MEMORY MANAGEMENT, RESOURCE ALLOCATION, AND COMMUNICATIONS IN UTS

UTS Section
Current Systems Department
12 September 1972

I. Memory Management

1. Memory Layout

1.1 Resident Monitor

- Characteristics
- Size
- Location

1.2 Non-Resident Monitor

- Characteristics
- Size
- Location

1.3 User Space

- Context
- DCBs
- Blocking Buffers
- Program Area
- Common Area
- Dynamic Area

2. Memory Allocation

2.1 User Services

- Memory Management
- Overlayed Programs
- Load-and-Link

2.2 Swapping

- Characteristics
- Allocation
- I/O Supervisor interface

2.3 Shared Processors

II. Resource Allocation

1. Resource Types

. Core

. Disk

. Peripherals

. Temporary Disk

2. Allocation for Job Initralization

3. Allocation at Job Step

III. Communications

1. Introduction

2. SYSGEN options

. Terminal types

. Networks of lines

. Line Protocals

. Translation

. Inclusion of communication routines

3. Communications Interfaces

. User with devices

. Monitor with communication routines

4. Operating Modes

4.1 Message Mode

. Line type support

. Terminal type support

. Remote Batch Terminal Support

4.2 Character Mode

. Line type support

. Terminal type support

. Timesharing Support

I. Memory Management

1. Memory Layout

The following paragraphs describe the layouts of the UTS monitor and user spaces in memory. The resident monitor is that portion of the operating system which is always in core memory. The non-resident monitor includes all functions that are placed in memory only when they are required. There are three mechanisms for placing non-resident functions in memory:

Physical overlay - placing programs in memory by physical address

Virtual overlay - placing a requested monitor service program in a user's virtual memory

Ghost job - an asynchronous monitor task performed in its own virtual space as if it were a user program

In addition, many system functions are carried out by slave mode user-style processors which operate in user space and conform to all user limitations and restrictions (e.g., no master mode instructions). Finally, the general layout of virtual memory space is described from the user's point of view.

All of the programs required by each user, i.e., the resident monitor, non-resident monitor, user program, and shared processors, are combined into a single 128K virtual memory space by means of the map. Five kinds of programs can be "shared", i.e., one copy can be used by more than one user program. The five kinds of shared programs are:

A monitor virtual overlay

A command processor

A shared processor

An overlay of the shared processor

A shared library or debugger

Use of the map to construct a virtual memory layout achieves several advantages:

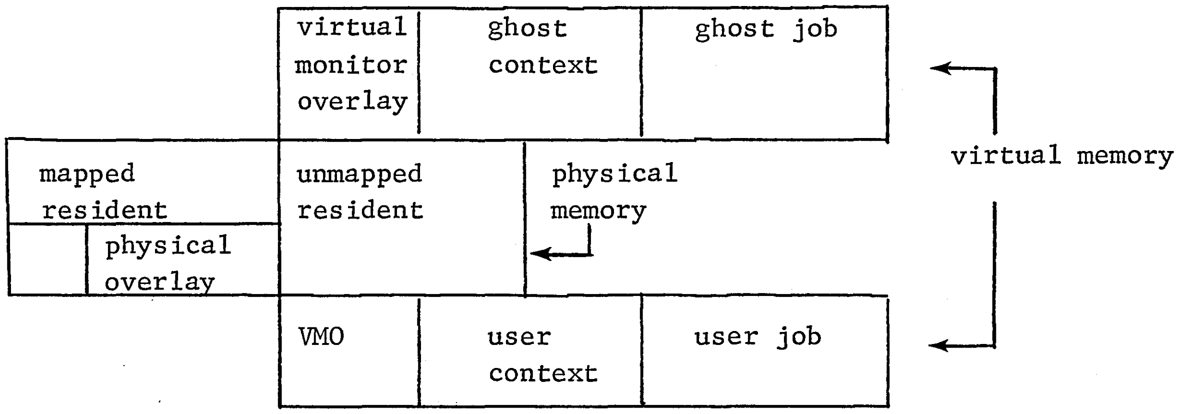
Any available physical memory pages may be used (to 512K on Sigma 9)

The pages need not be contiguous (avoids shuffling)

No program relocation is required

Processors and libraries may be shared among all current users

In conjunction with virtual memory management, the UTS swapper allows the total size of all running programs, both batch and on-line, to exceed the size of real memory.



1.1 Resident Monitor

The core-resident portions of the UTS monitor occupy low-addressed physical memory. These routines are loaded together, enter memory at system boot time, and are replaced only during recovery. Routines in this region run unmapped when servicing interrupts or mapped when carrying out the explicit requests of the user (his CAL). Some routines, such as I/O buffer allocation routines, must be executable both mapped and unmapped; therefore, much of the resident monitor is mapped "one-to-one" so that virtual and physical addresses are identical in this area. Some interrupt handlers and other routines which operate entirely unmapped appear in physical memory but not in virtual space, making more virtual space available to the user.

The routines chosen for residence in UTS are those modules which have a high incidence of use. From time to time, measurements of the running system indicate low-usage modules that can be moved to non-resident status. This technique has gradually reduced the physical resident size over the last few releases, providing space for larger user programs and added monitor facilities, e.g., RMA features.

The modules which are currently resident in UTS are as follows:

a. Basic I/O enqueueing and device-handling routines

b. Communications management routines

c. Symbiont/cooperative (peripheral I/O buffering) routines (partial)

d. Scheduling (tasks) and swapping routines

e. Memory management routines

f. Job step control routines

g. File management routines (high use portions)

h. Accounting and performance monitoring routines (data-gathering parts)

i. Error logging and diagnostic program interface routines

j. Machine fault handling routines

The total size of the resident monitor varies at SYSGEN with the options selected: number of users, number and type of devices attached, number of buffers for I/O, and routine selection such as performance monitoring and certain terminal communication routines. A large system (say 100 users and many devices) would have a resident of about 31,000 words, a medium time-sharing system about 26,000 words, and a minimum system requirement of 20-22,000 words. Partially explored techniques show promise of reducing this to 16-18,000 words.

