Ed Bryan
17 Sept 1972

MEMORY MANAGEMENT, RESOURCE MANAGEMENT, AND COMMUNICATIONS MANAGEMENT IN THE XEROX OPERATING SYSTEM

I. MEMORY MANAGEMENT

1.0 Core Layout

1.1 Resident Monitor

- Characteristics
- Size
- Location

1.2 Non-resident Monitor

- Characteristics
- Size
- Location

1.3 User Space

- Context
- Program Area
- Local Dynamic Area
- Common Dynamic Area
- Blocking Buffers, etc.

2.0 Memory Allocation

2.1 User Services

- Memory Management
- Program Overlays
- Program Loading

2.2 Swapping

- Characteristics
- Swapper Allocation (core and disk)
- I/O Supervisor Interface

2.3 Shared Processors

1.0 Core Layout

XOS uses the capabilities of the SIGMA Memory Map and Access Protection features to define a resident and a non-resident portion of the monitor. These are depicted in Figure 1.

1.1 Resident Monitor - The resident portion of the monitor resides at the low end of core memory and is mapped 1:1 (ie., the virtual address is the physical address). This resident portion is itself divided into 3 parts: (1) that portion below module ITCBBST ("Internal Task Control Block for the Basic-System Task") which executes either mapped or unmapped; (2) that portion above module ITCBBST which executes only unmapped; and (3) the INITIALIZATION and DEBUG module portion of the resident monitor which is only physically resident at system initialization (boot) time. This portion is "clobber code", needed only for initialization and patching purposes, which is subsequently overlaid in real and virtual memory by user programs. This technique allows user programs which always execute using the memory map, to effectively "overlay" a large portion of the resident monitor. Thus, user programs are not penalized virtually for a portion of the memory occupied by the resident monitor. The resident monitor will vary in size from 12K (a batch-only system) to 19.5K (full timesharing plus TAM character and message mode support) .

1.2 Non-Resident Monitor - The non-resident monitor (NRM) executes at the high end of virtual memory, uses the memory map, and consists of several user-specified segments or overlays. Each overlay is made up of one or more elements. Each element is made up of one or more assembly modules. A user is associated with an NRM element by requesting a monitor service via the CAL1 trap mechanism. A resident monitor module handles the trap, decodes the request and transfers control to the appropriate NRM module. All inter-module branching is accomplished via a call to a resident "Non-Resident-Monitor Handler" module. This module either verifies that the element (of which the requested module is a part) is already loaded into memory or causes this element to be loaded into memory (of course, references to resident monitor modules require no such loading). The memory map is then loaded to reflect the location of the NRM element and the call is complete. When one of these elements that was loaded into memory is no longer in use, it remains in memory, but is marked "disengaged". The resident monitor maintains statistics on the frequency of use of these "disengaged" elements, and when additional memory is required, the least frequently used element(s) are overlaid by the program or element that requires space. Using this technique, XOS is able to make use of any "unused" memory (even if it had been allocated as a resource but had not been "gotten" yet by the various active tasks). This will reduce the number of requests for loading NRM elements. Note in the above discussion that the unit of loading is the element not the segment. Although the

