Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Overview and Index

Technical Manual

90 19 84A

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245
213 679-4511

XEROX

Xerox Universal Time-Sharing System (UTS)

Sigma 6/7/9 Computers

Overview and Index

Technical Manual

First Edition

90 19 84A

September 1973

Price: $6.00

Printed in U.S.A.

NOTICE

This publication provides an overview of UTS and an index to the complete set of UTS technical manuals. The overview reflects the C01 version of UTS. The index reflects the B01 version. However, the index is largely applicable to the C01 version as well.

RELATED PUBLICATIONS

Title	Publication No.
UTS Basic Control and Basic I/O Technical Manual	90 19 85
UTS System and Memory Management Technical Manual	90 19 86
UTS Symbiont and Job Management Technical Manual	90 19 87
UTS Operator Communication and Monitor Services Technical Manual	90 19 88
UTS File Management Technical Manual[t]	90 19 89
UTS Reliability and Maintainability Technical Manual	90 19 90
UTS Interrupt Driven Tasks Technical Manual[t]	90 19 91
UTS Initialization and Recovery Technical Manual	90 19 92
UTS Command Processors Technical Manual	90 19 93
UTS System Processors Technical Manual	90 19 94
UTS Data Bases Technical Manual	90 19 95

[t]Not published as of the publication data given on the title page of this manual. Refer to the PAL Manual for current availability.

CONTENTS

INTRODUCTION

This document is designed to give the technically-oriented reader, who is assumed to have a general knowledge of large computer operating systems, an overview of UTS. It is assumed that the reader is familiar with the use of the system, knowing both the kinds of service which are provided and the language elements which the user uses to request these services. He should come away from the reading with a general knowledge of how UTS accomplishes the various requests made of it. He should also come away with an idea of the parts into which the system is divided, both functionally and physically. Finally, he should be able to understand where to look, both in the technical documentation and in the listing of the code itself, when there is a need for more detailed knowledge.

As can be seen from the table of contents, this overview comprises six major sections:

> The introductory section (of which this paragraph is a part) skims lightly over the system as a whole describing the services it provides, the salient characteristic of its implementation, the operating systems on which it is based, and the hardware which is required for operation.
>
> The second section describes the concepts fundamental to UTS operation. It introduces some of the vocabulary used throughout the technical documentation of the system.
>
> In section three are gathered descriptions of how UTS formats all the storage elements under its control: core memory in both physical and virtual forms, secondary storage used for UTS residence and user swapping space, RAD and disc storage used for files of stored data, and the contents of the source system tape are included.
>
> Section four divides the system into functional groupings and describes the general techniques used to accomplish those functions.
>
> Section five reviews the functional structure of section four giving module-by-module names, sizes, and description of function performed.
>
> Finally, in section six, the processors which, together with the UTS monitor, make up the total system are functionally reviewed.

The UTS Operating System

UTS is a multiple-user Sigma 6/7/9 operating system providing service for a maximum of 50-200 concurrent on-line terminals (a physical limitation of 512 lines is imposed by the hardware; system logic limits the number of concurrent terminals to 250; response and throughput impose a practical limit of 50-200 terminals depending on the load submitted) and full multiprogrammed batch processing services with full resource control. It includes BPM-compatible management of consecutive, key-indexed (ISAM-like), and random (direct) files (on either fixed-head disc (RAD), disk pack, or magnetic tape). These files are use-protected by password and access designation. A symbiont (spooling) system services the low-speed peripherals (card equipment and line printers) asynchronously with other CPU functions to buffer I/O to and from secondary storage.

Central to the operation of the system is the secondary storage, used for monitor and processor residence, symbiont buffers, swapping, and user information files.

Users at the terminals may create, modify, compile, execute, and symbolically debug programs on-line in BASIC, FORTRAN, COBOL, METASYMBOL, and other languages. Through terminal batch entry the user may submit tasks to batch processing, where COBOL, SORT/MERGE, MANAGE, and other processors are available. Any program may be run in either on-line or batch environments. Memory mapping allows reentrant processors (which may be overlaid and may contain initial data areas) to be shared by terminal and batch users. Other shared processors of UTS are EDIT (a context editor), DELTA (a DDT-like machine-language debugger), FDP (a FORTRAN debugging package), a program loader and link-editor, PCL (a device-to-device transmission and conversion program), and both batch and on-line executive-level command processors. The system can easily admit additional shared processors for other languages or for specialized user services added at each installation. Batch jobs may be inserted either at the central site, from remote batch terminals, or from on-line consoles. On-line terminals make use of the output printers and punches via the symbiont mechanism; they may also access tape drives and private disk packs.

Map access controls and write locks secure the system from its users and the users from one another. Through the map the full virtual address range is available for user programs, I/O buffering, shared libraries, and the operating system on machines with less than maximum memory. Multilevel queue scheduling for execution and swapping assures rapid response and overlap of computation with file I/O swapping. The map makes possible multiple user programs and shared processors in core, which contributes to efficient operation through the overlap of CPU execution with I/O. The map obviates the need for core shuffling or compaction.

A comprehensive performance monitoring facility which instruments and displays a wide variety of internal counts and timings allows an installation manager to examine current operation and adjust system performance.

Continuous operation is maintained by automatic error detection, reporting, and recovery. System recovery, which includes automatic failure analysis, maintains integrity of user files while providing automatic restart within one to three minutes.

Printers, punches, card readers, and tapes are maintained with time-shared diagnostics during system operation. System services allow on-line diagnostic programs for maintenance of all peripheral devices concurrent with system operation.

UTS is delivered as a package which includes the following:

1. An operational system tape for a standard configuration.

2. A tape containing compressed decks, symbolic updates, and binary versions of each system module.

3. Tapes containing symbolic, binary, and object modules for the following language processors: BASIC, METASYM OL, FORTRAN IV, SORT/MERGE, the Extended FORTRAN IV Library, ANS COBOL, and 1401 Simulator.

4. A full set of user and operations manuals for the system and language processors.

5. A set of test cases to exercise and verify proper system operation.

6. A delivery document (-11 or -61) describing it all.

Salient Characteristics

Some especially noteworthy characteristics of UTS are the following:

1. Full use of hardware page mapping (equivalent to a relocation register per page) to provide for location of a user's program and data in an arbitrary set of physical core pages (512 words each). This makes it possible for a variable number of different sized program partitions to be concurrently resident in core memory and for the number and size of partitions to vary dynamically from moment-to-moment.

2. Use of the map to share the code portions of reentrant processors among concurrent users with attendant savings in core requirements and associated overhead.

3. Division of all programs into procedure and data areas separately protected with execute-only and read/write access codes. Access codes and write locks are used to protect users from another, to protect the system code from the user, and to prevent the system from writing in its own procedure area.

4. Identical treatment at the execution level of batch and on-line programs, which provides for multiprogramming of batch programs and of batch with on-line, and for file sharing between batch on-line programs.

5. Swapping of user programs as a whole (rather than demand paging) as regulated by the swap scheduling algorithm. Unmodified pure procedure is never swapped out.

6. A multi-level queue scheduling discipline, which provides a common algorithm controlling both execution and swap scheduling and which allows separate scheduling of terminal I/O, file I/O, interactive CPU requests, batch/compute-bound execution, and other special situations. Terminal I/O, for example, has a higher priority than file I/O or compute-bound execution.

7. Full overlapping of user and swap I/O with CPU execution through scheduling, provided that there is enough core in which to do the overlapping.

8. Complete automatic recovery system with primary attention to preservation of user files provides fast restart following hardware malfunction.

9. Ability to create an installation-specific command processor to efficiently pass control to a subsystem and field all exits, errors, etc.

10. On-line diagnostics for card reader, card punch, line printers, tapes, and disk packs.

11. A comprehensive file management system which includes three organizations:

    Random (direct)

    Contiguous pre-allocated set of 512-word granules accessed by relative granule number. Content is managed entirely by the user program.

    Consecutive

    A collection of variable length logical records physically blocked into granules by the system. Access is tape-like: sequential, forward, reverse or spacing. Allocation is dynamically limited only by the size of physical devices on the system.

    Key-indexed (ISAM-like)

    Collection of variable length logical records each of which has an associated key (name). Access is either by key or sequentially or a mixture. A tiered tree index provides for fast access by key to any record. Allocation is dynamically limited only by the size of physical devices on the system.

Lineage

UTS is the latest member of a family of operating systems, or monitors, for the XEROX 6/7/9 line of computers. Because each is built upon its predecessor, each takes advantage of much of the experienced code of the preceding systems. From time to time portions of the monitor are rewritten to add facility, improve performance, enhance maintainability, reduce size, or some combinations of these. When this happens the common line makes it possible to apply the improvement to all monitors in the line. Broad-brush characteristics of each system are given below.

BCM, the Basic Control Monitor, provides device handlers for XEROX peripheral devices and an I/O enqueueing routine which synchronizes requests and provides for error recovery. Two monitor families distinguished by their file management systems, arose from this common ancestor.

RBM, the Real-time Batch Monitor, added simple job scheduling for batch jobs, and a basic file management system as well as real-time services. A new version of the I/O queueing routines and device handlers were added which improved real-time performance. They also replaced their counterparts in BPM, BTM, and UTS.

BPM, the Batch Processing Monitor, is a major full-service operating system for a single stream of batch jobs. Real-time services allow concurrent process control and other high response needs. Symbionts concurrently spool card-to-disk and disk-to-printer or punch. A full file management system is included with access methods for consecutive files, indexed sequential files (called KEYED), and pre-allocated direct files (called RANDOM). A Control Command Interpreter (CCI) processes the job control language to allow the user to call processors for compilation, assembly, loading, and execution, and to assign logical I/O units (DCBs) to physical devices or files on RAD or disk pack.

BTM, the Batch Timesharing Monitor, added to the full BPM batch service a single fixed partition of memory for terminal users. Editing, debugging, and various interactive languages serve the terminal user through a terminal command language. Since BTM does not make use of the memory map, it may be used on Sigma 5, 6, 7, 8, or 9 computers. It is limited by its two partition design.

UTS utilizes the hardware memory map to provide for a variable number of variable-sized memory partitions that do not require relocation after being moved into physical memory. Having several user programs in core increases the probability that the system can find concurrent computing to overlap with swapping and file I/O. The map also makes it possible to share the code portions of processors (e.g., BASIC, FORTRAN) in concurrent use. Because the executing partitions need not be confined to on-line users, UTS contains a basic multiprogramming facility for batch jobs. Up to 16 simultaneous batch streams are multiprogrammed with full control over physical resources, such as tapes, to prevent inter-job lockup. New and improved processors for on-line interactive use are provided in UTS.

Typical Hardware

A typical UTS hardware configuration would include the following:

- Sigma 6/7/9 CPU with 256-page map
- 64K-128K word core memory
- High speed swapping RAD
- File RAD and/or disk pack
- Tape units
- Card punch, card reader, line printer
- Operator's console
- 8 - 512 teletype, typewriter, or CRT terminals

CONCEPTS

Jobs

The UTS scheduling unit is the JOB or USER (see below). As each terminal user calls up or as a batch job is selected for execution, the job becomes active. For each active job, UTS maintains in core records in the form of user-associated tables that allow the job to be scheduled and swapped. Also, associated with each active job is a Job Information Table (JIT) which is the first page of each job - both in core and on the swapping RAD. It contains accounting information, memory map, swap storage addresses, and other information.

There are three kinds of jobs in UTS: BATCH, ON-LINE, and GHOST.

Batch jobs arrive via the input symbiont from a local card reader, a remote card reader, or an on-line terminal. They may be scheduled in the same way as on-line jobs, or in other ways, at the discretion of the system manager. The Control Command Interpreter (CCI) is the shared processor that reads and acts upon the control command stream (!commands) for batch jobs.

On-line jobs are terminal-initiated and generally assume interaction with a user at a keyboard-type device. The Terminal Executive Language (TEL) processor handles control commands for on-line jobs. Additionally, a user may build his own command processor.

Ghost jobs are operator- or program-initiated by naming the program load module to be "forked" to and do not have card or terminal input streams, although they may read command files or take commands from the operator's console. Ghost jobs are used in UTS for the following: initialization, operator key-in commands, file backup, hardware error log processing, certain diagnostics, performance monitoring, secondary storage (file space) granule allocation, multibatch scheduling, and remote batch and input symbiont processing.

User, User Number, User ID

The term USER is often used to describe a UTS job. Users are either terminal users, batch jobs, or ghost jobs. Each user is assigned a unique number at job entry which is carried in his JIT, printed on terminal page headings, and listed with every user-associated message that is typed at the operator's console. The number is also referred to as the user ID (or system ID) and is used by the operator to send messages, to abort or otherwise affect the user's job. A different, but associated value, user number, is used to index scheduler control tables when jobs are active.

Job Step

Each job, whether under terminal control or submitted through the batch stream, is divided into a set of sequential increments called job steps. For example, a FORTRAN compile and execute job divides into three job steps: a compilation, a load operation, and the execution.

Common information carried across all steps is the accounting and limit information carried in JIT (CPU time, elapsed time, pages out, cards in, tapes used, RAD space accumulated, etc.), and DCB assignment information carried in a special RAD record called the ASSIGN/MERGE record. The latter is the accumulation of information from all the ASSIGN cards or SET commands which have occurred previously in the job stream.

At each job step, control returns to the user's associated command processor. For batch jobs, all control cards occur between job steps and are read by CCI. For on-line, TEL reads and acts on all commands issued to it between steps and, in certain cases, during interruptions within job steps.

At the end of each job step, the user's core memory areas are released to the system's common pool, as are the corresponding spaces on the swap device. Thus, only the JIT accounting information, COOP buffers, and the DCB ASSIGN/MERGE records (plus files created by the steps) are carried from step to step.

Virtual Memory

Virtual memory is the logical memory seen by the user or other mapped program running under UTS. Instruction addresses of the program are virtual memory addresses. During program execution a hardware map register relates each virtual memory page (user addresses) to a page in real physical core memory. UTS keeps track of physical memory and assigns it as appropriate to individual users by establishing the contents of the map. Physical pages are associated on user program request either for an explicit page or implicitly when a program is called for and requires memory for residence. Unassigned pages are filled with the physical page address of a write-protected monitor page. This protects the system from erroneous references in master-mapped routines.

The map frees the monitor to choose any physical page to satisfy a request for virtual space at a given location. Thus, programs remain at the same virtual (logical) location and requirement for moving programs in core and relocating them are removed. A program may be placed in <u>any</u> available collection of physical memory pages.

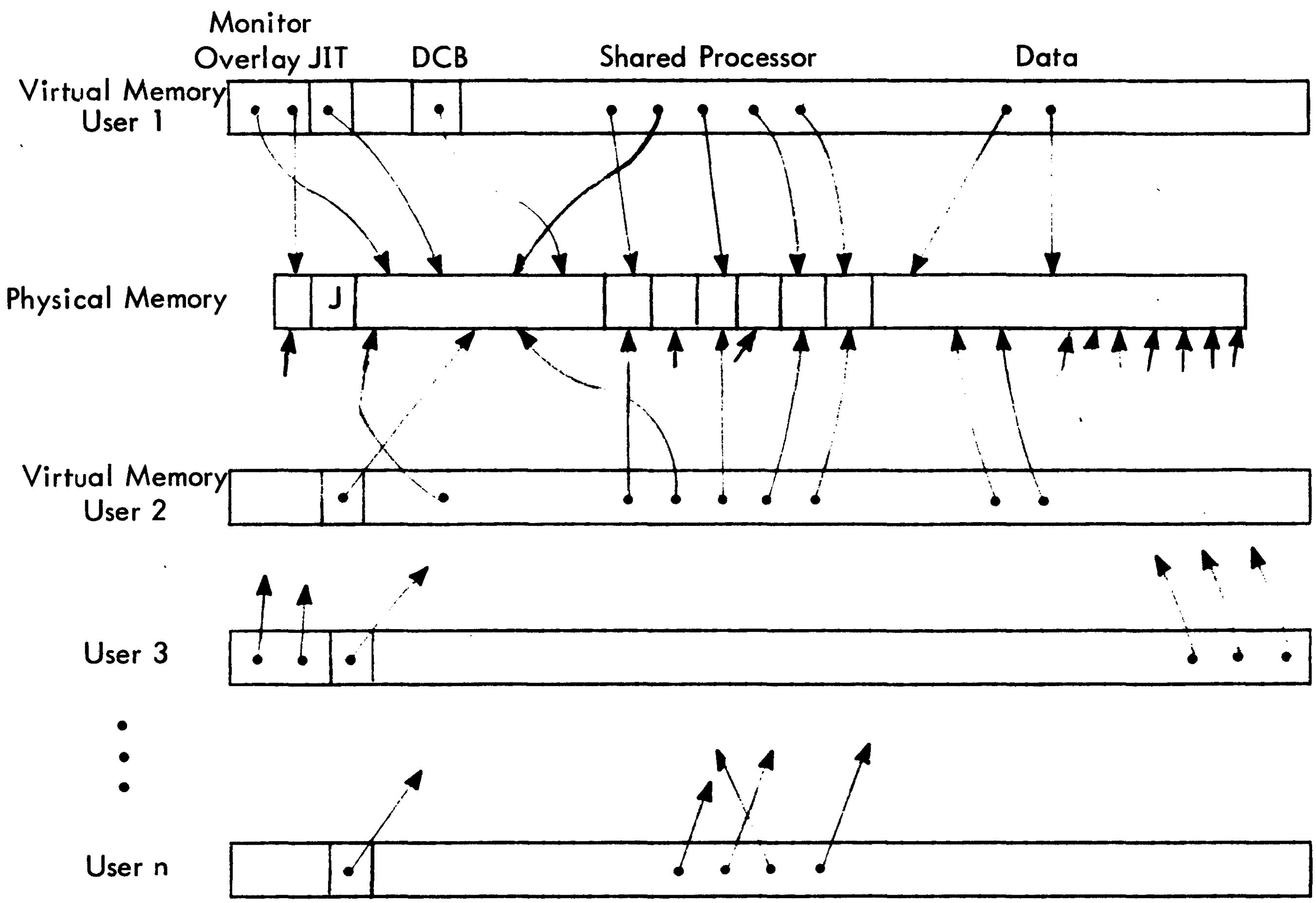
Mapping also permits sharing of the pure program procedure portions of commonly used system processors. (It is also possible to share data areas but this feature is only used for monitor data.) Programs requesting shared processors are connected via the map to a single in-core copy. Separate data areas are provided for each instance of execution of a shared processor. Programs which do not modify themselves may be shared in this map-reentrant way by separating them into data and pure procedure sections.

UTS takes full advantage of the extended memory capabilities offered with the Sigma 9, and may use up to 512K words to hold the monitor and user programs. User program area size may be as large as 64K and additionally have up to 12K of context area.

UTS has over 40 shared processors including ordinary shared processors, their overlays, monitor overlays, shared command processors, a shared debugger, and shared run-time libraries. Shared processors may be added or replaced during system operation by use of the processor DRSP.

Figure BB-1 shows how several users, each with his own virtual memory, might be mapped when they are all in the same physical core.

Figure BB-1 - Relation of Several Users' Virtual Memory to the Sigma Physical Memory



J is the physical JIT for unmapped programs.

Map cells for:

Virtual Pages not used are set to X'20';
Virtual Pages assigned a swap image but not yet a core page contain X'22'.

Each user has a 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 in physical memory. Similarly, 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.

JITs

The Job Information Table (JIT) is the central record keeping place for information related to each job. Accounting information, the memory map image, disc addresses for the job's image on swap image on swap device, the I/O command chain used for swapping, a DCB for terminal use (M:UC) and one for miscellaneous functions (M:XX), control command buffers, and the user-related push stack are some of the important elements stored in this table.

JIT is mapped. The CPU accounting clock ticks subjectively into one user's JIT or another depending on how the map is set. The monitor pushes temporary data into a user-related stack depending on how the map is set. In fact, much of the monitor, the file system for example, need not be and is not aware of which user it is working for, rather it is mapped to the appropriate user via the hardware map.

A master JIT exists in the physical space corresponding to the virtual space where all JITs are located. This JIT is used by all unmapped programs, the symbiont system, and interrupt processing, for example. All CPU accounting for symbiont operation is, therefore, recorded in the master JIT.

Each user is assigned a JIT in order to create the job. Depending on the source of the job, a JIT may be created which is appropriate to 1) an on-line, 2) a batch, or 3) a ghost program. The JIT for KEYIN, the operator's command language, holds in its push stack the entire program for KEYIN operation: a call for the KEYIN overlay and a self-destructive exit.

The JIT disc address is the scheduler's "handle" which allows retrieval of the job when needed from the swap device. This address is kept in a core-resident table along with the job-scheduling information.

Shared Programs

There are six distinct kinds of shared programs in UTS:

1. Ordinary shared processors (FORTRAN, BASIC, PCL, LOAD)

2. Overlays of the ordinary shared processors

3. Special shared processors (TEL, LINK)

4. Shared debuggers (DELTA)

5. Public libraries (FORTRAN run-time library, FDP)

6. Monitor overlays (OPEN, labeled tape routines, KEYIN)

Ordinary shared processors 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.

Although user programs may have large complex tree structures in both data and procedure sections, ordinary shared processors are restricted to a single overlay level in the procedure area only. However, they may have any number of overlays within that level. All changeable data must be in the root segment (unlike the overlays of unshared programs, which may have data in the overlays). Data is initialized at the same time the shared processor is called, and thereafter is associated with each user of that processor and swapped in and out with him.

Shared processors of other than ordinary type may not have overlays.

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.

UTS processors may be shared processors when they are named during SYSGEN and contain shareable pure procedure (reentrant code) or when they are added during system operation using the program DRSP. Data areas of the processor which will be user-associated are initialized at first entry. A shared processor has the following special charastics:

1. Its name is known to the system at SYSGEN time or provided by DRSP and is stored in resident tables.

2. It has dedicated residency on swap storage established at system initialization or by DRSP.

3. A single copy of the pure procedure is shared by all requesting users.

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 RAD 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.

Public Programs

A program whose load module is in the :SYS account is a public program in the sense that it may be called either by a control card containing the ! symbol and the program name or by entry of the program name in response to a TEL prompt (!) for commands. Each user of a public program has his own copy of the program. If a program name refers both to a shared processor and to a load module in :SYS, then the shared copy is used.

Files and Accounts

Upon the basic physical I/O management routines of UTS/BTM/BPM systems is built a file management system which is used not only by the users and processors of the system but also by the system itself. Read, write, open, close, and other command directives of this "file system" are issued by users and processors via CAL instructions. The monitor itself may issue CALs as a user does or may BAL directly to the routines through internal interfaces.

With minor exceptions, all temporary storage needed by the monitor is managed by this file system.

Files may be either consecutive or key-indexed and consist of a variable number of variable length records. Records may be read from key-indexed files by name or in a sequential manner. Unlike the file management of many systems, space is acquired from a general pool and files may expand indefinitely in size restricted only by the physical size of the secondary storage available.

A third type of file, called RANDOM, pre-allocates a fixed amount of space at open time and is read or written addressing by relative granule number. This type of file is not used by the monitor for any of its I/O.

All files are divided into and cataloged by account. Authorization to read or write a file within a given account is granted on an account basis. Each user must establish an account under which he runs at logon time.

Logon account, therefore, establishes control with respect to the file system and should not be confused with accounts established by the installation for fiscal purposes or with the "accounting" records produced at the end of each job to record time, core use, I/O activity, and other resource utilization. Accounting routines which gather this information have nothing to do with file accounts.

Star Files

Processes within the monitor, including the loaders and CCI, which require files of temporary intermediate information place this information in files which are called star files. These files are special with respect to their handling by file management since they are not entered into the file directory, and are special in their naming convention and in handling at job logoff.

The file name of star files is constructed of three characters: the first two are the halfword user ID which is included to assure that the file has a name unique in the system. (Two distinct files will therefore be created and used by a shared processor or monitor component executing concurrently for two different users.) The third character of the file name is assigned to the process using the file. The file named idD for example is a file used by the monitor batch debugging facility to temporarily save MODIFY and SNAP commands. Note that the star file names are often referred to with the ID in lower case and the following character in upper case to indicate that ID is substituted at file creation time.

Star files and their use in UTS are as follows:

idB   Binary file of ROMs from card input formed by CCI (and the tree table) so that the Loader may make its two passes.

idD   Batch debugging commands - MODIFYs, SNAPs, etc.

idL   Load module output file created by LOADER or LINK when a LM file is not explicitly named.

idG   Assembler or compiler output ROM file used when the GO option is specified. The default file assignment of the M:GO DCB.

idR   Assembler ROM output for LINK if no explicit file is given. R is exactly equivalent to B with respect to the file system.

idT   File containing the names of all files which have been marked for release at job end by the M:TFILE operation.

idN   Load and Link files

Libraries

There are three kinds of program libraries provided in UTS:

1. Relocatable Object Module (ROM) libraries (computer or assembler output) which may be private to a user's account or public by placement in the system account.

2. Load Module (LM) librarios (loader output) which may also be either publicly or privately held (these are formed by the Loader in :DIC and :LIB files as described in the UTS System Management Reference Manual).

3. Shared libraries (in absolute form) which are publicly shared by all concurrent users.

Association of libraries with a user program is carried out by one of the loaders, either the one-pass on-line loader, LINK or the two-pass overlay loader, LOAD. LINK does not include LM loading in its capabilities. Both loaders associate programs with the shared libraries either on explicit command or implicitly by knowing that certain unsatisfied references can be found in a particular library (e.g., 9INITIAL is to be found in the FORTRAN run-time shared library).

Shared libraries are created and absolutized at SYSGEN time. They consist of three elments each:

1. The instructions (pure procedure) of the library routines which will be the shared part,

2. An unitialized data area which provides local library context to each user at a fixed virtual address, and

3. A symbol table (REF/DEF stack) which enables the Loader to provide direct linkages to the library from the user program.

Two shared libraries are supplied with each UTS system: a standard set of FORTRAN run-time routines (excepting only complex and hyperbolic functions), and the same standard set, together with the FORTRAN Debug Package (FDP).

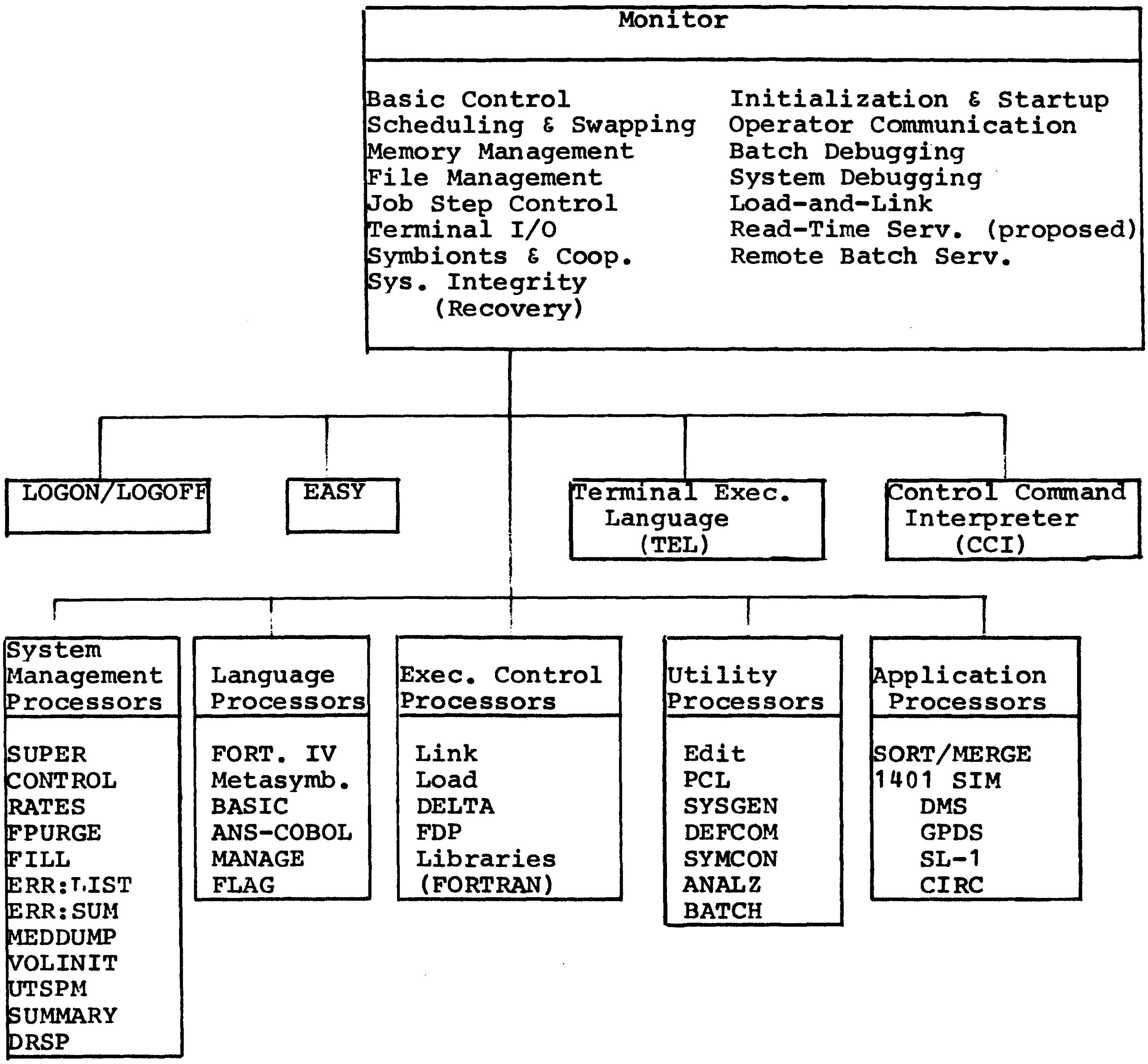
UTS Structures

The UTS Operating System may be divided into the resident monitor with its overlays and the processing programs without which it would be skeletal.