1.2 Non-Resident Monitor

Three methods are used for non-resident UTS monitor routines:

Virtual memory overlays for monitor services requested by a user program

Ghost jobs for job independent "parallel" monitor tasks allocated their own virtual memory

Physical overlays for initialization, master mode debugging and automatic recovery from system failures

a. Virtual Monitor Overlay

These groups of routines perform specific user-requested services. They are one of the several kinds of shared processors in UTS and are handled by the common shared processor mechanism. See Section 2.2 and 2.3. They are "linked" into the users virtual memory by means of the memory map; one copy of each routine is "shared" by all users requiring the routine. This is done by "mapping" the physical routine into each user's virtual memory when the routine is needed. More than one overlay may be physically resident in memory, if appropriate. Physical memory not currently in use by users is used by monitor overlays. When overlays are physically resident, a call requires only a map change, no swap I/O.

Monitor overlays are a maximum of 3,000 words in length although the average is near 1500 words. They are located at virtual memory addresses 32-35K, although they may be located anywhere in physical memory. Eight current overlay segments cover functions as follows:

1.	DEBUG	Services user batch debug requests.	1300
2.	OPEN	Services user file open requests.	3000
3.	CLS	Services user file close requests.	1500
4.	MUL	Builds tree'd indices for keyed (ISAM) files.	1000
5.	LTAPE	Services user I/O to monitor-formatted tape.	1000
6.	KEYN	Services operator console requests.	1500
7.	LDLNK	Services user inter program transfer requests.	800
8.	MISOV	Services infrequently used calls.	2500

Linking of an overlay into a user's virtual memory is controlled by a centralized routine, T:OV, which provides for two types of entries to overlays: a "branch" type and a "branch-and-link" type. The latter remembers the last overlay in use and therefore provides for transfer from overlay to overlay and controlled return. As with other shared processors, the monitor keeps a use count with each overlay which allows it to use memory occupied by overlays for other purposes when they are not currently in use.

b. Ghost Jobs

Ghost jobs are a general mechanism for providing non-resident programs and data areas. A ghost job may be a monitor subroutine that is disguised to look like a user job; it has its own Job Information Table and its own 128K virtual memory space. A "ghost job" is a way of tapping the unlimited virtual memory available to the UTS system. In effect, anytime the system needs more space, it can initiate a "user" in a new 128K virtual memory space. The swapping I/O mechanism brings the ghost job into memory to perform the needed task - usually a short infrequently needed system service. A ghost job can be run at a privilege level which permits it to use data and routines in the resident monitor.

Six monitor tasks are currently handled by ghost jobs in UTS:

1. Initialization. The program GHOST1 reads the system boot tape, modifies (patches) system files, places them in the system account :SYS, and establishes the shared processors in absolute form on the swapping device. If the initialization is the result of a recovery, then proper closing of user files and crash dump analysis is provided.

2. Secondary File Storage Allocation. The program ALLOCAT carries allocation tables for public file and symbiont (peripheral buffering) storage. It is activated periodically to refresh small resident tables of available file granules which are used between activations of ALLOCAT. This technique removes the allocation tables from the resident, an especially effective technique for systems with a large number of disk packs, and provides for their safety (on the swapping device) in the event of loss of core memory contents. The frequency of use of ALLOCAT can be adjusted for optimum efficiency by changing the resident table sizes; a typical activation frequency is 1-2 minutes.

3. Multibatch scheduling and remote batch processing. The routines and tables associated with these functions, contained in the ghost RBBAT, provide for the prescan of the batch job stream to acquire resource requirements and for matching of these resources to the batch partition parameters for job scheduling as described in Section II. The input job queues are protected against memory failure, since copies of the queues are stored on the swapping device.

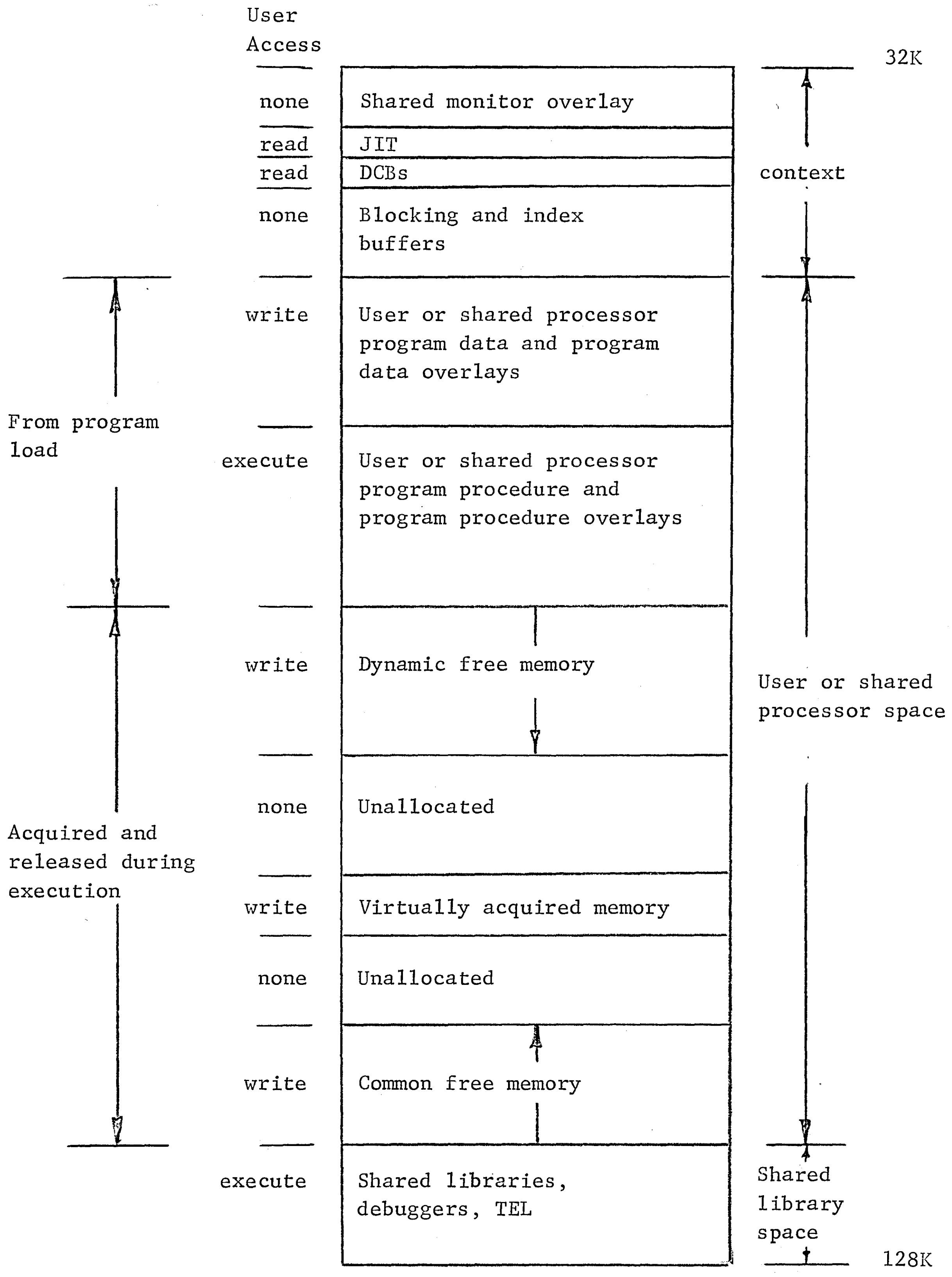
4-6. ERRFIL, FILL, ANLZ. These three ghosts handle production of the hardware error log, automatic file backup services, and crash dump analysis.

