3-10-1987

Partitioning and protecting control program function through virtual storage

Thomas J. Murray

Rochester Institute of Technology
School of Computer Science and Technology

Partitioning and Protecting Control Program Function Through Virtual Storage

by
Thomas J. Murray

A thesis, submitted to
The Faculty of the School of Computer Science and Technology,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.

Approved by: James Heliotis
Professor James Heliotis

Andrew Kitchen
Professor Andrew Kitchen

Peter G. Anderson
Professor Peter Anderson

March 10th, 1987

vised sample statement for granting or denying permission to eproduce an RIT thesis.

itle of Thesis PARTITIONING AND PROTECTING CONTROL PROGRAM FUNCTION THROUGH VIRTUAL STORAGE

Thomas J. Murray hereby (grant, eny) permission to the Wallace Memorial Library, of R.I.T., to eproduce my thesis in whole or in part. Any reproduction will ot be for commercial use or profit.

Or

_____ prefer to be ontacted each time a request for reproduction is made. I can be eached at the following address. _____

_____

ate 3/18/87

ABSTRACT

Many operating systems are endowed with the ability to access every byte of addressable memory in the processor complex. Unfortunately, a control program so designed creates an exposure of the following nature: It is possible for an errant section of operating system code to unwittingly modify (and perhaps destroy) either its own storage or the storage of others. This thesis addresses this problem and discusses the design and implementation of a control program in which many such attempts at destroying storage are detected and suppressed. This is accomplished by placing separate control program functions in separate virtual storages. The thesis also expounds upon the notions of protection and security within the operating system itself, and details a quasi capability mechanism that regulates the access to shared objects and ensures the invocation of one control program function by another is authorized.

TABLE OF CONTENTS

PART I: INTRODUCTION

1.1 STATEMENT OF PROBLEM

In many existing operating systems, all supervisory routines are programmed to execute in supervisor state and have read/write access to the entire real machine address space. This design point is an invitation for disaster, for such designs are capable of destroying themselves as well as the user processes they support. Very often the cause of a system failure cannot be detected in a design of this type; dump analysis merely reveals the result of an earlier disaster.

Consider a failure of the following nature. Part of the routine operation of an operating system module involves updating information in a control block. The module is coded to compute the address of a field in the control block, and that computation uses a value that has become invalid. (The source of corruption is not important for this illustration; suffice it to say the module is about to perform a calculation using bad data.) The module performs the address calculation which normally yields the address of the desired control block field. In this case, the resulting address does not point at the desired object. The module continues execution and performs its maintenance activities, which blindly modify storage at the bad address as though it were the desired location.

Erroneous storage accesses of this type inadvertently destroy memory that may be allocated to some other process. The effects of this type

of error are potentially many, and a great deal of time may pass before any of them become visible, if ever. Note that a system failure may not occur. However, a crash may be preferable to the potentially widespread and pernicious effects triggered by the generation and propagation of erroneous results. Finally, if the error eventually does manifest itself as a system failure, any trace of the perpetrator might be long gone.

This sort of destruction is made possible by a design that grants itself a master storage protect key (thereby effectively bypassing key-controlled protection facilities), and the ability to access any location in the central store. Closer inspection of the error process reveals a two-step sequence: 1) It was possible for a module to generate an illegal address, and 2) It was possible for that module to modify storage at that address, which was not allocated to it. It is unlikely that any scheme can be devised to prevent the first step of this sequence, and this is unfortunate, for it would preclude the second step. The reason step one cannot be prevented is because it can occur under two circumstances: the case in which the module is given bad data, which it uses during address generation, or the case in which the validity of the input data is of little relevance, for the module contains logic errors which can generate an illegal operand address.

Until such time as programs can be verified correct, operating systems will be vulnerable to the first step of the error process. However, the second step of the process can be arrested, given that the underlying hardware provides an address translation mechanism for the support of multiple virtual address spaces.

Statement of Problem

1.2 THESIS OBJECTIVES

The goal of this thesis is the design and implementation of a virtual storage operating system that compartmentalizes control program function. Each function will be encapsulated in a separate address space and will be activated in translate (relocate) mode. This design will effectively isolate disjoint components; consequently, programming errors in one component will not damage the storage of another. Unfortunately, there is no sure way to prevent a module from destroying its own storage. But this system will force bounds checking on every storage reference a module makes, and this will prevent the use of unallocated storage that would otherwise go undetected in systems not so designed.

A software mechanism will be provided to facilitate the transmission of shared objects (e.g. a control block) between address spaces that must communicate. Additional program mechanisms will ensure that communication of the type requested is permissible. For example, perhaps two modules of the operating system, e.g. the I/O first-level interrupt handler and process scheduler, are authorized by design to exit to another operating system routine, e.g. the dispatcher. The protection mechanisms provided by this project would grant control transfer requests of this nature. But an attempt to exit to the dispatcher by an unauthorized component would be detected and rejected. This software authentication/authorization technique is used to construct a system that recognizes and enforces a precisely specified control flow relationship among all the modules of the operating system.

As much of the system as is possible will run in problem state, and all system code will execute disabled for asynchronous interrupts. Careful consideration was given to a highly interruptible design. However, the system constructed for this project will not have the sort of realtime characteristics or requirements that would benefit from a design of that type, so a monolithic monitor [2] approach is adequate. While pondering the details of an interruptible structure, I gradually came to the realization that although my design would be able to withstand an interrupt at any point, I would still be queuing a deferred activation record for the required interrupt handler and immediately resuming the interrupted code. At some point in the design, I would serially dequeue each queued request, thereby activating an interrupt handler. The net effect of this technique is serial processing of each interrupt request. A monolithic design achieves the same results minus the queuing logic and save-area maintenance associated with the interruptible scheme. Granted, an uninterruptible design may have a severely negative and/or limiting effect on some areas of operating system performance, e.g. input/output subsystem throughput. However, the performance of the operating system constructed for this project is not severely hampered by this trait and it is expected that the applicability of this approach will be evaluated anew when applying the techniques of this project to a system with stated performance objectives.

1.3 CONCEPTUAL FRAMEWORK AND TERMINOLOGY

Further discussion of this project is likely to become most confusing to even knowledgeable readers unless a certain amount of stage-setting takes place. The following sections explore critical topics in detail, aiming towards a lucid, coherent and cohesive explanation of the thesis objectives.

1.3.1 PROGRAMMING ENVIRONMENT: SYSTEM/370

The term System/370 refers to the definition of a model-independent machine architecture implemented by a wide range of computer systems manufactured by International Business Machines Corporation. This overwhelmingly complex architecture is painstakingly documented in IBM System/370 Principles of Operation [4], to which the reader is ultimately referred. Applications of the architecture are broad, ranging from business to scientific settings. Its widespread use pervades the industrial, commercial and academic sectors.

System/370 defines a myriad of obvious, obscure and arcane features. Its salient features include sixteen general purpose registers, four floating-point registers, sixteen control registers, a program status word, byte-addressable storage, six interrupt classes, problem-state and supervisor-state execution, virtual storage capabilities, and a key-controlled storage protection scheme. References to operands in storage are typically specified in a base register, displacement instruction format. Most instruction formats allow for the use of an index register as well. There are neither indirect nor program-counter

relative addressing modes. The basic architecture incorporates 24-bit addressing, and a fullword of storage is 32 bits, as are each of the 16 general purpose registers. Real storage is logically organized as a linear sequence of bytes, but is physically organized, managed and protected in 4096 byte blocks known as page frames. Pages are grouped into segments, and segment size is customarily 16 pages. A page of storage can contain executable code or data or both; storage is not organized into code and data segments. The architecture also includes various clocks and timers, the ability to selectively disable and enable certain of the interrupt classes, multiprocessor facilities (shared storage interlocks and processor signalling), the ability to enter an idle wait state, and a powerful program debugging facility (program-event-recording). System/370 also defines an extremely robust input/output system based on a channel, control unit, device hierarchy. Up to 4096 I/O devices may be installed and placed under program control.