As shown in Figure BB-2, the processors may be thought of on two levels: first, on the executive level, are the command processors. These shared processors, of which TEL, CCI, LOGON, and EASY are examples, pass control to other processors on error command. They are returned to in the case of errors and aborts or exits in the other processors; secondly is a level containing user programs, language processors, utility programs, and management control processors. On this level, any special privileges required are granted to the user job.

The monitor and all processors except the application processors, language processors, FDP, libraries, are termed the control program and are those programs delivered with a UTS release. (As a matter of convenience, the latest versions of the FORTRAN Library and the language processors Meta-Symbol, FORTRAN, and BASIC are included in a UTS release.)

Figure BB-2 - UTS Logical Structure



Dynamic Structure

Another way of viewing UTS is through the dynamics of its operation. Here we see three levels: the slave program level, the monitor service level for carrying out the users' requests, and the scheduling level where the decision for next user is made.

Slave Level

This level includes all programs that run in the MAPPED, SLAVE mode (parts of some specifically privileged programs on this level may run in master mode). Batch and on-line user programs, with their shared public libraries, language processors, such as FORTRAN and COBOL, and the special processors of the system, such as CCI, TEL, LINK, and DELTA, all fall into this category. Programs operating at the slave level are always mapped and are protected from others in core by the access codes and write-locks of the hardware. Monitor services for I/O and other services are provided via CAL instructions which pass control to the monitor service level.

Monitor Service Level

The second logical level of UTS provides for service of CAL instructions, processing of machine traps, I/O interrupts, clock interrupts, and external interrupts. Operation at this level is always in the MASTER mode and may be either mapped or unmapped. Code at this level is largely resident in core memory and is divided into data and pure procedure sections. Write locks are set so that the procedure area can never be written even by the monitor itself. After the called service program is executed, exit is made to the scheduling level.

Scheduling Level

The third logical level of UTS controls scheduling of machine operations by making an appropriate selection for a swap between the swapping device and core memory, followed by selection of the next user for execution. Map, access codes, PSD and general registers are then loaded and control goes to the selected slave program.

This logical organization of UTS is shown in the diagram of Figure BB-3.

Figure BB-3 - UTS Dynamic Organization

	Users	Processors	Processors with Special Status	
Slave Level	On-line jobs Batch jobs Ghost jobs	BASIC FORTRAN EDIT COBOL PCL	CONTROL Diagnostics ERR:LIST ERR:SUM DRSP	TEL,CCI,EASY LINK, LOAD DELTA, FDP, RUNNER LOGON/LOGOFF Libraries, SUPER

Asynchronous Events (unmapped) | User Requests (mapped)

Monitor Service Level

Traps
- Program & machine errors; System recovery

Clock
- Time slice quantum;

I/O
- IOQ
- Processing for error recovery & retries;
- Start more I/O

External
- COC Char. interrupts
- COC Terminal Read & Write

CALS
- COC Terminal Read & Write
- STEP Exit CALs, Aborts, Special Exits (Associate Process, Link, Run, Program, Fetch, DCB MERGE
- Read, Write, Open, Close — IOQ
- Get & Replace Pages; Control Access;

Scheduling Level

Entered with event codes describing activity.
Four general duties:
1) Schedule for swap
2) Schedule for execution
3) Load map and access codes
4) Return to selected slave program

CORE, SWAP RAD, FILE, AND SYSTEM TAPE LAYOUTS

Core Memory

UTS makes full use of Sigma 6/7/9 mapping hardware, access protection, write locks, and Sigma 9 extended memory in allocating available physical core pages to users. Physical core pages are allocated to users at their request. At system boot time the physical size of the actual memory is determined by referencing all memory and linking existing pages into an available pool. Thus, it is possible to remove core from service by turning off the physical boxes so long as the available physical memory is contiguous from address zero.

Use of the map obviates the need for program relocation or physical moves. Full protection is provided, not only of the monitor from the users but also of one user from another, the monitor from itself, and each user from himself. All programs including the monitor itself are divided into procedure and data. The procedure area is protected by write-locks or access codes, or both, against inadvertent stores.

The strategy of write-lock usage to protect master mode programs are as follows:

See the Sigma 7 Reference Manual for a complete description of locks and keys, but remember that a key is associated with each program through the PSD and a lock is attached to each core memory page. Keys and locks control only store accesses. A key of 00 fits any lock; a lock of zero is "unlocked"; otherwise, the key must match to permit a store.

1. A key of 11 is never used nor is a lock of 10.

2. The monitor operates with a key of 01 and thus may store in

   a. its own data area (lock = 01).
   b. any batch, on-line, or shared processor core (lock = 01).
   c. a reserved area for resident real-time data (lock = 00).

   It may not store in

   a. its own procedure (lock = 11).
   b. pure procedure of resident real-time (lock = 11).

3. User programs operate with a key of 00 but in mapped/slaves mode so that protection is provided by the access controls.

4. A key of 10 is reserved for resident real-time. It may store only in its own data area (lock = 00). It may not store anywhere else (lock = 01 and 11).

5. Write-locks are initialized only once (at system startup) and are not changed thereafter except when running under control of EXECUTIVE DELTA where they are used to enable data breakpoints.

A typical layout of physical memory is shown in Figure BC-1.

The access code of each virtual memory page controls references made by slave mode programs (user programs and shared processors). Full access and map images are retained in the JIT of each user and are loaded when the user gains control. TEL, CCI, and LOGON are given special write access to JIT and other job context areas.

In examining the virtual and physical memory layouts to determine the protections, the reader should recall that although the map applies to all addressing operations when the map bit of the PSD is on, address protection depends on the master/slave bit. In slave mode, the access test is made first and then the write-key write-lock test. In master mode, the access test is skipped.

The layout of virtual memory that applies to user programs and ordinary shared processors is shown in Figure BC-2. Virtual core addresses shown are those appropriate for a typical system. More (or less) physical core may be established for the resident monitor at SYSGEN time depending on installation needs, such as the requirement for special device handlers or other options. The bound at which the one-pass Loader (LINK) places the user program is adjustable by assembly parameter in LINK.

Typical contents of the various areas together with number of pages used are as follows:

Context Area	Available Area	Special Area
Job Information Table (1-2)	User programs, data, and symbol tables.	Special shared processor and data:
DCBs (1-n)	Ordinary shared processors including:	LINK
File Buffers (4-n)	Root segment	DELTA
	Initial Data	TEL
COOP Buffers (0-2)	Overlay Area	FDP
Monitor Overlay (1-6)		Public Libraries

Virtual pages which have no physical core page associated and are mapped into a resident monitor page (20) that is write-locked and protected by the no-access (11) code. Thus, slave mode programs are denied access through the access mode, and attempts to store at these virtual addresses by a master mode program are protected by write-locks.

System Residence and Swapping RAD

In UTS, the system resides on the swapping RAD or disk pack. Allocation of components of the operating system on this system device is accomplished at the time the system is booted from a PO tape. The initial portions of the RAD contain enough information to accomplish a complete restart after quiescence or a recovery in event of system failure.

This device is also allocated dynamically to individual user jobs as they are swapped between bursts of activity which require core residence and use of the CPU or an IOP.

System Storage

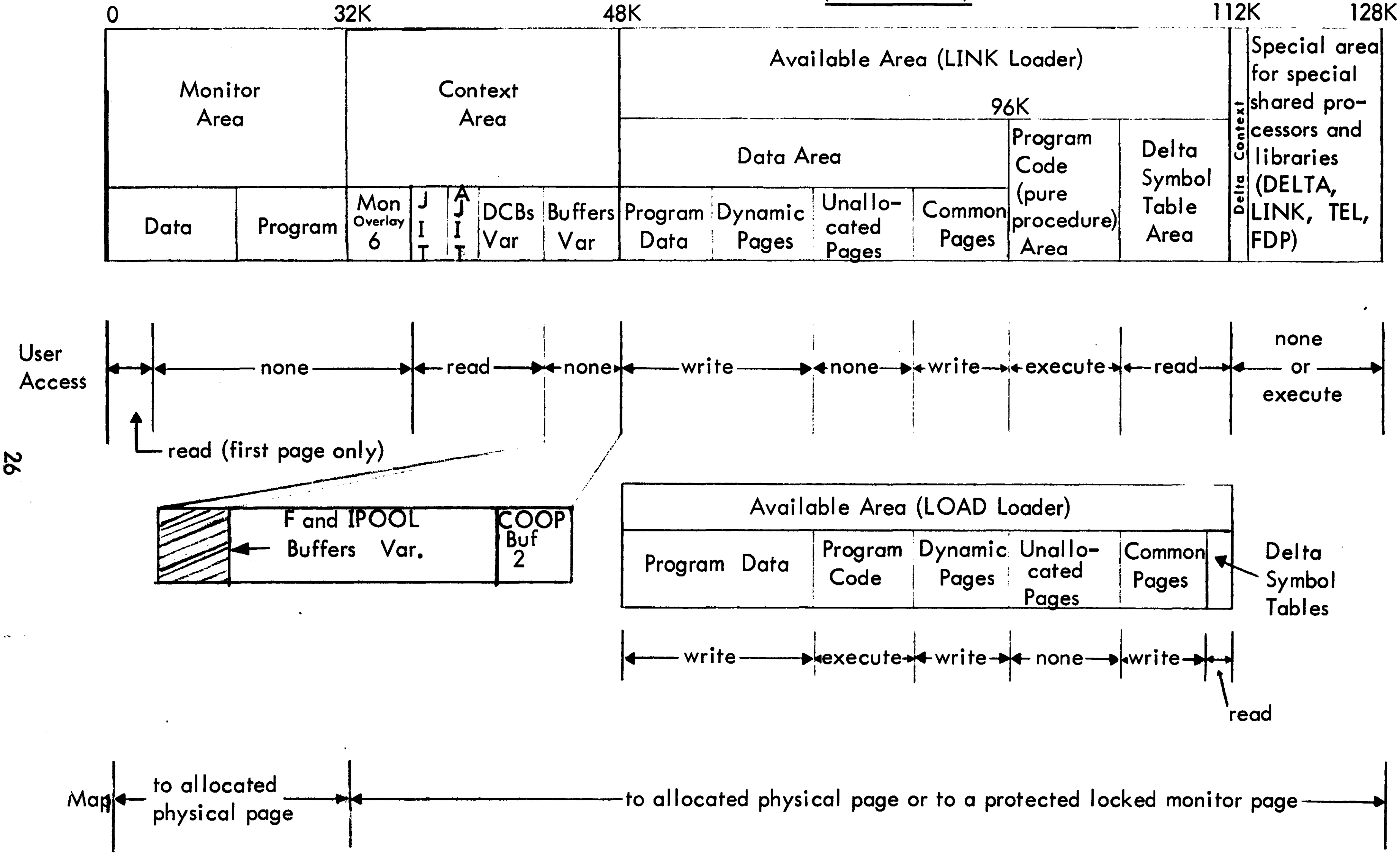
Table BC-1 lists the system components and shared processors appearing on the system/swap device. Two categories are listed: the area provided by the boot-from-tape process, and the area constructed from system files by the initializer GHOST1. This latter area is used by recovery for a core dump area and is reconstructed by the initializer following each recovery. The remaining portion of the system device is dedicated to user-swap space.

	0				end
Contents	Resident Monitor		On-line Jobs, Batch Jobs, Shared Processors, Nonresident Monitor Overlays (Master mode)	Resident real-time programs and data (Proposed)	
	Data	Program		Data	Program
Keys	01		00	10	
Locks	01	11	01	00	11
Mapping	Mapped or Unmapped		Mapped	Unmapped	
Use Mode	Master		Slave	Master or Slave	

Note that the system is protected from users by access codes, not locks and keys.
Note that key = 11 and lock = 10 are never used.

Figure BC-1 - Typical Memory Layout (not to scale)

Figure BC-2 - Typical User - Program Virtual Memory Layout
(Not to Scale)



Access Codes	11	none - no access of any kind permitted	01	execute - execute and read access
	10	read - read access only	00	write - write, execute, and read permitted

Table BC-1 - Contents of System Portion of the Swapping RAD

A. Items Written During System Boot

1. Disc bootstrap routine (Sectors 0-1).

2. Space for ALLOCAT JIT, AJIT (Sectors 2-5).

3. Master JIT (Sectors 6-7).

4. ALLOCAT data, including HGPs, the granule allocation bit maps.

5. ALLOCAT procedure - the granule allocation ghost program.

6. GHOST1, the system initializer.

7. Space for new or replaced monitor overlays (six pages each per MOSPACE).

8. Nine monitor overlays - Open Files (OPEN), Close Files (CLOSE), Label Tape (LTAPE), Operator Keyins (KEYIN), Load-and-Link (LDLNK), Batch Debugs (DEBUG), multilevel index creator (MUL), Device and Type CALs (IODTYPR), and miscellaneous routines (MISOV).

9. RECOVERY, the system failure recovery and restart routines.

10. XDELTA, the executive system debugger.

11. UTS Monitor Root, in absolute core image format.

B. Items Written by GHOST1.

The shared processors are built according to specifications in monitor tables provided by SYSGEN. XEROX shared processors established automatically by SYSGEN are as follows:

CCI, TEL, LOGON
LOGON, LOADER
BASIC, METASYMBOL, FORTRAN
EDIT, PCL, DELTA, BATCH
FILL, RUNNER
GHOST1, DRSP
FORTRAN Public Library, FDP

Swapping Storage

Users (batch and on-line) are removed from core to a dedicated area of secondary storage (RAD or disk pack) when core storage is required for higher priority users.

A bit table (SGP) is used to keep track of the availability of each granule (two sectors = 512 words) on the RAD. In this table, a zero is used to indicate that the granule is in use (assigned to a user) and a one is used to indicate that the granule is available. Users are assigned, in groups of four, a sufficient number of page-size granules to accommodate their current use. 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 swap device when the user is not in core. The disc location of the JIT is kept in core by the scheduler. The device layout is such that sufficient time is available after the user's JIT arrives from the swap device for the system to set up the I/O command chain contained therein for swapping the reaminder of the user program.

The amount of secondary 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 space allocated for swapping and the total size of all active users.

If the swap device is a disk pack, each user is allocated one or two cylinders during SYSGEN. The system still uses the RAD SGP and allocates swapping storage in terms of granules. The exception is the swap I/O routine which obtains the user's cylinder number from a resident table and epecially sets up disk pack command lists to perform I/O to continuous granules on cylinders.

Symbionts and Files

RADs and disk packs are divided into page size (512 words) granules. Each RAD or pack except for the system (swap) RAD is divided into a symbiont area (PER) and a file area (PFA). At SYSGEN, the proportion of each kind of storage on each device is specified. Once generated the PER and PFA are not exchangeable; they form separate allocation pools, except that when PER is exhausted, PFA is used for symbiont space.

For each device, SYSGEN provides an allocation table which contains a bit per granule on the device. These tables are collectively referred to as the HGP, although technically, HGP, the Head of Granule Pool, is a cell containing the address of the first of a linked chain of allocation tables. Also, contained in each allocation table are pointers dividing the PER and PFA area and constants defining the number of granules per track and other device-specific parameters. These allocation tables reside in and are manipulated by the ghost program, ALLOCAT, which is called occasionally to fill or empty stacks of available granules in core memory. Granules required for file addition or released when files are deleted are taken from the stacks of available granules. When the stacks' contents exceed pre-established thresholds, then the ALLOCAT Ghost is called to refresh them to an optimum level.

File Structure

A file may be organized as consecutive, keyed, or random. In a consecutive file, the records may be accessed only in the sequence in which they were orginally written. In a keyed file, each record has an associated name or key. Records in a keyed file may be accessed directly by specific key values or sequentially, according to their order in the file. A random file consists of contiguous granules rather than a group of records. Random files are accessed by granule number relative to the beginning of the file.

A disk file resides on the Monitor's secondary storage. UTS uses both the RAD and disk pack devices for secondary storage. Any combination of these devices can be defined for a UTS system at SYSGEN time. A disk pack device has dismountable volumes and can be declared either a public or private device at SYSGEN time, while a disk device, not having dismountable volumes, can only be declared a public device. A public disk pack has only one volume that can be recognized by UTS, and that volume must be mounted at all times while the system is active. A private disk pack device has any number of dismountable volumes that can be recognized by UTS. The Monitor requires that only those volumes needed for execution of the user's job be made available and be mounted. A public file resides on public devices (RAD and/or disk pack); a private file resides on private disk pack volumes.

A private volume set is defined as a collection of removable volumes that the user has grouped together containing any number of files with any type of organization (consecutive, keyed, or random). All files in a private set must belong to the same account. A private volume set is identified by the volume serial numbers specified in the SN option of the !ASSIGN command when the first file is written on the set. Volumes may be added to the set by entering a new volume serial number in the SN list, but a volume may not be removed.

Keyed and consecutive file space is allocated on a demand basis as the file is being created or updated, therefore such files do not necessarily exist in contiguous areas on a RAD or disk pack device and can exist on many different physical devices. Random file space is allocated when the file is opened for output. The size of a random file can never be changed.

Access to user files is via a hierarchy of disk-resident Monitor files. Figure BC-3 shows the structure of system-managed files. The top file is an Account Directory, which contains a directory of all accounts that have public user disk files. There is one account directory for all public files in the system (the Public File Account Directory). Each account has its own file directory, which contains a directory of all files in the account. Each file has a File Information Table (FIT), which is part of the file directory for random files and part of the file itself for keyed and consecutive files, and contains all the information necessary to open a file, such as its organization, location, password, etc.

To locate a public file, the public account directory is searched for the file account number. The account number entry contains the disk address of the account's file directory. The file directory is searched for the file name. The file name entry contains the disk address of the file's FIT. The FIT contains the disk address of the file.

Private files are located via AVR and MOUNT logic. A keyed file consists of two parts: a Master Index and a set of data granules. The data granules contain the records in the file, which are packed in granule-size blocks. Data granules do not contain any system information. The Master Index is a collection of hierarchical levels of index blocks where the entries in a higher level point to index blocks at the next lower level, and the entries in the lowest level point to data records.

A consecutive file consists of granules containing the data of the records preceded by four bytes of control information per record, generally. A random file is devoid of system information. Record management and format of the file is the user's responsibility. Besides the security checks required for access to a file, the only checks made by the system are to prevent the user from reading or writing past the limits of the file. Functionally and operationally, a random file is a collection of contiguous granules on the specified device type. However, if a random file is larger than a disk pack in size, the file will extend beyond volume boundaries (if private) or device boundaries (if public).

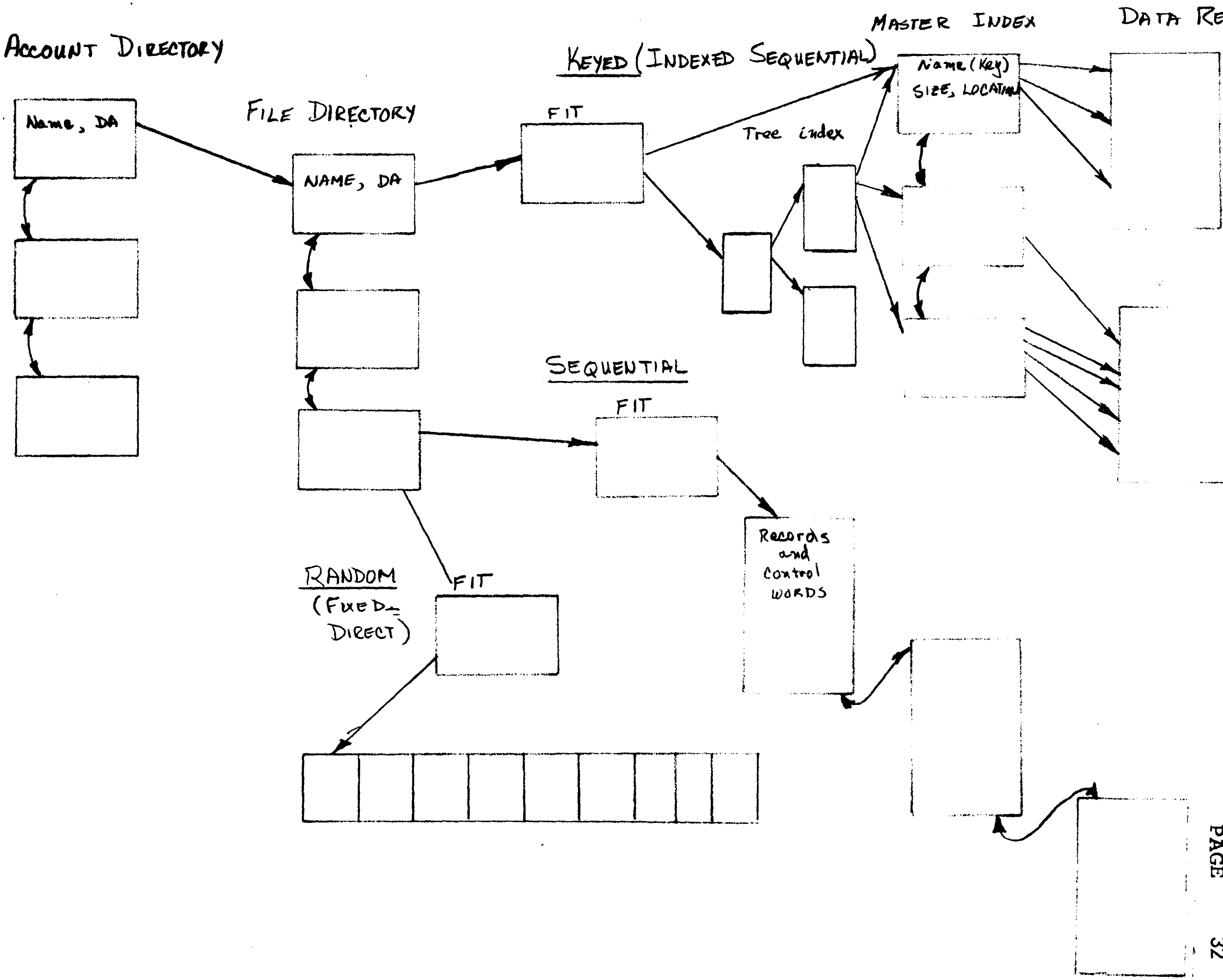


Figure BC-3 - UTS/BTM/BPM File Structure

System PO Tape Contents

The system tape, called a 'PO tape' for reasons lost in antiquity, contains all data needed to begin UTS operation. The tape contains ready-to-run load modules for the monitor, its overlays, and the processors of the operating system. It may contain any other files which the installation desires and includes when the tape is written (DEFed). The tape is structured into two parts. Prior to the first file mark are records absolutely required in getting the system into operation: the monitor, its overlays, EXEC DELTA, recovery, ALLOCAT, and the elements of the initialization program, GHOST1. Following the first file mark, the tape is in standard labeled tape format and contains load modules for all remaining parts of the system. The tape may contain any modules or files whatever. Only those preceding a null file named LASTLM are copied to the system device file structure during system initialization.

The system tape may contain any necessary number of records prior to the formatted part and still be a valid standard format tape because of the label tape identification procedure (AVR sequence). In this sequence, the tape is rewound, forward spaced to the first file mark, backspaced two records, and read forward to find the tape label. Thus, the label is found independent of the number of records preceding the first file mark.

Table BC-2 lists the records on a UTS PO tape.

Table BC-2 - Contents of UTS PO Tape

A. Unformatted Area Records

Tape Boot
Monitor Root in one-page records
System information record containing
version and creation date
EXEC DELTA Head
EXEC DELTA Data*
ALLOCAT Head
ALLOCAT Data*
ALLOCAT Procedure
GHOST1 Head
GHOST1 DCBs (load module protection type 2)
GHOST1 Data*
GHOST1 Procedure (load module protection type 1)
Overlay Head
Overlay Data*
Recover Head — Repeated for the nine overlays:
MISOV,IODTYPR,OPEN,CLOSE,LBLT,KEYIN,
Recover Data* — DEBUG,DLNK,MUL

B. Standard Labeled Tape Formatted Area

:LBL
:ACN
First Physical End-of-File
File records for all system load modules and
other needed files (SYSTEM PROCs, Error
Message Files, etc.)
LASTLM File
Other files as desired
:EOT

*Data is protection type 0 of the load module.

MONITOR FUNCTIONAL STRUCTURE

This section describes the UTS monitor's functional capabilities together with the broad strategy which is used to accomplish each. The outline of this section is echoed in the following section, BE, which reviews the system module by module giving details of the function provided by each, together with approximate physical size.

The broad categories and services provided by each are as follows:

1. Basic I/O

   This section describes the operation of routines which centrally queue all requests for I/O, provide device-specific handling of each request, service I/O interrupts, and buffer and manage all terminal I/O requests.

2. System Management

   This section describes the operation of those portions of the monitor which are responsible for scheduling execution and swapping of user programs, managing core and swap RAD memory, and controlling the sequencing of jobs from step to step.

3. Symbionts and Cooperatives

   The routines described in this section provide for buffing of input and output between user programs and low-speed peripherals (card readers, card punches, line printers, and remote batch terminals).

4. System Services

   This section describes routines which relate to the system as a whole. Areas covered are: initialization, recovery, operator communications, accounting, performance monitoring, system debugging, and hardware error logging.

5. User Services

   The routines described in this section carry out services at the explicit request of user programs. Covered are file management, the load-and-link commands, and batch debugging commands.

1. Basic I/O System

The code grouped in the 'basic control' category includes (a) the routine which queues up requests for I/O activity and handles the I/O interrupt, (b) the basic device I/O handling routines, and (c) the UTS terminal I/O and buffering routines. The first two sets are nearly identical for the BPM, BTM, and UTS systems. The I/O queue routines and handlers are also close cousins to those used in BCM and RBM.

Data used by these routines are largely generated by SYSGEN, including the Device Control Tables (DCTs) and RAD Granule Maps (HGP) in the module IOTABLE, the Queue Tables (IOQ) in M:CPU, and the terminal I/O tables in M:COC.

a. I/O Queueing and Device Handlers

The Basic Input/Output System which is common code to RBM, BPM, and UTS provides a simple interface between all parts of the operating system and the external peripheral devices. It stacks or 'queues' the requests for service rather than waiting for each operation to complete before returning to the caller. When a request is completed, the caller is notified via certain parameters in the DCB, or the caller may specify the address of a subroutine to be executed at this time (called the 'end-action' routine). It is capable of receiving requests for input at any time or from any place in the system and dispatching them in a manner which is independent of other operations concurrently being executed by the system. Error recovery procedures are invoked when necessary and do not require any additional specifications from the caller.

Requests are normally serviced in the order in which they are received. In a real-time system, requests are serviced by task priority. Precautions are taken to prevent any major service to lower priority requests when a higher priority task is active.

Standard techniques within the handlers provide centralized recovery from errors and device malfunctions Operator intervention is enlisted when required, for example, to reinsert a card read with error or to take action on unrecoverable device failure.

There are two basic entries to IOQ: a standard entry in which the I/O commands are prepared by IOQ and the handlers, and an entry in which the entire I/O command list is supplied by the caller.

Few restrictions are placed on buffer size or location. Facilities are included for gather-write/scatter-read operations (data chaining), and provision is made to allow construction of IOP command lists outside of the basic I/O. For standard tape, RAD, and Pack I/O, a monitor buffer is obtained in which data chained I/O command lists are built according to the actual physical core locations of the record requested. A maximum of 8K words is allowed for tape requests.

UTS 'blocks' I/O requests if the calling process is mapped, i.e., a user service. Operation is discontinued for this user and the system turns to the next.