c. Physical Overlays

Routines required only at initialization or recovery are in this class. Initialization routines which process the beginning of the system tape and provide for its modification (patching) are resident only until the process is complete. The memory used is recovered as soon as initialization is complete. On crash, special recovery routines physically overlay the monitor to take a dump and save critical tables for entry to system re-initialization.

1.3 User Space

The virtual memory of the machine as seen by a user (both on-line and batch) or by a shared processor is shown in the figure on the next page. The arrangement provides for the several types of shared processors. Access codes of the memory map coupled with writelocks protect the users from one another, the system from the users, the system from itself when operating in behalf of the user, and the user from errors in this own code which otherwise might step on this own code, DCBs, buffers, or library routines. Total available virtual memory is 96K words: 16K in context (DCBs, buffers, and job information), 64K program and data space, and 16K shared library space. Buffers for I/O are pooled in the context area. This generally means that a smaller amount of core is required for a job when compared to a fixed number of buffers for each file opened. A particularly important and unique feature of UTS is that both batch and on-line users are treated uniformly by the system with respect to memory management. Programs see the same virtual memory regardless of on-line or batch status. As described in the next section, the monitor provides service CALs which enable the user to acquire and release memory during the course of his execution.


User Access
32K
none
Shared monitor overlay
read
JIT
read
DCBs
context
none
Blocking and index buffers
write
User or shared processor program data and program data overlays
From program load
execute
User or shared processor program procedure and program procedure overlays
write
Dynamic free memory
User or shared processor space
none
Unallocated
Acquired and released during execution
write
Virtually acquired memory
none
Unallocated
write
Common free memory
execute
Shared libraries, debuggers, TEL
Shared library space
128K

2.1 User Services

Services of the UTS monitor available to the user program for management of core memory fall into three broad categories:

Explicit commands for acquiring and releasing pages of core memory

Various services for calling in the parts of segmented programs

Services which allow transfer to and return from independent programs

2.1.1 Memory Management

User programs may acquire and release memory during their execution by means of the commands listed below. The memory management routines acquire or release both core pages and corresponding pages on the swapping device. The allowed limit on amount of memory acquired is controlled by system defaults, installation set limits on individual users, and the resource requests of batch jobs.

GP,FP	These two services get and free a given number of pages from the dynamic memory area, always proceeding from lower to higher addressed contiguous pages.
GCP,FCP	These are similar services except that requests are satisfied from high to low addresses in common memory. Common memory remains intact as the communication area for programs called via M:LINK and M:LDTRC.
GL	This service allows the user to determine where available dynamic and common memory is located.
GVP,FVP	These services allow a program to get or free explicitly addressed virtual pages anywhere in unallocated memory. They are especially useful for programs which must build more than two tables of unknown size, since the usual table movement operations are no longer required. For example, APLX uses them in conjunction with trap control to demand page user work spaces.

CVM	This service allows a user to map specific physical pages into his virtual address space. It is useful for examination and change of the resident monitor (for example, performance parameters) and for inter-program communication through dedicated physical memory.

2.1.2 Program Overlays

User programs may be overlayed with multi-level overlay trees in both procedure and data segments. Three methods are available for the user program to manage the calling of his overlays. First he may make explicit calls on the monitor for the desired segment (SEGMENT loading). Second, via a loader option, he may ask that the monitor bring in segments on each branching reference to another segment (BREF loading). And, third, also via a loader option, he may ask that the monitor bring in a segment on _any_ reference made to another segment (REF loading).

The core memory not in use when a large overlay is replaced by a small one is returned to the system for use by monitor overlays or other user jobs.

2.1.3 Program Loading

User programs may completely overlay themselves with another program and communicate with the called program through use of M:LINK and M:LDTRC procedure calls. Communication is accomplished through the general registers and common dynamic storage. An M:LINK saves the image of the calling program on a file for return on exit from the called program while in M:LDTRC simply transfers. This feature is commonly used in calling SORT from COBOL programs and in calling the loader from SYSGEN's PASS3.