This paper will discuss particular architectural features at length where appropriate, especially the addressing of main storage and the processing of interrupts. Again, the reader is urged to consult [4].

1.3.2 PROCESS VERSUS ADDRESS SPACE

The notion of a process is of fundamental importance to the discussion of operating systems. However, this convenient notion is usually defined only enough to give the reader an intuitive feeling as to how such an entity might be implemented. For pedagogical purposes it is not necessary to discuss the implementation details surrounding this

all-important concept. But because this project involves the construction of an operating system for a System/370 environment, it is essential to present the details of process implementation.

Attempts to define a process vary in their use of mental devices, such as "the thing which is dispatched", or "the executing image of a program". For "model" definitions, see [1, 2, 6], among others. Along with this notion of executing instructions, we are usually told that the definition includes elements of the machine architecture that comprise process state, i.e. an image of the machine's registers and program counter, and perhaps some logical state and descriptive data, e.g. process name and whether it is running, ready or blocked. This state information is necessary because seldom does a process run to completion during its present dispatch interval; control of the machine might be taken away from it for a variety of reasons, and the interrupted process can be restarted from the saved state information at a later time. This definition, or slight variations of it, are widely used and accepted. The textbook definitions make no attempt to associate any particular function with a process, and rightly so.

While this project supports user processes in the conventional sense, the structure of the actual operating system is not process-oriented but rather is monolithic. That is, this operating system consists of a collection of address spaces (not processes), each performing a precisely specified function. Conceptually, an address space is merely a consecutive sequence of integer numbers together with the specific transformation parameters which allow each number to be associated with a byte location in storage [4]. The integer numbers are otherwise known

as virtual addresses; the numbering sequence starts at zero and increases by one. Each virtual address designates a particular byte of data within an address space. The data may be an instruction or part of an instruction, or an operand or part of an operand. Thus, an address space can be said to contain a program or programs and data. If you like, you may think of an address space as containing the program(s) and data of a process, but a process is much more than an address space. These two terms are not synonymous, as the following section explains.

1.3.3 THE STRUCTURE AND ROLE OF THE ADDRESS SPACE

Multiple address spaces can and do coexist in most computer systems with virtual storage capabilities. Each address space has its own set of virtual addresses, however this set is rarely unique. A particular virtual address belonging to one process usually designates a real main storage location not accessible to code running in other address spaces. However, two or more address spaces can refer to the same real storage locations, each by the same or perhaps different virtual addresses, if the transformation parameters for each address space are properly configured. This is the technique used to accomplish code and/or data sharing. However, this project does not make use of shared code, and data is shared by transmitting it from one address space to another, not by direct sharing of the type described.

An address space exists once a segment table and zero or more page tables have been constructed for it in main storage. Built and maintained by operating system software, these tables serve to establish the relationship between virtual and real addresses; the dynamic address

translation (DAT) facility of the CPU refers to these tables during the address generation process. DAT decomposes each virtual address referred to by code executing in an address space, and maps the virtual address to a corresponding real storage location. The resulting real storage address is a function of the virtual address, the DAT translation algorithm, and the information contained in the translation (segment and page) tables. Notice that at this point, the address space exists but has no intrinsic function; it contains neither code nor data, and therefore certainly does not qualify as a process. However, the address space can own resources: it has "empty" page frames of real storage assigned to it. If another address space contained a loader program, and the loader address space was given access to the same storage frames owned by the newly created address space, it would be possible for the loader address space to place code and data into the storage frames of the new address space. Having done this, the loader could give up access to the frames it momentarily shared with the new address space, and a new executable address space would now exist in the system.

Again, realize that an address space physically exists in a computer system with virtual storage capabilities when a segment table has been created for it. In System/370, a segment table consists of a contiguous sequence of fullwords in main storage. Each fullword is called a segment table entry and has a particular format to be discussed later. In general, though, each segment table entry points at a page table, and a page table consists of halfword entries, each pointing at a page frame of real storage. In this system, each address space is represented by a separate, unique segment table resident in main storage at

all times. Therefore, the address of an address space's segment table in real storage is fixed, and this address conveniently serves as the identity of the address space. (In some operating systems, memory management routines shuttle translation tables between main and auxiliary (disk) storage over time. Under such a scheme the address of a segment table is not fixed. The operating system constructed for this thesis identifies each of the address spaces in the system by segment table address. Having these addresses remain static simplifies programming requirements and avoids the added time required to move the tables and update the changing segment table addresses.)

This is another perfect point at which to compare and contrast process with address space. In general, a process will consist of an address space as described, and will have an associated appendage located elsewhere in real storage for the storage of state information. In addition to state information, the appendage will most likely contain a pointer to the process segment table. Typically, each appendage represents the existence of a process, and the appendages are chained together forming a master list of all the processes in the system. In such a system there is usually a process dispatcher that has access to this master list, and its function is to search the list looking for a dispatchable candidate. Once found, the resources of the machine are turned over to this candidate; the candidate process is said to be dispatched. In popular architectures providing virtual storage capabilities, the act of dispatching a process consists of at least the following steps: 1) loading a segment table register with the address of the process's segment table, 2) restoring the saved machine register

image, and 3) restoring the saved PSW image, simultaneously engaging the DAT facility or its equivalent.

The operating system constructed for this project is monolithic, and each trip through it consists of a series of control transfers initiated by the occurrence of some event. Each complete path really consists of the serial execution of some of the system's components. More specifically, one address space can call another much like a subroutine. The flow of control passes logically, albeit indirectly from one address space to an entry point in another.

It is possible to envision a system in which pieces of the control program are actually (possibly autonomous) processes. That is, each system function would be performed by a process or group of processes, and the processes of different functions could execute concurrently. These operating system processes would be scheduled for execution independent of each other, and while the processes that must execute to complete a particular function may very well have serial execution requirements, the execution of the process chain for a function could be interspersed with the execution of processes for other functions.

A design point of this system simply requires that the transfer of control from one system address space to another be effective immediately. Care must be taken to preserve the serial execution execution requirement of system address spaces; these address spaces must execute exactly and uninterruptibly in their order of invocation. This requirement considerably simplifies the design of the project by avoiding many of the problems with the hypothetical process-oriented system.

Conceptual Framework and Terminology

Finally, unlike a process, the called system address space in this project cannot be dispatched in the traditional sense, as upon each invocation it is effectively a virgin with no associated saved state information.

1.3.4 LINKAGE BETWEEN ADDRESS SPACES

Recall that in this project each system address space provides a particular function. A typical address space consists of one particular module of code and its associated data areas. The code and data reside in the pages of real storage allocated to the address space. Each operating system function will be isolated from every other function in the system; each function will be encapsulated within an address space. The flow of control within the running system will therefore consist of the activation of different address spaces, each performing a well-defined function. For example, suppose the project is running and the address space housing the I/O supervisor (IOS) is currently active. IOS decides that it needs to build a control block representing the I/O request it is processing. It must call the free storage manager--another operating system module--to obtain storage for the control block. Conceptually, it issues what amounts to a CALL instruction for the free storage manager. In reality, the CALL is implemented as a supervisor call (SVC) instruction. In System/370, the SVC instruction generates an SVC interrupt. This means that the address space executing the SVC immediately loses control to an operating system module known as the SVC first-level interrupt handler. The interrupt

handler must save the caller's state (i.e. return linkage) in its save-area, and then activate the called address space. At this point, the called function executes; the caller does not. The "suspended" caller must not be thought of as a blocked process, but merely as a body of code that has called upon the services of another. Eventually, the called function completes and RETURNs to the caller, again by issuing an SVC instruction to effect the transfer of control. Both CALL and RETURN have the effect of immediately activating one address space on behalf of another. Note that the address space being called or returned to is not dispatched in the traditional sense; there is no state to restore which is the principal function of a dispatcher.