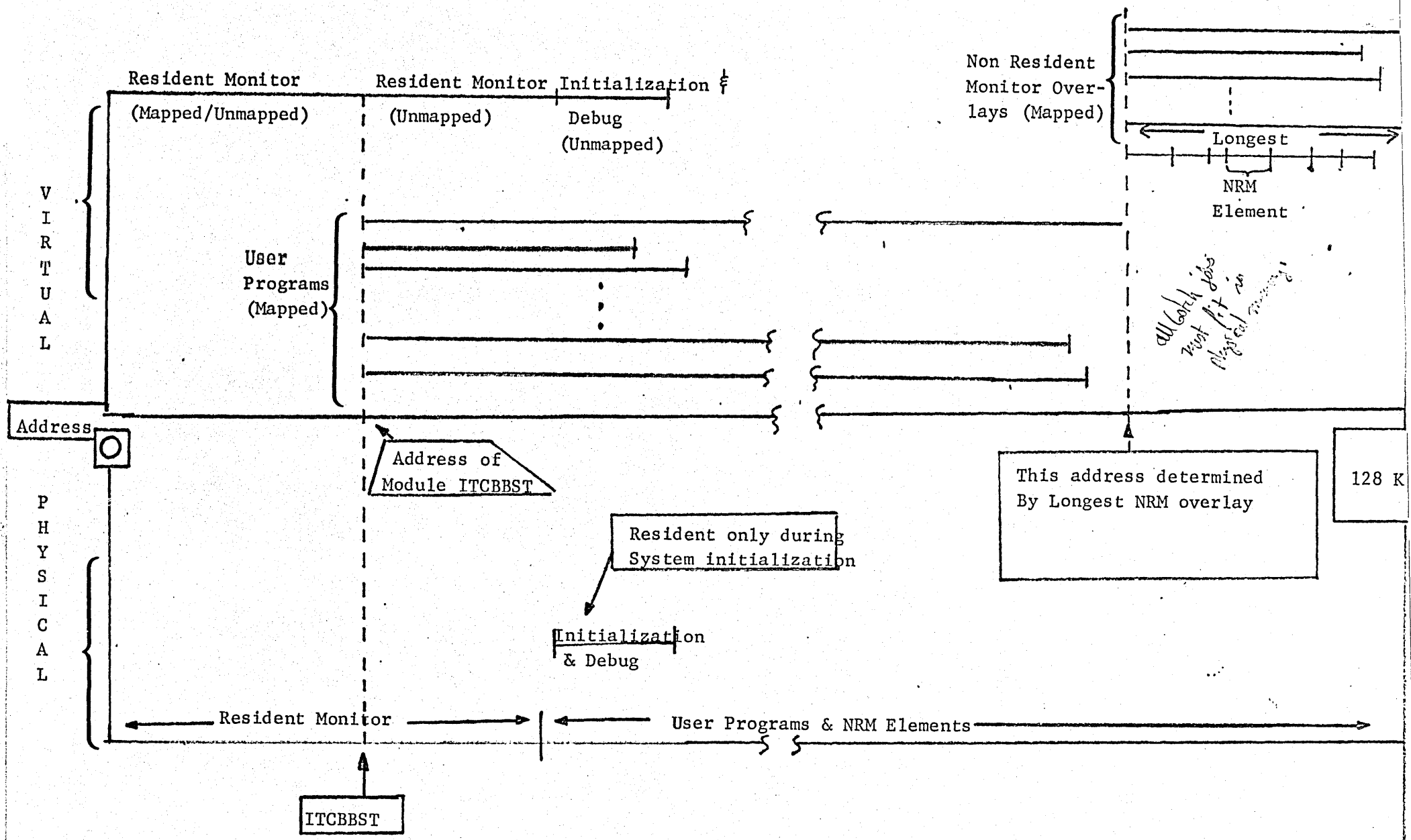


Figure 1 (XOS Virtual & Physical Memory)

1.2 Non-Resident Monitor continued.....

NRM segments will normally begin virtually at 116K and extend to 128K, XOS needs only to reserve physical memory equal to the largest element (1.5K in XOS-A01; 1K in XOS-B00) in order to insure that physical memory will always be available for loading the NRM.

1.3 User Space - Figure 2 depicts the portrait of an XOS user. A user is located virtually beginning at the monitor's ITCB address. Approximately 105K of virtual memory is available to user programs (ie., any user may execute a program whose size is 105K assuming physical memory is available). This may be increased at system generation by shortening the longest NRM overlay thereby giving the users more virtual memory. A user's ITCB page is his context page; among other things it contains the user's memory map image and access control image. A user is allocated core based upon the value stated on a control command. To this value the system adds 2 pages: an ITCB page and a common page (see Figure 2) which is reserved as a work page for monitor services. The user's program is loaded into core starting at the virtual page address above the ITCB page. Pages not loaded with the user's program, but logically allocated to the job, are available to the user via the memory management user services (see Section 2.1 below). Also, until these pages are physically allocated to the job, they are available for loading NRM elements.