2.1.4 Ghost Jobs

Privileged programs may also initiate any processor in the system account as a ghost job through a procedure call within the program. This is used for file backup, crash analysis, and in logging of hardware errors.

2.2 Swapping

In UTS, the swapping mechanism is a general system service. It not only provides for the interchange of on-line users in the traditional time-sharing sense, but also provides for the association and reading in of shared processors, including monitor overlays, and for the system asynchronous parallel tasks, ghost jobs. In a properly tuned batch-only UTS system, the swapper would only be used to bring in shared processors as they are requested. Out swaps of batch jobs would only occur when the need to run a ghost job arises.

In a time-sharing system, particularly with only a few users, all of core might be allocated to batch jobs. In this case, swaps of a batch job would be required to bring in the occasional time-sharing user for execution. During periods when no time-sharing user needs service, the entire memory is devoted to batch jobs.

A "write-ahead" operation allows a user program to be swapped during write I/O except, of course, for the buffers involved in the I/O.

Core management includes the parallel management of swap space. When a core page is requested, a swap page must also be acquired. Similarly, a release of core requires release of swap space. In order to provide for fast swaps, space acquired must be contiguous or nearly so, to that already allocated. Further, the program pure procedure is always placed last on swap devices so that it need not be written out if it is unchanged. The required rearrangement of swap space is carried out by a rearrangement of I/O commands rather than by a physical move. A swap is not forced but eventually occurs in the normal course of events, if required.

A bit table (SGP), is used to keep track of the availability of each storage unit on the swapper. In this table, one bit is used per granule or group of granules to indicate whether it is free or in use. Users are assigned a sufficient number of granules to accommodate their current use. That is, swap space is conserved by dynamic allocation of only the needed amount from a common pool. The assignment is done in such a way that command chaining of the I/O can order the granules to be fetched for a single user with a minimum latency. That is, each user's pages are spread evenly over the set of available granules so that data will be transmitted in every disc sector passed over when the user is swapped.

The records of disc granules associated with each user are kept in the user's Job Information Table (JIT), which is kept on the swapper when the user is not in core. The disc location of the JIT is kept in core for fast access. The swapping device is arranged so that sufficient time is available after the user's JIT arrives from the swapper for the system to set up the I/O command chain contained therein for swapping of the remainder of the user program.

The amount of swap storage assigned to swapping is a parameter of SYSGEN. The number of active (batch and on-line) users that the system can accommodate is limited by the size of swap space allocated by SYSGEN for swapping.

Experience has shown that one 7212 is sufficient for the system residence plus the swapping requirements of 50-60 users. The second 7212 will accommodate 75-90 additional users.

Swap I/O is given preference by carrying it out at the highest I/O priority level. Priority assignments for I/O in UTS are:

1) Swap I/O
2) File Open/Close I/O
3) User I/O
4) Symbiont I/O

2.3 Shared Processors

The procedure portion of most processors, libraries, and monitor overlays is shared among all concurrent users. For example, if four FORTRAN coded programs are running concurrently, only one copy of the library is required in core. This yields a savings of 13.5K words -- enough core to run another job. About 40 processors and overlays are shared in UTS, which results in particularly efficient uses of core memory. In addition to the core saving, a speed improvement is gained since they are fetched for execution directly by the swapping mechanism rather than the slower retrieval via the file system. Using DRSP, these shared processors or overlays may be replaced during system operation with updated versions without disturbing on-going work. Current users of the replaced processors continue to use the old version until they terminate. Installations may take advantage of this core-efficient facility by adding their commonly used processors and libraries to the system as shared processors either during SYSGEN or dynamically, via DRSP.

There are six distinct types of shared processors in UTS:

1) Ordinary shared processors (FORTRAN, BASIC, PCL, APL, METASYMBOL)
2) Overlays of the ordinary shared processors
3) Command processors (TEL,CCI,LOGON,EASY)
4) Shared debuggers (DELTA)
5) Public libraries (FORTRAN run-time library, FDP)
6) Monitor overlays (OPEN, Labeled tape routines, DEBUG)

Ordinary shared processors and their overlays occupy the same virtual memory as user programs. Special shared processors, shared debuggers, and public libraries occupy (and are overlaid in) dedicated high virtual memory and may be associated with user programs or ordinary shared processors. The processors CCI, TEL, and LOGON which require store access to JIT are granted that special privilege.

Shared processors are not limited to programs provided by Xerox. The facilities may be effectively used whenever a program has a high probability of common usage. Service bureaus, for example, may use the mechanism for proprietary packages, and corporate installations may use it for programs with a high frequency of use. Shared processors may be written in any language - FORTRAN, COBOL, Metasymbol, etc.

Command processors may also be installation written to provide for isolated user communities which do not have to know the full system -- transaction processing communities for example.

Any program which meets the restrictions may be established as a shared processor by naming it at SYSGEN, which causes the file copy of the program from the :SYS account to be written on the swapping device and its name placed in shared processor tables in resident monitor core during system initialization. The program is then available through high-speed swapping I/O. DRSP accomplishes a similar task during system operation.