The CALL/RETURN primitives in this project perform obvious functions. There are, however, times when a CALL without RETURN is desired. This feature will be known as EXIT. EXIT processing differs from CALL in that return linkage to the caller is not saved; EXIT simply activates the target address space. CALL without RETURN is used when an executing module such as the I/O supervisor completes its function and needs to activate another operating system component, such as the dispatcher or scheduler. It does this by simply EXITing to the next stage of the system.

Notice that a called address space has no saved state information. An address space is not invoked by resuming it, as would be the case if the address space represented a process. Later in this paper, it will be explained how an address space and supporting data structures will work together to support user processes in the traditional sense. However, the actual operating system code consists of many address

spaces which are not treated as processes. Control flows from one address space to the next through the use of CALL, RETURN and EXIT. Invocation of one system address space by another does not result in the creation and dispatch of a process. It merely results in a reconfiguration of the hardware control registers governing DAT; the segment table register is reloaded with the address of the called address space's segment table, and the PSW is loaded with the address space's virtual initial PSW (which specifies DAT 'on'). As a result, a new address space comes to life within the machine. With the exception of parameters, if any, the initial state of this new address space consists entirely of a virtual PSW designating the virtual address of the first instruction of the function implemented by the address space.

1.3.5 COMPARTMENTALIZATION AND ACCESS CONTROL

In order to effectively implement the CALL primitive as specified, there must be something in the system that knows about each address space and its associated initial state. There happens to be an address space containing a program known as the authentication routine. The authentication routine knows the name and virtual entry point address of every system address space. The name of an address space actually turns out to be a mnemonic identifier, but the real identity of an address space is given by the address of its segment table. The primary function of the authentication routine is to determine whether a request is authorized. The modules of this operating system are authorized to issue requests for particular services and resources. These authorizations are static and are assigned as required during the design of the system. The authentication routine protects the system from acci-

dental or malicious attempts to activate certain of the control program functions, and from unauthorized access to protected data structures. This facility forces strict adherence to the design specifications, thereby preventing unwarranted design changes. It also increases the integrity of this system by denying unauthorized access to both code and data.

The following example demonstrates the utility of this feature. It is sometimes convenient to substitute the term "module" for "address space." With respect to this project, each system address space contains one module of the operating system, so the terms can be considered synonymous.

Generally, associated with each callable module of the system is a list of the names of the address spaces permitted to invoke it. Similarly, authorization lists are defined for shared objects (tables, lists, etc.) Suppose, for whatever reason, the designer of the system realized that the only module that needed to call the free storage manager was the I/O interrupt handler. This restriction becomes part of the design of the system, and the authentication routine can be used to strictly control access to the free storage manager.

The technique used to enforce access rights is simple. An entry for the free storage manager is placed in a table owned by the authentication routine. Further, the entry identifies the I/O interrupt handler as an authorized caller. The I/O interrupt handler's identity is internally represented by the address of its segment table, as that value uniquely represents a particular address space. When an

address space issues a CALL to a module in another address space, it loses control (via the SVC interrupt handler) to the authentication routine. The caller's identity is accessible to the authentication routine, i.e. it knows the address of the caller's segment table. To validate the request, the authentication routine locates the table entry for the called address space, and subsequently searches its authorization list for the caller's identity (the address of its segment table.) If found, the CALL is performed.

The authentication routine can be used to control access to virtually any object in the system. Consider a trace table. Many operating systems have a region of storage set aside for tracing. As the system runs, various modules place trace entries into the table. The entries provide a very recent history of system activity, which is sometimes useful during failure analysis. Naturally, the trace table should be carefully protected to ensure its contents are accurate. If any part of the trace were inadvertently destroyed, its contents would be useless to the person reading the system dump. Obviously, the trace table is a perfect candidate for protection. In the system defined by this project, the trace table can be fully protected simply by adding an entry for it to the authentication routine's authorization table. The trace table is then assigned its own address space and is managed by a small, well-defined and exhaustively tested program. Every module that needs to record trace information is added to the authorization list for the trace table address space. Then, modules simply CALL the trace facility, passing the trace entry as a parameter. Because the trace table is isolated in its own address space, it cannot be accessed di-

rectly by any program in any other address space. Moreover, only authorized modules are permitted to create trace entries.

The management of some shared objects present problems. Consider a shared linked list. When a module requests access to a list, the pages of storage allocated to the list can easily be attached to the module's address space at the virtual address of the first available page. Assuming the structure of the list is correct, the module can examine and manipulate the list, e.g. it can add and delete elements. Customarily, the list elements would be chained together by virtual address, which, as far as code running in the address space is concerned are real addresses. The problem with this sort of list structure is that once the list is constructed (or more specifically, once it contains more than one element), it becomes non-relocatable. That is to say, when another module (or the same module) requests the list, it must be attached to the address space at the same virtual addresses, because it has specific virtual addresses embedded in it. Because the size of an address space is unknown/unlimited, it is not a guarantee that the same pages of an address space will always be available over time for such purposes; it may turn out that the pages may be allocated to other objects. If this is the case, linked lists must be managed differently. The problem can be solved by placing the list in its own address space, along with code that manages the list. Modules that need access to the list simply call the list address space, passing the required information and specifying the desired list operation, e.g. GET NEXT, ENQUEUE, FIND, INSERT, etc.

In general, then, this project will provide access to shared objects in the following manner: An object, along with code to manage it will

reside in an address space, and other address spaces will be authorized to request execution of the code that performs the desired operation on the object. This method is discussed in section 2.3.7. It is effective because the caller has access to neither the object nor the the code that manipulates it. Technically, an address space has execute-only rights to the code that manages the object. Note that there may be several operations defined on an object, each provided by by a separate entry point in its address space. An entry point might perform a read or write operation on an object. For example, a linked list enqueue operation writes a new list element, while a list find operation performs a read-only search and returns the requested item. So object access rights, while not explicitly encoded are implicitly defined by the semantics of the object manipulation entry points an address space is authorized to invoke. Each operation defined on an object in such a way may be thought of as an authorized capability.

The question arises as to how an unauthorized request for a service or object is handled. Presumably, a module making a request cannot proceed until the request is satisfied. In this project, the authentication routine is used exclusively by modules of the operating system. If a module is denied a service or object, one must assume that the module cannot complete its function, as it would not be asking for things it absolutely did not need. Since the module cannot continue, the operating system cannot possibly continue. Therefore, an unauthorized request results in abnormal termination of the operating system.

Clearly there are several key advantages to access control and compartmentalization. Modules are prevented from destroying other

modules or data areas. Services and data can be protected from misuse. Design specifications can be rigidly enforced. This third point is of importance not only during the initial development phase, but also later when change teams are correcting new problems or adding functional enhancements. However, it is important to point out that the integrity of the system lies in the hands of the person changing the authentication routine's table. For this reason, it is important that the original design be well-documented in a particular manner. Suppose that a change team wants to add a new function. Its implementation requires that a module be permitted to make a call that was heretofore unauthorized. At any point in time, the set of defined authorization lists describes legal relationships between the components of the system. This set has a complement which can be partitioned into two disjoint subsets: a set which contains authorization lists which would be valid but have not yet been defined (both legal extensions to existing lists and new lists) and a set which contains lists which, if admitted would create deadly exposures.