The inherent differences between peripheral devices are accounted for by the insertion of device-oriented code (handler) for each type of device in the system. A well-defined handler interface allows addition of new handlers with a minimum of difficulty. Also, a number of subroutines are available which perform common hander functions.

Handlers are added to the monitor root as a result of a SYSGEN PASS2 DEVICE command which names the device, its addresses, and its handler. This causes the handler to be added to the standard file of handlers which initially includes the handlers for the operator's console, the card reader, the line printer, the RAD, and nine-track tape.

b. Logical I/O Channels

A channel is a data path connecting one or more devices with the CPU, only one of which may be transmitting data (to or from memory) at any time.

Thus, a magnetic tape controller connected to an MIOP is a channel. But one connected to an SIOP is not, for in this case, the SIOP itself fits the definition. Other examples of channels are a card reader on an MIOP, a keyboard/printer on an MIOP or a RAD controller on an MIOP.

Input/Output requests made on the system are queued by channel. This method facilitates starting a new request on the channel when the previous one has completed. The exception to this rule is the 'off-line' type of operation such as rewinding of magnetic tape or arm movement of certain moving arm devices. If this type of operation is started, an attempt is always made to start a data transfer operation as well. Thus, the channel is always kept busy, if concurrent requests are available.

By using logical channels to separate devices on a physical channel (MIOP), the IOQ routine may be used to prevent data overruns when more devices are connected than can be handled by the MIOP simultaneously.

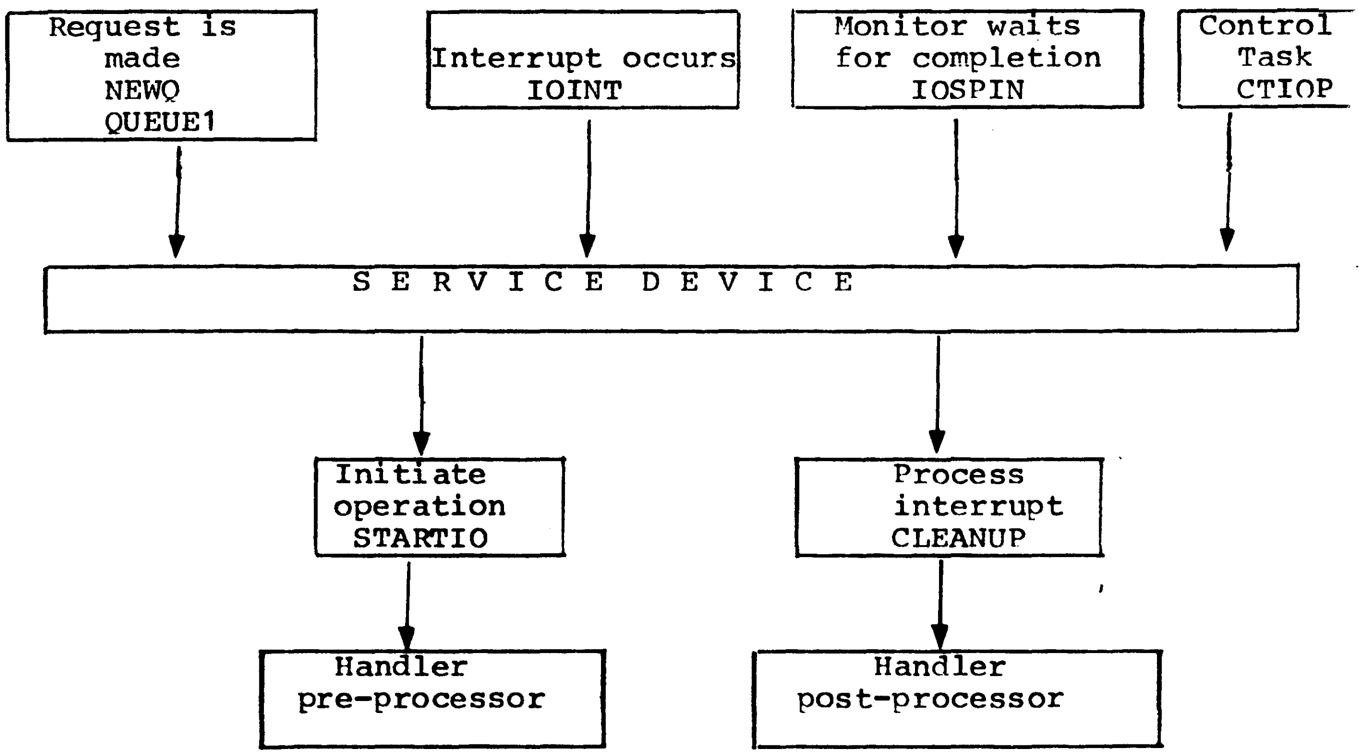
In addition to assigning a logical channel(data path) to a group of devices, it is possible to define two logical channels for a group of devices where the hardware permits. Thus, requests to use any of the devices will be honored as soon as either channel (data path) is available for data transmission. This facility is commonly referred to as 'device pooling'. Thus, for example, two controllers can simultaneously have any two of eight disk packs; whereas, without the feature, each controller would be able to serve any one of four. Obviously, the former case is more efficient, in general.

Since requests on a channel are normally "chained" by the I/O interrupt, there must be a means whereby any action on a request which is deferred by priority may be resumed at a later time. This provision is the 'Control Task', usually the lowest level external interrupt in the system. When action is deferred, the device code is entered into the Control Task stack and its interrupt is triggered. When it becomes active it will call the scheduler for the device in question. In a system created with no Control Task, the console interrupt will be triggered instead. The console interrupt receiver is designed to perform Control Task functions when there is no external interrupt assigned for this purpose.

There are two major parts involved in the processing of an I/O request: start (done by STARTIO) and cleanup (done by CLEANUP). The start consists of building the IOP command list and executing the SIO instruction, while the cleanup consists of testing for errors and notifying the caller of the completion. For a given request, the time at which a start of cleanup is done is determined by the I/O scheduler (called Service Device or SERDEV).

System Flow

The center of I/O activity is the scheduler, Service Device. This routine starts all operations and processes their interrupts (cleanup). Thus, Service Device must be called whenever certain key events occur or when other special conditions are present in the system. The figure below shows the downward flow of control from some of the most important areas of the I/O system.



Service Device is a highly independent routine in the sense that it can be called at any time from anywhere in the monitor. It is called whenever there is any chance that a start or cleanup can be done for a given device. Some examples of when Service Device is called are as follows:

1. When a request is queued (start may be performed for the next request in the queue).

2. After an I/O interrupt has occurred (cleanup may be done).

3. After a cleanup has been done (a start may be performed for the next request in the queue).

Device-dependent routines are provided for building command lists and testing for errors. STARTIO calls the 'handler pre-processor' to do the former, while CLEANUP calls the 'handler post-processor' to do the latter. These two parts constitute the device handler for any given peripheral and are provided in separate assembly modules.;9

Information pertaining to requests, devices, and channels is maintained in a series of parallel tables produced at System Generation Time. The first entry (index = 0) in each table is reserved for special use by the system. Three groups of tables are used 1) to carry individual I/O requests, 2) to carry status and control information for each device, and 3) to group the requests for each logical channel.

IOQ, Request Information

These tables contain all information necessary to perform an input/output operation. When a request is made on the system, data is transferred from the controlling DCB and/or registers into one element in each of the parallel IOQ tables. This set of elements forms a 'queue entry'. The entry is then linked into the channel queue below other requests of higher or the same priority.

DCT, Device Control

The device control tables contain fixed information about each system device (unit level) and variable information about the operation currently being performed on the device.

CIT, Channel Information

These tables are used primarily to define the 'head' and 'tail' of those entries which represent the queue for a given channel at any time. A channel queue may have more than one entry active at any time (such as several tapes rewinding while another reads or writes).

c. Terminal I/O (COC)

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.

The routines support teletypes, ASCII-compatible CRTs, and 2741's for most common speeds, formats, and character encodings. Where full-duplex terminal are available, type-ahead is supported - the user may type input while output is ongoing or before a read request is received. Paper tape units are supported for both full- and half-duplex terminals. Translation of characters may be suppressed to provide arbitrary binary I/O.

Recognition of special characters to allow simple character-delete and line-delete editing functions, mode settings to control echoplex operation, tab simulation, code set restriction, and other activities are included.

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 messages to and from a pool of four-word blocking buffers. All these data are included in the module COCD and in M:COC, which is provided by SYSGEN as a result of processing the :COC control card. Initialization of 7611 lines is accomplished by the routine COCI, which is needed during system initialization, recovery, and power fail-safe restart.

The COC routines are resident in the monitor root and consist of four main parts plus common subroutines, all assembled as a single unit:

1. Output 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 the terminal.

2. System Management

Four groups of routines are associated with this activity: a) those that record the significant events which occur during operation and schedule user execution and swapping from them, b) those that centrally manage core and RAD or Pack memory, allocating and releasing pages of core and granules of secondary storage on demand, c) those that properly sequence the operation of a job between its individual steps, and d) those that associate and release monitor overlays in a job's virtual memory space.

a. Scheduling and swapping

The routines in this group control the overall operation of the system. Inputs to these routines, together with the current state of users as recorded by the scheduler, are used to change the position of each user in the scheduling state queues. It is from these queues that selections are made for both swapping and execution. Swaps are set up by the selection of a high-priority user to be brought into core and by pairing this user with one or more low-priority users to be transferred to swap storage. Similarly, the highest priority user in core is selected for execution.

Scheduler Inputs

System activities are reported by direct entry to the scheduler, which makes changes to user state state queues through a logical event signaling table. The scheduler records inputs by changing the user the user state and other information associated with the user. In general, a table-driven technique is used. The received event is on one coordinate of the table and the current state of the user is on the other. The table entry thus defined names the resulting state or the routine to be executed in response to the given event-state combination. Since the number of events and states is large, the table technique aids in debugging by forcing complete specification to all the possibilities. Inputs to the Scheduler are listed in Table BD-1.

The Scheduler also receives control at execution of each CAL issued by a user program that is requesting monitor service. All these entries (Table BD-2), the special entries from the executive language processors, and entries from internally reported events drive the scheduling of the system. Other entries to the Scheduler occur following each trap, each interrupt, and the end of each clock quanta.

Scheduler Output

The scheduling routine performs two major functions during the time it is in control of the computer. The first is to set up swaps between main core memory and swap storage in such a way that high-priority users are brought into core to replace low-priority users transferred to swap storage. The actual swap is controlled by the swapper according to specifications prepared by the Scheduler according to priority state queues described in the next section. Given a suitably large ratio of available core to average user size (greater than 4), the Scheduler can keep swaps and compute 100 percent overlapped.

The second function is to select a user for execution according to the priority state queues and the rules for batch processing. The rule is simple: the highest priority user whose program and data are in core is selected.

Table BD-1 - Events Received by Scheduler

EVENT	MEANING
E:ABRT	Operator-aborted user.
E:AP	Associate shared processor with user.
E:ART	Associate real-time job (not used).
E:CBA	COC buffer available.
E:CBK	Break signal received.
E:CBL	Number of output characters system limit.
E:CEC	TEL request: Y received.
E:CFB	COC buffer available.
E:CIC	Terminal input message complete.
E:CRD	Read terminal command received.
E:CUB	Number of output characters = system limit.
E:DPA	Swap page available.
E:EI	External interrupt event (unused).
E:ERR	Operator errored user.
E:IC	I/O complete.
E:IIP	I/O started and now in progress.
E:IP	Request permission to start I/O.
E:KI	User back in core.
E:KO	User kicked out of core.
E:NC	Cannot get requested core pages.
E:ND	Cannot get requested swap page.
E:NOCR	Initiate user requesting open or close.
E:NRD	Job exit until next external interrupt (unused
E:NSYMD	No symbiont disc space.
E:NSYMF	No symbiont file entry.
E:OCR	User request to do open or close.
E:OFF	User hung up or logged off.
E:QA	Q for access (e.g., for access to tape or disk pack).
E:QE	Quantum end.
E:QFAC	No file granules available for user.
E:QMF	Master I/O function count exceeded.
E:SL	Sleep time for user.
E:SYMD	Symbiont disc granule is now available.
E:SYMF	Symbiont file table entry is now available.
E:UQA	De-Q for access (e.g., for access to tape or disk pack).
E:UQFAC	ALLOCAT has filed granule stacks.
E:WU	Wake up time for user.

Table BD-2 - Service Request Input to Monitor

SOURCE OF INPUTS	SERVICE REQUEST ENTRIES
User program (through monitor service calls)	1. Terminal input/output request.
	2. Input/output service calls for RAD, disk pack, or magnetic tape.
	3. Wait (sleep) request.
	4. Program exit (complete).
	5. Core request (for common, dynamic, or specific pages).
	6. Program overlay request.
	7. Debug requests.
	8. Requests for control of breaks, traps, timing, etc.
Executive Processor	1. Name of system programs (shared or not) to be loaded and entered (implies deletion of any current program).
	2. Continuation signal
	3. LINK load-and-go exit.

User State Queues

State queues form a single priority structure from which selections for swapping and execution are made. The state queues form an ordered list with one and only one entry for each user. The position in queue is an implied bid for the services of the computer. As events are reported to the Scheduler, individual users move up and down in the priority structure. When they are at the low end, they are prime candidates for removal to secondary storage. This latter feature, that of having a definite priority for removal of users to swap storage, is an important and often overlooked aid to efficient swap management. It avoids extraneous swaps by making an intelligent choice about outgoing as well as incoming users.

In addition to these primary functions, user state queues have other functions:

1. Synchronizing the presence in core of the user program and data with the ability of I/O devices.

2. Queueing user program to be 'awakened' at a pre-established time.

3. Queueing requests for entry and use off processors.

4. Managing core memory.

5. Queueing requests for buffers in core or on RAD.

6. Queueing requests for several non-reentrant services.

A list of the state queues is given in Table BD-3.

Table BD-3 - Scheduler State Queues

STATE NAME	MEANING
AB	Users waiting for a COC buffer.
BAT	Batch compute-bound users under segregated batch scheduling discipline.
BK	Users who have high BREAK.
C	High-priority compute-queue (used for associating processors and some special cases of memory and swap storage management).
COM	Compute-bound users
CU	Current user of the CPU.
CP	Users waiting for a core page.
DP	Users waiting to be allocated a swapping page.
EC	Users queued for entry to TEL (they have hit Y^c).
ERR	User jobs errored by the operator.
IOC	Users with I/O complete.
IOW	Users with I/O in progress.
IOMF	Users queue because of excessive current I/O count.
IR	Users with complete terminal input messages.
NRRT	External interrupt received (not used).
OCU	Users waiting to open or close a file while another open or close is in progress (non-reentrant portions only).
OFF	Operator aborted user or user hung up.
ON	Users queued for the log-in process.
QA	Users queued for access to an I/O device.
QFAC	Users queued for ALLOCAT managed granules.
SYMD	Users queued for symbiont disc space.
SYMF	Users queued for symbiont file table entry.
TI	Users typing input and in core.
TIO	Users typing input and user not in core.
TOB	Terminal output users - in core (more characters than the system limit are ready for typing).
TOBO	Same as TOB except user is not in core.
TOC	Users ready to continue terminal output (the number of characters remaining to be typed is less than a system limit).
W	Users waiting for a specified 'wake up' time.

NOTE: The actual names of the scheduler state queues are those given above prefixed with the letter 'S'.

Scheduler Operation

The scheduling queues may be divided into four categories:

1. READY Queues (SB:EXU)

   Jobs in one of these state queues are ready for execution if in core or ready to be swapped in if not. Through some event, they have indicated a present need for the CPU.

2. ACTIVE Queues

   Jobs, in one of the states CU or IOW, are currently running either using the CPU or one of the IOPs.

3. WAITING Queues (SB:SWP)

   These jobs have no present need for the computer and are not in core.

4. OUT-OF-IT Queues

   These jobs have no present need for the computer and are not in core.

Table BD-4 shows the queue list used for selection of users to be brought in for execution and the queue list used for execution of users to be moved to the swap device. HIR (High-In-core-Ready-to-run) is a condition set when an in core user is in one of the READY Queue states (actually a count of such users).

Table BD-4 - Ready and Waiting Queue Lists

READY QUEUES			WAITING QUEUES
NRRT		High Priority	SYMF
ON			SYMD
OFF			W
ERR			QEI
EC	HIR		QA
BK			DP
IR			TI
TOC			TOB
C			
IOC			OCU
COM			
BAT			
		Low Priority	

To select users for execution, the scheduler searches a list of the state queues, the READY list, in order to find the highest priority user in core memory. The highest priority user is served first. Thus, for example, interrupting users are served before those with an active input message (both of these take precedence over users with unblocked terminal output), then come on-line compute-bound users and, finally, compute-bound batch jobs. Note that users in order states have no current requests for CPU resources. Note also that as each user is selected for execution, the state queue of the user is changed to CU. When the quantum is complete, the highest priority queue which the user can enter is the compute queue. Users that enter any of the high (above COM) priority states receive rapid response, but only for the first quantum of serivce. Thereafter, they share service with others in the compute queue.

A similar selection procedure is used to set up users for swapping. First, the highest priority in the READY list who is not in core is selected and his size requirement (including the requirement for shared processors not in core) is determined. Second, users are selected from the WAITING list until enough space is freed until enough space is freed by these users and their shared processors to provide for the user selected for swapping. If a single user can be found to swap out, then a single rather than multiple swap is chosen. No swaps occur until a user that is out of core enters a high-priority queue (READY Queue). No execution selection occurs prior to the end of the minimum compute quanta. No execution selection occurs prior to the end of the full compute quanta unless the HIR signal is set.

Two lists resulting from this selection are presented to the swapper. One list contains the user (or users) to be swapped out and the other contains the user to be swapped in, the shared processors that must accompany the user, and the current free core-page list.

Priority queues are arranged from high to low in order of increasing expected time before the next activation. This ensures that the users that are least likely to be needed are swapped out first, while the users most likely to require execution are retained in core. For example, the swap algorithm operates so that compute users remain in core and use all available compute time while the interactive users are swapped through the remaining core space whenever the following three conditions exist:

1. There is room in core for three user programs.
2. Two users are computing steadily.
3. Other users are doing short interactive tasks.

In order to prevent deadlocks and to provide for round robin scheduling of the compute-bound queue, the swap algorithm also provides for a search through the READY Queue list in inverse order up to the level of the inswap user for a set of outswap users.

Thus, users whose programs have just issued a terminal input request will be swapped out before programs which have blocked on terminal output. Both of these will precede programs blocked by file I/O requests, and the final selection will be made in reverse order through the queue of compute-bound users.

For file I/O, programs are blocked from the time the I/O command is issued until it is complete. Terminal input is similar. Output to the terminal is no wait until about four seconds of typing have been accumulated in system buffers. It is then blocked; unblocking occurs when one-half second remains.

Since users' programs are of different sizes, it may be necessary to swap out more than one program to make room for the incoming program, although a detail of the selection algorithm causes it to preferentially select a single outswap program if one adequate size (including any associated shared processors) can be found on the WAITING Queue list.

The layout of programs on the swap device is made by selecting four pages (always a 512-word granule) at a time from a common pool, but preferential allocation occurs for pages which will maintain nearly continuous sector-by-sector allocation. This technique keeps swap time short while preserving a general allocation scheme. Programs are allocated to storage with the pure procedure portions ordered last so that the procedure portions do not have to be transferred from core to swap storage when a copy already exists on the device.

Note that the queues CU, IOW, TOBO, and TIO do not appear in either list. Thus, the users in these states are not selected either for execution or for swapping, nor is unnecessary overhead expended in their search.

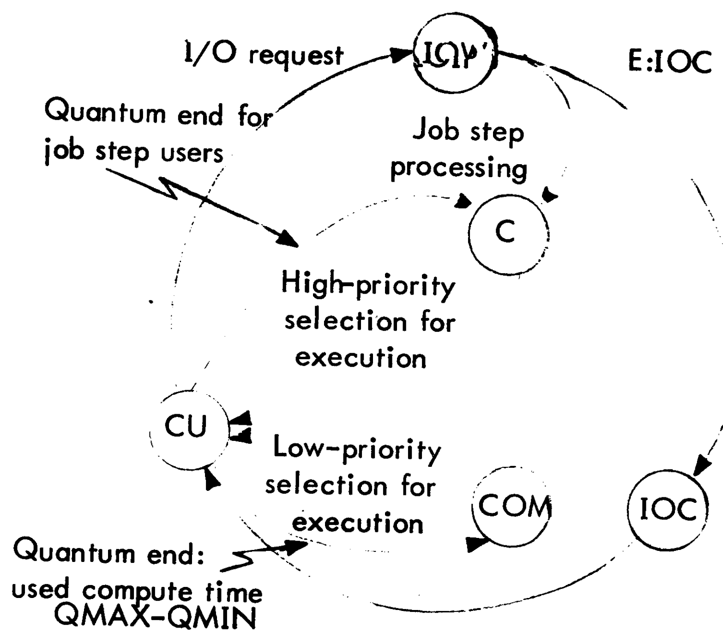
Two examples of typical interactive use are illustrative of the scheduling operation. The first example traces scheduling operations for a simple, short interactive user request. At the time the request is typed, the user is in the typing input (TI) queue. His program, which has probably been swapped, remains on swap storage until the COC routines receive an activation chracter. Receipt of this character is reported to the scheduler and causes a change in state of the user to input received (IR).

The scheduler finds a high-priority user not in core and initiates a swap removing a low-priority user (if necessary) and bringing in the one just activated. On completion of the swap, the scheduler is again called and now finds a high priority user ready to run. Given that the current user has completed his minimum quanta, the user's state is changed to CU, the program is entered, and the input command is examined by the reading program. The cycle in this example is completed by preparation of a response line and a request to the monitor for more input, which changes the user's state to TI again, making him a prime candidate for removal to swap storage.

The second example illustrates an output-bound terminal program. This program moves through the state cycle TOB-TOC-CU as output is generated by the program. The COC routines signal when the output limit has been reached, thus causing the program to be delayed while output is transferred to the terminal. In a typical operation, four to six seconds of typing is readied in buffers each time the user program is brought into core and executed. During the typing time, the program is not required in core and the CPU resources can be given to other programs.

I/O Scheduling

I/O scheduling is designed to give job step I/O a very high priority to provide good terminal response. Other I/O is permitted to run as fast as possible until the user has accumulated a full maximum quantum of CPU time, at which point the user is placed at the bottom of the compute queue. The scheduling scheme is illustrated in Figure BD-1.



An I/O-bound user cycles through the queues CU, IOW, IOC, and CU until he exhausts his time quantum at which time he cycles through the compute (COM) queues. This ensures that a single I/O bound user does not dominate the system. I/O that occurs at job step time (that done by CCI, TEL, and the program fetch logic) proceeds through the higher priority C queue. If the number of concurrent I/O operations for a user exceeds a specified limit, the user is blocked in state IOMF until some of them complete.

Reentrancy

The scheduler permits job-to-job switching only at certain carefully controlled points within the monitor. At these points control is explicitly given to the scheduler for job switching. The scheduler also receives control on asynchronous events from traps and interrupts (this code is completely stack-reentrant in the unmapped stack), but it enforces a logical disable of monitor operations by returning to the point of interrupt if the trap or interrupt occurred with the monitor in control. This scheduler-enforced logical disable allows critical monitor operations, such as a file index update to run to completion before permitting another user job to proceed and possibly interfere with the incomplete activity.

Batch Jobs

Two ways of scheduling batch jobs which result in quite different fractions of machine time devoted to batch processing are reasonable in this priority structure. Both are provided in UTS, and the mode of opration may be selected by the installation manager.

The first scheduling technique keeps the batch job stream in a separate queue (BAT) that has a lower priority than the interactive compute queue indicated in Table BD-3. Thus, batch jobs get service only when no interactive user has a request. Estimates from current systems indicate that 10 to 20 percent of compute time is available to batch processing on a system supporting between 20 and 30 concurrent users in prime shift. During nonprime time, 80 percent or more of CPU time is available to batch jobs.

The second method of scheduling cycles batch jobs through the interactive compute queue, where each job receives an equal fraction of the available time. It is usual in on-line systems for 5 to 20 percent of the on-line users to be computing at any one time. Thus, as much as one-half of prime time, plus 80 percent of nonprime time, could be devoted to batch background operation. In this scheme, batch jobs can be biased to get a different quantum than on-line user, thus permitting the installation manager to control the actual percentage of computer time devoted to batch processing.

b. Memory Management

These routines control the allocation of physical core memory, maintain the map and access images for each user, service the get and free page CALs, and manage the swapping space.

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 device so that it need not be written out if it is unchanged. These two requirements make necessary a shuffling of space on the swap device and corresponding adjustment of memory maps and swap command list when a new data page is acquired.

Frequently no new core pages are available when requested. In this event, memory management must allocate the swap space and not the core space by the 'get virtual, no physical' process and cause an entry to the swapper to provide the needed extra page(s) through its normal swap scheduling algorithms.

Physical Core Allocation

Allocation of core memory pages to a user at his request depends on the actual size of the machine as determined during initialization, the current size of the user including all needed shared processors and the management set limits on user size. Details of the calculations are given below.

The following table describes how physical memory is reserved for system functions in UTS:

AMOUNT (in pages)	USED FOR	HOW ESTABLISHED
(JITLOC+511)/512	Resident Monitor	SYSGEN
9	XDELTA	Answering "Y" to DELTA during initialization request.
6	Longest Overlay (OPEN)	Initialization
3	KEYIN Procedure	Initialization
1	KEYIN JIT	Initialization
1	Monitor JIT	Initialization
1	Each Symbiont Device	Initialization

The above table shows that an 80K system with three symbiont devices and a 27K monitor will have 41.5K in which to run user programs if XDELTA is requested, and 46K if it is not.

In addition, pages must be reserved for the context area and other things, as follows:

PAGES	PURPOSE	HOW ACQUIRED
1	JIT	Logon
1	AJIT	Allocated when N pages are acquired and is never released once allocated. N is 32 for Σ7 and 13 on Σ9 greater than 128K.
n	DCBs	Job step time, from user program.
m	IPOOL/FPOOL Buffers	Job step time. A minimum of two IPOOL and two IPOOL are required; i.e., three pages.
2	CPOOL Buffers	Automatic for batch jobs, reserved if an on-line user has symbiont access in his account.
8	TEL	Reserved if user is on-line.

Note: n may be obtained from the LOADER map and is never program-dependent.

m may be altered using !POOL card; otherwise, system defaults are assumed. these defaults are defined at SYSGEN time and may be altered using CONTROL.

Therefore, the maximum user program size run on-line on the previously mentioned system, with two pages of DCBs and the minimum allocation of file buffers (three pages) would be 33K with XDELTA and 37.5K without. The maximum size of the same program in batch would be 37K with XDELTA and 41.5K without.

An increase in physical memory will increase the maximum size of a user program up to a point (less than 128K) where the limiting factor is the virtual memory layout. The first 32K of virtual memory is dedicated to the Monitor. The context area which includes monitor overlays, buffers, DCBs, JIT, and AJIT follows in the next 16K of virtual memory. The next 64K is set aside for user programs, and the last 16K of virtual memory is allocated to special shared processors and shared libraries. 64K is available for user program pure procedure and data, and 12K is available for user context (DCBs, buffers), not including JIT and AJIT - maximum program size is 76K.

On Sigma 6 and Sigma 9 configurations with 128K or less, an AJIT is required when the user size exceeds 32 pages. On Sigma 9 configuration over 128K, this threshold is 13 pages due to the larger memory map.

c. Job Step Control

The collection of monitor resident routines called STEP is entered between major segments of a job or an on-line user's session. Entries are made whenever ERROR, EXIT, or ABORT CALs are executed or when a new shared processor or new program must be fetched. When command processors (CCI, TEL, or LOGON/OFF) exit, they do so with coded information in registers which are used to associate a shared processor or fetch a prepared load module. (This exit is known as an interpretive exit.) Prior to either type of fetch, the user's core and swap RAD space are returned to the available pool to be reacquired during the fetch. Following the fetches, all DCB assignments associated with the user are merged into the DCBs acquired in the latest fetch. Required initialization of JIT is completed.

Following an exit by LINK from the load phase of processing a RUN command, step control sets up the loaded program, core image for execution, including the association of required shared debuggers and public libraries.

Exit from CCI, TEL, and LOGON/OFF includes two other 'interpretive' exits. The first, to simply continue the current activity, and the second, to do the final cleanup after LOGOFF exits. The latter includes a test for completion of a batch job. If the job is completed, entry is made to the batch scheduler for selection of another batch job for processing.

I/O, issued by STEP in order to fetch programs and processors at user request, is handled as a special high priority in order that good response time be achieved in these cases.