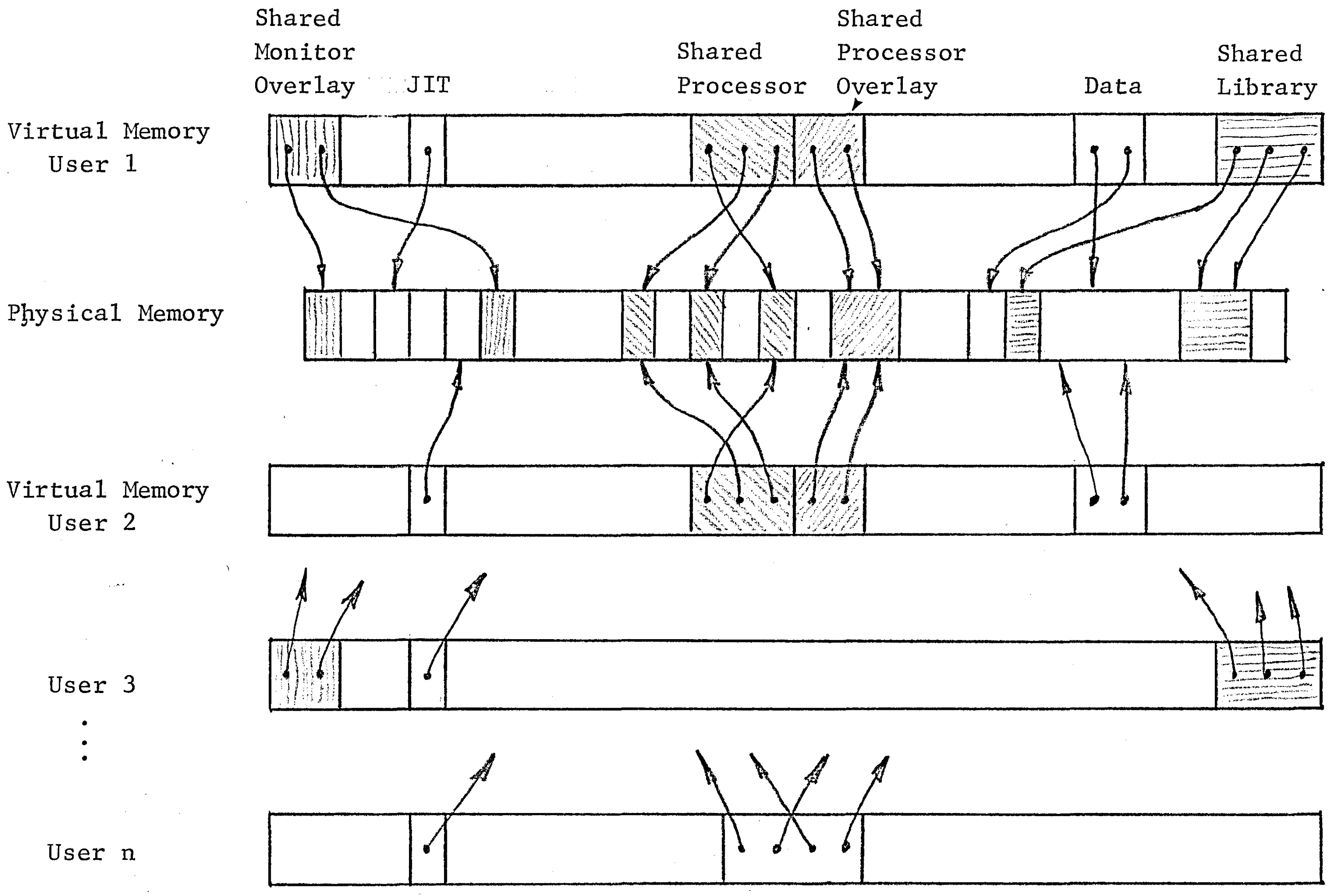
The file copy of the program is retained for recovery purposes and may be run as an unshared program under DELTA for development and debugging purposes. If the load module in the :SYS account is replaced, the shared copy of the program on the swapping device is updated to the newer version in the event of a system recovery.

Shared processors are absolute, live in system resident space on the swapping device, and are swapped in as needed with user or associated via the map if already in. The swap occurs during swap in the first swap phase in which the JIT and all of the required shared processors are brought in. (In the second phase, the user's program and data are brought in.)

The association and loading of shared processors is especially efficient not only because they are absolute program images but also because the resident secondary storage address is directly known and therefore does not require access via the file system.

A single program may have up to four shared processors associated at one time: a monitor overlay, an ordinary shared processor, an overlay of the ordinary shared processor, and a shared library or debugger. The shared library or debugger may be associated or disassociated dynamically using a monitor service CAL.

The following chart shows how the virtual memory of several users of shared processors relates to physical memory.



Relation of Several Users' Virtual Memory

to the Sigma Physical Memory

Virtual Pages not used are set to X'20' with no access; Virtual Pages assigned aoswap image but not yet a core page contain X'22' with no access. These are the page numbers of write protected portions of the resident monitor.

Each user has separate JIT, DCBs, Data, and other memory areas which are private to him and his execution. User 1 and User 2 share a single processor as indicated by the fact that their maps point to the same places. They also share an overlay for the same processor. User 1 and User 3 share a library; User 1 and User 3 share a single monitor overlay. User n has his own private program resident in the same virtual space which Users 1 and 2 are using for a shared processor.

II. Resource Allocation

This section describes the UTS facilities for allocation of system resources--tapes, packs, core, and time to the on-line and batch users of the system. Allocation is done not only for efficiency but also in such a way as to prevent deadly embraces in which two or more jobs make impossible requests (e.g., two jobs each holding one tape asks for one more on a machine with only two tapes).

UTS provides up to 16 concurrent batch processing partitions and guarantees that the total resources, such as tapes, private packs, core, etc., used by active partitions do not exceed the physical resources of the machine (except core and time) and the limits set by the installation manager. UTS classifies jobs dynamically into one or more of the 16 partitions depending on the resources required and the stated resource ranges of the partitions. Each partition, through its resource ranges, represents a job class; however, UTS is unique in that it can and will run--in priority order-- the job class for which resources are available. This feature results in greater than usual throughput when jobs are submitted in an arbitrary manner. Conversely, critical jobs may have their resource requirements and priorities established in such close match to a partition class that immediate execution of critical jobs is assured.

Partition definitions in UTS act as a screening process for job selection and do not define a fixed boundary partition in the traditional sense. Jobs acquire core dynamically as needed up to the specified limit. Any excess core is available for running other jobs. Batch jobs are not core-resident from start to finish but are scheduled and swapped as system load dictates, which assures maximum system utilization consistent with good on-line response time. Resources may be released by a job at each job step, e.g., tapes unused in subsequent steps.