In XOS, all monitor service work space (such as access method blocking buffers, I/O tables, etc.) is obtained from the user's virtual memory with read-only access. For instance, when a user opens a file using an assisted access method, the system will obtain the space for the blocking buffers from the user's Common area (based upon the DCB parameters at OPEN-time: ie., number of buffers, size of buffers). This eliminates the need for the monitor to maintain its own pool of blocking buffers for allocation to users.

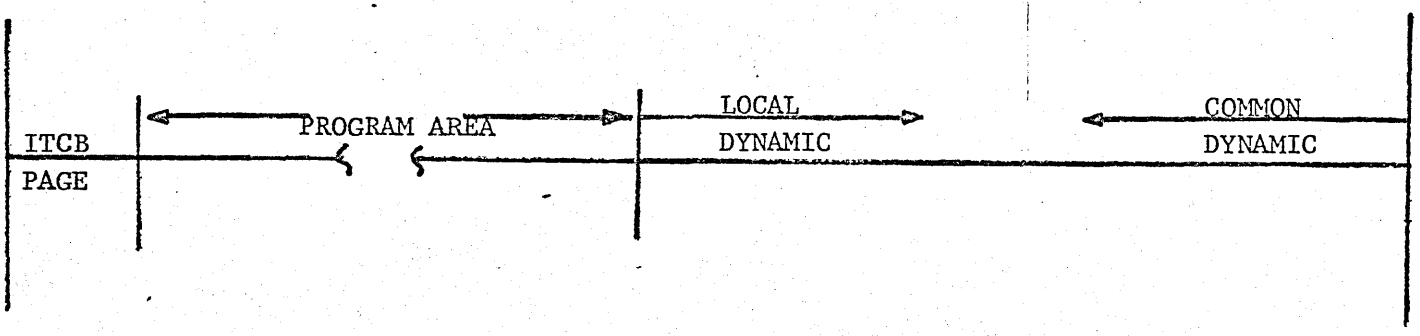
2.0 Memory Allocation

2.1 User Services

2.1.1 Memory Management - XOS provides the user with the capability of managing the core memory that has been logically allocated to the task. The user may:

- Get Limits of Dynamic Space: Obtain the number of contiguous unallocated virtual pages between the highest address in the Local Dynamic and the lowest address in the Common Dynamic areas.

FIGURE 2. (XOS VIRTUAL USER PROFILE)



2.1.1 Memory Management continued . . . .

- Get/Free Pages in the Local Dynamic Area: request (or free) virtual pages in the Local Dynamic area (ie., allocate/deallocate a physical page to a virtual page in the users memory map).

- Get/Free Space in the Common Dynamic Area: obtain (or free) a block of "all-access" memory from the Common Dynamic Area. As blocks are freed they are chained into either of two chains (pointed to by two entries in the ITCB) depending upon access type ("all access" or "read-only access") for subsequent re-allocation to the user. As an entire page is freed it is returned back to the system.

- Dynamically create a DCB at Execution Time: In order to avoid having to build and allocate all DCB's at assembly time, 1 skeletal DCB may be coded into a user program or compiler, filled in at execution-time and moved to read-only Common Dynamic storage as needed.

2.1.2 Program Overlays - User programs may have an overlay structure. Overlay segments will be loaded into memory either automatically by one segment's referencing another segment (REFERENCE loading) or under direct program control whereby a user explicitly loads an overlay segment via a monitor service (SEGMENT loading).