This complement set must be considered when adding new function to any system using the design described by this thesis. One might be tempted to casually authorize a caller, but there may be good reason why one module should not call another. The developers must thoroughly research why the proposed call is not currently authorized, and how the proposed authorization will affect the design. This research is necessary to avoid straying from limitations and/or restrictions present in the original design. Alterations must be clearly documented; well-documented restrictions will accelerate the research phase.

Conceptual Framework and Terminology

Finally, a word about the protection provided by this project in comparison with the protection provided by certain high-level modular programming languages, e.g. Concurrent Euclid (CE) [3]. The block structure of these languages allows the programmer to create self-contained modules. Any code and/or data defined in a module is invisible to other modules written in that language. The syntax of the language requires that any variable or entry point that must be accessed by a module other than that which defines it be "exposed". In CE, an exposure is specified by the EXPORT statement. This feature provides a degree of protection and is necessary but not sufficient to provide access to a shared object. Any module that needs an exposed object must formally request access to that object. In CE this is accomplished by IMPORTing an object exported in another module. Further, shared variables can be imported read-only or read-write. Thus, in CE procedures and data may be shared by exporting and importing them where needed. Modularization and access capabilities are specified by the syntax and enforced during compilation. No user-written CE code can subvert the language's protection mechanisms.

In contrast, this project is written in an assembler language that does not define any protection schemes. Yet the degree of protection provided by this operating system is much stronger than that of CE. For example, a CE program could call an assembler routine. The latter could inspect and/or alter any portion of the CE system since they both reside in the same address space. It is trivially easy to subvert the inherently secure CE system since no run-time authorization techniques exist. In this project, snooping is impossible since each function is isolated from every other function. CE also does not control access

to exported objects. An exported procedure or variable can be imported by any CE module. This project requires that address spaces be authorized by name to activate other modules. And while data items themselves are not directly shared, the code which manages them is. Similarly, access to that code is limited to authorized address spaces, and this effectively regulates access to the data items. This technique allows the invention of capabilities that must be assigned to address spaces.

1.3.6 INTER-ADDRESS SPACE COMMUNICATION

Undoubtedly the system address spaces in this project will need to communicate amongst themselves. Since the storage of each module is isolated from every other module, it is not possible to pass data through common, or shared areas. This project facilitates the transmission of data from one address space to another through the use of a special system address space called the transmitter module. Each operating system module is designed to have one page of its address space designated as a parameter buffer. When a module wants to call another address space and pass it data, it places the outbound data in the buffer and CALLs the target address space; the CALL is issued with an option that indicates data is to be passed by the transmitter. The transmitter module is special in that it can gain access to the caller's and receiver's parameter pages. Once these pages have been attached to it, it issues a single instruction to copy the outbound (source) page to the inbound (sink) page. These parameter pages have no specific format, although caller and receiver obviously must agree on their use. Note that data can be transmitted from one address space to another only

during the CALLing process. It is neither necessary nor desirable to permit a module of this system to pass data to an address space for later use (as might be done in a message-based approach).

1.4 THE SYSTEM FROM A USER'S PERSPECTIVE

Operating systems are built to make unusable hardware usable, so that it may run work, or computer jobs for people. Many operating system components and functions are hidden from the casual user; they are "under the hood", so to speak. However, certain OS features or subsystems present a usable view of the system--a personality--to those who come in contact with it. Most systems provide interfaces catering to the human needs of the programmer/user; the interface makes the machine usable by the outside world. The fact that the operating system exists to manage resources is often overshadowed by the view that the operating system exists to make the machine available to users.

This project is concerned primarily with the structure and function of the operating system. It is an experiment in the use of compartmentalized storage. The operating system supports user processes, but they do not see a functionally rich OS layer beneath them. In this project, the user processes are merely a means to an end, supplying a constant background load that exercises the components of the system. Simply, the users of this system are nothing more than (perhaps simpleminded) address spaces that are dispatched to consume time and issue requests for services from the control program software. They are true processes: They have saved state information, their execution

can be interrupted and safely resumed, and they compete with one another for the resources of the machine.

1.5 IMPLEMENTATION REQUIREMENTS

This project is a control program designed to run on a machine with virtual storage capabilities. Naturally, it needs access to the entire range of processor facilities. This includes execution in supervisor state and the issuing of many privileged instructions. Customarily, and for good reason, institutions do not make their real hardware available to students for projects of this nature. For this reason, the project will run under the IBM Virtual Machine/System Product High Performance Option Control Program (VM/SP HPO). VM/SP HPO provides each user a virtual machine environment that furnishes the full spectrum of System/370 architectural features. See [7] for an introduction to the virtual machine concept and capabilities provided by this operating system. [8, 9, 10] provide in-depth coverage on the use and method of operation of VM/SP HPO. Briefly, VM is a control program that supports many concurrent users, each running the operating system of his choice. Each user can run any IBM or user-written operating system on top of VM, including VM itself. (Taking this concept to its extreme, it is even possible to run VM on top of VM on top of VM, etc., up to many levels.) VM is an extremely powerful operating system development tool, as many OS developers can each be modifying and running their own copy of the OS concurrently, instead of each waiting a turn to run their system stand-alone on a real machine.

This project will be written entirely in the IBM OS/VS-DOS/VSE-VM/370 Assembler Language [5], and will run on top of VM as described. Proof of proper operation of the system is likely to be in the form of instruction traces, trace table output and storage dumps.

PART II: DESIGN AND IMPLEMENTATION

2.1 VIRTUAL STORAGE: THE DESCRIPTION OF AN ADDRESS SPACE

Much of sections 2.1.1 and 2.1.2 provide a condensed description of the dynamic address translation process defined by System/370 machine architecture. Where appropriate, the use of these features by the operating system specifically defined by this thesis is described. For exacting detail the reader is referred to [4].

2.1.1 DAT TABLES: DATA STRUCTURES DESCRIBING AN ADDRESS SPACE

In System/370 an address space is the set of all storage addresses available to a program in execution. This set consists of either virtual or real addresses depending on the mode of CPU operation. If the CPU is operating in translate mode, each program-generated storage address is considered virtual and is subject to a virtual-to-real translation process before main storage is accessed. In System/370 this translation is performed by the hardware but requires software-created translation tables present in main storage. If the CPU is not operating in translate mode, each program-generated storage address in not subject to translation of any sort. Instruction and operand effective addresses generated in this mode are used in their original form to access storage. Hence, DAT tables are not needed. Out-of-range addresses will be properly detected by the storage controller; notification of this exceptional event will be posted by an interruption.

In this project, nearly every module of the control program resides within an address space and executes in translate mode, i.e. executes with "DAT on". Consequently, each address space is defined not only by the code and data which constitute its functional properties, but also by its DAT tables through which each and every program-generated effective (virtual) address is mapped to a corresponding real address.

The DAT tables consist of one segment table and several page tables. The format of each (see [4]) varies slightly in accordance with the real storage features implemented by a particular CPU model. The relationship between these tables and other essential DAT components defined by System/370 is shown in Figure 1 on page 27.