The installation may define a priority increment to be used to increase the priority of bypassed jobs, thus giving them credit for longevity in the queue. Special treatment is accorded an F priority job, i.e., it is guaranteed to be the next job selected, depending only on the availability of the required resources.

The paragraphs above and the remainder of this section discuss resources which are of concern to users of UTS. In addition to these external resources, UTS also uses and manages a number of internal resources. Internal resources are characterized by their period of use which is short enough so that it is practical to wait for availability without noticeably affecting response time. Generally, internal resources are allocated from a pool the size of which can be adjusted via SYSGEN in order to minimize the number and duration of the waits in queue for them. Queueing for these resources is handled as part of the general queueing mechanism which is the UTS execution or task scheduler. Thus, a user job is queued for one of these resources just as it is queued on the more frequent events of terminal I/O, file I/O, computation, etc.

Internal resources managed by UTS are:

- Terminal I/O (COC) core buffers
- Core memory pages
- Swapper pages
- File or symbiont secondary storage granules
- Symbiont secondary storage granules
- Symbiont file table entries
- I/O enqueueing table entries
- Tape and private disk pack mount requests
- Use of the non-reentrant file open and close routines

1. Resource Types

UTS resource allocation and multibatch scheduling are based on specification and control over five resources that are specified as requirements by each entering job on a LIMIT card.

CORE	The amount of core memory required for the job.
TIME	The time required to run the job.
9T	The number of nine track tape drives required.
7T	The number of seven track tape drives required.
SP	The number of private disk pack spindles required.

A later release will add disks A and B and 1600 bpi tapes as distinct resources.

In addition, scheduling is controlled by three other parameters specified on the LIMIT card.

ORDER	All previously entered jobs with this account number must run prior to this job.
MOUNT	Specifies which packs or tapes must be pre-mounted and for each pack whether it is permissable to share its use with other jobs.
ACCOUNT	Specifies that no other job with this account number may run concurrently.

2. Allocation at Job Initiation - Job Selection

In UTS, the selection of a batch job for execution depends on three factors:

1) The resources required by the job as determined by analysis of the JCL (the JOB and LIMIT cards in particular).

2) The installation manager-determined attributes of each partition. These attributes are established during SYSGEN and may be changed dynamically during system operation in order to match changing system loads. They are given in the form of allowed value ranges for each of the five resources listed above: core, time, nine track tapes, seven track tapes, and disk pack spindles.

3) A set of global resource maximums which control the total allowance to batch programs taken together, and to on-line programs taken together (for example, the number of nine track tapes allowed for all on-line users, or the total amount of core allowed all batch jobs). In addition, the total number of simultaneous batch jobs and the total number of on-line jobs may be controlled. These parameters are also determined at SYSGEN time but may be modified dynamically by the installation manager using CONTROL, a system program provided for this purpose.

By setting the partition attributes, the installation manager defines job "classes". For example, he may have one partition-class for tape jobs, three for small short jobs requiring no tapes or packs, and one for large jobs. It is important to note that the installation manager controls the efficiency of the system by dynamic setting of the partition attributes and the global maximums. Users do not classify their jobs themselves other than by describing resources. The internal ability of the system to run any job that fits available resources without an artificial class constraint improves total throughput in a changing demand environment.

All jobs entering the system are assigned to a partition on the basis of the five resource parameters. They are passed through the multi-dimensional sieve formed by the partition parameters and marked according to the partition in which they fit. A job may fit in more than one partition.

The multibatch scheduler is called upon to select a job for execution at each appropriate point: 1) whenever a batch job is completed, 2) whenever a new job enters the system, and 3) whenever the installation manager makes a change in the global limits or the partition parameters. (The latter requires that all waiting jobs be re-fitted to the partitions.)

At these points, the scheduler scans the input job queue in priority order and selects the first job which fits a vacant partition and is consistent with current global limits. This technique is particularly effective in selecting jobs for execution which match the available resources and thus tends to maximize total system throughput.

Special treatment is given to jobs carrying the highest priority (F). This "super" priority allows the installation manager to guarantee that the job carrying it will be run next. The scheduler will not run any job which uses resources required by the super priority job -- resources are held for it until it can run.

3. Allocation at Job Step

Any job in UTS may use system service calls to release to the system resources which he no longer needs. This may be done at any time within a job as well as at the end of a job step.

Core memory is returned to the system free memory pool at the end of each job step and reacquired by the next step as needed up to the limit established by the job. Unused memory may be used by the system for monitor overlays or other jobs.

UTS does not permit jobs to acquire resources beyond the initial request during the job's execution since this might result in a deadly embrace which could only be resolved by aborting one of the jobs.

III. Communications

1. Introduction

In UTS, communications management is a logical part of the system's general I/O supervisory routines. The mechanism is designed to provide for "device independence" -- that is the actual device connected to the user's read and write requests is not determined until execution time when the coupling or assignment is made (in the DCB) of the users request to the device. In all cases, the monitor, through it's I/O supervisor, provides for the actual I/O by building command lists appropriate to the device and enqueueing the requests when channels are busy at request time. Translations of characters, where needed, is also provided so that the user sends and receives messages composed of standard EBCDIC characters.