2.1.3 Program Loading - User programs may completely overlay themselves with another program by calling on either of two monitor services:

- LINK - allows an executing program to dynamically request the loading into memory of, and transferring control to, another program while preserving the state of the calling program for a later return. Common dynamic storage is unmodified thereby permitting program-to-program communication.

- LDTRC - "load and transfer control" functions exactly like LINK (above) except that the calling program is not preserved.

2.2 Swapping

The basic XOS Batch Multiprogramming System does not require a swapping mechanism in order to support its multiprogramming operations. The core memory resources are allocated to as many tasks as possible. These tasks are then maintained in the same physical core and executed on a prioritized demand basis based upon the SIGMA Priority External Interrupt System. Swapping is a logical extension of the basic multiprogramming system used to support timesharing user tasks. Swapping occurs between central memory and a standard file which may be located on any system RAD or disk pack. The installation allocates this file via a System Utility Program which uses normal file management facilities. When the Timesharing Task is not active, this file space may be deleted and its space allocated to user or system files. In addition, when the timesharing task is initiated by the operator, this file is dynamically sub-divided into contiguous user sections such that when swapping for a user does occur, the channel program need only contain one SEEK command. The swapper achieves priority over other I/O operations by virtue of its direct interface with the I/O Supervisor and the fact that the Timesharing Standard System Task operates at a hardware priority interrupt level (specified at System Generation) higher than that of batch or timesharing user tasks. The Timesharing Task allocates all timesharing user tables dynamically (in its own Common Dynamic Area) at the time it is activated by the operator, thereby eliminating the need for keeping these tables in core when the Timesharing Task is not active. The swapping of timesharing users allows the simultaneous operation of a large number of users whose combined virtual memories exceed 128K words. The timesharing user's core image is identical to that of a batch user and is managed by the user in the same way.

fixed alloc from? not over cylinder boundaries? Flawed tracks ~~[illegible]~~ size limit

2.3 Shared Processors

In XOS the NRM elements function as a type of shared processor. Any NRM element in core memory may be associated with any number of executing tasks. There will never be more than one copy of an NRM element in core although several users may be associated with (or mapped into) that element thereby conserving physical memory pages.

II. RESOURCE MANAGEMENT

1.0 Resource Types

2.0 Allocation at Job Initialization

3.0 Allocation at Job Step

II. RESOURCE MANAGEMENT

The management of the XOS system resources provides the basis for the scheduling of the various tasks supported by XOS:

- Standard System Tasks such as the Symbionts, Telesymbionts and Timesharing Task
- Foreground User Tasks such as the Control Command Interpreter (CCI) and user real-time programs
- User Tasks such as compilers, user programs and utilities

1.0 Resource Types - The Scheduler, Job Management and Task Management provide a generalized method of resource allocation and control. A resource is defined to be anything that has an associated Resource Control Block (RCB). It may be a table, program data file, peripheral device, or any other entity that requires controlled access. An RCB describes a resource quantitatively (by indicating the maximum number of units of the resource that are simultaneously allocatable and the actual number of units that are currently available). An RCB also indicates the queue of tasks awaiting access to the resource. There are RCB's associated with the following allocatable resources in XOS:

- Sharable files
- Input symbiont entries
- Output cooperative entries
- System disk space for input symbionts
- Global space on system disks
- Core memory
- Temporary disk space
- Pseudo disk volumes (account volumes) on the system disk
- Card Readers
- Card punches
- Line printers
- 1600 bpi tape drives
- 800 bpi tape drives
- 7-track tape drives
- Private disk pack drives
- Telesymbiont transmission lines
- Message mode transmission lines

.../continued

- Character mode transmission lines
- Groups of transmission lines
- Any of the system tables that are dynamically changed (ie., I/O device tables, scheduler tables, etc.)

2.0 Allocation at Job Initiation - A job is normally entered into the system by the input symbionts. There are other methods of entering jobs and/or tasks such as operator keyins, but regardless of the method of introduction, all incoming job or tasks are processed by the scheduler task to allocate the required resources. The symbiont method is presented here because it is the most common and most comprehensive.