3. Symbionts, Cooperatives, and Multibatch Scheduling (RBBAT)

a. Symbionts/Cooperatives

Records sent to and received from the low-speed peripherals (CR,CP,LP,PL,RBT) are buffered to RAD or pack through the symbiont-cooperative routines. Four stages are readily identifiable.

First, input jobs from the CR or RBT are blocked by the input symbiont into disc unit records and written in the peripheral storage area (PER). This process is carried out asynchronously with respect to other tasks in the system and, once started, is interrupt-driven until completion. Initiation is accomplished by operator command for CR and is automatic for RBT. The input symbiont recognizes !JOB cards for CR and RBT and treats them as beginning-of-file and end of previous file (if any), recognizes !FIN cards for CR and RBT and treats them as end-of-stream, and recognizes !RB cards for RBT and treats them as beginning-of-file/end of previous file as with !JOB cards. At file end, the file starting disc address is passed to RBBAT, the symbiont file ghost job, for entry into the batch tables.

Second, when a user issues a read directed to the card reader, the operation is intercepted by the input cooperative. This routine reads and deblocks the records for presentation to the reading program, which is not allowed to read past the end of the symbiont file containing his own job. Initially, the multibatch scheduler selects the job to be run by placing the job and resource information in the GET tables. The batch user is started and the !JOB card CCI read causes this information to be placed in the user's JIT. Thereafter, records of the file are passed to the user on subsequent reads.

Third, the output cooperative, which is an intercept routine acting on all output directed to symbiont devices, blocks records into buffers, and writes them to secondary storage. Separate symbiont files are built for each type of output (print and punch). Upon user signal ('superclose', usually at end of job), the file is cloosed by entering it into the RBBAT queue via the add output file communication.

Fourth is the interrupt-driven task (the output symbiont), which reads symbiont files and writes the symbiont device. Output symbionts are started automatically when RBBAT senses that there is work to do, the device is idle and, otherwise, capable of processing the output.

Symbionts use, for buffer memory, pages obtained from the general pool of physical memory. This restricts maximum user size in that a user must not be allowed to exceed the available physical memory left while symbionts are active. The cooperatives use similar buffer and control memory pages from the user's virtual space. The buffer management routines get memory and restrict size appropriate to the mapped/unmapped (cooperative/symbiont) condition on entry.

Symbiont files are selected by the Multibatch Scheduler (MBS) portion, RBBAT, for input and output by resource, priority, system id, and control information maintained by RBBAT. Priority by symbiont files which originates from the job card (or on-line user default) may be changed by the operator, who may also delete files. Control information (e.g., remote batch hold) is specified by the user. Figure GA-1 shows the symbiont and cooperative big picture.

b. Multibatch Scheduler

Inputs

- o Job description (resource requirements) from JOB and LIMIT cards. This information is carried in input symbiont tables which reside in the RBBAT.
- o Partition definitions (permissible ranges of resource values) created by SYSGEN in resident tables and modifiable dynamically during system operation using CONTROL.
- o Maximums, also carried in resident tables and changeable via CONTROL, which limit the total use of each resource by all batch (or on-line) jobs taken together.

New Job selection initiated whenever:

1. a job completes exeuction.
2. a new job is entered.
3. partition definitions are changed.
4. operator command !S is issued.
5. Resources are released (by an on-line job or by a CAL which releases resources).
6. Clock routine which checks a flag set by certain cases of resource releasing.

Scheduling Algorithm

1. Identify all available partitions (not executing, not locked).

2. Find the highest priority job which fits one of the available partitions.

3. Verify that execution would not exceed established maximums.

4. Failing 3, increment job priority and go to Step 2.

5. Verify that order and account parameters do not preclude running the job.

6. Run the job selected if all tests have been passed.

7. Go back to Step 1, unless:

   a. The job was 'F' priority and not selected.
   b. No partitions are available.
   c. All jobs in the input queue have been processed.

4. System Services

a. System Initialization

UTS initialization routines accomplish three major functions: booting from a system PO tape, booting from system resident secondary storage, and recovery-restart. The functions are accomplished by common routines which distinguish recovery from booting by zero contents of cell 2A which is always filled in during a device boot by the hardware.

The initialization routines fall into three physical groups: first, the routine INITIAL which initializes trap and interrupt cells and loads locks and access images both for booting and recovery; second, the routine BOOTSUBR which provides for monitor patching and system storage initialization; and third, the initialization job, GHOST1, which copies the system tape to the system account, provides for GENMOD patches to processors, and completes system storage initialization. The last two processes which have similar functions are divided in order to remove as much code as possible from the monitor root to job status even though, in this case, it is a master mode job. BOOTSUBR completes just enough initialization of the system to enable it to run its first job, GHOST1, which completes the initialization task. Figure BD-1 summarizes the initializarion process.

INITIAL

This routine is entered immediately after a tape or disc boot has read in the monitor's root or after recovery has done the same thing. Its purpose is to preset the hardware for system operation. It accomplishes this in the following order:

1. the unmapped JIT is moved from assembled location to execution location;

2. external interrupt cells are preset to zero;

3. the trap and interrupt cells 40 through 5F are initialized;

4. the memory locks are set to 01 everywhere except the code portion of the monitor, which is set to 11;

5. the virtual memory map is preset in one-to-one correspondence with physical memory;

6. access is preset to read-only for virtual page zero and to no-access for the rest;

7. I/O interrupts are enabled for tape boot; CLOCK4 counter for disc boot; and

8. GETHGP is called to read in XDELTA if initialization is from disc.

FIGURE BD-1 - Initialization Overview

Tape Boot | Disc Boot | Recovery

INITIAL

Move master JIT to Execution Location.
Zero external interrupts
Set up 40 through 5F.
Load Locks, Access.
Enable I/O interrupts.
Enable CLOCK4 counter interrupts.

BOOTSUBR
MONINIT

Check and set assigns for C, LL, DC, COC.
Print and type patch numbers.
Type sense switch setting assignments.
Set up location 2B with proper monitor type.
Read in XDELTA.
Read card reader via XDELTA, patch root.

SWAPINIT

Copy ALLOCAT, GHOST1, Monitor Overlays
XDELTA, RECOVER to swapper.
Set up monitor tables with disc addresses.

WRTROOT

Write monitor root to swapper.
Write bootstrap on swapper.

GETHGP (Get XDELTA)

Set up memory size info.
Turn off symbionts.
Enable all interrupts.

GHOST1

Ask about DELTA and keep or no; release core of INITIAL and BOOTSUBR.
PASS0 to read and patch (GENMOD) processors.
RECOVER2 for shutdown of open files.
SYSMAK: copy shared processors to swap RAD
Request date and time from operator.
Write start record in ERRLOG.
Initialize COC.
Turn on symbionts
Log on Analyze to process crash dump
Start scheduling batch jobs by start of RBBAT ghost job; interpretive exit to FILL.

BOOTSUBR

Three subroutines of BOOTSUBR are then called if the initialization is from a PO tape: MONINIT, SWAPINIT, WRTROOT.

MONINIT, the first subroutine of BOOTSUBR, carries on the initial dialogue with the operator:

1. it requests from the operator new device addresses for card reader, line printer, system resident swapper, and COC, providing dynamic reconfiguration for these devices;

2. it prints the patch segment numbers and sense swith setting both on line printer and on the operator's console;

3. it sets location 2B with monitor version and type; version comes from the monitor information record generated on the PO tape by DEF;

4. it reads in EXEC DELTA and initializes the monitor cells which locate it;

5. it then passes control to EXEC DELTA to read the card reader for monitor patches, interpret them, and place them. If the ** card is read, a flag is set to control the 'boot-under-the-file-structure' operation, in which the PO tape is not read.

SWAPINIT, the second subroutine of BOOTSUBR, initializes the system portion of the swapping RAD. Enough monitor elements must be placed to be able to run the first job - the GHOST1 initializer. During copying to the swapper of monitor overlays, ALLOCAT, the elements of GHOST1, XDELTA, and the RECOVER overlays, the card reader is read by DELTA for patches to them; monitor tables which record overlay swapper locations are set up. This setup defines the portion of system RAD which must be intact to accomplish recovery. Recovery uses the system swapper from this point on (that which will be occupied by the shared processors) to save the crash core dump.

WRTROOT, the third subroutine of BOOTSUBR, writes the monitor root which is now fully initialized and patched to the system swapper. The routine also writes the disc bootstrap routine onto system swap storage.

INITIAL's Entry to GHOST1

Final activity carried out before entry to GHOST1 includes:

1. scanning memory for existing physical pages which are linked into an available memory page pool;

2. enabling of all interrupts (COC lines are not scanned by the clock interrupt routines until later when the input external interrupt locations are set up);

3. temporary disabling of the symbionts so that GHOST1 will use printer and card reader directly.

INITIAL exits through the job initialization logic calling for startup of GHOST1.

GHOST1, The System Initializing Program

This master mode job contains all initialization and recovery functions which can be run as a job (as distinct from those functions which must be imbedded in the monitor root). The program takes differential action on recovery (cell 2A = 0) and on disc and tape boots. Major elements included in GHOST1 are as follows:

1. RECOVER2, which is entered only in a recovery situation to replace dynamic system information like the date, to provide accounting summaries for all interrupted jobs, to copy files that were open in update mode and could not be closed normally, and to copy the core dump from the swapper to a permanent file,

2. PASS0, which copies PO tapes to files, including the application of GENMOD patches,

3. SYSMAK, which reads the shared processors from files and prepares absolute copies on the swapper.

As shown in the schematic flowchart of Figure BD-2, GHOST1 first asks if EXEC DELTA is required. If not, or if there is no answer within six seconds, the physical memory used by EXEC DELTA (from about 60-64K physical) is released to the physical page pool and the 'Lees-watering-hole' entry to EXEC DELTA at location 4E is disabled.

A check of location 2A determines whether recovery (2A = 0) or boot is intended. RECOVER2, PASS0, and SYSMAK are entered, as shown.

Following a date-time request, if booting, common logic is entered which:

1. writes a startup (or recovery) record into the hardware error file,

2. initializes terminal I/O by starting COC I/O and turning on all line receivers,

3. turns on the symbiont system,

4. logs on a ghost job for Analyze (if recovering) to process the crash dump,

5. enters the batch job scheduler to start jobs still in the input symbiont queue after a recover, and, finally,

6. exits through the monitor's interpretive exit logic to activate FILL for possible reading of backup tapes.

The flowchart Figure BD-3 shows PASS0's main execution line.

SYSMAK copies the shared processors listed in the monitor table P:NAME from files to locations on the swapper with addresses, sizes, and start addresses placed in the monitor shared processor tables.

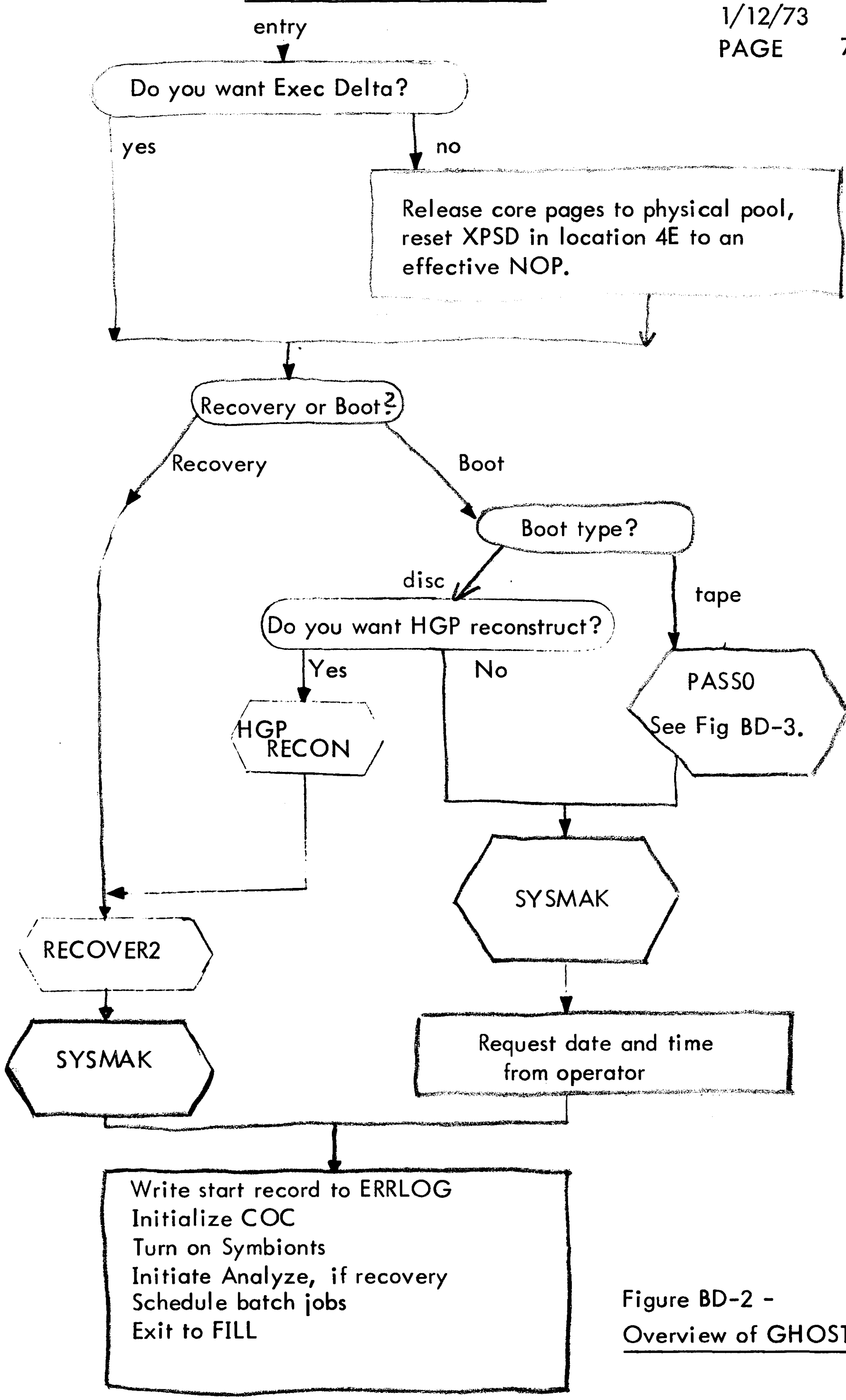
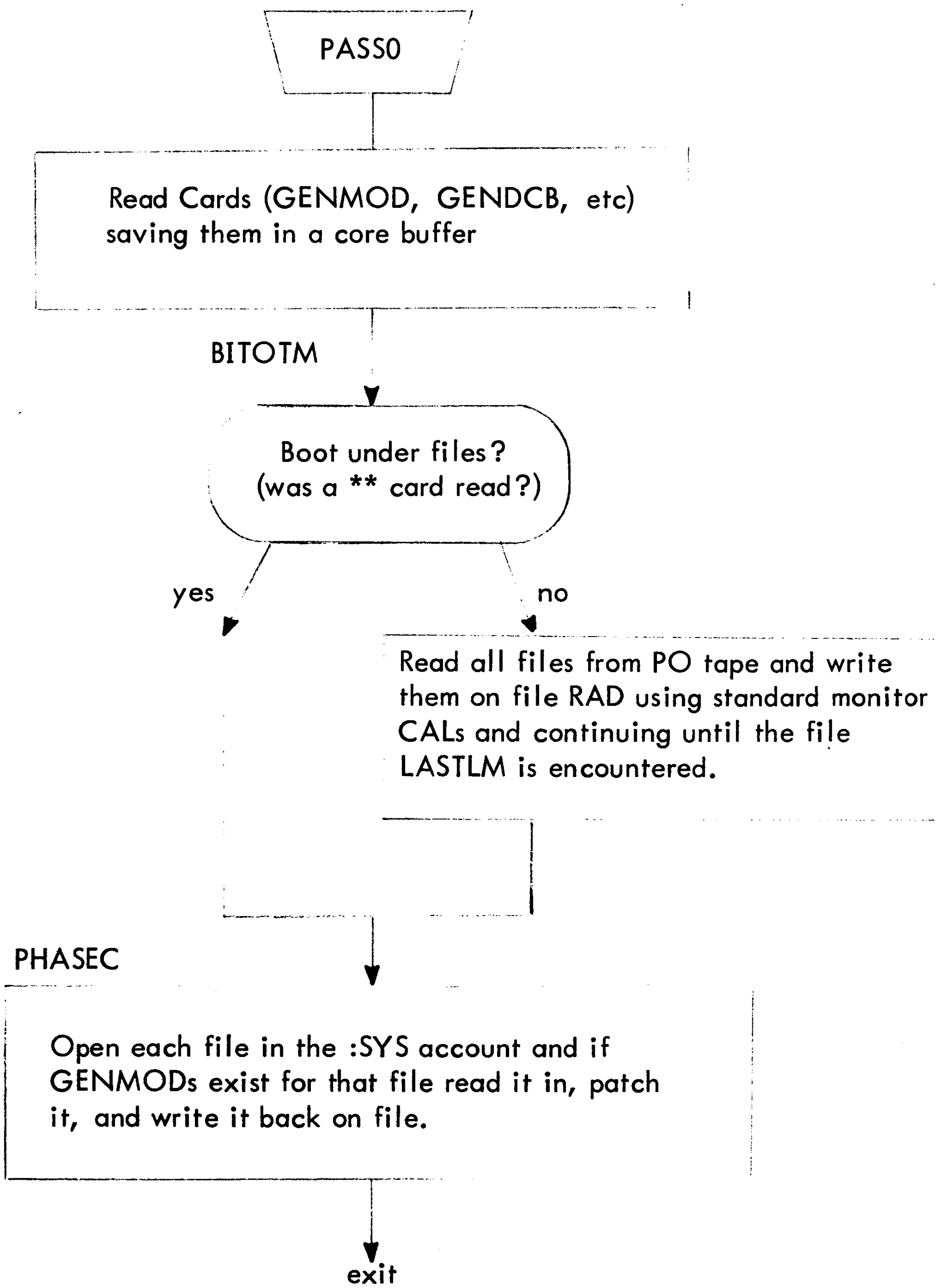


Figure BD-2 - Overview of GHOST1

Figure BD-3 - PASS0 Overview



b. Operator Communications

The machine operator communicates his instructions and requests to the system through key-ins at the operator's console. This 7012 console is a TTY-like EBCDIC transmitting device connected to an MIOP. It is usually designated TYA01. Since the device may be used in only one direction at a time, the operator must signal his desire to type by pressing the PCP interrupt button. He is prompted for input with a !, carriage return terminates the control message, and EOM deletes it for a retry.

When the PCP interrupt button is pressed, IOQ recognizes the request and starts the console read operation into a dedicated buffer. On completion of the message, the ghost job for KEYIN is initiated. The pre-established JIT for this job is read, and the initial environment is pulled and executed as is normal for job beginning. For the KEYIN job, the program is contained entirely in the registers. The two-instruction program calls for association and entry into the KEYIN overlay and for job deletion after return.

The KEYIN overlay reads the input message from its fixed buffer, interprets it and acts on the commands. The overlay structure is used in order to provide convenient direct entry to monitor routines and to the monitor tables which KEYIN is directed to change or display.

c. Accounting and Performance Monitoring

CPU execution accounting is carried out by the incrementing of the CLOCK4 timer. This clock ticks each 2ms into a cell in the JIT. Addressing is subjective, that is, the JIT of the current user is selected by the setting of the memory map. When the map mode is not on, the time increments are accumulated into the monitor's JIT located at the same physical address that is occupied virtually by user JITs.

Thus, when the CPU is executing for a given user, whether in his program or in the monitor cting at his request, time ticks are directed to his JIT via the map. When the monitor is operating unmapped in servicing I/O or terminal character interrupts, processinq traps, providing symbiont I/O or scheduling jobs - all general services which are not simnlv allocatable to a single job - the time ticks are accumulated to overhead cells in the master unmapped JIT.

Two other breakdowns are performed on the CPU time accumulated for each user. The two breakdowns result in four separate CPU time accumulations. Time is separated at the CAL boundary accumulating time used by the user program and monitor time used to carry out his CAL requests. Monitor service and program time are carried separately also for UTS shared processor execution and other proqram execution. This is slightly different than BPM/BTM which counts processor execution for all programs coming from the :SYS account. COBOL is the important processor which is not shared and is therefore accounted for as a user program.

Performance monitoring is carried out as an integral part of the UTS system. Subroutines and count-incrementing instructions are embedded in the monitor at appropriate places. The counts which they accumulate and the program to display these counts are described in detail in the UTS System Management Reference Manual.

Approximatelv one page of memory is devoted to accumulation of data on system operation. In order to keen the memory required small some reduction of the data is done at the time of gathering. Along with sums and counts for averaging, certain data is accounted for by adding into an appropriate cell of a distribution histogram.

d. Automatic Recovery

The system recovery function is provided to restore UTS to operational status very quickly following an unrecoverable failure, which may be either hardware or software caused. Some examples are memory parity error by the hardware or an illegal memory reference trap because of software error. Each reported error is checked to determine whether the entire system is in danger (unmapped mode errors) or if only one user is affected (mapped mode errors). In the latter case, that user is logged off, or failing that, deleted, and system operation continues. In the former case, recovery is entered. Recovery consists of cleaning up all open-ended information (both user and system-oriented information) and restarting the system at initialization. If this occurs all terminal users must log on again and the current executing batch job(s) must be resubmitted. Any job partially read through the card reader must be reinserted. Jobs already submitted but not yet in execution are saved and need not be resubmitted. The recovery routine is entered whenever hardware and software errors are detected. Manual entry is also provided for use by the operator when the system cannot automatically recover, such as if low core erased or the system loops.

When the recovery routine is entered, none of the normal operating system is assumed to be operating. Most routines of the normal system required for recovery are duplicated in the recovery routine, but for automatic recovery a small resident recovery driver is required intact. This driver brings in the bulk of the recovery routine, overlaying the pure procedure portion of UTS. Certain monitor tables are also required intact. This is verified where possible. If the recovery process cannot be completed, the operator is instructed to reload the system from the PO and file backup tapes.

The recovery routine performs the following functions:

1. Displays cause of failure.

2. Takes a full core dump for later analysis.

3. Closes all open files using default options.

4. Packages or releases all partial symbiont files.

5. Packages error log.

6. Informs users of interruption.

7. Saves time, date, error log pointers, accounting information, symbiont file directory, and RAD granule stack contents.

8. Restarts system and restores items saved above.

When any functions cannot be performed, these are noted on the operator's console. If the function is considered minor, recovery continues. If it is connected with file operations, the file identification is noted and recovery proceeds.

If recovery determines that the RAD allocation tables (HGP) or File Control Tables (CFU) have been destroyed, then a routine is called to rebuild the HGP by reading through the entire file hierarchy, recording RAD and pack addresses as it proceeds. While this technique cannot repair or replace file elements which have come unlinked during the failure, it does provide a much faster restart mechanism than reloading of files from tape (about 15 minutes, as opposed to one to five hours, depending on reload technique and file size).

e. System Debugging

Although much system debugging is carried out by other means and with other tools, UTS carries with it a master interactive debugger called EXECUTIVE DELTA. Language features of this debugger are virtually identical to those of user DELTA as described in the UTS Time-Sharing Reference Manual.

EXECUTIVE DELTA carries with it an elided symbol table for the monitor and may be entered through location 4E. EXEC DELTA does not use (and therefore depend on) monitor I/O and thus, may be used to examine, change, set breakpoints and otherwise completely control the operation of the system whenever such steps are necessary for detailed debuggging or development activities. (For most crash analysis on running systems, the dumps taken by recovery and reported by ANALYZE are adequate for finding problems.)

EXECUTIVE DELTA is loaded with the monitor's REF/DEF stack and placed on the system PO tape by SYSGEN. One of the first tasks of the boot routines is to bring in EXEC DELTA and place in physical memory at approximate location 60-64K. During the boot processes it may be used to make symbolic patches to the system either entered from the console or from the card reader. At the end of the boot process the operator has the option of retaining DELTA for possible later use or releasing it and returning its physical space to system use. Once released EXEC DELTA cannot be regained except through the recovery process.

f. Error Logging, Diagnostic Device Access

Recording of hardware errors for analysis by customer engineers is carried out by a special procedure designed to minimize the possibility of losing the record of the errors. Each device error, watchdog timer trap, memory parity error, device timeout, etc., together with system startup and recovery records and software-detected inconsistencies which might have been caused by hardware errors are recorded by the resident error logging routine into a pair of 64-word core buffers which are then transferred to RAD in a simple linked chain. A special CAL may be used to read this file and a routine, ERR:FIL, is provided with the system read this special file and, using standard file management operations, transfer it to a standard managed file, ERRFILE. ERR:FIL is called as a ghost program each time five error records are accumulated. In file form, the records are accessible to customer engineers and to two standard system programs, ERR:LIST and ERR:SUM, for listing and summarizing the error file contents. Descriptions of these programs and of ERRFILE record formats are given in the UTS System Management Guide.

Also provided for customer engineers is a privileged method for opening I/O directly to a device, bypassing the symbiont operation. Thus, diagnostics may be run on-line during system operation to diagnose, test, or PM the peripherals. In this special mode, the AIO, TDV, and TIO status information from the device are returned directly to the program via the DCB. Error and failure records are still recorded in the error log and privilege-controlled CALs allow direct reading and writing of the special error file. Alternately, the diagnostic program may cause ERR:FIL to transfer records to the standard file, ERRFILE, by issuing a ghost job initiation CAL, and read the records from that file.

5. User Service

This category encompasses most of the monitor routines which are called at the explicit request of user programs, both batch and on-line, through CAL instructions. The major categories are: a) file management service for reading and writing of files on tape, RAD, and disk pack; b) load-and-link services; and c) batch debugging services. Also in this category but not explicitly described in this overview, are routines for the UTS-specific CALs, trap control and timer CALs, the user program overlay segment loading CAL, error log read and writing CALs, and the job entry CAL.

a. File Management

This category includes routines which manage the contents of and access to physical files of information. Included are the functions of indexing, blocking and deblocking, management of the pools of granules on RADs and disk packs, labeling, label checking and positioning for mag tape, formatting for printer and card equipment, and controlling access to and simultaneous use of a hierarchy of files.

Four subgroups are identifiable:

1. Basic routines for reading and writing files and physical devices.

2. Routines for opening and closing files.

3. Routines to service the CALs requesting position changes in files or on tape (PFIL, PRECORD, REW, WEOF, PEOF) and those requesting DCB changes for device DCBs (all the M:DEVICE CALs).

4. Routines to service labeled tape.

The primary storage areas used by file management are the DCBs and buffer areas in user virtual memory, and the CFUs in resident core which control simultaneous file usage. Also in resident memory are 'monitor buffers' from MPOOL, which are used primarily for preparing operator console I/O. Occasional use of DCT and IOQ tables occurs.

All physical I/O is accomplished via the basic I/O routine, IOQ. Entries to the file management routines are via the CAL receivers, CALPROC and ALTCP.

b. Load-and-Link Command

This set of monitor routines is contained in the overlay, LDLNK, and processes the M:LINK and M:LDTRC CALs. They allow processors to pass control back and forth from one to another in either a subroutine or transfer-of-control fashion. COBOL object programs and the MANAGE processor use SORT as a subroutine via M:LINK; PASS3 of SYSGEN uses the Loader in a similar way. Communication between caller and callee is via information stored in COMMON memory and in registers.

When an M:LINK is issued, the entire program and context, including open DCBs but not the COMMON memory area, is saved in the star file idN where N is a binary number incremented for each M:LINK. All memory except COMMON is released and control passes to a point in STEP to associate the indicated shared processor or fetch the named program. The parameter N is passed to the called program to identify the saved program for possible return.

Two possible actions are available for M:LDTRC. The first is like M:LINK except that the current program is not saved. The second occurs when the request names a program file, idN, preserved by a previous M:LINK. Current memory pages are released and the file idN is read in. The file idN is released and the program entered at its return point just following the M:LINK.

Cleanup is necessary for the saved program images after program exit or abort and processing of any PMDs. This need is indicated by a nonzero value of the link counter, N, in the rightmost byte of the JIT cell, J:RNST. Each idN file is read and all DCBs therein are closed, the file is released, and finally, N is zeroed.

c. Batch Debugging

Batch debugging services include program MODIFY commands, execution output and test via SNAP, IF, AND, COUNT, etc., CALs and control cards, and postmortem dumps through PMD commands.

These commands are read, processed, and executed by the coordinated action of the processors CCI and RUNNER, the root element STEP, and the monitor overlay DEBUG. The processors read and prepare tabular forms of the commands while the monitor elements carry out the indicated actions.