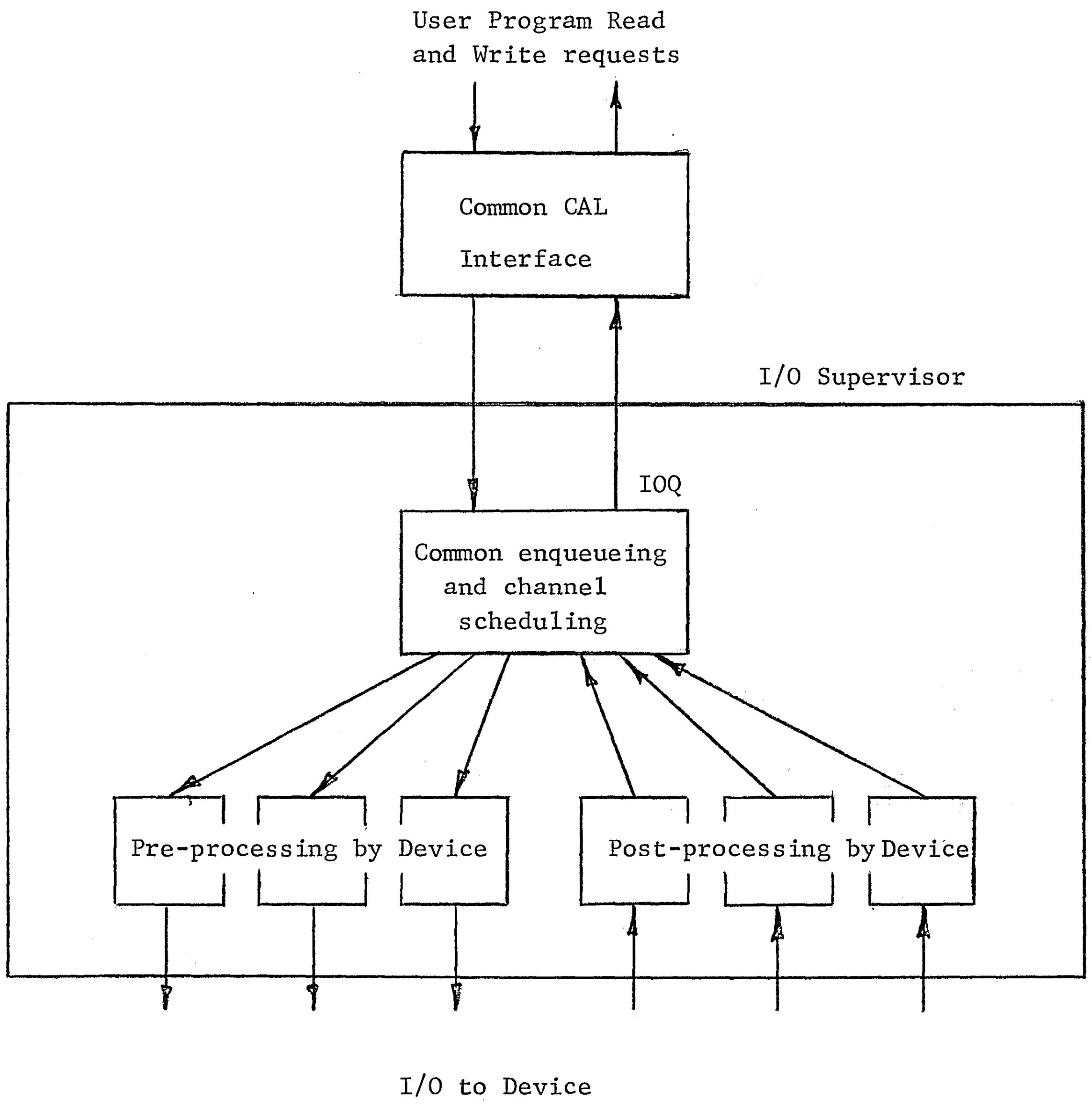
When new devices are added to the system, new "handlers" are required which provide the proper I/O commands and translations for the device. Two sub-parts of each handler are required: a pre-processor to initiate each I/O request, and a post-processor to recognize and recover from errors.

The relationship is outlined in the figure to follow.

System generation facilities allow for the inclusion of handlers for new devices. (Some recently devised methods allow for dynamic addition of handlers.) Devices are grouped according to I/O channels which may be logical as well as correspond to the physical I/O channels. In the special case of character mode terminals operating through the 7611 COC hardware, the handler itself may be modified during SYSGEN to established the characteristics of the individual terminals on 7611 lines. To the system and user, each terminal appears as a separate device.

Most systems, including UTS, treat devices as resources which may be acquired and used by programs. An exception is usually made for the card reader and line printer which are special in that they are "controlling" devices, card for input, and printer for output. A JOB card appearing at the card reader is special; it means "create a job".

UTS is unique among operating systems in that it treats character oriented terminals as "controlling" devices. A job is created when a terminal line calls up. Message mode communication devices are treated as acquirable resources in UTS just as in other systems, not as controlling devices.


User Program Read and Write requests
Common CAL Interface
I/O Supervisor
IOQ
Common enqueueing and channel scheduling
Pre-processing by Device
Post-processing by Device
I/O to Device

The fact that terminals in UTS are logical job stream sources means that programs written for batch mode operation may be run from terminals without change. The system automatically makes the necessary logical association and translation for the device.

Note, also, that because of shared processors, a program may be connected to many terminals at one time without the need to acquire them all and "sub-monitor". UTS automatically provides separate context for each terminal by treating it as a separate job even though the same program is shared by them all.

2. SYSGEN Options

In UTS, SYSGEN facilities which relate to communications are most complete for character mode I/O. The options available cover:

Terminal types - 7012, TTY 33, 35, 37, IBM 2741

Translation tables - ANSCI, APL, Selectric, EBCD, standard

Line protocols - full, half duplex TTY, exchoplex, 2741

I/O address, DIO address for up to eight 7611 units

Number of lines

Amount of buffering

Block and unblock limits on I/O

Inactive terminal timeout values

Hardwire/data set

Because of the hardware, line speeds from 110-1200 baud are supported as well as the formats 8/10, 8/11, and 7/9. Software dynamically supplies on a demand basis, the differing buffering requirements of various line speeds. Assembly time options for the character mode handler allow a reduction in size of 700 locations by eliminating page formatting and performance monitoring routines.

The remote batch message mode communication routines may be added to a UTS system through SYSGEN.

Although not a SYSGEN option, the installation manager may control the program, processor, or command sub-executive which is to be automatically entered following logon.