A symbiont task will read the job and place it on the symbiont chain for that class (there are seven possible classes with their priorities and limits established at System generation). The Command Card Interpreter (CCI) is invoked and executes as a Foreground (privileged) User Task. CCI performs a pre-process of the job just entered by the symbiont task to check the job control cards (JCL) for errors, build the resource profile, and compare the requested types and amounts of resources for the job against the System generated maximums allowed for that class to see if the job should even be considered for scheduling. If JCL errors are detected or class limits are exceeded then the job is only scanned for notification of further errors and removed from the system by CCI. An errored job is not presented to the scheduler task. CCI places verified jobs on the scheduler chain after appending to the source image a series of tables defining the resources required and defining the actions to be taken by job management and job step management. The same images are not read again by job or job step management; the tables created by CCI are used to manage the job. The scheduler task is then activated to attempt to initiate the job. CCI and the scheduler operate together to maintain the class scheduling priority established at System generation. The scheduling philosophy of the classes is as follows:

Does this limit
a) # of job steps?
b) # of ASSIGNs per job?
c) # of files, tapes, etc.?

Class	Internal Class Priority	Multiprogrammed
P*	0-7 priority for each job	Yes
T	0-15 priority for each job	No (only one of this class active at a time
A-E	First in-First out	No (only one of each of these classes acti at a time)

* The priority order of the classes is specified at System generation, the classes are stated in this order for ease of presentation.

The Scheduler maintains a quantitive table of estimated usage of all types of available resources. There are entries in this table for real core pages, temporary disk space and device types. It is from this table that the Scheduler determines whether a job may be initiated. A job selected by the Scheduler is initiated only if (1) the number of units for each resource required to start the job (as determined by CCI and recorded in the job entry on the scheduling chain) does not exceed the corresponding number of units of that resource assumed to be available and (2) any devices explicitly requested by symbolic address required to start the job are available for allocation. If conditions (1) and (2) are both satisfied, the Scheduler decrements the corresponding entries in the resource availability table. Note that this is used as a mechanism to inhibit initiation of multiple jobs whose resources in combination exceed the total available for allocation; in this way, the Scheduler insures that, should multiple jobs request their maximum number of resources simultaneously such resources will be available.

When the Scheduler determines that the maximum number of units of any resource needed by any one step is available, it initiates the job. The actual allocation of these resources is performed by job step management : core memory is allocated and MOUNT messages are sent to the operator; logical disk space for temporary files was allocated via the Scheduler's decrementing the estimated resource availability table while physical disk space for temporary files is allocated at the time the files are OPEN'ed and as they are being built.

The release of any resource is left to the user through the manner in which he constructs his ASSIGN control commands and the manner in which he closes DCBs. Resources may be released during a job step (by closing a DCB), at the end of a job step (when the system closes any open DCB's), or at the end of a job (when all resources are returned to the system).

3.0 Allocation at Job Step - Normally, the resource profile produced by CCI reflects, for any particular resource, the maximum number of units of that resource needed by any one step. However, by using the RESOURCE and/or the SLIMIT control commands, the user can cause a resource profile to be constructed which allows the job to be initiated with the Scheduler having verified the availability of the resources needed only for the first step. In this event, the job step management routines activate the Scheduler to verify that any resource in excess of those originally verified as available at the time the job was initiated are in fact available for the job step. The event control block (ECB) mechanism is used

...../continued

between the Scheduler and the job step management routines to ensure that the user task associated with the job step must wait until the availability of the supplementary resources is verified. Should a task be required to wait on the availability of some resource, all currently allocated resources are returned to the system for the duration of the wait, thus avoiding the possibility of a deadly embrace situation. The operator is notified when a task is placed in a wait state awaiting a resource and reminded of this fact periodically.

How do I get them back? are my tapes repositioned? Tapes left positioned over job steps?