The process begins when CCI reads the MODIFY, SNAP, PDM, etc., cards which follow the RUN command in the JCL stream. A RUN table is built from the information on the RUN card and left in high virtual memory for use by RUNNER and STEP. For each card read after the RUN card, a record is written into the star file, idD. A flag is left in JIT to indicate the presence of PMDs and a count of the number of other debug cards is left in the run table and CCI exits indicating the required load module fetch to STEP.

The fetch portion of STEP calls the special shared processor, RUNNER, as an aside in order to process the idD file. RUNNER reads the file and creates two tables in core, the first of which contains location and contents values corresponding to MODIFYs and SNAPs. The second table contains FPTs for the debug CALs. PMD and PMDI records are left in the idD file.

The head and tree of the load module requested (as recorded in the RUN table) by the original fetch are read by RUNNER, the size of the pure procedure area is determined, the two tables are moved into position just above it, and the head and tree records are updated to reflect the additional pages (if any) and the LM start address. The page containing the Run table is released.

STEP interprets the final exit from RUNNER and, after completing the load module, fetch places the MODIFYs and SNAPs in the appropriate locations in the user program as indicated in the RUNNER prepared tables. The user's program is then placed in execution.

When SNAPs, IFs, COUNTs, etc., are executed, the CAL receiver associates the DEBUG overlay which provides the dumps and other required operations.

On final exit from the user's program, if either the flag indicating idD presence is set, or if the program exits with an error or abort indication, then STEP associates the DEBUG overlay. The TELUSER portion of this overlay processes error and abort codes into messages and appropriate dumps, while the PMD portion processes PMDs from the idD file, provides the indicated dumps, and releases the idD file.

Return to STEP is made for the remainder of the job step shutdown procedure.

MONITOR PHYSICAL STRUCTURE

This section summarizes the UTS monitor by listing and functionally noting each of the system modules. The modules are summarized in six functional categories, then each category is detailed, module by module, as to function and size. Finally, the utility processors (as distinct from language processors, which are delivered with the system) are listed by function and size. Sizes and exact module content are approximate only; they are accurate for a particular version of UTS. The gross size of the system can also be estimated from the size of the compressed source files (280 files totaling 2400 granules) and from the size of a typical :SYS account (175 files totaling 3100 granules), although this later value is highly dependent on individual installation desires.

Modules are grouped by place of residence in four categories:

1. MONITOR ROOT - These routines are loaded together, enter the machine at system boot time, and are never replaced except during recovery.

2. VIRTUAL OVERLAY - These groups of routines are required to perform specific user serivces. They are loaded with the REF/DEF stack of the monitor root and communicate directly with it. They run in master mode but are mapped. They act as map reentrant shared processors - only one copy is required for all users. More than one overlay may be physically resident in the CPU if appropriate in light of cumulative user size and processor association.

3. PHYSICAL OVERLAY - Three kinds are used: a) monitor initialization code booted with the root but where space is reclaimed after startup; b) space is physically reserved permanently for execution of DELTA if that debugger is selected at boot time; and c) the recovery routines are loaded over code of the root monitor.

4. PROCESSOR - The utility routines of UTS are mostly user-style programs running in slave mode and mapped. Some of the programs are shared processors and others are ordinary unshared ones. Two exceptions are the initialization program, GHOST1, which runs in master mode in order to patch the monitor and to establish shared processors on the swapper with direct execution of I/O commands, and the granule allocation program, ALLOCAT.

Root size is summarized in Table BE-1.

Table sizes are detailed in Table BE-2.

Typical size of modules in loading order is given in Table BE-3.

Differences between a large and a minimum monitor are given in Table BE-4.

The major SYSGEN parameters which control root size are given in Table BE-5.

Table BE-1 - UTS Root Size

Code		
BASICS	6900	
System Management	6100	
Symbionts and COOP	1700	
System Services	400	
User Services	5300	
	20,400	
Tables		
Fixed Size	1400	
Variable Size	8000	Large 128 user system (small system = 2800) in variable tables; a difference of 5200)
TOTAL	29,800	

Table BE-2 - UTS-C01 Resident Tables

NAME	INITIAL DEF	DECIMAL SIZE	SYSGEN COMMAND	DESCRIPTION
Fixed Size Tables and Assembly Parameterized Tables				
SSDAT		598	--	Ghost job tables, swapper skeleton command list and disc address, swapper page pools, swap scheduler tables
PPP	PPP	4		Physical page, pool data
PMDAT	PMDAT	218		Performance monitor counters and distribution
HGPSTK	BUFSPD	424		Granule allocation stacks, pointers, comm. buffers, etc.
CFUD	CFUD	6		Parameters definitions for Packs and RADs (sectors/track, etc.)
M:OLIMIT	SL:OTIME	14	:OLIMIT	Default limits (print, punch, time, core, etc.) for on-line
M:ELIMIT	SL:ETIME	14	:ELIMIT	Limits (print, punch, time, core, etc.) for exit control
M:BLIMIT	SL:BTIME	14	:BLIMIT	Default limits (print, punch, time, core, etc.) for batch
COMBAT	GI:SDA	74		Contains GETI tables and RBBAT symbiont and MBS communication buffers.
		1350		
SYSGEN-Generated Variable Size Tables				
M:COC	COD:LPC	1100	:COC	Terminal I/O control tables and buffers
M:SPROCS	P:NAME	550	:SPROCS	Shared processor control tables
M:IMC	S:CUAIS	650	:IMC	System control parameters, user tables for scheduling, etc.
M:CPU	MPOOL	4370	:UTM	MPOOL, CPOOL, IOQ Tables, CFUs, PPUT, Sigma 9 PSDs
IOTABLE	IOTABLE	1150	:CHAN,:DEV	DCT, CIT, OPLABEL, TPMEN, AVR Tables, Remote Batch Control Tables, Swapper configuration definitions, HGP skeletons, private HGPs
M:SDEV	SSTAT	34	:SDEV	Symbiont control tables
M:PART	PL:LK	138	:PART	Multibatch partition control tables
		8000		

Table BE-3 - Typical Contents of UTS-D00 In Loading Order

Name	Decimal Size	Name	Decimal Size	Name	Decimal Size
Begin	100	CRDOUT	90	OUTSYM	361
SSDAT	598	PLOT	14	INSYM	212
PPP	4	SKD	728	SUSPTERM	24
PMDAT	218	7TAP	102	SYMSUBR	50
SLIMS	0	DPACK	140	IORT	757
COCD	48	COC	1982	RDF	2345
COCI	76	TSIO	432	WRTF	1158
Tables	623	ANSTP	256	WRTD	622
M:COC	1100	S9TRAP	170	PFSR	73
M:SPROCS	550	2741 Tables	384	INITIAL	246
M:IMC	650	ERHNDLR	394	JIT	512
M:OLIMIT	14	FBCD	21	BOOTSUBR	964
M:BLIMIT	14	SSS	2388		
M:ELIMIT	14	STEP	1906		
REQDC	64	MM	1018		
CFUD	6	CALPROC	203		
RECORD	2	ALTCP	542		
CHK	28	PM	264		
M:CPU	4370	T:OV	214		
M:BIG9	0	IOQ	1346		
IOTABLE	1150	ENTRY	202		
M:SDEV	34	BUFF	58		
COMBAT	78	GRAN	260		
M:PART	138	GRANSUB	263		
HGPSTK	424	ADD	86		
INITRCUR	100	SUB	19		
GPHGP	18	AVR	160		
CLOCK4	155	COOP	312		
ACCT	52	SUPCLS	296		
Handlers	419	SACT	163		

Modules Contain Tables, Write Permitted ↔ Write-Protected Code

Resident Only ↔ Initialization Only

Table BE-4 - Differences Between a Large and Minimum Resident Monitor

Large Resident	29,800 Words
Minimum Resident	23,500
	6,300
COC Without 2741, etc.*	800
Handlers: DISK, ANSTP	400
COC Tables & Buffers 128 Lines	1,100
Symbiont Tables, CFUs Monitor Buffers, Patch	4,000
	6,300

*SYSGEN options will remove 384 words of 2741 translation tables from the monitor load. To recover code for 2741 handling, the COC module must be reassembled. A total of 760 locations may be saved in COC by eliminating 2741 code, page heading, logic, buffer checking, and performance monitor entries.

Table BE-5 - UTS-D00 Monitor

Size Increases due to SYSGEN Parameters

MODULE	FACTOR	SYSGEN KEYWORD
M:COC	4 words per buffer	BUFFERS
	5-3/4 words per line	LINES
M:SPROCs	9-1/2 words per shared processor entry. 10 if Disc Swapping or BIG9 10-1/2 if Disc Swapping and BIG9.	:SPROCs entries
M:IMC	7 words per user	MAXG+MAXB+MAXOL
	2-1/4 words per ghost job	MAXG
M:CPU	34 words per MPOOL	MPOOL
	8 words per IOQ	QUEUE
	19 words per CFU	CFU
	6-1/4 words per tape if ANS system	:DEVICE
	1 word per input symbiont file	INFILE
	1 word per output symbiont file	OUTFILE
	1-1/4*((AVGSER*16)+3+17 words	AVGSER
	1/4 word per physical page (1/2 word if BIG9)	CORE,(BIG9)
	18 words for Sigma 9 PSDs	SIG9,BIG9
	Patch Space	MPATCH
	2-1/4 words per RBT device	:DEVICE
IOTABLE	13-3/4 words per DCT	One per :DEVICE
	2 words per CIT	One per :CHAN
	3-1/2 words per tape and private pack (AVR)	One per PRIV + tape
	8 words per public HGP	One per pack or RAD
	20 words per private pack HGP	One per PRIV
	1 word per DCT+AVR IOCTQ	
	6-74-word CLIST per device	One per device: Punch - 74 SKD - 74 DP - 12 Other - 6-8
	2-1/2 words per RBT device	:DEVICE

M:SDEV	6-3/4 words per symbiont device	:SDEVICE names
M:PART	7-3/4 words per partition	Maximum n in PART,n
S9TRAPS	169 words for Sigma 9 traps	SIG9,BIG9

	Root	Virtual Monitor Overlay	Physical Monitor Overlay	Ghost Job or Processor
Basics				
Control & I/O	3300			
Device Handlers	1100			
Terminal I/O & Buffering	2500			
	6900			
System Management				
Scheduling & Swapping	2388			
Memory Management (Core & Files)	1589			680
Job Step Control	1906			
Monitor Overlay control + CHK	242			
Multibatch Scheduling Symb. File Handling and Remote Batch	1700			3800
	8825			
System Services				
Initialization	113		1200	10700
Operator Communications		1800		
Accounting & Performance Monitoring	300			
Recovery			7000	
System Debugging			4400	
	400			
User Services				
File Management	5300	11900		
Load & Link commands		800		
Batch Debugging commands		1700		1900
Other User Services		3600		
	5300			
Tables	9400			
TOTALS	29800	19800	12600	17100
Minimum Size	24,600			

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
(Basic Control)		3300		Trap & Interrupt Handlers; I/O Queueing
	ALTCP	542	886	Secondary CAL1 Processor; trap processing
	CALPROC	203	358	CAL receiver and distributor (direct for CAL1,1 CAL1,2)
	CLOCK4	155	323	Clock 3 handler (time of day, timed-events)
	TABLES	623	671	Constants, dates, error log routine & buffer, WD trap memory parity interrupt, file account directory index
	IOQ	1346	2028	Central I/O queueing and dispatching
	ENTRY	202	258	Central XPSD receivers; routines for traps and interrupts
	PFSR	73	121	Power fail-safe recovery
	S9TRAPS	170		Trap & interrupt handlers for the Sigma 9
(I/O Device Handlers)		1100		Device-specific I/O start & recovery routines
	HANDLERS	419	687	RAD, printer, card reader, 9-track tape, operator console
	PTAP	(143)	172	Paper tape handler (not teletype terminal top)
	PLOT	(14)		Plotter handler
	7TAP	(41)		Seven-track tape handler
	MTAP	(71)		Nine-track tape handler
	MAGTAP	(162)		Common mag tape routines
	CRDOUT	90	128	Card punch handler
	DPAK	140	296	Disk pack (7242) handler
	FBCD	44	54	Hollerith to EBCDIC (026 to 029)H conversion
	TSIO	432	.683	Swapper I/O routines
	DPSIO	(688)		Swapper I/O routines for disk pack swapping

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
(Terminal I/O Handler)		2500	2791	
	COC (1364)	1982	2378	Teletype terminal (7611) handler and buffering routines including 2741 code
	COCI	76	143	Initialization for 7611
	COCD	48	622	Data areas for terminal I/O, not generated by SYSGEN
		384		2741 Translation tables
(System Management)		5950	7945	Scheduling, swapping, memory management step control
	MM	1018	1899	Memory management - core & swap RAD pages
	BUF	58	146	Core buffer management
	GRAN	260	454	File & symbiont granule management
	GRANSUB	253	328	Granule management subroutines
	SSS	2388	3602	Scheduler for swap & execution; swapping
	STEP	1906	2482	Job step control - exits, program fetch, assign merge
	T:OV	214	430	Monitor overlay association
	CHK	28	248	System consistency checking
(Symbionts & Cooperatives)		1675		RAD buffered and queued I/O for printers and card equipment
	ADD	86	161	Move input information to JIT.
	COOP	312	554	Input cooperative & common routines for cooperatives
	INSYM	212	400	Input symbiont for card reader
	OUTSYM	361	571	Output symbiont for punch & printer; deletes symbiont files
	REQDC	64	142	Disc and core allocation for symbionts and cooperatives
	SACT	163	488	Start & restart requests for buffers or I/O
	SUSPTERM	24	35	Type suspended & terminated messages
	SUPCLS	296	437	Close output coop files; output coop routines
	SYMSUBR	50	109	Miscellaneous symbiont routines

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
(Initialization)		1325		Boot from tape or RAD; space reclaimed from root after root
	BOOTSUBR	964	956	Initializer & patch monitor portion of swap RAD - all space reclaimed
	INITIAL	246	285	Turns on system & initiates GHOST1
	INITRCVR	100	121	Initialization or recovery entry
	GPHGP	18	57	Read XDELTA
KEYIN (operator communications)		1850		Operator Command Processor, Virtual overlay
	DELPRI	52	110	Delete symbiont files & change priorities
	DISPLAY	507	465	Display key-ins
	IOREC	30	86	Device I/O recovery key-ins
	KEYN	1190	1716	All other key-ins
	KEYSUB	68	139	Symbiont command analyzer
(Other System Services)		300		System instrumentation, Root resident
	ACCT	52	88	CPU accounting
	PM	264	568	Performance monitoring
	RECORD	2	187	System event trace recorder & buffer
RECOVER		7050		Recover from crashes, physical monitor overlay
	CYCUSR	2560	1324	UTS-specific - process users
	RCVCTL	2750	590	Recovery control
	SYMFILS	660	523	Symbiont file recovery
	TSTHGP	1071	980	File system recovery
XDELTA		4400		Executive (monitor) debugger - dedicated physical, if used.
	DELSYMS SYMTAB	730	357	Symbol table for Exec DELTA
	XDLT XDELTA	3661	4808	Debugger

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
(File Management)		5350		File & tape routines, root resident
	AVR	166	304	Tape volume recognizer
	CFUD	6	53	RAD address & sector size definitions
	IORT	760	1175	Common routines for reads & writes; interprets FPTs
	ANSTP	256	350	Special handling of AVR for ANS tapes
	RDF	2345	3430	Read RAD files; common routines for file operations
	WRTD	622	919	Write device other than file or labeled tape
	WRTF	1158	1592	Write RAD files
LTAPE		4775		Labeled tape operations; virtual overlay
	ARDL	250	359	Read labeled tape reverse
	LBLT	1289	1669	Write labeled tape & general purpose labeled tape out routines
	RDL	243	430	Read labeled tape records
OPEN		3000		All open operations; virtual overlay
	OBSE	291	490	Open subroutines: scan FPT, check names, file security checks
	OPLO	213	363	Open labeled tape - output
	OPN	1610	2074	Open files & device DCBs except tape
	OPNL	887	1201	Open labeled tape - input
	OPNTP	12	56	Open free form mag tape
CLOSE		2860		Monitor overlay for all close operations
	CLS	1998	2721	Close DCBs
	DLT	867	1160	Delete records and files
MUL	MUL	1330		Monitor overlay which creates treed indices
	MUL	1046	1389	Create treed indices for keyed files.
	OBSE	291	490	Security checks, etc.
LDLNK	LNKTRC	850	1103	Load-and-link, load-and-transfer; virtual overlay

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
DEBUG		1700		Batch debug commands; virtual overlay
	DEBUGTV	11	28	Entry vector for debug overlay
	PMD	370	990	Postmortem dumper
	TELLUSR	500	664	Batch error message generator
	SNAP	250	430	Execution time routines for debug CALs
	DUMP	590	622	Core dump routine for SNAP, etc.
IODTYPR		1225		
	TYPR	808	1043	Tape mount and dismount, messages
	IOD	320	508	M:DEVICE & M:SETDCB CALs
MISOV		2360		
	UCAL	600	1000	UTS CALs
	TRAPC	150	280	Trap control CALs
	RDERLOC	140	190	Read and write special error log file
	T:DSMNT	200	280	Print tape dismount messages at logoff
	T:JOBENT	350	580	Symbiont file insertion CAL
	TFILE	100	145	Record temporary file name for release at end of job
	TIM	120	200	Time CALs
	POS	400	580	Positioning operations
	SEGLD	320	470	Load overlay for user program
(Data Tables)		9400		
	SSDAT	600	411	GJOB tables, swapper shell command lists, miscellaneous tables
	PPP	4	145	Physical page pool data
	PMDAT*	218	218	Performance monitoring buffers
	HGPSTK	424	51	Granule allocation stack, points, communication buffers, etc.
	CFUD	6	53	
	M:OLIMIT*	14	*	Default limits (print, punch, time, etc.) to on-line
	M:ELIMIT	14	*	Limits (print, punch, time, etc.) for exit control
	M:BLIMIT*	14	*	Default limits (print, punch, time, etc.) for batch
	COMBAT	74		Contains communication buffers for RBBAT
	M:COC*	1100	**	Terminal I/O command tables from :COC SYSGEN command

96

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
	M:SPROCS*	550	**	Shared processor control tables
	M:IMC*	650	**	Installation management system control & job scheduling (user) table parameters (:IMC card)
	M:CPU*	4370	*	I/O control tables, CFUs, core page tables, queues, MPOOLs, CPOOL, patch
	IOTABLE*	1150	*	I/O control tables from :CHAN, :DEVICE SYSGEN cards, HGP
	M:SDEV*	34	*	Symbiont control tables from :SDEV card
	M:PART*	138		Partition tables

*Data and Tables generated by SYSGEN. No compressed or source corresponds to the ROMs.

**Size depends on SYSGEN parameters - this example is a 45-user system for the Xerox Data Center which is approximately typical.

(function) LM Name	ROM Compressed	D00 Size	No. Lines	Description
GHOST1		10675		System initialization - tape boot or recovery. A master mode user.
	BITOTM	220	178	Move modules from boot tape to file RAD
	CCIO	1180	1414	Reads PASS0 control cards
	CLS1	273	403	Character scan routines
	GHOST1D	465	228	Ghost 1 driver
	MODIFY	523	748	Subroutines for GENMODs
	PHASE A	362	482	Process GENCHN, GENOP, GENDCB
	PHASE B	496	770	Process GENMODs, GENDICTs - builds tables
	PHASE C	328	990	Executes changes as dictated by PHASE B
	PODCBS	296	84	DCBs for GHOST1
	RECOVER2	1360	1331	Restore systems data saved by RECOVER
	SYSMAK	860	1033	Initialize swap RAD with shared processors
	ACCTSUM	1760	1850	Produce accounting for jobs shut down during recovery
	MAILBOX	180	166	Recovery messages to users
	HGPRECON	3180	3090	Rebuild HGP tables for recovery
	RCVRIO	413	510	I/O routines for Rrecovery
ALLOCAT		680		Ghost jobs which allocate RAD and pack space
	ALYHD	8		Granule counters, master account directory pointer
	M:HGP*			The HGP maps for granule allocation
	ALLYTL	66	9	List of AD granules & first name in each
	GRANSUB	253	328	Granule allocation routines
	ALLYCAT	352	460	Control module - comm. buffers, stack-adjusting, counting
		3810		
RBBAT	RBBATM	3085	2955	Symbiont file control
	MBS	370	481	Multi-batch scheduling
	RBBATR	350	394	Symbiont file recovery

*Data and Tables generated by SYSGEN. No compressed or source corresponds to the ROMs.

UTS UTILITY PROCESSORS

PROCESSOR LMN	APPROX. TOTAL SIZE	DESCRIPTION
ANALZ	4254	Crash Dump Analyzer
BATCH	869	Terminal Batch Job Entry
CCI	7177	Batch Control Command Interpreter
CONTROL	3546	Installation Management Displays and Controls
DEF	3961	System Tapewriter for SYSGEN
DEFCOM	200	Extracts REF/DEFs from LM
DELTA	3810	User Debugging Language
DRSP	3700	Dynamic Replacer of Shared Processors
EASY		GE Mark II Command Processor
EDCON	2642	Compressed Deck to Edit File
EDIT	4288	Editor for Symbolic Files
ERR:FIL	1817	Hardware Error Logging
ERR:LIST	3451	Hardware Error Log List
ERR:SUM	1331	Hardware Error Summaries
ERRMWR	230	Centralized Error Message Filewriter
FILL	3496	File Save, Restore, and Auto PURGE
FPURGE	3207	File Save/Restore Program
LABEL		Prelabels ANS tapes
LINK	3798	On-line/On-pass Loader
LOAD	8138	Overlay Program Loader (Link-Editor)
LOCCT	809	Loader Command Tablewriter
LOGON	2570	Job/User Logon/Logoff Control
MEDDUMP	12700	Pack and RAD Surface (Cylinder-by-Cylinder Dump)
PASS2	11647	SYSGEN Monitor Table Compiler
PASS3	2468	SYSGEN Loader Runner
PCL	3121	File & Device Copying Utility
PFIL	1800	Position File Control Command
RATES	516	Charge Rate Table Creator
REW	1800	Rewind Control Command
RUNNER	1900	Debug Command Preprocessor
SUPER	2350	User Authorization File Maintenance
SUMMARY	21800	Performance Monitor History File Processor
SYMCON	1144	LM REF/DEF Stack Manipulator
TEL	3767	Terminal Executive Language
UTSPM	9600	Performance Monitor
WEOF	1800	Write-End-of-File Control Command

BPM MODULES	UTS EQUIVALENT	DESCRIPTION
LDPRGM EXIT PRGMLDR	STEP	Load and Execute Programs MXXX, M:ERR, M:EXIT Load Programs
MEMALOC	MM	Core & Swap RAD Management
LNKRT LNKLDTRC LNKIO	LNKTRC	Load & Link CALs
CHKPT CHPTDCBM	None (TEL - SAVE/GET	Checkpoint
REC CNTC REC COOP REC FILE REC BTM	RECOVER	System Recovery
MONSEGLD	T:OV	Monitor Overlay Control
COOPRES COPNRES SYMCR SYMPPRTY SYMCOM	COOP SUPCLS INSYM OUTSYM KEYSUB	Symbionts & Cooperatives

UTS PROCESSORS

The UTS Operating System consists of a monitor and a number of associated processors (Figure BF-1). The monitor provides overall supervision of program processing and the associated processors provide specific functions, such as compilation, execution, and debugging.

Processors operate in slave mode and thus request all I/O and other master mode services through monitor CALs like an ordinary program. CCI, TEL, and LOGON have store access to JIT in order that they may update accounting and other information stored there. These programs (command processors) also have a special interpretation applied to their EXIT CALs to provide the mechanism for calling other programs or processors into service. Special EXIT interpretation also applies to LINK to provide the load-and-go facility of the RUN command.

All processors are independent loads except those that use JIT which are loaded with the JIT definitions. Many shared processors are single assemblies. Exceptions are CCI, PCL, and the Public Libraries which consist of many assemblies. Further, processors may be shared - that is, a single copy is established at system boot time in absolute form on the swapping RAD and then shared by all concurrent users. An ordinary shared processor may have a single level overlay structure; that overlay is also shared among all concurrent users. Processors may be special - that is, they reside in the highest 16K of virtual memory. This is because the user's program already occupies or may soon occupy the remainder of virtual memory.

Public Libraries, DELTA (the on-line debugging language interpreter), LINK (the on-line Loader), RUNNER (the batch debugging language preparation program), and TEL (the on-line executive language interpreter) reside in the special shared processor area.

Processors may require that the user have a certain privilege level in order to run. Examples are CONTROL, DRSP, ERR:SUM, ERR:LIST, RATES, and SUPER.

Five kinds of shared processors may be associated with a given user at one time: 1) an ordinary shared processor, 2) the ordinary processor's overlay, 3) a monitor overlay, 4) a public library, and 5) a debugger (DELTA is the only current possibility). TEL may be associated and used without forgetting the other processor associations. DELTA and Public Libraries may be used by the same program but breakpoints may not be set in the library nor can DELTA make use of the library subroutines.

Processors

Processors are illustrated in Figure BF-1 at two levels. The upper level contains executive language and related processors, and the lower level, all other processors. These processors are defined in the following paragraphs.

Executive Language Processors

The three processors in this group are: LOGON, TEL, and CCI. The first two of these processors are available to on-line users only and the last to batch users only. It is also possible to implement other command processors, such as UTS-EASY.

LOGON

LOGON admits on-line users to the system and connects the user's terminal either to TEL or to an alternative processor, such as BASIC that has been selected by the user. User authorization is established by reading the file USERS for a record keyed by the concatenation of the LOGON account and name. LOGON also disconnects a user from the system and does the final cleanup and accounting (reference: Section PC).

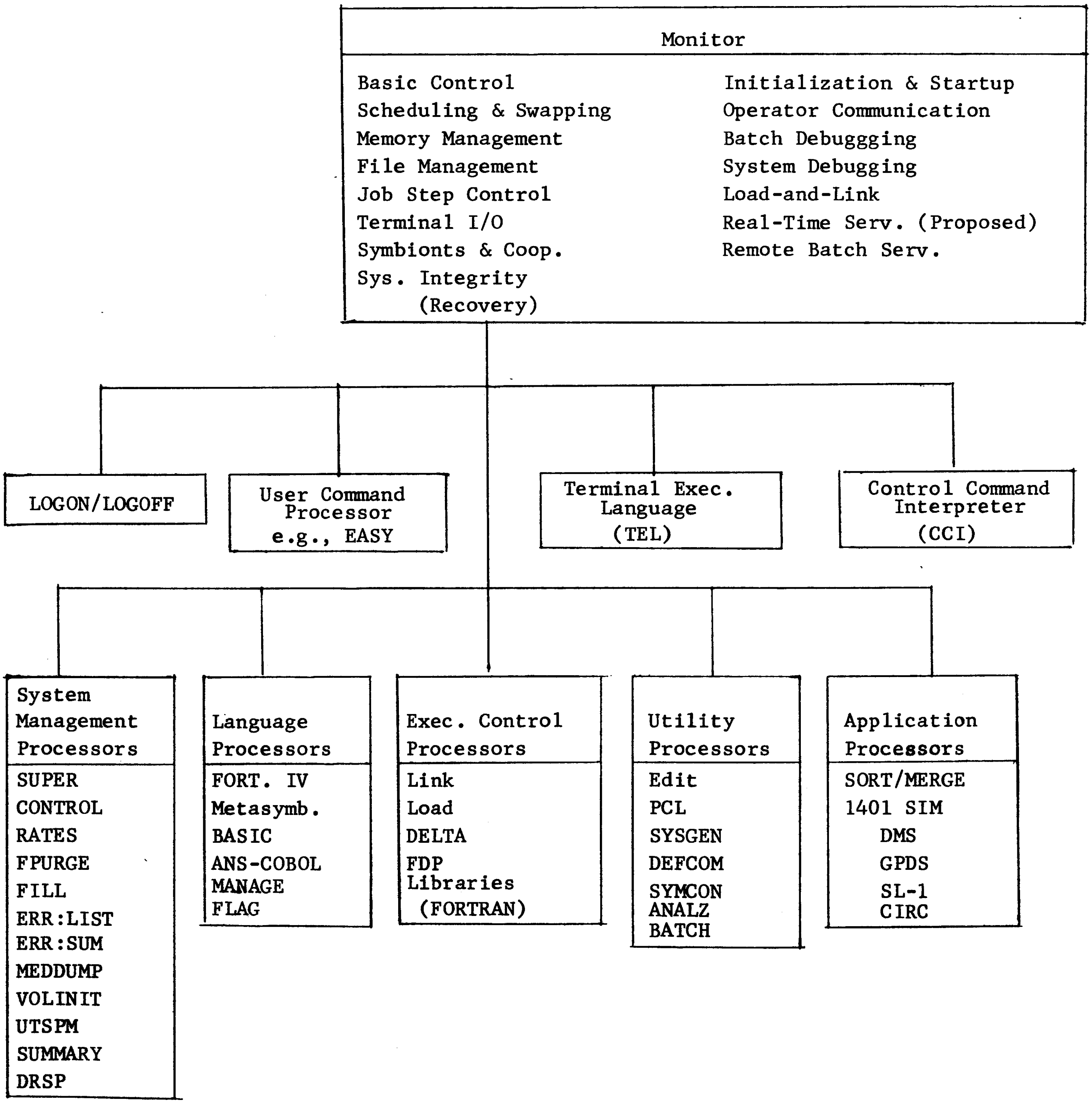
Terminal Executive Language

The Terminal Executive Language (TEL) is the principal terminal language for UTS. Most activities associated with FORTRAN and assembly language programming can be carried out directly in TEL. These include such major operations as composing programs and other bodies of text, compiling and assembling programs, linking object programs, initiating execution, and debugging programs. They also include such minor operations as checkpointing on-line sessions, determining current user charge status, and setting simulated tab stops (reference: Sigma 7 UTS/TS Reference Manual, Publication No. 90 09 07).