This project is implemented on a processor which organizes its storage in identically-sized segments, each consisting of 16 pages. Further, page size is fixed at 4096 bytes (4K), making the size of a segment 64K bytes. Specification of page and segment size is referred to as the translation format and is established by a program-supplied bit setting in control register 0. (Although there are sixteen control registers in System/370, only two are related to DAT and are of interest here. These are control registers (CREGs) 0 and 1. Each System/370 control and general purpose register is 32 bits wide. Bit positions are numbered from 0 to 31 starting with the LEFTMOST bit.) The translation format is specified by the value of bits 8-12 of control register 0. Setting bit 8 to one and bits 9-12 to zero informs DAT that the machine storage is organized into 64K segments with a 4K page size. This information is used by DAT during virtual address dissection, as it explicitly identifies which bit groupings in a 24-bit virtual address

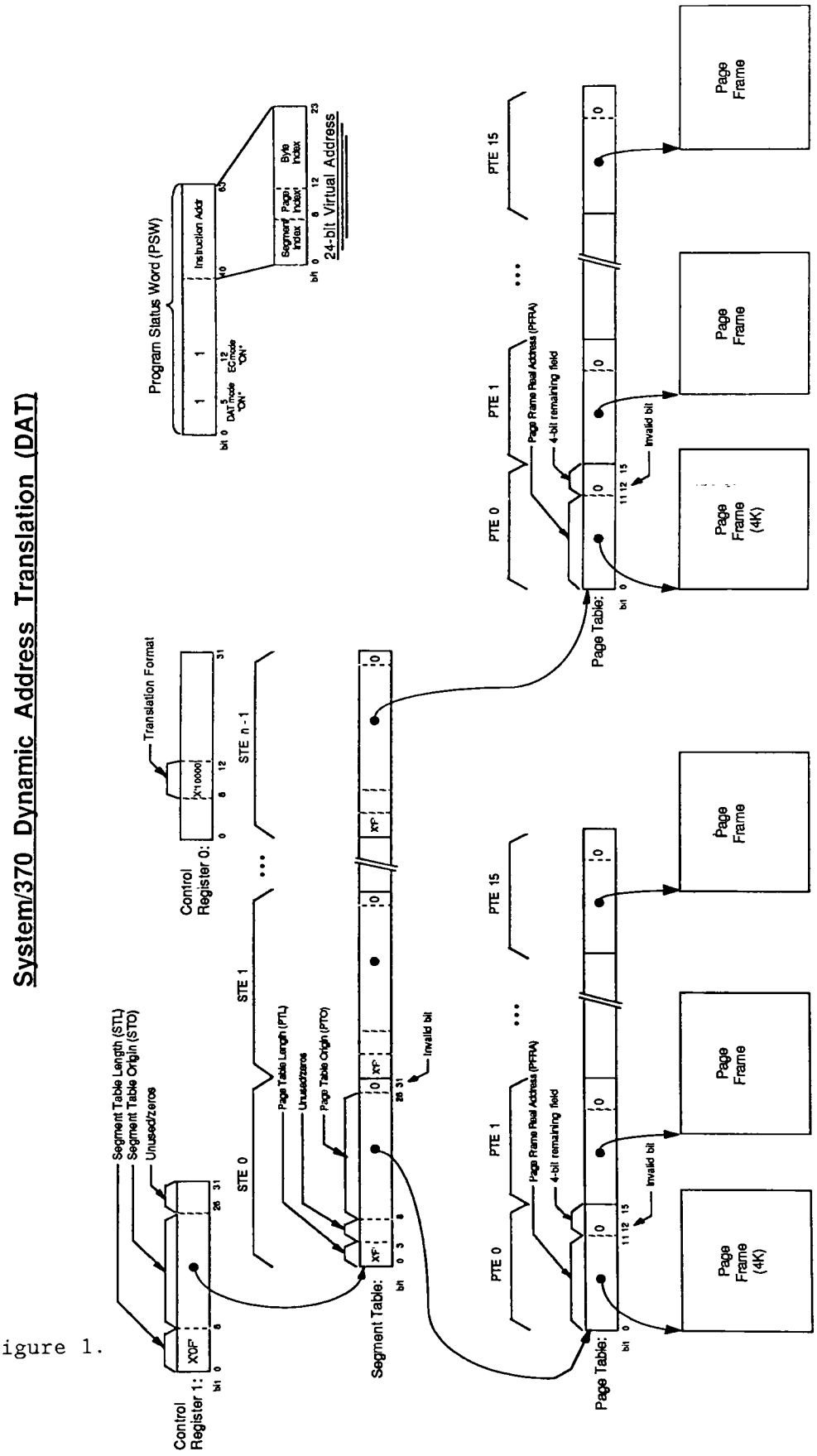

System/370 Dynamic Address Translation (DAT)
Program Status Word (PSW)
Instruction Addr
24-bit Virtual Address
Segment Index
Page Index
Byte Index
Control Register 0:
Translation Format
Control Register 1:
Segment Table Length (STL)
Segment Table Origin (STO)
Unused/zeros
STE 0
STE 1
STE n-1
Segment Table:
Page Table Length (PTL)
Page Table Origin (PTO)
Invalid bit
Page Table:
PTE 0
PTE 1
PTE 15
Page Frame Real Address (PFRA)
4-bit remaining field
Page Frame
Page Frame (4K)

Figure 1.

identify segment, page and byte indexes. Control register alteration is a privileged operation which can only be performed in supervisor state. The translation format in CREG 0 is established once during system startup by a privileged instruction.

A segment table consists of a contiguous sequence of fullwords in main storage. The length of the segment table determines the size of the address space it represents. Control register 1 points at a segment table and specifies its length in units of 16 segment table entries (STEs). More precisely, bits 0-7 of control register 1 designate the primary segment table length (PSTL), bits 8-25 designate the address of the segment table in main storage and are referred to as the primary segment table origin (PSTO), bits 26-30 have no assigned meaning and bit 31 is a space-switch-event-control bit. Use of the word "primary" in these descriptions hints at the possibility of "other" segment table designations. In fact, a System/370 feature known as Dual Address Space allows the specification of two simultaneously active segment tables--primary and secondary--which permit an application instruction to access information in two different address spaces. Additional control registers designate secondary segment table and address space control information. This project does not make use of Dual Address Space so there is no need qualify entities "primary"; subsequent references omit this term. Along these lines, the space-switch-event-control bit is meaningless outside of Dual Address Space and is ignored by DAT.

It should be clear that an address space has a minimum size. Segment table length is indicated in units of 16 STEs. Each segment can have

up to 16 pages, so a minimum-length segment table specifies an address space consisting of 256 pages, or a one-megabyte virtual storage. Note that an 8 bit STL allows for a segment table specifying a 256*16*16 = 65536 page (256 megabyte) address space. However, the 24-bit virtual address in System/370 allows for a maximum 256*16 = 4096 page (16 megabyte) address space. Thus, in practice the STL ranges from X'00' to X'0F' with each increment between representing an additional megabyte of virtual storage.

Notice the STO consists of 18 bits. During translation DAT concatenates the STO with six zero bits on the right to form a 24-bit real address. This address specifies the real main storage address of the segment table. It should be clear that every segment table must begin on a 64-byte boundary This alignment requirement is consistent with the minimum size of a segment table (64 bytes).

Each STE has the following format: Bits 0-3 specify the page table length (PTL), bits 4-7 are ignored and should be zeros, bits 8-28 are known as the page table origin (PTO) and point at the page table associated with this segment, bit 29 is the segment-protection bit, bit 30 is the common-segment bit and bit 31 is the segment-invalid bit. Neither bits 29 nor 30 are used by this project. The PTL specifies the length of the page table in terms of the number of pages in the segment. PTL ranges from X'0' to X'F' and specifies the number of page table entries (PTEs) in this segment. Each page table entry is two bytes long and will be discussed shortly. A fully-configured page table consists of 16 PTEs and occupies 32 bytes of real storage. During translation, DAT generates the 24-bit real address of the page table by concatenating

three zero bits on the right side of the 21-bit PTO. Consequently, every page table must begin on an eight-byte (doubleword) boundary. This boundary alignment is not consistent with the minimum size of a page table, as was the case with segment table alignment.