Permanent allocation of resources to T.S. jobs limits flexibility.

What guarantee is there that a job which has given back resources ever gets to run again? -- He is held out by a stream of jobs using just enough to prevent his execution.

III. COMMUNICATIONS MANAGEMENT

1.0 Introduction

2.0 System Generation Interface

- 2.1 Network Definitions
- 2.2 Line Protocol
- 2.3 Translation Specification
- 2.4 Monitor Structure Definitions for Communications Management

3.0 Communications Management Interface

- 3.1 User Interface
- 3.2 Monitor Interface

4.0 Operating Modes

- 4.1 Message Mode
  - Line Support
  - Terminal Support
  - Remote Batch Interface
- 4.2 Character Mode
  - Line Support
  - Terminal Support
  - Timesharing Interface

III. COMMUNICATIONS MANAGEMENT

1. INTRODUCTION

Communications Management is an optional modular component of XOS; it includes the Telecommunications Access Method (TAM), supporting monitor level functions and parametric System Generation user specifications.

Since TAM is a logical interface to communications input/output, the same communications capabilities are extended to all users. The user may be a normal user task or a system task such as telesymbiont control or timesharing control. The Communications Management option is used by XOS itself to provide for communications control for telesymbionts and timesharing.

The supporting monitor level functions consist of such items as resource allocation allowing the specification of groups of lines or linking of lines as allocation units; clock routines allowing for character mode buffer size optimization and line timeout processing for both character mode and message mode and the Input/Output Supervisor allowing message mode post processing to occur at a specific priority interrupt level.

System Generation provides parametric support for Communications Management by allowing the definition of communications networks; allowing information to be supplied regarding line protocol; enabling the specification of translation tables and allowing monitor structure definition and interface for TAM modules.

2. SYSTEM GENERATION INTERFACE

The user may define a collection of lines and stations to be a single logical network. A single user Data Control Block (DCB) may be used to control a network or group of networks. The network is specifiable to the line/terminal/component level. All four combinations of nonswitched switched and bipoint/multipoint lines are supported. The implicit polling/selection order is established by the network definition. However, the polling order may be explicitly stated or dynamically changed by the user.

Supervisory sequence values for line protocol characters may be specified at System generation. Default values are provided on a device type basis. For message mode the presence or absence of longitudinal parity is specified. For character mode the definition of special function characters is allowed and the desired echoplex procedure is established.

The translation tables are defined by the user at System generation. Standard translation tables may be selected and then partially altered or totally changed by the user. The user specifies the translation procedure to be selected for each line specification.

The structure of the XOS Monitor is controlled by the user at System generation. This philosophy is extended to the selection and placement of the Communications Management modules. In addition, the message mode and character mode portions of the Communications Management component are distinct and may be separately selected or rejected for incorporation into the XOS Monitor.

The required residency of the interrupt handlers for Communications Management does not decrease the user virtual size. Since the interrupt handlers operate in real mode they may exist in real memory that coincides with user virtual memory. The connection between the interrupt locations for character mode input/output interrupts and the interrupt location for message mode post processing priority is parametrically stated at system generation.

3. COMMUNICATIONS MANAGEMENT INTERFACE

Communications Management services via TAM are requested by the user by standard XOS system procedures in a Metasymbol program, the same as any other access method. The XOS system procedures that pertain to TAM are listed at the end of this section. The network or networks are allocated by user supplied assignments to a line or group of lines. Line groups may be defined at System generation or by user assignment. The network(s) polling list may be open-ended (linear) or wrapped (circular) and it is attached to one DCB by the open procedure. TAM will manage the allocated lines in a "multiplexing" fashion that is transparent to the user.

FORTRAN M COBOL?

Communications Management module interface with the XOS Monitor conforms to the standard Non-Resident Monitor procedures. The priority interrupt level of message mode post processing is established at System generation.