Control Card Interpreter

The Control Card Interpreter is the batch counterpart of TEL. It provides the batch user with control over the processing of batch programs just as TEL provides on-line users with control over the processing of on-line programs. Authorization for batch jobs is obtained by reading the USERS file and final job exit is through LOGOFF (LOGON).

Figure BF-1 - UTS Logical Structure



System Management Processors

System management processors furnish the manager of a UTS installation with on-line control of the system. Eight system management processors are supplied: SUPER, CONTROL, RATES, DRSP, FPURGE, FILL, ERR:LIST, and ERR:SUM.

SUPER

SUPER gives the installation manager control over the entry of users and the privileges extended to users. Through the user of SUPER commands, the installation manager may add and delete users, specify how many central site magnetic tape units a user will have. He may also grant certain users, such as system programmers, special privileges, e.g., examining, accessing, and changing the monitor. All commands result in creation or modification of the file USERS in account :SYS.

CONTROL

The CONTROL processor provides control over system performance. UTS has a number of performance measurements built directly into the system. Commands of the CONTROL processor enable the installation manager to display these measurements and to "tune" the system as needed by setting new values for the parameters that control system performance.

RATES

The RATES processor allows the installation manager to set relative charge weights on the utilization of system services. Specific items to which charge weights may be assigned include the following:

1. CPU time
2. CPU time multiplied by core size
3. Terminal interactions
4. I/O CALs
5. Console minutes
6. Tapes and packs mounted
7. Page-date storage
8. Peripheral I/O cards plus pages

FPURGE

The FPURGE processor allows the installation manager, through the computer operator, to purge unwanted files from the system. Specifically, FPURGE provides for:

1. Purging (releasing all unwanted user files from RAD storage.
2. Loading (restoring) RAD storage with files that were created and saved under the Batch Time-Sharing Monitor (BTM), or under UTS.
3. Printing (on the line printer) the names of all files on RAD storage by account number.

(Reference: Sigma 7 UTS/OPS Reference Manual, Publication No. 90 16 75)

FILL

The FILL program executes as a ghost program to provide for the safety of file information. This program writes backup copies of files on a system-owned magnetic tape. In addition, a facility is provided for the automatic deletion of expired files and a semi-automatic (operator-initiated) purge of inactive files in the event of a critical shortage of available file storage.

The FILL ghost is scheduled by a file called BACK:SCHED in account :SYS. This file may be created or modified during system operation to suit the requirements of individual installations. If the schedule is not frequent enough for some users, the user may employ terminal command !BACKUP to request that a specific file be added to the current backup tape.

The backup schedule specifies the frequency of three types of backup which are necessary to keep the physical amount of tape at a minimum to speed recovery while holding loss of filed data to a minimum.

The three types of backup in ascending frequency of operation are as follows:

1. SAVEALL - Saves all files currently known to the system.

   This provides a starting point for recovery (FILL) and allows the release of all previous backup tapes.

2. INCREMENTAL - Saves all files that have been created or modified since the last INCREMENTAL (or SAVEALL, whichever is later). During a recovery or initial load, these tapes are processed by FILL after the SAVEALL tape has been processed.

3. SQUIRREL - Saves all files that have been created or modified since the last backup of any tape. These tapes provide for a minimal loss of data but occupy a large volume of tape; they are therefore replaced periodically by the INCREMENTAL tapes.

In case of a catastrophic failure during which the information on the RAD is lost, recovery routines instruct the operator to request execution of FILL. The FILL program reads the various sets of backup tapes in sequence by date/time and thereby restores the backed-up files to the latest version available.

ERR:LIST and ERR:SUM

All hardware malfunctions occurring during UTS operation, whether recovered or not, are recorded in a special RAD storage file which is periodically copied into two standard UTS files (ERRFILE and SUMFILE) by a ghost program (ERR:FIL) that is initiated automatically for that purpose. The resulting files may be listed and summarized by the two programs, ERR:LIST and ERR:SUM. These files are also available for on-line preventive maintenance of the system and for diagnosis and prediction of hardware malfunctions.

The ERR:LIST program examines the error file (ERRFILE) for malfunction records that were written during the specified time period and produces a formatted listing of these records with (optionally) a summary of the records for that period. The formatted listing is complete with headings and formatting necessary for easy reading and use by field personnel.

ERR:SUM produces a complete one-page summary of errors accumulated in the error file.

Language Processors

Language processors translate high-level source code into machine object code. Five processors are of special importance (XDS Extended FORTRAN IV, Meta-Symbol, MANAGE, ANS COBOL, and BASIC) and can be used in both on-line and batch mode.

Execution Control Processors

Processors in this group control the execution of object programs. Two of the processors (LINK and DELTA) can be used in on-line mode only. Load can be used in batch mode only. The FORTRAN Debugging Package (FDP) can be used in either batch or on-line mode.

LINK

LINK is a one-pass Linking Loader that constructs a single entity called a load module which is an executable program formed from relocatable object modules (ROMs). LINK is designed to make full use of mapping hardware. It is not an Overlay Loader. If the need for an Overlay Loader exists, the Overlay Loader (LOAD) must be called by entering the job in the batch stream (reference: UTS/BP Reference Manual, Publication No. 90 17 64).

LOAD

LOAD is a two-pass Overlay Loader. The first pass processes:

1. all relocatable object modules (ROMs).

2. the protection types and sizes for the control and dummy sections of the ROMs.

3. defining expressions for definition and references (primary, secondary, and forward references).

4. loads from libraries as requested.

The second pass forms the actual core image and its relocation dictionary, and produces the executable program in Load Module (LM) form.

DELTA

DELTA is designed to aid in the debugging of programs of the assembly language or machine language levels. It operates on object programs and tables of internal and global symbols used by the programs but does not require that the tables be at hand. With or without the symbol tables, DELTA recognizes computer instruction mnemonic codes and can assemble machine language programs on an instruction-by-instruction basis. The main purpose of DELTA, however, is to facilitate the activities of debugging by:

1 examining, inserting, and modifying such program elements as instructions, numeric values, and coded information (i.e., data in all its representations and formats).

2. controlling execution, including the insertion of breakpoints into a program and requests for breaks on changes in elements of data.

3. tracing execution by displaying information at designated points in a program.

4. searching programs and data for specific elements and subelements.

Although DELTA is specifically tailored to machine language programs, it may be used to debug FORTRAN, COBOL, or any other program. DELTA is designed and interfaced to UTS in such a way that it may be called in to aid debugging at any time, even after a program has been loaded and execution has begun (reference: UTS/TS Reference Manual, Publication No. 90 09 07).

FORTRAN Debug Package

The FORTRAN Debug Package (FDP) is made up of special library routines that are called by XDS Extended FORTRAN IV object programs compiled in the debug mode. These routines interact with the program to detect, diagnose, and in many cases, repair program errors.

The debugger can be used in batch and on-line modes. An extensive set of debugging commands are available in both cases. In batch operation, the debugging commands are included in the source input and are used by the debugger during execution of the program. In on-line operations, the debugging commands are entered through the terminal keyboard when requested by the debugger. Such requests are made when execution starts, stops, or restarts. The debugger normally has control of such stops.

In addition to the debugging commands, the debugger has a few automatic debugging features. One of these features is the automatic comparison of standard calling and receiving sequence arguments for type compatitility. When applicable, the number of arguments in the standard calling sequence is checked for equality with the receiving sequence. These calling and receiving arguments are also tested for protection conflicts. Another automatic feature is the testing of subprogram dummy storage instructions to determine if they violate the protection of the calling argument (reference: Sigma 7 FORTRAN Debugger Reference Manual, Publication No. 90 16 77).

Utility Processors

The processors in this group perform such functions as editing, sorting, and transferring data between RAD storage and central site peripheral devices. One of the processors (EDIT) can be used in the on-line mode only. Three processors (PCL, SYMCON, and ANALYZ) can be used in both batch and on-line mode. The remaining processors can be used in batch mode only.

EDIT

The EDIT processor is a context editor designed for on-line creation, modification, and handling of programs and other bodies of information. All EDIT data is stored on RAD storage in a keyed file structure of sequence number variable length records. This structure permits EDIT to dirctly access each line or record of data.

EDIT functions are controlled through single line commands supplied by the user. The command langugage provides for insertion, deletion, reordering, and replacement of lines or groups of lines of text. It also provides for selective printing, renumbering records, and context editing operations of matching, moving, and substituting line-by-line within a specified range of text lines. File maintenance commands are also provided to allow the user to build, copy, merge, and delete whole files (reference: UTS/TS Reference Manual, Publication No. 90 09 07).

Peripheral Conversion Language

The Peripheral Conversion Language (PCL) is a utility subsystem designed for operation in a batch or on-line environment under UTS. It provides for information movement among card and paper tape devices, line printers, Teletype terminals, magnetic tape devices, disk pack, and RAD storage.

PCL is controlled by single-line commands supplied through on-line terminal input or through command card input in the job stream. The command language provides for single or multiple file transfers with options for selecting, sequencing, formatting, and converting data records. Additional file maintenance and utility commands are provided (reference: UTS/TS Reference Manual, Publication No. 90 09 07).

SORT/MERGE

The XDS SORT/MERGE processor provides the user with a fast, highly efficient method of sequencing a nonordered file. SORT may be called as a subroutine from within a user's program or as a batch processing job by control cards. It is designed to operate efficiently in a minimum hardware environment. Sorting can take place on from one to sixteen keys; each individual key field may be sorted in ascending or descending sequence. The sorting technique used is that of replacement selection tournament and offers the user the flexibility of changing the blocking and logical record lengths in explicitly structured files to different values in the output file.

The principal highlights of SORT are as follows:

1. Sorting capability allows either magnetic tapes, RADs, or both.

2. Linkages allow execution of the user's own code.

3. Sorting on from one to sixteen key fields in ascending or descending sequence is allowed. Keys may be alphanumeric, binary, packed decimal, or zoned decimal data.

4. Records may be fixed or variable length.

5. Fixed length records may be blocked or unblocked.

6. RADs may be used as file input or output devices, or as intermediate storage devices.

7. SORT employs the read backward capability of the tape device to eliminate rewind time.

8. User-specified character collation sequence may be used.

9. Buffered input/output is used.

(Reference: Sigma 6/7 SORT/MERGE Reference Manual, Publication No. 90 11 99.)

1400 Series Simulator

The 1400 Series Simulator provides an economical and effective solution to the program conversion problem that arose because of a change in hardware. This interpretive program is designed to execute 1400 series object programs automatically as if they run on a 1401, 1460, or 1440. Thus, an existing level of computing capability can be maintained while new processing methods that take advantage of the new, more powerful Sigma equipment are designed and implemented.

The 1400 Series Simulator simulates object code produced by SPS, FORTRAN, Auto-coder, RPG, and utility routines. Almost all 1400 operations may be simulated except for I/O operations in which hardware differences make total simulation impossible. Full 1400 operator capabilities are provided (reference: Sigma 5/7 1400 Series Simulator Reference Manual, Publication No. 90 15 01).

SYSGEN

SYSGEN is made up of several processors that are used to generate a variety of UTS systems tailored to the specific requirements of an installation. The SYSGEN processors are: PASS2, LOCCT, PASS3, and DEF. PASS2 compiles the required dynamic tables for the resident monitor. generation. PASS2 compiles the required dynamic tables for the resident monitor. LOCCT and PASS3 respectively file away and execute load cards to produce load modules for the monitor and its processors. DEF writes a monitor system tape that may be booted and used (reference: Xerox Universal Time-Sharing System (UTS)/SM Reference Manual, Publication No. 90 16 74).

DEFCOM

DEFCOM makes the DEFs and their associated values in one load module available to another load module by using a load module as input and by producing another load module that contains only the DEFs and DEF values from the input modules. The resultant load modules of DEFs can be combined with other load modules. DEFCOM is used extensively in constructing the UTS monitor and the shared run-time libraries (reference: UTS/BP Reference Manual, Publication No. 90 17 64).

SYMCON

The Symbol Control Processor (SYMCON) provides a means of controlling external symbols in a load module. Its primary function is to give the programmer a means of preventing double definitions of external symbols, but it may also be used to reduce the number of external symbols. For example, if certain load modules cannot be combined because their control tables are too large, the size of the tables may be reduced by deleting all but essential external symbols (reference: UTS/BP Reference Manual, Publication No. 90 17 64).

ANALZ

ANALZ provides the system programmer with a means of examining and analyzing the contents of dumps taken prior to system recovery. It is called automatically by the system initializer following a recovery and is executed as a ghost job. It may also be called by the operator to analyze tape dumps when recovery is not possible, or by an on-line user to examine dumps or the currently running monitor.

ANALZ performs three major functions:

1. It runs a series of monitor integrity checks on the contents of a core dump to determine what caused the crash.
2. It provides formatted dumps of the monitor's tables at the time of recovery.
3. It permits, via commands, the examination of dumps and the examination and change of the monitor.

BATCH

The Terminal Batch Job Entry (BATCH) processor inserts the contents of a RAD file into the symbiont input job queue. After insertion, the user is notified of job ID and queue position relative to the currently executing job.

BATCH functions are controlled by a TEL or CCI command line in which the user has specified the FID(s) to be inserted.

The status of a previously inserted job may be checked via the JOB command in TEL. Batch is an ordinary shared processor consisting of a single assembly.

LABEL

LABEL processor tapes with ANS header sentinels and readies them in a protected shop so they may be AVRed.

DRSP

DRSP controls the addition, deletion, or replacement of shared processors, shared libraries, and monitor overlays during normal system operation. Current users of a replaced processor, library, or overlay continue to use the old copy while additional users are associated with the new version (reference: UTS/SM Reference Manual, Publication No. 90 16 74).

INDEX TO UTS TECHNICAL MANUALS

The following pages contain two indexes to the complete set of UTS technical manuals. The first is an index by item and the second is an index by module. The two indexes are preceded by a key that indicates the volume numbers in which the various section numbers are located.

KEY TO INDEXES

Sections Included in Volume	Volume Number	Title
B, BA, BB, BC, BD, BE, BF	90 19 84	Overview and Index
C, CA, CB, CC, CD D, DA, DB, DC	90 19 85	Basic Control and Basic I/O
E, EA, EB, EC, ED, EE F, FA, FB,	90 19 86	System and Memory Management
G, GA, GB, GC, GD	90 19 87	Symbiont and Job Management
H, HA I, IA, IB, IC, ID	90 19 88	Operator Communication and Monitor Services
J, JA, JB, JC, JD, JE, JF, JG, JH, JI, JJ, JK, JL, JM, JN, JO	90 19 89	File Management
K, KC, KD, KE, KF L, LA, LB, LD, LE, LF, LH W, WA, WB	90 19 90	Reliability and Maintainability
M	90 19 91	Interrupt Driven Tasks
N, NA, NB, NC, ND, NE, NG O, OA, OB, OC, OD, OE, OF OG, OH	90 19 92	Initialization and Recovery
P, PA, PB, PC	90 19 93	Command Processors
Q, QB, QC, QD, QE R, RA S, SC, SD, SE U, UB, UC, UD, UE, UF	90 19 94	System Processors
V, VA, VC, VD, VE, VF, VG, VH, VI, VK, VL, VM, VN, VO	90 19 95	Data Bases

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
BOOTSUBR	BOOTSUBR	NB	MONITOR BOOT SUBROUTINES
BOOTSUBR	OVERVIEW	BD	SYSTEM INITIALIZATION MODULE
BPM	BPM	UE	TO ASSEMBLE MONITOR SERVICE PROCEDURES
BPMBT	BPMBT	90 18 77	WRITE BPM/BTM BASE SYSTEM TO PO TAPE
BTM	SYSGEN	90 18 77	PROCESSES BTM (BPM ONLY)
BUFCHAIN	STEP	EB	CHAINS COOPERATIVE AND FILE BUFFERS
BUFGRAN	BUFGRAN	FA.03.02	SYSTEM BUFFER-GRANULE MANAGEMENT
BUFOUT	ANALZ	LF	WRITE BUFFER OUTPUT
BUILDCDT	SDEVICE	90 18 77	SET UP MASTER PLIST
C:NOPROC	PMDAT	VJ	# OF TIMES A PROCESSOR WASN'T IN CORE
C:PROCREQ	PMDAT	VJ	NUMBER OF TIMES PROCESSOR REQUIRED
CAL	OVERVIEW	BD	USER REQUEST FOR MONITOR SERVICE
CALPROC	CALPROC	CB	DECODE CALS 1,2
CASSIGN	JIT	VA	SEE J:CASSIN
CBINT	SYMCON	SE	INTERPRET EXPR. STACK CONTROL BYTES
CC PLISTS	SYSGEN	90 18 77	SYSGEN CONTROL COMMAND SCAN PLISTS
CCA	DEFROM	90 18 77	SAME AS CCE
CCBEF	JIT	VA	BIT 8 SET SAYS CNTL. CMD.
CCE	DEFROM	90 18 77	WRITE OUT PO TAPE
CCI	CCI	PA	CONTROL CARD INTERPRETER
CCI	OVERVIEW	BF	CONTROL CARD INTERPRETER
CCIR	CCI	PA	CCI, EXECUTIVE ROUTINE
CCITABLS	CCI	PA	DATA TABLE MODULE
CCI2	CCI2	ND	PASS2 CONTROL CARD PROCESSING
CCLFLAGS	JIT	VA	
CCLOAD	PASS2CCI	90 18 77	LOAD PASS2 PROCESSORS
CCLTFLGS	JIT	VA	
CCL0	SSS	ED.01	ROUTINE TO ORDER CL AFTER OSAC
CDPO	JIT	VA	BITS 0-14 ARE CURRENT DEBUG PAGES OUT
CEXT	JIT	VA	CURRENT EXECUTION TIME
CHANNEL	UBCHAN	90 18 77	SET UP CHANTOL ENTRY FOR :CHAN COMMAND
CHARACTERISTC	OVERVIEW	BA	SALIENT CHARACTERISTICS OF UTS
CHARNX	SYMCON	SE	SCAN INPUT COMMANDS
CHARROUT	CCI	PA	SYNTAX ANALYSIS SUBROUTINES

120

FØR ITEM	IN MØDULE	SEE SECTIØN	CØMMENT
CHARSCAN	SYSGEN	90 18 77	SEARCH FØR DELIMITER
CHECK	CHK	KC	USER CØNSISTENCY CHECK
CHECKPØINT	UCAL	IA	ØN LINE USER CHECKPØINT FACILITY
CHEKNAME	ABS	90 18 77	CHECK PRØCESSØR NAME
CHK	CHK	KC	SYSTEM CØNSISTENCY CHECK RØUTINE
CHKDØ	STEP	EB	CHECKS VALID PRØCESSØR ASSØCIATIØN
CHKDCBN	TEL	PB.03	VALIDATE DCB NAME
CHKNAM	PASSØRAM	90 18 77	DETERMINE PØSSIBLE DELETIØN ØF FILE
CHSTSCAN	SYSGEN	90 18 77	GET NEXT STRING
CIC	JIT	VG.02	BITS 0-15 ARE CARD INPUT CØUNTER
CIT1	IØTABLE	VG.02	BYTE, QUEUE CHAIN HEAD
CIT2	IØTABLE	VG.02	BYTE, QUEUE CHAIN TAIL
CIT3	IØTABLE	VG.02	BYTE,BIT 0 SET IMPLIES CHANNEL BUSY
CIT4	IØTABLE	VG.02	WØRD, 0 ØR Q RØUTINE FØR THIS CHANNEL
CJØB	TABLES	VB.03	CURRENT USER JIT ADDR AND PRIØRITY
CKRAD	RCVCTL	KB.05.02	VALIDITY CHECK ØF DISC ADDRESSES
CL	SSS	ED.01	SWAPPER CØMMAND LIST
CLEANUP	IØQ	DA.01	PERFØRM PØST-INTERRUPT PRØCESSING
CLØBBER TEL	RUNRAM	LB.01	CØNTAINS LØC & VALUE FØR DEBUGS
CLØCK 3 INT	CLØCK4	CD	CLØCK 3 INTERRUPT PRØCESSØR
CLØCK4	CLØCK4	CD	CLØCK 3 INTERRUPT PRØCESSØR
CLSFILS	TSTUGP	KB.02.07	CLØSE FILES
CLS1	CLS1	ND	CHARACTER SCAN RØUTINES FØR PASS0
CNST	JIT	VA	CURRENT NUMBER ØF SCRATCH TAPES
CNVDEC	FRGD	90 18 77	CØNVERT EBCDIC TØ HEXADECIMAL
CNVHEX	FRGD	90 18 77	CØNVERT EBCDIC HEXADECIMAL TØ HEXADECI
CØC	CØC	DC	CØC HANDLER
CØCBP	CØC	VG.05	BYTE ADDR ØF NEXT INPUT CHAR BY LN #
CØCCRLF	CØC	DC.01.04	PUT CARRIAGE RETURN AND LINE FEED IN B
CØCD	CØCD	DC	TABLES FØR CØC HANDLER
CØCGETP	CØC	DC.01.04	GET I/Ø BUFFER FRØM CØC BUFFER PØØL
CØCHC	CØC	DC.01.04	DETECT AND REPØRT HANG-UP AND DIAL-UP
CØCI	CØCI	DC.01.04	INITIALIZATIØN ØF 7611
CØCICP	CØC	DC.01.04	MAINTAIN VALUE ØF CARRIER PØSITIØN

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
COCINIT	COC	DC.01.04	INITIALIZE COC
COCINP	INITIAL	NA	COC TABLE INITIALIZATION FROM BC
COCIP	COC	DC.01.02	PERFORM COC INPUT CHARACTER PROCESSING
COCMINT	COC	DC.01.04	INITIALIZE LINE MODE CONTROL BYTES
COCMU	COC	DC.01.01	MOVE INPUT MESSAGE FROM COC TO USER BU
COCNBUF	COC	DC.01.04	REPORT CAN'T-FIND-BUFFER EVENT
COCOC	COC	VG.05	# OF OUT BYTES LEFT BY LINE #
COCODE	ANALZ	LE.01	DISPLAY COC TABLES
COCOFF	COC	DC.01.04	INITIALIZE LINE FOR LOGGING OFF
COCOP	COC	DC.01.03	PERFORM COC OUTPUT INTERRUPT PROCESSIN
COCPCIB	COC	DC.01.04	PUT OUTPUT CHARACTER IN COC BUFFER
COCPUTBL	COC	DC.01.04	PUT I/O BUFFER IN FREE COC BUFFER CHAI
COCRD	COC	DC.01.01	INITIATE PROCESSING OF TERMINAL READ R
COCRIC	COC	DC.01.04	REPORT EVENT TO SCHEDULER
COCRICXU	COC	DC.01.04	RECORD INPUT COMPLETE
COCSCIB	COC	DC.01.04	STORE INPUT CHARACTER IN INPUT BUFFER
COCSO	COC	DC.01.04	START COC OUTPUT OPERATIONS
COCTERM	COC	VG.05	BYTE,TERMINAL TYPE BY LINE #
COCWR	COC	DC.01.01	INITIATE PROCESSING OF TERMINAL WRITE
CODE	COCD	VG.05	BYTE, SEE COCOC, INPUT EOM CONDITIONS
COMBINE	PCL	703027	CHECKS FOR VALID OPTION COMBINATIONS
COMLIST	BASHANDL	DA.02	BUILD COMMAND LIST
COMPARE	ANALZ	LE.01	COMPARE RUNNING MONITOR OR DUMP LOCATI
COMRET	FRGD	90 18 77	SET CONDITIONS FOR VALUE IN DECIMAL
COMRETA	FRGD	90 18 77	OBTAIN VALUE
CONTROL	CONTROL	QA	ON-LINE PERFORMANCE MONITOR AND CONTRO
CONTROL	OVERVIEW	BF	INSTALLATION MANAGER TOOL
CONV	LOCCTRAM	90 18 77	CONVERT ERROR/ABNORMAL CODE TO EBCDIC
CONV	PASS3RAM	90 18 77	CONVERT ERROR/ABNORMAL CODE TO EBCDIC
COOP	COOP	FA	INPUT/OUTPUT COOPERATIVES
COOP BUFFERS	COOP	FA	BUFFERS IN USER VIRTUAL MEMORY
COOPERATIVES	OVERVIEW	BD	USER INTERFACE WITH PERIPHERAL I/O
COOPERATIVES	SYMB/COO	FA	USER LEVEL PACK & UNPACK PERIPH. I/O
COOPFILS	SYMFILS	KF.04.05	CLOSE CO-OP FILES ASSOCIATED WITH JIT

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
DCT3	IOTABLE	VG.01	BYTE, LEGAL OPERATIONS BY DCT INDEX
DCT4	IOTABLE	VG.01	BYTE, TYPMNE INDEX BY DCT INDEX
DCT5	IOTABLE	VG.01	BYTE, SWITCHES BY DCT INDEX
DCT6	IOTABLE	VG.01	BYTE, QUEUE HEAD INDEX BY DCT INDEX
DCT7	IOTABLE	VG.01	HWORD, ADDRESS OF COMMAND LIST BY DCT
DCT8	IOTABLE	VG.01	WORD, ADDRESS OF PREPROCESSING ENTRY
DCT9	IOTABLE	VG.01	WORD, ADDR OF POST-PROCESSING ENTRY
DEBUG TABLE	RUNROM	LF.01	CONTAINS SYS CREATED DEBUG FPTS
DEBUGGING	OVERVIEW	BD	MONITOR AND BATCH TOOLS FOR
DEBUGR	CCI	PA	DEBUGGING COMMAND (SNAP,PMD) PROCESSOR
DEBUGTV	DEBUGTV	LP	TRANSFER VECTOR FOR DEBUG ROUTINES
DECBIN	TEL	PR.03	CONVERT EBCDIC TO BINARY
DECCNV	ABS	90 18 77	CONVERT DECIMAL SIZE TO HEX
DECSCAN	SYSGEN	90 18 77	GET DECIMAL STRING
DEF	SYSGEN	90 18 77	WRITES PO TAPES
DEFCOM	DEFCOM	SD	LOAD MODULE REF/DEF STACK EXTRACTION
DEFCOM	OVERVIEW	BF	PREPARE LIBRARIES- REMOVE CODE FROM LM
DEFRDCC	DEFROM	90 18 77	PROCESS NEXT CONTROL CARD
DEFX	SDEVICE	90 18 77	SET UP DEF PLIST FOR MODIFY ROUTINE
DELPRI	DELPRI	HA	DELETE FILES FROM SYMFILE AND DISC
DELTA	DELTA	LA	CONVERSATIONS PROGRAM DEBUGGING PROC.
DELTA	OVERVIEW	BF	ASSEMBLY LANGUAGE DEBUGGER
DELTA	MONFIX	LG.01.02	DELTA MAY BE USED TO DEBUG A BOOTFILE
DELTA INTERFC	DELTA	LA	DELTA INTERFACE WITH PROCESSORS
DELTAGET	ANALZ	LE.01	GET SUBROUTINE FOR DELTA
DELTAPUT	ANALZ	LE.01	PUT SUBROUTINE FOR DELTA
DEVTRAN	PCL	703027	CHECKS FOR VALID DEVICE ID CODE
DIAGNOSTIC OP	CLS	KD	OPEN SYMBIONT DEVICE FOR DIAGNOSTICS
DIAGNOSTIC OP	COOP	KD	OPEN SYMBIONT DEVICE FOR DIAGNOSTICS
DIAGNOSTIC OP	IOQ	KD	OPEN SYMBIONT DEVICE FOR DIAGNOSTICS
DIAGNOSTIC OP	OPN	KD	OPEN SYMBIONT DEVICE FOR DIAGNOSTICS
DISASSDEL	ANALZ	LE.01	DISASSOCIATE DELTA
DISCIO	BASHANDL	DA.03	RAD I/O HANDLER
DISPLAY	DISPLAY	HA	DISPLAY SPECIFIED MONITOR INFORMATION