If a segment has no associated page table, then the segment-invalid bit (bit 31) of the STE describing that segment must be set to one. This bit is referenced by DAT and must be zero for normal translation to proceed. If DAT discovers an attempt to reference an invalid segment, it stops the translation and posts a program interrupt indicating segment translation exception. This condition might be indicative to the operating system of a segment fault condition. In this project, certain of the segments in a segment table are simply unused/unassigned but must be represented because of the minimum segment table length requirement. These uninstalled segments are denoted by turning on the segment-invalid bit in the corresponding STE.

As mentioned earlier, a page table consists of from one to sixteen page table entries. Certain segments may only consist of one page, as is the case numerous times in this project. Thus, the PTL in the segment's STE is set accordingly (to zero). The format of each sixteen bit PTE is as follows: Bits 0-11 specify the page frame real address (PFRA, also known here as the frame prefix), bit 12 specifies the page-invalid bit, bits 13-14 are the extended-storage-address bits and are not used in this project, and bit 15 is unassigned and ignored. The frame prefix cites the leftmost 12 bits of the address of a page frame origin in real storage. During translation, DAT fetches the frame prefix from the PTE and appends 12 zeros to the right of it to generate the 24-bit address

of a page frame in real storage. Since page size is 4096 bytes = 2**12, it is not necessary to store the low-order 12 zeros of the frame address in the PTE. It is also clear that boundary alignment for a page frame is 4K bytes, which is in accordance with the size of a page. If a page of real storage is assigned to this PTE, the page-invalid bit should be zero. If DAT detects a page-invalid bit set to one during translation, it terminates the translation process and posts a program interruption citing a page translation exception condition (i.e. a page fault). This project makes extensive use of virtual addressing but does not implement a complete virtual storage subsystem, i.e. it does not employ memory management techniques that move pages and/or segments of data between primary and secondary memory. Rather, all code and data are permanently assigned to pages of storage, so the occurrence of any page or segment fault interrupt would indicate a severe programming error.

This completes the description of the data structures used by dynamic address translation. Together these structures define what is referred to as the address space of a program running in translate mode. The length of the segment table defines the set of virtual addresses belonging to the address space. However, it is possible that not every virtual address in this set maps to a real address. This is because certain of the segments represented in the segment table may not be installed and are therefore marked invalid. And an installed segment may contain less than 16 pages--either the page table is "short", as indicated by the PTL, or perhaps the PTL designates a full-sized page table but certain of the PTEs are invalid. Consider a full 16 megabyte address space defined by a segment table consisting of 256 STEs. It

could be that every STE is marked invalid, in which case none of the 16 million virtual addresses in the address space map to a real address. Thus, the size of the address space is best thought of in terms of the size of the set of virtual addresses defined by the segment table. The size of an address space should not be confused with its main storage allocation. It simply declares the range of virtual addresses DAT should attempt to translate for this address space.

In this project, each address space is represented by a segment table of minimum size, which places the size of the address space at 1 megabyte. Nearly every address space consists of two pages; the largest address space consists of only four pages. The one megabyte minimum is overkill, and real storage is wasted supporting unneeded STEs. However, the design must conform to the specifications of System/370, which are not ideally suited to the structure of this operating system. The Analysis portion of this paper (section 2.5) suggests System/370 alterations which, if applied would substantially improve the performance of this project and unify its design.

2.1.2 USING DAT AND DAT TABLES TO ACCESS STORAGE

DAT tables sketch the relationship between segments and pages. Control register 0 defines segment and page size which establishes the format of a virtual address. Control register 1 points at a segment table, and the current machine PSW specifies whether DAT is on or off. The DAT translation algorithm uses all of these items to translate a program-generated 24-bit virtual address into a 24-bit real address. Here, a virtual address is any address generated explicitly or implic-

itly by a running program, used to fetch an instruction or operand or as the target of a store type instruction. An example of implicit virtual address generation is the calculation of the next sequential instruction address computed by the instruction fetch microroutine at the end of each fetch cycle. The current PSW next address field is updated with this value. Explicit virtual address generation includes operand and branch addresses resulting from base-index-displacement address calculations. Each of these storage effective addresses is a virtual address that must be translated.

Dynamic address translation is collectively implemented by a set of processor microroutines. When the current PSW specifies the extended-control (EC) mode (bit 12, the EC mode bit, is one), the machine is running in EC mode and PSW bit 5 is the DAT mode bit. If the DAT mode bit of an EC mode PSW specifies DAT on (bit 5 is a one), all effective addresses are considered virtual and must be submitted to DAT for translation. (In basic-control (BC) mode (bit 12 = zero), DAT is not defined and is not available. BC mode is synonymous with System/360, and has been retained as a mode of operation on machines with EC mode capabilities merely to maintain compatibility with programs written to run under System/360. Dynamic address translation is just one of the extensions to System/360 found in the extended-control mode of operation.)

When presented with a virtual address, DAT begins the translation process shown in Figure 2 on page 34.

