may access its data files by specifying which elements it wishes to process. DMS provides a series of functions to accomplish this.

There are four major functional areas in DMS; the access method, segment processor, request manager, and the file and family protection system.

THE DMS ACCESS METHOD

The DMS access method consists of a group of modules which map the complex logical structures in the data base to and from their physical representation on direct access storage devices. The access method supports the extreme flexibility in data manipulation afforded to the user program, while at the same time equalling or exceeding the performance of more conventional data accessing techniques. Data integrity is protected by internal access method routines. The access method also logs the information necessary to recover from unexpected system failures.

The access method structures segments into families. Each family is normally stored in one physical block on a direct access device. Both families and individual segments may, however, occupy several physical blocks if necessary. The modules which structure segments into families and retrieve segments from the data base are known as data block management modules.

The access method identifies a family for the purpose of physical access by its relative block number within the file, and its relative family number within the block. These are converted to physical addresses by the access method which then requests I/O operations from the resident operating system. Because these addresses are relative, pointers between families and files need never be updated in any kind of file maintenance procedures between file reorganizations.

The access method is designed so that the physical structuring of segments into families is independent of the technique used to access families. This means a family may be accessed in several

different ways. Direct access by root segment is possible via the access method index handling routines. These routines also provide access sequentially by root segment beginning anywhere in a file and proceeding forwards or backwards from there. A family may be accessed directly via a pointer from a family in another file: this allows a multiple indexing capability. Physical sequential access without regard to root segment sequence is available for rapid retrieval in situations where order is unimportant. Files may also be organized randomly, and accessed using a randomizing routine.

THE DMS REQUEST MANAGER

The request manager modules process user program requests for data management services. It converts these requests into the appropriate format for the access method. The request manager locates and maintains the user program's position within the data base, and converts to and from its compressed format.

There are two main tasks performed by the request manager; scan processing and element processing. The scan processor modules establish the hierarchical path through the data base that contains the data elements.

Requested by the user program, the element processor modules establish the unique set of segments, which satify the user program's request.

Both the scan processor and element processor modules convert user requests for data elements into internal requests for the segments which contain them. These internal requests are then passed to another set of modules - the segment processor.

THE SEGMENT PROCESSOR

The segment processor modules handle internal requests for segment level manipulation. They manage the segment work area (SWA) where segments are constructed from data elements for insertion into the data base, and where retrieval segments are stored so requested elements can be extracted from them.

The segment processor modules handle segment-level requests from the request manager routines. They manage the segment work area (SWA) where segments are constructed from data elements for insertion and where retrieved segments are stored so requested data elements can be retrieved from them. The segment processor keeps all the segments in the SWA linked together by segment type (retrieval, insert/update, or delete). These SWA "chains" are linked to the appropriate entries in the segment descriptor table. Each segment in the SWA has a header which describes where the segment belongs in a family. The segment processor creates and maintains this header. When the segment work area fills with segments, the segment processor attempts to clear it by releasing all non-insert/update segments.

When the segment processor receives a retrieval request for a segment from the request manager, it first checks the segments in the SWA to determine if the segment has already been read. If the request can be satisfied by a segment already in the SWA, the segment processor returns control to the request manager. Otherwise it builds the required control blocks and calls the access method to retrieve the segment. When the segment processor receives segments from the request manager for insert/update, it stores them in the SWA until the update is completed. The segment processor then arranges the segments in the correct physical file sequence and passes them to the access method, one segment at a time, for insertion

inot the data base.

The segment processor routines also handle pointer segments. For retrievals, this means making two requests to the access method; one for the pointer and one for the segment pointed to.

DMS DATA PROTECTION

The file and family protection modules of DMS provide several levels of data base protection. Data integrity is assured during access by a system of validation routines. In the event of an abnormal termination by the user program, the DMS protection modules release control of the files held by the terminating program to insure the integrity of the affected files.

The data protection modules provide the basis for data base recovery in the event of a system failure. They write a record of each data base update transaction (inserts, deletes, and changes) to the system capture file. The remainder of the modules in the file and family protection system are the file recovery modules. These provide the capability to restore the data base in the event it becomes damaged during a system failure. The backout module removes the transactions which were only partially completed. Other modules apply transactions from the capture file to a backup copy of the data base.

DATA BASE DEFINITION AND DBCB GENERATION

After a data base has been designed for an installation, and loaded

using the data base load utility programs it must be made ready for use. There are two steps in this procedure.

(1) The data base definition process creates two system files which contain descriptive information about each data element and segment in the data base. Each entry in the data dictionary contains a data element name, accessibility specifications, compression type, the external and stored lengths, the location of the data element in the segment, and some data validation information. The segment dictionary entries identify each segment and its hierarchical position within the file. These dictionaries contain the information used by the data management system to locate data elements requested by an application program. (2) The DBCB generation process creates a data base control block (DBCB). DBCBs specify which data elements are accessible to an application program. Each program which accesses the data base has a DBCB associated with it.