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
E:ND	MM	GA.01	EVENT REPORTED BY MEMORY MANAGEMENT
E:ND	SSS	EA	EVENT 9, CANT GET REQUESTED DISC PAGES
E:NRD	SSS	EA	EVENT 15, REAL TIME JOB EXIT
E:OFF	SSS	EA	EVENT 11, USER HUNG UP
E:QA	SSS	EA	EVENT 13, Q UP FOR I/O DEVICE ACCESS
E:QE	SSS	EA	EVENT 7, QUANTUM END
E:SL	SSS	EA	EVENT 12, SLEEP TIME FOR USER
E:UQA	SSS	EA	EVENT 14, UN Q FOR I/O DEVICE ACCESS
E:WU	SSS	EA	EVENT 10, WAKE UP TIME FOR USER
EDCON	EDCON	NONE	BATCH PROCESSOR FOR EDIT FORMAT FILES
EDCON	OVERVIEW	BF	BATCH UTILITY FOR EDIT USERS
EDIT	EDIT	NONE	CONTEXT EDITOR
EDIT	OVERVIEW	BF	CONTEXT EDITOR
END ACTION	RDERLOG	IA	WARNING ON USE OF END ACTION
END ACTION	SYMB/COB	FA	END ACTION DRIVEN I/O ROUTINES
ENTRY	ENTRY	CA	ENTRY AND EXIT FOR PROCESSING CALS
EOCCSCAN	DEFROM	90 18 77	FIND END OF CURRENT CONTROL COMMAND
EOCCSCAN	PASS1ROM	90 18 77	SEARCH FOR END OF CONTROL COMMAND
EOCCSCAN	PASS3ROM	90 18 77	SEARCH FOR END OF CONTROL COMMAND
EOMTTME	COCD	VG.05	HW, TIME OF END OF MESSAGE BY LINE #
ERLFLAGS	JIT	VA	SEE J:CASSIN
ERO	JIT	VA	B15-31 IS ERROR OVERRIDE ADDR
ERR:FIL	ERR:FIL	KE.02	PROGRAM TO COPY ERRORLOG TO KEYED FILE
ERR:LIST	ERR:LIST	KE.05	ERROR LOG FORMATTING & LISTING PROGRAM
ERR:LIST	OVERVIEW	BF	LIST ERROR LOG
ERR:SUM	ERR:SUM	KE.03	ERROR LOG SUMMARY PROCESSOR
ERR:SUM	OVERVIEW	BF	LIST ERROR LOG
ERRCONTL	PASS1ROM	90 18 77	SPECIAL PASS1 ERROR ROUTINE
ERRDELIM	PASS1ROM	90 18 77	SPECIAL PASS1 ERROR ROUTINE
ERRLFLGS	JIT	VA	SEE J:CASSIN
ERRLOG	RDERLOG	IA	USER PROGRAM INTERFACE TO ERROR LOG
ERRLOG	TABLES	KE.01	ERROR LOGGING ROUTINE
ERRMSG	ERRMWR	UB	SYSTEM ERROR MESSAGE FILE
ERRMSGE	TEL	PB	ERROR MESSAGE FILE SUBROUTINE

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
ERRMWR	ERRMWR	UR	ERROR MESSAGE FILE CONTROL PROCESSOR
ERRNAME	PASS1ROM	90 18 77	SPECIAL PASS1 ERROR ROUTINE
ERROR	PCL	703027	RECORDS ERROR CONDITION
ERROR LOG	OVERVIEW	BD	RECORD DEVICE FAILURES
ERROR REPORT	MONFIX	LG.01.04	RECORD OF ALL ERRORS MAY BE REQUESTED
ERROUT	PASS1ROM	90 18 77	DISPLAY ERROR MESSAGE AND EXIT PASS1
ERRSEQ	PASS1ROM	90 18 77	SPECIAL PASS1 ERROR ROUTINE
EVENTS	OVERVIEW	BD	EVENTS RECEIVED BY SCHEDULER
EXITCL	ANALZ	LE.01	EXECUTE NORMAL EXIT TO MONITOR
EXITSYSW	PASS1ROM	90 18 77	EXIT SYSWRT
EXNEXT	SYMCON	SE	SET REGISTER TO EXPR. STACK ITEM
EXPAND	TEL	PR.03	EXPAND COMPACTED A/M TABLE ENTRY
EXPR	SYMCON	SE	STACK PRODUCED BY LOAD
EXPRX	SDEVICE	90 18 77	SET UP EXPR PLIST FOR MODIFY ROUTINE
FBCD	FBCD	NONE	FORTRAN BCD CONVERSION
FDP	OVERVIEW	BF	FORTRAN DEBUGGER
FETCH	STEP	ER	ASSOCIATE UNSHARED PROCESSOR ROUTINE
FETCH3	STEP	ER	REPORTS ABORT CODE A6 TO TEL
FID	CONVENTN	AR.01	FILE IDENTIFICATION-NAME.ACCT.PASSWORD
FID	TEL	PR.03	BREAK COMPLEX FID
FILE DIRECTRY	OVERVIEW	BC	CHAIN OF FILE NAMES AND FITS
FILENAME	ANALZ	LE.01	SET FID INTO ASSOCIATE PROCESSOR CAL
FILENM	PASS1ROM	90 18 77	PROCESS FILE OPTION
FILENT	TEL	PR.03	CREATE SHORT FORM P-LIST
FILL	FILL	KA.02	RESTORES USER'S FILES FROM BACKUP TAPE
FILL	OVERVIEW	BF	RESTORE FILES
FILTRAN	PCL	703027	SYNTAX ANALYZER FOR FILE IDENTIFIER
FINDEND	LOCCTROM	90 18 77	FIND END OF LOCCT TABLE
FINDENDX	LOCCTROM	90 18 77	CHECK FOR VALID ROM IN LOCCT TABLE
FINDEOC	ABS	90 18 77	SEARCH FOR CONTROL CARD END
FINDNAME	PASS1ROM	90 18 77	FIND SPECIFIED FILE
FINDROMX	LOCCTROM	90 18 77	OBTAIN NEXT ROM TABLE FROM TREE TABLE
FINDRPAR	ABS	90 18 77	SEARCH FOR RIGHT PARENTHESIS
FIT	OVERVIEW	BC	FILE INFORMATION TABLE-FILE ATTRIBUTES.

FØR ITEM	IN MØDULE	SEE SECTIØN	CØMMENT
FIXARG	PCL	703027	TABLE SEARCH SUBRØUTINE
FLAG	CØCD	VG.05	FLAGS CØNTRØL OUT CAUSED BY INPUT CHAR
FLAGS	:USERS	VN.01	USER'S PERIPHERAL DEVICE ACCESS
FLØP	TEL	PB.03	ØPEN INPUT FILE
FPMC	MM	GA	INDICATES VP IS AVAILABLE FØR USE
FPØØL	JIT	VA	SEE J:FPØØL
FPURGE	ØVERVIEW	BF	SAVE,RESTØRE,PURGE,LIST FILES .
FRGD	SYSGEN	90 18 77	PRØCESSES FRGD,INTLB
FRGDØP	FRGD	90 18 77	PRØCESS FRGD PARENTHETICAL EXPRESSIØNS
FX	JIT	VA	BITS 15-31 ARE FILE EXTENSIØN BITS
GENABS	PHASEC	ND	NØP
GENCHN	PHASEA	ND	PRØCESS PASSO GENCHNS
GENDCB	PHASEA	ND	PRØCESS PASSO GENDCBS
GENDEF	PASS3RØM	90 18 77	BUILD DEF PLIST FØR MØDIFY RØUTINE
GENDICT	PHASEB	ND	PRØCESS PASSO GENDICTS
GENDICT	PASS3RØM	90 18 77	BUILD DICT PLIST FØR MØDIFY RØUTINE
GENEXP	UBCHAN	90 18 77	SET UP IØTABLE EXPRESSIØN STACK
GENFILE	LØCCTRØM	90 18 77	GENERATE PERMANENT FILE FØR LØCCT TABL
GENHAN	PASS3RØM	90 18 77	GENERATE HANDLERS FILE FØR MIMØN LØAD
GENHANDL	PASS3RØM	90 18 77	GENERATE HANDLERS FILE
GENMD	PHASEB	ND	PRØCESS PASSO GENMDS
GENØP	PHASEA	ND	PRØCESS PASSO GENØPS
GENRØØT	PASS3RØM	90 18 77	GENERATE RØØT LØAD MØDULE
GENT0	FRGD	90 18 77	PRØCESS, CØDE=0, TYPE CØNTRØL TABLE EN
GENT1	FRGD	90 18 77	PRØCESS, CØDE=1, TYPE CØNTRØL TABLE EN
GENT2	FRGD	90 18 77	PRØCESS, CØDE=2, TYPE CØNTRØL TABLE EN
GENT3	FRGD	90 18 77	PRØCESS, CØDE=3 TYPE CØNTRØL TABLE EN
GENT4	FRGD	90 18 77	PRØCESS, CØDE=4 TYPE CØNTRØL TABLE EN
GET PAGE	ANALZ	LE	GET SPECIFIED PAGES FRØM DUMP FILE
GETADDR	ANALZ	LE	ØBTAIN DUMP PAGE CØNTAINING SPECIFIED
GETARG	PCL	703027	CØMMAND SCANNER
GETCHAR	BATCH	SC	SCAN ARGUMENT FIELD ØF JØB CØMMAND
GETCHST	SYSGEN	90 18 77	INTERNAL STRING GETTER
GETCØM	LØCCTRØM	90 18 77	GET ØRIGINAL LØCCT TABLE FRØM STØRAGE

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
IVO	MM	GA.01	INSERT VP AND PP
J:ABC	JIT	VA	FLAGS AND STUFF
J:ABUF	JIT	VA	LOCATION OF ASSIGN BUFFER IF IN MEMORY
J:AC	MM	GA	INITIALIZED BY MEMORY MANAGEMENT(JIT)
J:ACCN	JIT	VA	ACCOUNT NUMBER (DWORD)
J:ADCBTL	JIT	VA	(7 W) START OF DCB NAME TABLE IE. M:UC
J:AJ	JIT	VA	ADDITIONAL JIT'S ADDRESS
J:AJ	MM	GA.01	SET UP BY MEMORY MANAGEMENT
J:AMR	JIT	VA	DISC ADDRESS OF ASSIGN MERGE TABLE
J:ASSIGN	JIT	VA	LIMIT FLAGS
J:ASSIGN	RUNROM	LB.01	BIT 14 INDICATES PRESENCE OF PMDS
J:BUP	JIT	VA	FIRST PAGE # OF USER AREA
J:CASSIN	JIT	VA	BITS SET TO DIRECT ERROR OUTPUT
J:CBPOOL	JIT	VA	HEAD OF COOPERATIVE CONTEXT BLOCK POOL
J:CCBUF	JIT	VA	(20 WD) CONTROL COMMAND BUFFER
J:CFLGS	JIT	VA	CURRENT FLAGS ASSOC. WITH JOB
J:CFLGS	LNKTRC	RC	INFO SET UP FOR T:ASP
J:CL	JIT	VA	COMMAND LIST FOR DISC (4 WD/DISC REF)
J:CL	MM	GA	INITIALIZED BY MEMORY MANAGEMENT(JIT)
J:CLE	JIT	VA	NUMBER OF WORDS IN COMMAND LIST
J:CLE	MM	GA	INITIALIZED BY MEMORY MANAGEMENT(JIT)
J:CLL	JIT	VA	PAGE # OF JOB CONTEXT LOWER LIMIT (JIT
J:CLMN	JIT	VA	TEXTC OF CURRENT PROGRAM NAME (3 WD)
J:CLMP	JIT	VA	TEXTC OF CURRENT PROGRAM PASSWORD (3 W
J:CLP	JIT	VA	POINTER TO DESTROY WORD OF COMMAND LIS
J:CLPA	JIT	VA	COMMAND LIST PHYSICAL ADDRESS
J:CLS	JIT	VA	SAVED WORD OF COMMAND LIST
J:CPPO	JIT	VA	SEE CPPO
J:CPROCS	JIT	VA	PROCESSOR ASSOCIATION INDEXES
J:CTIME	JIT	VA	EXECUTION TIME FOR PROCESS CURRENTLY R
J:CUL	JIT	VA	PAGE # OF JOB CONTEXT UPPER LIMIT
J:DBPOOL	JIT	VA	HEAD OF COOPERATIVE DATA BLOCK POOL
J:DCBLINK	JIT	VA	ADDR OF SECOND PART OF DCB NAME TABLE
J:DDLL	JIT	VA	PAGE # OF PROGRAM DYNAMIC DATA LOWER L

FØR ITEM	IN MØDULE	SEE SECTIØN	CØMMENT
J:DDUL	JIT	VA	PAGE # ØF PRØGRAM DYNAMIC DATA UPPER L
J:DELTAT	JIT	VA	USED FØR TIMING EXECU. ØVERHEAD ØR IØ
J:DLL	JIT	VA	PAGE # ØF PRØGRAM DATA LØWER LIMIT
J:DUL	JIT	VA	PAGE # ØF PRØGRAM DATA UPPER LIMIT
J:DWSK	JIT	VA	STACK PTR DW FØR USE BY TEL
J:EUP	JIT	VA	LAST PAGE # ØF USER AREA
J:FPØØL	JIT	VA	ADDRESS ØF FIRST AVAILABLE BLØCKING BU
J:GST	JIT	VA	SIZE AND LØC ØF GLØBAL SYM TABLE
J:INTENT	JIT	VA	ENTRY ADDR TØ USERS CØNSØLE INTERRUPT
J:INTR	JIT	VA	NUMBER ØF INTERACTIØNS
J:IPØØL	JIT	VA	ADDRESS ØF FIRST AVAILABLE INDEX BUFFER
J:IST	JIT	VA	MAX SIZE AND LØC ØF INT. SYM TABLE
J:JAC	JIT	VA	2-BIT ACCESS TABLE FØR USER (12 WØRDS)
J:JIP	JIT	VA	SEE JIP
J:JIT	JIT	VA	JØB INFØRMATIØN TABLE
J:JIT	JIT	VA	START ØF JIT
J:LMN	JIT	VA	NAME ØF LAST LMN BUILT IN TEXTC (3 WØR
J:LMP	JIT	VA	PASSWØRD ØF LAST LMN BUILT IN TEXTC (2
J:LØCK	JIT	VA	FLAGS, BIT0 SET LØCKS USER IN CØRE
J:MRT	JIT	VA	MAXIMUM RUN TIME
J:NFPØØL	JIT	VA	NUMBER ØF BLØCKING BUFFERS
J:NIPØØL	JIT	VA	NUMBER ØF INDEX BUFFERS
J:ØPT	JIT	VA	ØPTIØN BITS IN USE
J:PLL	JIT	VA	PAGE # ØF PRØGRAM LØWER LIMIT
J:PTIME	JIT	VA	TØTAL PRØCESSØR EXECUTIØN TIME
J:PUL	JIT	VA	PAGE # ØF PRØGRAM UPPER LIMIT
J:RATE	JIT	VA	NØT USED
J:RNST	JIT	VA	JØB RUN STATUS
J:START	JIT	VA	STARTING ADDR ØF CURRENT PRØGRAM
J:T	JIT	VA	USED FØR PERFØRMANCE RECØRDING
J:TCB	JIT	VA	ADDR ØF TCB
J:TELBUF	JIT	VA	ADDR ØF TEL BUFFER
J:TELFLGS	JIT	VA	FLAGS USED BY TEL
J:TIC	JIT	VA	USED FØR PERFØRMANCE RECØRDING

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
M:ALDCB	M:ALDCB	VB.04	ACCOUNTING LOG DCB
M:BIDCB	M:BIDCB	VB.04	BINARY INPUT DCB
M:BODCB	M:BODCB	VB.04	BINARY OUTPUT DCB
M:CDCB	M:CDCB	VB.04	CONTROL COMMAND INPUT DCB
M:CIDCB	M:CIDCB	VB.04	COMPRESSED INPUT DCB
M:CODCB	M:CODCB	VB.04	COMPRESSED OUTPUT DCB
M:DODCB	M:DODCB	VB.04	DIAGNOSTIC OUTPUT DCB
M:EIDCB	M:EIDCB	VB.04	ELEMENT INPUT DCB
M:EODCB	M:EODCB	VB.04	ELEMENT OUTPUT DCB
M:FPPC	M:CPU	VF	COUNT OF MONITOR FREE PAGE POOL
M:FPPC	MM	GA	MONITOR FREE PAGE POOL COUNT
M:FPPH	MM	GA	MONITOR FREE PAGE POOL HEAD
M:FPPT	M:CPU	VF	TAIL OF MONITOR FREE PAGE POOL
M:FPPT	MM	GA	MONITOR FREE PAGE POOL TAIL
M:GODCB	M:GODCB	VB.04	EXECUTION OUTPUT DCB
M:LIDCB	M:LIDCB	VB.04	LIBRARY INPUT DCB
M:LLDCB	M:LLDCB	VB.04	LISTING LOG DCB
M:LODCB	M:LODCB	VB.04	LISTING OUTPUT DCB
M:OCDCB	M:OCDCB	VB.04	OPERATOR'S CONSOLE DCB
M:PODCB	M:PODCB	VB.04	PUNCH OUTPUT DCB
M:SGP	MM	GA.01	FINDS SWAP GRAN POOL
M:SIDCB	M:SIDCB	VB.04	SOURCE INPUT DCB
M:SLDCB	M:SLDCB	VB.04	SYSTEM LOG DCB
M:SODCB	M:SODCB	VB.04	SOURCE OUTPUT DCB
M:UC	JIT	VA	WORD ADDR OF JOB TITLE (720 WORDS)
M:US	JIT	VA	WORD ADDR, SEE J:TITLE
M:XX	JIT	VA	SYSTEM DCB USED BY DELTA AND OTHER PRO
MAILBOX	MAILBOX	UC	DELIVERS MESSAGES TO USERS
MAILBOX	BACKUP	KA	SEND BACKUP AND FILL MESSAGES TO USERS
MAILBOX	RECOVER2	KP.07	FILE INCONSISTENCY MESSAGE TO USER
MAND	SNAP	LR.02	ROUTINE TO PROCESS AND CALS
MAP	TABLES	VR.03	SETS MAP BIT IN PSD AND RETURNS TO R1
MAPMODE	ANALZ	LF.01	LOAD MAP FOR SPECIFIED USER
MASK	ANALZ	LF.01	MASK USED IN SEARCH

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
MASTER INDEX	OVERVIEW	BC	KEY INDEX INFO FOR EACH FILE
MAXJITS	SSDAT	VC	MAX NUMBER OF TASK JITS IN SYSTEM
MAXOVLY	M:SPROCS	VE	MAXIMUM OVERLAY PROCESSOR #
MB:DWT	MM	GA.01.08	SWAP RAD TABLE= DW SIZE OF SGP
MB:GAM1	MM	GA.01.08	SWAP RAD TABLE= GRANULE ADDRESS MASK
MB:GAM2	MM	GA.01.08	SWAP RAD TABLE= GRANULE POOL WORDS/GRN
MB:GAM3	MM	GA.01.08	SWAP RAD TABLE= SHIFT POOL TO GRAN POS
MB:GAM4	MM	GA.01.08	SWAP RAD TABLE= SHIFT TRACK TO GRAN AD
MB:GAM5	MM	GA.01.08	SWAP RAD TABLE= SHIFT OF DA TO TRACK #
MB:GAM6	MM	GA.01.08	SWAP RAD TABLE= SECTOR ADDRESS MASK
MB:GPT	MM	GA.01.08	SWAP RAD TABLE= GRANULES PER TRACK
MB:PPUT	M:CPU	VF	LINK TO NEXT PHYSICAL PAGE IN CHAIN
MB:PPUT	MM	GA	PHY PG CHAINS SET UP IN IT
MB:PPUT	SSS	ED.01	USAGE TABLE CONTAINS SWAP PG CHAIN
MB:SWAPS	MM	GA.01.08	SWAP RAD TABLE= SHIFT GRAN POS TO SGPX
MCOUNT	SNAP	LB.02	ROUTINE TO PROCESS COUNT CALS
MDPS	JIT	VA	BITS 0-14 ARE THE MAX DEBUG PAGES OUT
MEMORY LAYOUT	OVERVIEW	BC	MONITOR, USER, LIBRARIES, MON. OVERLAY
MFL	JIT	VA	SEE J:ASSIGN
MIF	SNAP	LB.02	ROUTINE TO PROCESS IF CALS
MJCFLG	JIT	VA	
MM	MM	GA	MEMORY MANAGEMENT
MNST	JIT	VA	MAX NO OF SAVE TAPES ALLOWED
MODE	COCD	VG.05	BYTE, LINE MODE BY LINE #
MODF	FRGD	90 18 77	SET UP MASTER PLIST AND SUB-PLISTS
MODGEN	SYSGEN	90 18 77	SPECIAL LOAD MODULE BUILDER
MODIFY	MODIFY	90 18 77	BUILDS LOAD MODULES
MODIFY	SYSGEN	90 18 77	BUILDS LOAD MODULE (EXCEPT SECT00)
MODIFY	SUPER	QC	COMMAND
MODIFY PLISTS	SYSGEN	90 18 77	SYSGEN PLISTS FOR MODIFY (SEE BI.07)
MODULES	OVERVIEW	BE	LISTED WITH SIZE AND FUNCTION
MODULES	OVERVIEW	BF	LISTED BY FUNCTION WITH SIZES
MONDMP	RECOVER2	KB.07	FILE CONTAINING CORE DUMP FRM RECOVERY
MONFIX	MONFIX	LG	MONITOR DEBUGGING AND REPLACING

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
PASS3CHK	PASS3RAM	90 18 77	PUT VALUES INTO LOCCT TABLE
PASS3DEL	PASS3RAM	90 18 77	PROCESS DELETE OPTION
PASS3LCT	PASS3RAM	90 18 77	REFORM LOCCT FILE RECORDS INTO LOCCT T
PASS3NXT	PASS3RAM	90 18 77	GET NEXT CONTROL COMMAND
PASS3PAR	PASS3RAM	90 18 77	PROCESS CONTROL COMMAND PARAMETERS
PB:DCBSZ	M:SPROCS	VE	NUMBER OF PAGES OF DCB'S BY PROC #
PB:DSZ	M:SPROCS	VE	NUMBER OF PAGES OF PROC DATA BY PROC #
PB:HPP	M:SPROCS	VE	HEAD OF PHYSICAL PAGE CHAIN BY PROC #
PB:HPP	MM	GA	PROCESSORS PHY PG CHAIN HEAD
PB:HVA	M:SPROCS	VE	VIRTUAL PAGE # OF 1ST PAGE NOT USED
PB:LNK	M:SPROCS	VE	PROCESSOR # OF FIRST OVERLAY BY PROC #
PB:PSZ	M:SPROCS	VE	NUMBER OF PAGES OF PROC PROCEDURE
PB:PSZ	MM	GA	SIZE OF PROCESSOR
PB:PVA	M:SPROCS	VE	VIRTUAL PAGE # OF FIRST PAGE USED
PB:TPP	M:SPROCS	VE	TAIL OF PHYSICAL PAGE CHAIN BY PROC #
PB:TPP	MM	GA	PROCESSORS PHY PG CHAIN TAIL
PB:UC	M:SPROCS	VE	COUNT OF CURRENT USERS IN CORE BY PROC
PBT:LOCK	M:SPROCS	VE	DW, PROCESSOR LOCKED IN CORE BIT TABLE
PCCF	JIT	VA	BIT 9 OF J:RNST, PROCESSOR CONTROL CMD
PCCI	M:SPROCS	VE	PROCESSOR # OF CCI
PCL	PCL	703027	PCL EXECUTIVE
PCL	OVERVIEW	BF	PERIPHERAL CONVERSION LANGUAGE
PCLLIST	PCL	703027	LIST,DELETE,REW,SPE COMMAND PROCESSR
PCT	FRGD	90 18 77	SET CONDITIONS FOR PROCESSING CT VALUE
PCTQ	FRGD	90 18 77	SET CONDITIONS FOR PROCESSING CTQ VALU
PENT	TEL	PB.03	INSERT PARAMETER INTO SKELETAL DLIST
PER	OVERVIEW	BC	SYMBIONT AREA OF RAD
PERFORMANCE	OVERVIEW	BD	SYSTEM PERFORMANCE MEASUREMENTS
PFA	OVERVIEW	BC	FILE MANAGEMENT AREA OF RAD
PFCOM	FRGD	90 18 77	SET CONDITIONS FOR PROCESSING FCOM VAL
PFFPOOL	FRGD	90 18 77	SET CONDITIONS FOR PROCESSING FFPOOL V
PFIPOOL	FRGD	90 18 77	SET CONDITIONS FOR PROCESSING FIPOOL V
PFSR	PFSR	KF	POWER FAIL SAFE ROUTINES
PGSOUT	ANALZ	LE.01	DISPLAY HEAD,TAIL, AND COUNT AS CHAIN

FØR ITEM	IN MØDULE	SEE SECTIØN	CØMMENT
S:BDA	SSDAT	VC	LIST ØF BEGIN DISC ADDR. (SEE SB:ØSUL)
S:BDA	SSS	ED.01	FIRST DISC.ADR ØF USER SWAPPED ØUT
S:BFIS	SSDAT	VC	NUMBER ØF JØBS IN BATCH STREAM
S:BUAIS	M:IMC	VC	BATCH USERS ALLØWED ØN THE SYSTEM
S:BUIS	SSDAT	VC	CØUNT ØF BATCH USERS IN SYSTEM
S:CLP	SSS	ED.01	PØINTER TØ WØRD DESTRØYED IN USER'S CL
S:CLS	SSS	ED.01	WØRD DESTRØYED IN CL BY TIC
S:CUAIS	M:IMC	VC	CURRENT USERS ALLØWED ØN THE SYSTEM
S:CUIS	SSDAT	VC	CØUNT ØF USERS IN SYSTEM
S:CUN	SSDAT	VC	CURRENT USER NUMBER
S:EAF	SSDAT	VC	
S:ECL	SSDAT	VC	LIST ØF PTRS TØ END ØF CMND LIST
S:ECL	SSS	ED.01	END ØF CL FØR USER SWAPPED ØUT
S:EDA	SSDAT	VC	LIST ØF ENDING DISC ADDR (SEE SB:ØSUL)
S:EVF	SSDAT	VC	EVENT HAS ØCCURED FLAG
S:FPPC	SSDAT	VC	CØUNT ØF NØ. ØF FREE PAGES IN S:FPPT
S:FPPC	SSS	ED.01	CØUNT ØF SWAPPER'S FREE PHY PAGE PØØL
S:FPPH	SSDAT	VC	HEAD ØF SWAPPER FREE PAGE PØØL
S:FPPH	SSS	ED.01	HEAD ØF SWAPPER'S FREE PHY PAGE PØØL
S:FPPT	SSDAT	VC	TAIL ØF SWAPPER FREE PAGE PØØL
S:FPPT	SSS	ED.01	TAIL ØF SWAPPER'S FREE PHY PAGE PØØL
S:GJØBTBL	SSDAT	VC	DW, NAME ØF GHØST JØB BY GHØST JØB #
S:HIR	SSDAT	VC	CØUNT ØF HI-PRIØRITY JØBS READY TØ RUN
S:IDLE	SSDAT	VC	IDLE FLAG
S:ISUN	SSDAT	VC	INSWAP USER NUMBER
S:ISUN	SSS	ED.02	THE # ØF THE USER TØ PREPARE FØR EXEC
S:JCL	SSDAT	VC	CØMMAND LIST FØR READING JIT ØR AJIT
S:JCL	SSS	ED.02	WHERE CL BUILT TØ SWAP IN AJIT & JIT
S:JITERR	SSS	ED.02	RØUTINE HANDLES JIT SWAP ERRØRS
S:JSP	SSDAT	VC	(JIT SECTØR PØS. + SDLAY)/2
S:JSP	SSS	ED.02	SAVES JITS GRAN PØS 1ST SWAP IN
S:LUN	SSDAT	VC	LAST USER NUMBER
S:ØSS	SSDAT	VC	ØUTSWAP SIZE
S:ØUAIS	M:IMC	VC	ØN-LINE USERS ALLØWED ØN THE SYSTEM