3. Communications Interfaces

On-line programs communicate with character mode terminals through DCBs assigned to the terminal. The system automatically connects the standard DCBs to the terminal when a job is run on-line so no explicit action is required of the user. I/O is accomplished using normal Read and Write CALs in exactly the same way that the program would do I/O to card reader and printer. Message mode terminals are acquired by batch or on-line jobs through use of normal I/O assignments -- ASSIGN cards if batch or SET commands if on-line. Again, I/O proceeds through standard Reads and Writes directed through the DCB to which the terminal has been assigned.

When the program is established in execution, the assignments are placed in the DCB. These cause the I/O requested by the read or write to be directed to the proper communications management routine (the one indicated by the assignment)--either the general purpose character mode handler or a specific handler for the indicated message mode device.

4. Operating Modes

4.1 Message Mode

As explained above, message mode line and terminal support is achieved through the addition of handlers. The standard UTS system includes 7601/7670 remote batch support as described below. A handler for the BC/100/BC/200 is included in each UTS system at El Segundo although it is not a standard product. This handler provides for the unique protocols of that device including polling for multidrop lines.

Remote Batch Support - Remote batch support under UTS operates via a handler for the 7601/7670, and a set of service routines in the remote batch ghost program, RBBAT. The remote batch facilities also provide support for the DCT200 and compatible terminals such as the MOHAWK.

All system facilities available to local batch jobs are available to remote batch jobs; no modifications need be made to a job to run it from an RBT. Printed output is automatically sent to the submitting terminal when ready without operator intervention.

Terminals can be operated in either the attended or unattended mode, i.e., with or without an operator. In the unattended mode, the terminal is disconnected if transmission errors forbid communication.

Half- and full-duplex switched and unswitched communication lines are supported.

Workstations are authorized by the system manager dynamically with SUPER, which also allows him to define a maximum priority for input jobs and the right to use the system account.

The remote operator may (1) defer output of files and request that output later, (2) send and receive messages to and from the central site operator, (3) delete his input and output files and abort his running jobs, (4) obtain the status of any or all of his files, (5) adjust the parameters (e.g., paper size) of his peripherals.

Either the remote or central operator can switch output files from one workstation to another or from an RBT to the central site.

Central site batch or on-line jobs may direct their output to a remote batch workstation.

The complete format control capability of the remote printer is supported. Form control is supported. The remote operator is informed of forms changes and can control registration.

Input decks too big for the card hopper may be sent in shorter segments.

Output files currently printing can be suspended by the remote operator; they can then be deleted, restarted either at the stopping point or the top of the form, or saved for later output. While an output file is suspended, jobs may be entered into the system.

A simple procedure is provided to recover from all transmission errors, and no output is ever lost even in the case of complete device failure at the remote site.

Full error checking is done on all control commands from the remote site; "message files" inform the remote operator of errors in his jobs and remote control commands.

The message file provides a record of all jobs submitted by the workstation and their status, if requested.

Remote batch may be added to a UTS system simply by defining remote batch devices during SYSGEN.

4.2 Character Mode

Introduction - Terminal I/O COC routines are the read/write buffering and the external interrupt handling routines for I/O directed to user terminals. The read and write routines on the user-interface side translate characters to external form and buffer messages into linked, core-resident blocking buffers. Insertion of page headers, vertical format control (VFC), user headings, tab simulation, and other formatting tasks are performed.

The interrupt routines demultiplex incoming characters by line, translate to internal EBCDIC form, check parity, block messages into buffers, echo characters to the terminal, and test for valid end-of-message characters.

A routine entered periodically as a result of a clock interrupt scans all 7611 lines to detect data set hangup and data set answer to provide automatic logoff and logon, respectively.

The COC routines carry out their functions using information carried in a series of line-associated tables, processing both characters deposited by the 7611 hardware in a 'ring-buffer' and user program output messages to and from a pool of four-word blocking buffers.

The COC routines are resident in the monitor root and consist of four main parts plus common subroutines:

1) Input interrupt handler.
2) Output interrupt handler.
3) Code to process a user's write CALs directed to the terminal.
4) Code to process read CALs directed to terminal.

Support Functions - Terminal communication access routines provide support for a wide range of Teletypes (including all ASCII coded terminals) and 2741-type typewriters (APL and standard correspondence keyboards; EBCD and selectric code sets), and CRT terminals.

The communication routines automatically determine the type of a 2741 terminal when the first character (an asterisk) is typed and associate the correct translation table.

Type-ahead (input message queueing) is provided and allows a user to continue inputting commands without having to wait for the next computer prompt. Commands typed-ahead are synchronized with program output to assure a clear and meaningful output identical to non-type-ahead in form.

Character I/O employs buffers from a pool common to all users resulting in extremely low core storage requirements (four words/terminal). Input and output messages are queued independent of the user program -- the user need not be in core during his character I/O.

Automatic page headings are provided, with formatting control over both page width and length.

Simple line-editing commands for erasing characters and lines are available with the handling routines and thus are available to all programs and processors. For 2741 terminals, a backspace overstrike line editing mode is provided.

Both half- and full-duplex paper tape input and output are supported from on-line terminals. Full duplex input operates by starting and stopping the paper tape reader for each input message. Half duplex input, where the start-stop technique is impossible, buffers input ahead in the same manner as type-ahead. In both paper tape modes, rub-outs are treated as no characters. In half duplex,output is suppressed to prevent garbling at the remote printer. Untranslated input and output is provided via a "transparent mode", which allows preparation of paper tapes for milling machine control and control over special devices, such as plotters and vector-generating CRT displays.

Prompt characters, if specified, are automatically supplied for each program read. A unique prompt has been chosed for the terminal executive and for each processor function to insure easy user identification of the requestor.

Tab simulation is provided if desired, with explicit control over the choice of spaces or tabs for delivery to the reading program and the logical beginning point of the carriage.

The reading program may dynamically change the "break set"-- redefining which characters are to be treated as end-of-message characters.

Miscellaneous other controls restrict input to upper case, allow input of lower case characters from Teletypes lacking them, provide for control transfer to the executive, acknowledge the presence of the system, allow re-typing of the current input. All terminal control modes can be established by use of control keystrokes or by the program using CALs.