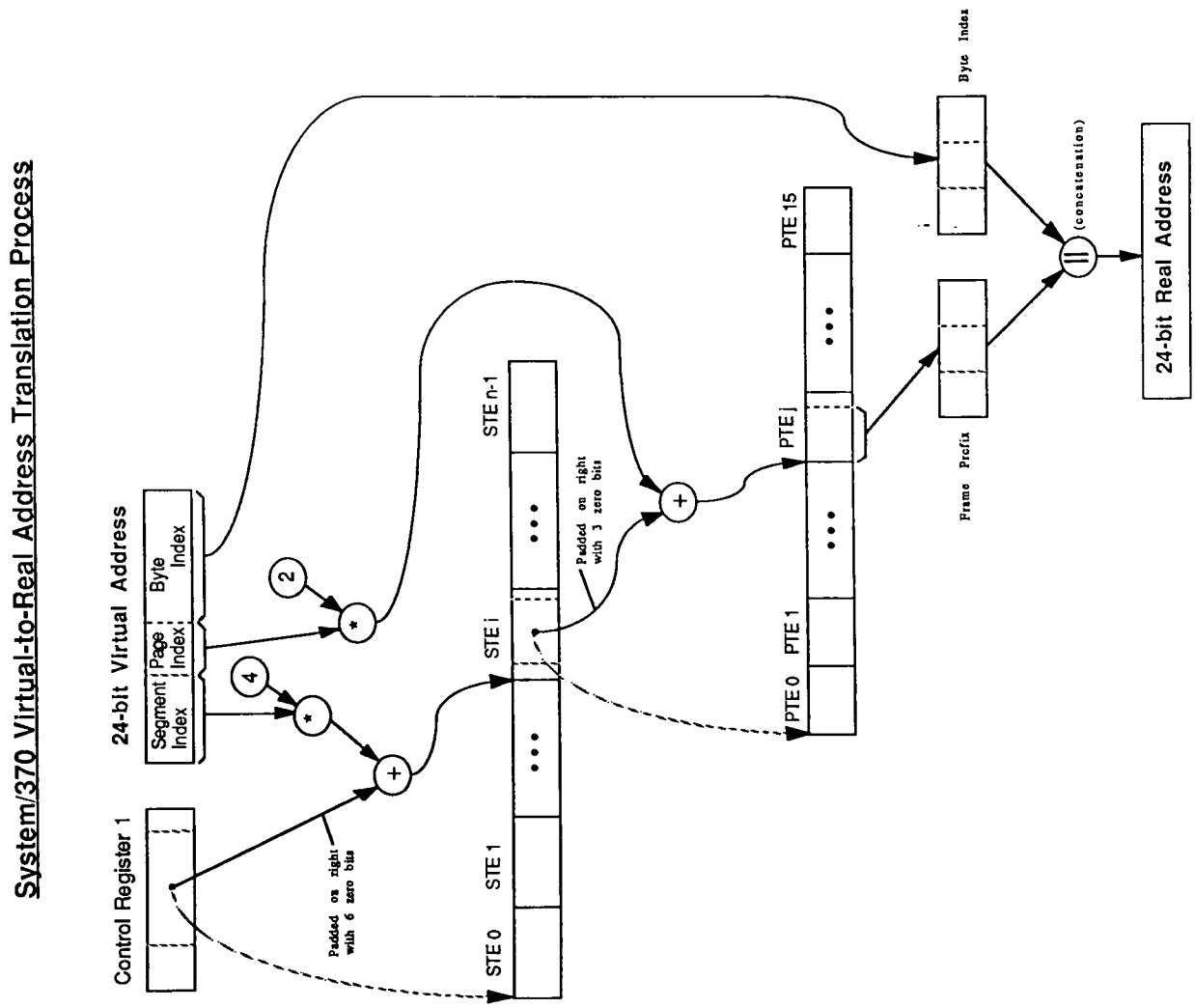

System/370 Virtual-to-Real Address Translation Process
Control Register 1
24-bit Virtual Address
Segment Index
Page Index
Byte Index
Padded on right with 6 zero bits
STE 0
STE 1
STE i
STE n-1
Padded on right with 3 zero bits
PTE 0
PTE 1
PTE j
PTE 15
Byte Index
Frame Prefix
(concatenation)
24-bit Real Address

Figure 2.

Using the translation format specified in control register 0, DAT determines the format of a virtual address. This project makes use of 64K segments and 4K pages. Under this scheme, a 24-bit virtual address has the following format: Bits 0-7 constitute the segment index, bits 8-11 constitute the page index, and bits 12-23 constitute the byte index. Now, DAT isolates the segment index. This value (which incidentally can range from X'00' to X'FF') will be used as an index into the segment table pointed at by the STO in control register 1. Before indexing into the segment table, DAT ensures the segment is within the table. It does this by comparing bits 0-3 of the segment index with bits 4-7 of the STL in control register 1 (remembering that bits 0-3 of the STL are meaningless as they designate megabytes of storage not available under a 24-bit addressing scheme). If bits 0-3 of the segment index exceed bits 4-7 of the STL, the virtual address is not defined within the address space, i.e. it is out-of-bounds. The translation process is terminated and a program interruption is generated indicating segment translation exception.

When the segment index is legitimate, it is multiplied by the length of a STE (4 bytes) to convert it into an offset. Now, DAT appends six zeros to the right of the 18-bit STO in control register one, thereby forming the 24-bit real address of the segment table. To this address it adds the offset to form the real address of the STE for this segment. Then, DAT fetches the 4-byte STE from storage. If the segment-invalid bit is one, the translation process is terminated and a program interruption is generated indicating segment translation exception.

Now DAT isolates the page index. This value (which ranges from X'0' to X'F') will be used as an index into the page table for this segment. But first, DAT compares the 4-bit page index with the PTL (bits 0-3 of the STE). If the page index exceeds the PTL, the referenced page has no corresponding page table entry, i.e. the page address is undefined. The translation process is terminated and a program interruption is generated indicating page translation exception. If the page index is within bounds, it is multiplied by the length of a page table entry (2 bytes) to form an offset to the PTE. Then, DAT appends three zeros to the right of the 21-bit PTO in the STE to form the 24-bit real address of the page table. The offset is added to this value to generate the real address of the PTE; DAT fetches the PTE at this address. If the page-invalid bit is one, the translation process is terminated, and a program interruption is generated indicating a page translation exception condition.

Finally, DAT isolates the byte index which ranges from X'000' to X'FFF'. From the PTE it extracts the frame prefix. The byte index is concatenated with the frame prefix on the right, and the result is the 24-bit real address corresponding to the program virtual address.

This completes the description of the System/370 dynamic address translation process.

2.1.3 WHY DAT IS EFFECTIVE AS AN ISOLATION TOOL

Any program running in translate mode is isolated from any storage not mapped by its translation tables. Among the many advantages of virtual

storage, the privacy it offers is of tremendous importance especially in a multiprogramming environment. Virtual storage has long been used to wall off user address spaces from the system as well as one another. But it can also be used to isolate distinct operating system functions. As is the case for multiple users, each module of this operating system is encapsulated in its own private address space. Each module has its own set of DAT tables which map its function into only those pages of storage assigned to it. Since there is only one address space active at any given time, all virtual-to-real translation proceeds through the DAT tables allocated to the active address space. This makes it impossible for code running in one address space to reference the storage of any other module (address space) in the system.

In System/370, the only other means of partitioning control program structure is through the use of storage keys. Key-controlled protection offers store and fetch protection, but its biggest problem is the size of the key space. The System/370 storage key contains several fields: Access control bits, reference bit and change bit. Of these, the four access control bits are taken together to form the protection key for the page of storage to which the six bit storage key is assigned. A total of sixteen different protect keys can be formed from the four access control bits. There simply are not enough possible combinations of protection keys to allow them to be used satisfactorily as an isolation tool. On the other hand, there is no practical restriction on the number of address spaces defined in a system. And if necessary, the storage within each address space can be further subdivided through the use of up to sixteen storage protect keys.

Virtual Storage: The Description of an Address Space

Now that we have an effective way to partition the control program, the system becomes impervious to external damage from other parts of itself. And just as important, a partitioned design allows the development of a control program that prevents its components from misuse, thereby rigidly enforcing system design specifications. The details of the control program mechanism that does just that will be covered soon.

2.2 RELATIONSHIP AMONG SYSTEM ADDRESS SPACES

2.2.1 HOW SYSTEM STRUCTURE IS ISOLATED AND CONTAINED WITHIN ADDRESS SPACES

Many operating system functions are disjoint. This project encourages code isolation by functional component. Thus, division of labor within the system becomes quite apparent. Independent functions must be cleanly separated so that they may be housed in separate address spaces. To partition a control program in this way requires the designer to, at the very outset of the design phase, very carefully identify every functional component needed by the system. Further, the scope, i.e. service(s) of each component must be clearly defined, as this helps to delineate the interface between modules of the same component that are, for purposes of protection housed in separate address spaces.

Once it could be shown that the proposed compartmentalization technique was viable, the design of this project proceeded using this design methodology. The most important task was that of determining exactly which functional components a rudimentary control program exemplifying

the isolation property would consist of. The design of the system is covered later in this paper; we need not concern ourselves with specific functions here. Suffice it to say this has been done and we have in front of us all the pieces of the operating system that will be housed in separate address spaces. This leaves us to deal with two fundamental issues: 1) How control passes from one address space to another, and 2) How address spaces are able to pass shared objects amongst themselves.

2.2.2 HOW CONTROL PASSES FROM ONE ADDRESS SPACE TO ANOTHER: CALL, RETURN AND EXIT

Earlier in this paper, the Authentication Routine (or simply, Auth) was identified as the vehicle facilitating transfer of control among system address spaces. Auth implements three transfer-of-control primitives: CALL, RETURN and EXIT. The semantics of these operations have already been sketched out. Now Auth's precise implementation of each will be discussed.