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
S:PCT	SSDAT	VC	TOTAL PAGE COUNT FOR SWAP IN
S:PCT	SSS	ED.02	PAGE COUNT TO SWAP IN USER & PROCESSOR
S:SCL	SSDAT	VC	SWAPPER COMMAND LIST TABLE
S:SCL	SSS	ED.01	WHERE CL'S BUILT TO SWAP OUT JITS
S:SCLP	SSDAT	VC	
S:SET	SSS	EA	STATE EVENT TRANSITION TABE
S:SIP	SSDAT	VC	SWAP IN PROGRESS FLAG
S:SIP	SSS	ED.02	RESET AT END OF SWAP IN, SWAP COMPLETE
S:SIR	SSDAT	VC	COUNT OF USERS TO BE SWAPPED IN
S:SWPCNT	SSDAT	VC	SWAP COUNTER FOR SWAP IDENTIFICATION
S:SWPCNT	SSS	ED.01	READ CHECK ID FOR NEXT OUTSWAP USER
S:TRNSVEC	SSDAT	VC	EVENT TRANS. VECTOR FOR EVENTS >=X'40'
S:USID	SSDAT	VC	USER SYSTEM ID
SA[illegible]	SSS	EA	STATE 18, USERS WAITING FOR COC BUFFER
SACT	SACT	FA	QUEUED SYMBIONT AND COOP RESTART
SAVE.GET	UCAL	IA	LIMITED ONLINE CHECKPOINT
SAVEALL	BACKUP	KA.01	TYPE OF AUTOMATIC BACKUP
SAVEREGS	INITRCVR	LD	REGISTERS SAVED FOR RECOVERY & ANALZ
SAVHGP	TSTHGP	KB.02.02	SAVE (FDA),(SSMI), AND (SMI) IN HGP
SAVINCOM	PASS3ROM	90 18 77	SAVE LOCCT TABLE FOR SYSTEM STORAGE
SAVSYM	SYMFILE	KB.04.01	SAVE SYMFILE AND SYMFSDA
SB:CQ	SSDAT	VC	COUNT OF USERS IN Q BY STATE #
SB:ETT	SSS	EA	EVENT INDEX INTO S:SET
SB:EXU	SSS	EA	LIST OF EXECUTABLE STATES
SB:FPL	SSDAT	VC	LIST OF PROCESSORS FREED BY OUTSWAP
SB:FPN	SSDAT	VC	NUMBER OF PROCESSORS FREED BY OUTSWAP
SB:GJOBFLG	SSDAT	VC	GHOST JOB FLAGS BY GHOST JOB #
SB:GJOBUN	SSDAT	VC	GHOST JOB USER NUMBER, BY GHOST JOB #
SB:HIR	SSDAT	VC	Q OF Q'S
SB:HIR	SSS	EA	LIST OF HIGH PRIORITY STATES
SB:HPP	SSDAT	VC	PROCS TEMP PHYSICAL PAGE CHAIN HEAD
SB:HQ	SSDAT	VC	USER # OF FIRST USER IN STATE Q
SB:NP	SSDAT	VC	NUMBER OF PROCESSORS TO SWAP IN
SB:NP	SSS	ED.02	INDICATE HOW MANY PROCESSORS TO SWAPIN

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
SEC	SSS	EA	STATE 3, USERS Q'ED FOR TEL (CONTROL E
SEGLD	SEGLD	EC	USER OR SHARED PROCESSOR OVRLAY LOADER
SELECT	PASS1RAM	90 18 77	PROCESS :SELECT COMMAND
SEND	CONTROL	QA	OUTPUT BUFFER
SENSE SWTCH 4	TSIO	DB	CHECK COMMAND LIST CHAIN
SERR	SSS	EA	STATE 18, OPERATOR ERRORED USER
SERVDEV	IOQ	DA.01	SERVICE I/O DEVICE
SETOREG	TSIO	DB	ROUTINE SETS REGS IN TSIO FOR NEWQ
SETALL	PASS1RAM	90 18 77	PROCESS ALL OPTION ON SELECT/UPDATE CO
SETGRANO	UBCHAN	90 18 77	BUILD HGP BIT MAPS FOR PFA AND PER
SETMODFY	UBCHAN	90 18 77	MANIPULATE LOAD MODULE
SFIND	CONTROL	QA	DETERMINES AN INDEX VALUE FOR NAME
SGP	MM	GA.01.08	SWAPPE GRANULE ALLOCATION POOL
SH:EDA	SSS	ED.01	LAST DISC ADR OF USER SWAPPED OUT
SH:JAJDA	SSDAT	VC	DISC ADDRESSES FOR JIT AND AJIT
SH:JAJDA	SSS	ED.02	DISC ADDRESS TABLE FOR S:JCL
SH:JDA	SSDAT	VE	DISC ADDRESS OF GHOST JOB JIT BY GHOST
SH:SDA	SSDAT	VE	SEEK DISC ADDRESSES REF'D BY S:SCL
SH:SDA	SSS	ED.01	AREA USED FOR DISC ADR FOR S:SCL
SHARED PROGRM	OVERVIEW	BB	PURE PROCEDURES SHARED BY USERS
SH:WPID	SSS	ED.01	HALF WORD IDS FOR READ CHECKING
SIOC	SSS	EA	STATE 12, USERS WITH I/O COMPLETE
SIOIP	SSS	EA	STATE 11, USERS WITH I/O IN PROGRESS
SIOW	SSS	EA	STATE 10, USERS WAITING TO START I/O
SIR	SSS	EA	STATE 5, USERS WITH TTY INPUT COMPLETE
SIZECHK	UBCHAN	90 18 77	KEEP TRACK OF DISC OPTIONS
SL:BB	M:IMC	VJ	BATCH BIAS
SL:BC	M:IMC	VJ	MAX CORE ALLOWED ANY BATCH USER
SL:CORE	M:IMC	VJ	MAX CORE ALLOWED SPECIAL PROCESSORS
SL:OC	M:IMC	VJ	MAX CORE ALLOWED ANY ON-LINE USER
SL:OT	M:IMC	VJ	MAX # OF TAPES ALLOWED ON-LINE USERS
SL:QMIN	M:IMC	VJ	MINIMUM QUANTUM
SL:QUAN	M:IMC	VJ	QUANTUM FOR COMPUTE BOUND USERS
SL:TB	M:IMC	VJ	# OF CHARS TO BLOCK TERMINAL OUTPUT

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
SL:UB	M:IMC	VJ	# OF CHAR TO UNBLOCK TERMINAL OUTPUT
SLAVE	OVERVIEW	BB	USER PROGRAM MODE,I.E.,NOT MONITOR
SLEEP CAL	UCAL	IA	M:WAIT, PROCESSED BY T:WAIT
SLIMS	SLIMS	NONE	ANCIENT NULL TABLE
SMAXOUT	SSDAT	VC	MAX NUMBER OF USERS SWAPPED OUT
SMBUIS	M:IMC	VC	MAXIMUM BATCH JOBS IN SYSTEM
SMUIS	M:IMC	VC	MAXIMUM USERS IN SYSTEM
SNAME	CONTROL	QA	INPUTS A STRING FROM TERMINAL
SNAP	SNAP	LB.02	EXECUTION TIME PROCESSOR FOR DEBUG CAL
SNDDX	M:SDEV	VI.01	SNDDX,0 IS NUMBER OF SYMBIONTS
SNDDX	M:SDEV	VI.01	BYTE, DCT INDEX BY SYMBIONT INDEX
SNRRT	SSS	EA	STATE 1, REAL TIME USERS
SNSTS	SSS	EA	NUMBER OF STATES IN SYSTEM
SOFF	SSS	EA	STATE 1C,OPERATOR ABORT OR USER HUNG U
SON	SSS	EA	STATE 2, USERS Q'ED FOR LOG ON
SORT/MERGE	OVERVIEW	BF	XDS SORT/MERGE
SOUT	CONTROL	QA	SAME AS SOUTA, BUT ALSO OUTPUTS BUFFER
SOUTA	CONTROL	QA	APPENDS A STRING TO OUTPUT BUFFER
SPACE	SYMCON	SE	UPSPACE A GIVEN NUMBER OF LINES
SPACES,SPACE2	ANALZ	LF	INSERT SPACES IN OUTPUT BUFFER
SPEC:HAND	SYSGEN	90 18 77	SPEC:HAND FILE STRUCTURE
SPECFIL	ANALZ	LE.01	OPEN M:EI TO RECOVERY FILE MONDMP
SPMAP	SSS	ED.02	ROUTINE PUTS PROCESSOR PHY PG IN CMAP
SPROCS	SYSGEN	90 18 77	PROCESSES SPROCS
SQA	SSS	EA	STATE 16, USERS Q'ED FOR ACCESS TO I/O
SQUIRREL	BACKUP	KA.01	TYPE OF AUTOMATIC BACKUP
SRCHF	SRCHF	FA	SEARCH SYMFILE TO DELETE FILE
SRCHTBL	PASS1RAM	90 18 77	SEARCH FILE OR STD TABLE FOR NAME
SRET	M:SDEV	VI.01	SYMBIONT RETURN ADDRESS BY SYMBIONT #
SS	JIT	VA	BITS 26 TO 31 ARE PSEUDO SENSE SWITCHE
SSDAT	SSDAT	VC	DATA BASE FOR SCHEDULER/SWAPPER
SSDAT	SSS	EA	DATA BASE FOR SCHEDULER
SSIG	M:SDEV	VI.01	BYTE, SIGNAL CHARACTER BY SYM #
SSS	SSS	EA	SCHEDULER AND SWAPPER

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
SSTAT	M:SDEV	VI.01	BYTE, SYMBIONT STATUS BY SYMBIONT #
STADDR	SDEVICE	90 18 77	CALL MODIFY ROUTINE
STAR FILES	OVERVIEW	BB	UNIQUELY NAMED TEMP FILES BY SYSTEM ID
STAR FILES	ACCTSUM	PC.01	RELEASE OF TEMPORARY FILES
START	SSS	EA	EVENT 48, ADD REAL TIME USER
STARTIO	IOQ	DA.01	INITIATE ALL I/O OPERATIONS
STATE	COCD	VG.05	BYTE, STATE OF LINE BY LINE #
STATE QUEUES	OVERVIEW	BD	STATE QUEUES ARE PRIORITY STRUCTURE
STATES	ANALZ	LE.01	GET STATE
STBA	SSS	EA	EVENT 46, COC BUFFER AVAILABLE
STCRD	SSS	EA	EVENT 4A, CRD TO CHECK FOR STIC CASE
STDNM	PASS1ROM	90 18 77	PROCESS STD OPTION
STDPA	SSS	EA	EVENT 44, DISC PAGE IS AVAILABLE
STEP	STEP	EB	MONITOR JOB STEP CONTROL ROUTINES
STI	SSS	EA	STATE C, USERS INCORE AND TYPING IN
STIC	SSS	EA	EVENT 49,IC WHEN USER IS CURRENT USER
STIIP	SSS	EA	EVENT 47,I/O IN PROGRESS
STIME	CONTROL	QA	RETURN TIME, IN SECONDS, SINCE SYSTEM
STIO	SSS	EA	STATE 1A, LIKE STI, BUT NOT IN CORE
STIP	SSS	EA	EVENT 43, GIVE IO START PERMISSION
STKO	SSS	EA	EVENT 4D, KICK USER OUT OF CORE
STNOP	SSS	EA	EVENT 40, NO OPERATION
STOB	SSS	EA	STATE B, TTY OUTPUT BLOCKED USERS
STOBO	SSS	EA	STATE 19, LIKE STOB, BUT NOT IN CORE
STOC	SSS	EA	STATE 6, USERS READY TO CONT. TTY OUT
STOFF	SSS	EA	EVENT 45, OFF PROCESS
STORVLP	PCL	703027	ADDS ENTRY TO VLP OF OPEN PLIST
STQA	SSS	EA	EVENT 52, Q FOR ACCESS TO I/O DEVICE
STREGS	SDEVICE	90 18 77	SAVE REGISTERS
STSABRT	SSS	EA	EVENT 51, SET ABORT FLAG
STSABRTC	SSS	EA	EVENT 4F, SET ABORT FLAG AND CHANGE ST
STSBK	SSS	EA	EVENT 41, SET BREAK FLAG
STSBKC	SSS	EA	EVENT 4B, SET BREAK AND CHANGE STATE
STSEC	SSS	EA	EVENT 42, SET EC FLAG

FØR ITEM	IN MØDULE	SEE SECTIØN	CØMMENT
T:FCP	MM	GA.01	FREE CØMMØN PG
T:FP	MM	GA.01	FREE PG
T:FPP	MM	GA.01	FREE PHY PG
T:FVP	MM	GA.01	FREE VIRTUAL PG
T:FVPM	MM	GA.01	FREEVP MASTER
T:GAJP	MM	GA.01	GET AJIT PAGE
T:GCP	MM	GA.01	GET CØMMØN PG
T:GHØST	UCAL	IA	RØUTINE TØ SEND ERR MSG IF GHØST ABØRT
T:GJØBSTRT	UCAL	IA	RØUTINE TØ START UP GHØST JØBS
T:GL	MM	GA.01	GET CØMMØN LIMITS
T:GNVNPI	MM	GA.01	GET N VP AND NØ PP
T:GNVPI	MM	GA.01	GET N VP AND PP
T:GP	MM	GA.01	GET PG
T:GPP	MM	GA.01	GET PHY PG
T:GVGPI	MM	GA.01	GET N VP GIVEN PP
T:GVP	MM	GA.01	GET VIRTUAL PG
T:GVPI	MM	GA.01	GET VP INTERNAL
T:GVPM	MM	GA.01	GET VP MASTER
T:IACU	MM	GA.01	INTERRØGATE AC IN USER'S IMAGE
T:INITJØB	UCAL	IA	RØUTINE TØ PRØCESS GHØST START CALS
T:JØBENT	T:JØBENT	IA	ENTER JØB IN SYMBIØNT STREAM
T:NAMECHK	T:ØV	EC	CHECK FØR VALID GHØST NAME
T:ØFF	SSS	FA	FØRCE A USER ØFF
T:ØV	T:ØV	EC	ASSØCIATE MØNITØR ØVERLAY
T:ØVER	T:ØV	EC	ASSØCIAT ØVERLAY - NØ RETURN
T:ØVERLAY	T:ØV	EC	ASSØCIAT ØVERLAY - REMEMBER RETURN
T:ØVERLAY1	T:ØV	EC	T:ØVERLAY WITH NAME IN REGISTERS
T:ØV2	T:ØV	EC	T:ØV WITH NUMBER SPECIFIED
T:PAC	MM	GA.01	PRØCESSØR ACCESS CØNTRØL
T:PGCHK	CHK	KC	MØNITØR ØR SWAPPER PAGE CHAIN CHECK
T:PGCHK	SSS	ED.01	CKS VALIDITY ØF MØN,SWAP,USER PG CHAIN
T:PRØCØV	T:ØV	EC	ASSØCIATE SHARED PRØC ØVERLAY
T:PULLA	SSS	EA	PULL AN ENVIRØNMENT TØ ALT ADDR
T:PULLE	SSS	EA	PULL AN ENVIRØNMENT

FOR ITEM	IN MODULE	SEE SECTION	COMMENT
T:RCE	SSS	EA	REPORT A COC EVENT
T:RE	SSS	EA	REPORT EVENT ON CURRENT USER
T:RECORD	SSS	ED.01	CREATES SWAP DEBUG INFO
T:REG	SSS	EA	REPORT EVENT AND GIVE UP CPU
T:REMEMBER	T:OV	EC	RECORD CURRENT SEG AND R11 FOR RETURN
T:RSTLMS	STEP	EB	RESET ALL JIT MEMORY POINTERS
T:RUE	SSS	EA	REPORT EVENT ON SPECIFIED USER
T:RUNDOWN	STEP	EB	INTERNAL ENTRY TO REINITIALIZE A USER
T:RVPI	MM	GA.01	RELEASE VP INTERNAL
T:RVSPI	MM	GA.01	RELEASE VP SAVE PP
T:SAC	MM	GA.01	SET ACCESS
T:SAD	MM	GA.01	SEARCH AND DISPLAY
T:SAVEGET	UCAL	IA	ROUTINE TO PROCESS SAVE/GET CAL(CHKPT)
T:SE	SSS	EA	SCHEDULE FOR EXECUTION
T:SELFDESTRUC	UCAL	IA	ROUTINE TO DISASSOCIATE MON OVERLAY
T:SENSE	SSS	ED	ROUTINE RETURNS RAD HEAD POSITION
T:SEXIT	TSIO	DB	ROUTINE USED TO RETURN TO CALLER
T:SGA	MM	GA.01	SWAP GRAN ALLOCATION
T:SGAJIT	MM	GA.01.08	SWAP GRANULE ALLOCATION WITHOUT A USER
T:SGR	MM	GA.01	SWAP GRAN RELEASE
T:SGRNU	MM	GA.01.08	SWAP GRANULE RELEASE WITHOUT A USER
T:SIO	TSIO	DB	ENTRY TO TSIO TO PERFORM SWAP I/O
T:SMMC	MM	GA.01	SET UP MMC
T:SMP	MM	GA.01	SET MEMORY PROTECTION
T:SNAC	MM	GA.01	SET N ACCESS
T:SS	SSS	EA	SCHEDULE SWAP
T:SSE	SSS	EA	SCHEDULE SWAP AND EXECUTION
T:SSEM	SSS	EA	SCHEDULE SWAP AND EXECUTION MAPPED ENT
T:STPMT	UCAL	IA	ROUTINE TO ESTABLISH PROMPT CHARACTER
T:SXAC	MM	GA.01	EXECUTE AC
T:SXMAP	MM	GA.01	EXECUTE MAP
T:SYS	RDERLOA	IA	ROUTINE TO GIVE SLAVE USER MASTER MODE
T:SYSLOAD	UCAL	IA	ROUTINE TO COMPUTE FTMF
T:TOTESZ	SSS	EA	CALCULATE USERS SIZE

IN MØDULE	FØR ITEM	SEE SECTIØN	CØMMENT
CRDØUT	CRDØUT	DA.03	CARD PUNCH HANDLER
CYCUSR	CYCUSR	KB.03	VERIFY USER TABLES, CLØSE USER FILES
DEBUGTV	DEBUGTV	LB	TRANSFER VECTØR FØR DEBUG RØUTINES
DEFCØM	DEFCØM	SD	LØAD MØDULE REF/DEF STACK EXTRACTIØN
DELPRI	DELPRI	HA	DELETE FILES FRØM SYMFILE AND DISC
DELTA	DELTA	LA	CØNVERSATIØNS PRØGRAM DEBUGGING PRØC.
DISPLAY	DISPLAY	HA	DISPLAY SPECIFIED MØNITØR INFØRMATIØN
DPAK	DPAK	DA.03	DISC PACK HANDLER
DSCIØ	DSCIØ	NØNE	REMØTE BATCH HANDLER
DUMP	DUMP	LB.05	CØRE DUMP RØUTINE
EDCØN	EDCØN	NØNE	BATCH PRØCESSØR FØR EDIT FØRMAT FILES
EDIT	EDIT	NØNE	CØNTEXT EDITØR
ENTRY	ENTRY	CA	ENTRY AND EXIT FØR PRØCESSING CALS
ERR:FIL	ERR:FIL	KE.02	PRØGRAM TØ CØPY ERRØRLØG TØ KEYED FILE
ERR:LIST	ERR:LIST	KE.05	ERRØR LØG FØRMATTING & LISTING PRØGRAM
ERR:SUM	ERR:SUM	KE.03	ERRØR LØG SUMMARY PRØCESSØR
ERRMWR	ERRMWR	UB	ERRØR MESSAGE FILE CØNTRØL PRØCESSØR
FBCD	FBCD	NØNE	FØRTRAN BCD CØNVERSIØN
FILL	FILL	KA.02	RESTØRES USER'S FILES FRØM BACKUP TAPE
GETF	GETF	FA	GET FILE FRØM SYMFILE
GHØST1D	GHØST1D	NC	GHØST 1 DRIVER
GPHGP	GPHGP	NG	READ/WRITE HGP TØ SWAP RAD (ALSØ XDELT
HANDLERS	HANDLERS	DA	REQUIRED HANDLERS
HGPRECØN	HGPRECØN	KB.08	HGP RECØNSTRUCTIØN DURING RECØVERY
INITIAL	INITIAL	NA	INITIALIZE MØNITØR
INITRCVR	INITRCVR	LD	INITIALIZE RECØVERY
INSYM	INSYM	FA	INPUT SYMBIØNT (CARD READER)
IØQ	IØQ	DA	BASIC I/Ø STARTER
IØREC	IØREC	HA	DEVICE KEYIN RØUTINES
JIT	JIT	VA	JØB INFØRMATIØN TABLE
JULIAN	JULIAN	UA	CØNVERT MØNITØR DATA-TIME TØ JULIAN
KEYN	KEYN	HA	ØPERATØR CØNSØLE CØMMAND PRØCESSØR
KEYSUB	KEYSUB	HA	KEYIN RØUTINES
LINK	LINK	RA	LØADER PRØGRAM

IN MODULE	FOR ITEM	SEE SECTION	COMMENT
PCL	FILTRAN	703027	SYNTAX ANALYZER FOR FILE IDENTIFIER
PCL	FIXARG	703027	TABLE SEARCH SUBROUTINE
PCL	GETARG	703027	COMMAND SCANNER
PCL	HEXDUMP	703027	HEXADECIMAL DUMP PROCESSOR
PCL	INTARG	703027	EDCDIC-BINARY DECIMAL CONVERSION
PCL	PCL	703027	PCL EXECUTIVE
PCL	PCLLIST	703027	LIST,DELETE,REW,SPE COMMAND PROCESSR
PCL	PRTERR	703027	PRINTS ERROR MESSAGES
PCL	RDWRT	703027	PERFORMS FILE COPY
PCL	STORVLP	703027	ADDS ENTRY TO VLP OF OPEN PLIST
PCL	TEXTARG	703027	CHECKS ARGUMENT LENGTH
PCL	UTILITY	703027	UTILITY AND CONVERSION ROUTINES
PFSR	PFSR	KF	POWER FAIL SAFE ROUTINES
PHASEA	PHASEA	ND	PROCESS GENOP,GENCHN, AND GENDCB
PHASEB	PHASEB	ND	TRANSLATE GENMD AND GENDICT
PHASEC	PHASEC	ND	COPY PO TO :SYS ACCOUNT, ADD GENMDS
PHASED	PHASED	ND	NOP, REPLACED BY SYSMAK
PM	PM	IB	PERFORMANCE MEASUREMENT ROUTINES
PMD	PMD	LB.03	ROUTINE TO PROCESS PMDS AND PMDIS
PMDAT	PMDAT	VJ	DATA BASE FOR PERFORMANCE MEASUREMENT
PODCBS	PODCBS	N	DCB'S FOR PASSO
PPP	PPP	NONE	ANCIENT NULL TABLE
PTAP	PTAP	DA.03	PAPER TAPE HANDLER
RATES	RATES	QB	CHARGE RATE CONTROL PROCESSOR
RCVCTL	RCVCTL	KB.01	RECOVERY MAIN CONTROL
RDERLOG	RDERLOG	IA	READ ERROR LOG
RECORD	RECORD	LF	EVENT RECORD ROUTINE AND BUFFER
RECOVER2	RECOVER2	KB.07	RESTORE SYSTEM TABLES
REQDC	REQDC	FA	DISC AND CORE ALLOCATION FOR SYMB,COOP
RUNROM	RUNNER	LB.01	BUILD DEBUG TABLES
SACT	SACT	FA	QUEUED SYMBIONT AND COOP RESTART
SEGLD	SEGLD	EC	USER OR SHARED PROCESSOR OVRLAY LOADER
SLIMS	SLIMS	NONE	ANCIENT NULL TABLE
SNAP	SNAP	LB.02	EXECUTION TIME PROCESSOR FOR DEBUG CAL

IN MODULE	FOR ITEM	SEE SECTION	COMMENT
SRCHF	SRCHF	FA	SEARCH SYMFILE TO DELETE FILE
SSDAT	SSDAT	VC	DATA BASE FOR SCHEDULER/SWAPPER
SSS	SSS	FA	SCHEDULER AND SWAPPER
STEP	STEP	FB	MONITOR JOB STEP CONTROL ROUTINES
SUPCLS	SUPCLS	FA	OUTPUT CARD, TERMINAL SYMBIONT FILE
SUPER	SUPER	OC	LOGON CONTROL PROCESSOR
SUSPTERM	SUSPTERM	FA	TYPE SUSPEND AND TERMINATE MESSAGES
SYMCON	SYMCON	SE	LOAD MODULE SYMBOL CONTROL PROCESSOR
SYMFILS	SYMFILS	KB.04.04	PROCESS SYMBIONT TABLES
SYMSUBR	SYMSUBR	FA	MISCELLANEOUS SYMBIONT SUBROUTINES
SYMTAB	SYMTAB	LA	EXECUTIVE DELTA SYMBOL TABLE
SYSGEN	ABS	90 18 77	PROCESSES ABS (BPM ONLY)
SYSGEN	BTM	90 18 77	PROCESSES BTM (BPM ONLY)
SYSGEN	DEF	90 18 77	WRITES BO TAPES
SYSGEN	FRGD	90 18 77	PROCESSES FRGD,INTLB
SYSGEN	IMC	90 18 77	PROCESSES IMC
SYSGEN	LOCCT	90 18 77	BUILDS LOCCT FILES
SYSGEN	PASS1	90 18 77	SYSGEN FILE MANAGER WRITES BO TAPES
SYSGEN	PASS2	90 18 77	SYSGEN TABLE BUILDER
SYSGEN	PASS3	90 18 77	LOADS MONITOR AND PROCESSORS
SYSGEN	P2CCI	90 18 77	READS AND ASSIGNS PASS2 COMMANDS
SYSGEN	P2COC	90 18 77	PROCESSES COC
SYSGEN	SDEVICE	90 18 77	PROCESSES SDEVICE
SYSGEN	SPROCS	90 18 77	PROCESSES SPROCS
SYSGEN	UBCHAN	90 18 77	PROCESSES CHAN,DEVICE,STDLB,OSTOLB
SYSGEN	XLIMIT	90 18 77	PROCESSES OLIMIT,BLIMIT,DLIMIT
SYSGEN	XMONITOR	90 18 77	PROCESSES UTM,MONITOR
SYSMAK	SYSMAK	NE	INITIALIZE SWAPPING RAD (PROCS,JITS)
T:JOBENT	T:JOBENT	IA	ENTER JOB IN SYMBIONT STREAM
T:OV	T:OV	FC	ASSOCIATE MONITOR OVERLAY
TABLES	TABLES	NONE	CONSTANTS, DATA
TAPECHST	TAPECHST	PD	SYNTAX SCAN UTILITY ROUTINES
TAPEFCN	TAPEFCN	PD	COMMAND FUNCTION PROCESSOR
TELLUSR	TELLUSR	LB.04	PRINT MONITOR ERROR MESSAGES TO BATCH

IN MODULE	FOR ITEM	SEE SECTION	COMMENT
TIM	TIM	IA	DATE/TIME CAL PROCESSOR
TOPRT	TOPRT	VE	SEG NAMES AND ENTRY POINT DISPLACEMENT
TRAPC	TRAPC	NONE	BPM CAL PROCESSOR
TSIO	TSIO	DB	ENTRY TO TSIO TO PERFORM SWAP I/O
TSIO	TSIO	DB	SWAPPER I/O ROUTINE
TSTHGP	TSTHGP	KB.02.01	VALIDITY CHECK OF HGP TABLES
UCAL	UCAL	IA	PROCESSES MISCELLANEOUS UTS CALS
UTS	UTS	UD	USED FOR ASSEMBLING UTS MONITOR
VDCB	VDCB	RA	
XDELTA	XDELTA	LA	EXECUTIVE DELTA
7TAP	7TAP	DA.03	7-TRACK TAPE HANDLER

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

XEROX

Reader Comment Form

We would appreciate your comments and suggestions for improving this publication.			
Publication No.	Rev. Letter	Title	Current Date

How did you use this publication? ☐ Learning ☐ Installing ☐ Sales ☐ Reference ☐ Maintaining ☐ Operating	Is the material presented effectively? ☐ Fully Covered ☐ Well Illustrated ☐ Well Organized ☐ Clear
What is your overall rating of this publication? ☐ Very Good ☐ Fair ☐ Very Poor ☐ Good ☐ Poor	What is your occupation?

Your other comments may be entered here. Please be specific and give page, column, and line number references where applicable. To report errors, Please use the Xerox Software Improvement or Difficulty Report (1188) instead of this form.

Your Name & Return Address

2190(12/72)

Thank You For Your Interest. (fold & fasten as shown on back, no postage needed if mailed in U.S.A.)

Staple

Staple

Fold

First Class
Permit No. 229
El Segundo,
California

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

Postage will be paid by

Xerox Corporation
701 South Aviation Boulevard
El Segundo, California 90245

Attn: Programming Publications

Fold