TAM System Procedures

- M:DCB — Enables user at assembly time to introduce any or all of the DCB parameters applicable to TAM
- M:MOVEDCB — Allows dynamic creation of a DCB in the common area by replication of an existing DCB.
- M:SETDCB — Allows modification of DCB parameters during program execution
- M:OPEN — Establishes the connection between the program DCB and the network by
  - verification of the explicit user defined lists and the lines assigned as resources
  - initialization of the network; initialization of the transmission device controllers and the line adapters (character mode)

- verification of the operational status of the intermediate telecommunications equipment
- creation in the user program of the list of components or terminals if an implicit list is required
- creation of the required communications tables between the access method and the I/O supervisor

- M:CLOSE — Closes the DCB and, optionally, the network. The close may be either temporary or definite.
- M:LIST — Requests at assembly time an explicit component or terminal polling or selection list. Lists may be linear or circular.
- M:MDFLST — Requests at execution time modification of a component or terminal list.
- M:WRITE, M:READ — Requests a transmission of data to or from a terminal, respectively. A user may also read in survey mode to detect any attention characters a terminal may have sent.
- M:CHECK — Requests a test for successful completion of a specific I/O operation
- M:DEVICE — Enables the users to specify a transmission code change or to perform a device specific operation such as:

BEL	send an alarm to a component
SUS	suspend transmission from a component
ABO	abort transmission from a component
IND	identify by index into a list the component on which the operation is to be performed
MOD	redefine working mode to EBCDIC or binary

4. OPERATING MODES

The Telecommunications Access Method (TAM) provides support for two operating modes:

- Message Mode
- Character Mode

Message mode operation is characterized by block transmission and the use of Input/Output Processors (IOP's). Character mode uses character transmission and the Direct Input/Output (DIO) interface.

4.1 Message Mode

The message mode portion of TAM (TMM) is designed to provide access to terminals requiring high transfer rates, block transmission or multidrop connection.

TMM provides "framing" of user output messages by attaching control sequences peculiar to the terminal type. On input, all such control sequences are removed and the user receives only the data portion of a message.

But only to the extent of change in binary value of frame characters. Different protocols are not supported eg. P3 Bisync, etc

Message mode operation is available to any user program observing TMM protocol. Message mode is also used by the Telesymbionts for submission of jobs from remote batch stations and reporting of output to those stations.

Remote batch operation is supported for the 7670 Remote Batch Terminal.

4.2 Character Mode

TAM Character Mode (TCM) provides access to contention terminals operating on the Character-Oriented Communications controller. TCM supports terminals operating with ASCII transmission code such as teletype-compatible visual display terminals. TCM also provides support for 2741-like terminals using Correspondence or Extended Binary Coded Decimal (EBCD) transmission code.

APL Keyboards ?

TCM provides a number of functions to facilitate program-terminal communication, including:

- Normal and abnormal input termination..reporting of end-of-text, tabulation, attention and BREAK conditions via anomaly returns to the user program
- Input editing functions: character erasure and line cancellation
- Output formatting: TCM attaches carriage control sequences, upon demand of the user program, to user output messages.
- Paper tape control: TCM starts and stops the terminal paper tape reader and punch, upon demand, to facilitate paper tape I/O. Implies Full duplex only.
- Upper/lower case input (2741) with the ability to force all lower case input letters to upper case
- Type-ahead processing (teletype) on a line-by-line basis as selected at SYSGEN. Type-ahead buffers are not part of TCM proper but are allocated from user space when a type-ahead line is opened.

per buffer line buffer allocation !
only one line?

Type-ahead processing and 2741 support are provided in a modular fashion. Type-ahead processing is centralized in one system module, 2741 support in another. A module need not be included in the system if the corresponding function is not desired.

TCM is used by the timesharing subsystem to provide the communication link with user terminals.

Timesharing user programs use the Timesharing Access Method (TSAM) which provides an ASAM-compatible terminal interface - TSAM, in turn, communicates the user requests to the timesharing manager task which communicates directly with TCM.