Consider one address space that needs to call upon the services of another. We are not interested in how the address space issuing the CALL got control of the processor; that could have happened by a CALL or an EXIT. In any event, a module executing within a system address space places the name of the address space it wishes to call in register zero (R0). Then it simply executes a CALL SVC. This SVC (or any SVC for that matter) results in the execution of the Auth address space. Auth is a highly-specialized system address space which receives control as

a result of an SVC instruction. The details on how Auth gets "activated" are discussed later in the section on system design.

Auth has a table which identifies every system address space by a 4-byte symbolic name. Having determined that a system SVC has been issued (as opposed to a user-issued SVC), Auth decodes the SVC request. It determines the SVC was a CALL, and initiates a search of its system address space table looking for the 4-byte symbolic name passed by the caller in R0. If the search for the address space fails, i.e. that name is not registered in the table, Auth terminates system operation.

Assume Auth locates a table entry for the desired address space. The table entry contains the real address of the segment table for the called address space, otherwise known as the target segment table origin, or target STO. It also contains the PSW key under which the called address space executes, and it contains the virtual address of the first instruction of the function in the address space. Now Auth has everything it needs to complete the CALL request: It knows the real address of the segment table defining the function's address space, it has an initial value for the instruction address portion of the PSW, and it knows what storage protect key the address space should operate with. Given the latter two elements, a PSW image can be constructed incorporating this information and specifying EC mode with DAT on. The structure of a system address space and its association with Auth is shown in Figure 3 on page 42 (the diagram also illustrates a user address space and its association with the user control block, covered later). Provided that Auth has the ability to cause both control register 1 to be loaded with the target's STO and the current PSW to be

Relationship Among System Address Spaces

replaced with the image constructed as described, the called address space can be given control of the processor. However, Auth has more work to do to properly carry out the CALL request.

Facilitating transfer of control involves the maintenance of linkage information so a called address space can return to its caller. The semantic actions of system CALL, RETURN and EXIT SVC requests are modeled within Auth by the constructs GOSUB, GOBACK and GOTO, respectively. Each of these makes use of a stack also defined within Auth. It seemed natural to re-express each high-level request in terms most meaningful to Auth. Auth knows that in order to complete a CALL it has to stack a savearea containing the caller's state information then activate the called address space. This stack push-activate combination is collectively referred to as a GOSUB operation, as it implies the called routine functions as a subroutine which eventually returns to its caller. Similarly, a RETURN operation consists of restoring the caller's saved state information from the top stack entry, popping the top entry and activating the address space just popped. This restore-pop-activate operation is known as GOBACK because it lets a called address space return back to its caller. Before discussing GOTO and the remaining details of GOSUB and GOBACK, RETURN and EXIT will be described.

EXIT is similar to CALL in that it causes control to be passed to a target address space. It differs in that an address space which receives control via EXIT cannot RETURN to the address space that activated it--no linkage to the activating address space is saved. EXIT is used in this system by a module that has completed its function, and the next phase of system operation requires that a new, separate func-

Address Space Structure

System Address Space

-or-

User Address Space

Defined by an entry in Auth's table of system/user service address spaces. The entry provides the following:

Each user is represented by a control block with the following general structure:

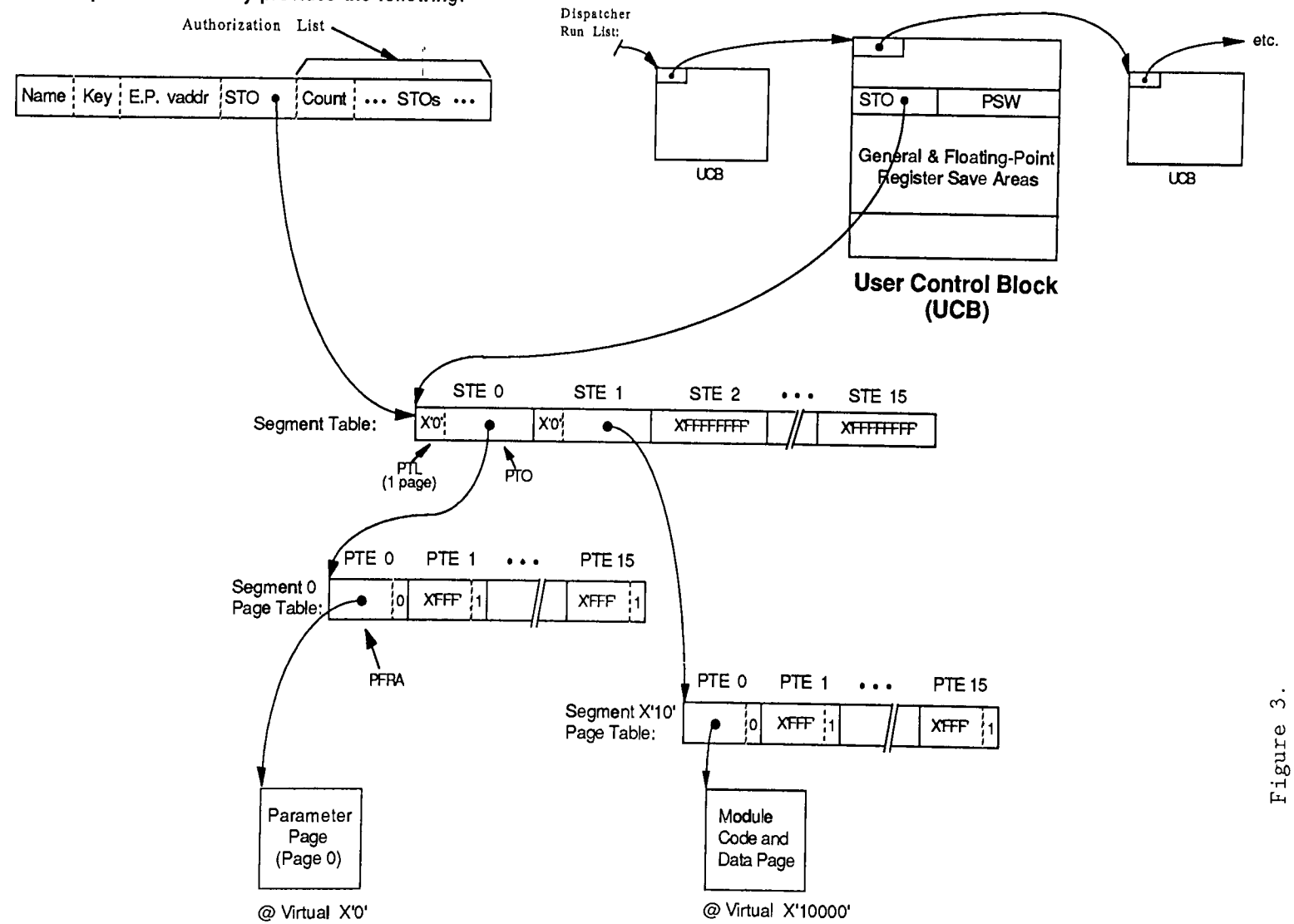


Figure 3. Relationship Among System Address Spaces

tion be activated. The need for EXIT will become apparent later in a discussion of overall system structure. An EXIT request is similar to CALL in that the "caller" places the name of the target address space in R0 and executes an EXIT SVC. Auth receives control, searches its table for the target, shuts the system down if the target is not found, otherwise performs GOTO processing.

RETURN processing differs from CALL and EXIT in that the address space doing the return does not specify the name of the address space to which it returns. It returns, naturally, to its caller, and that information is saved on the stack. The RETURN SVC therefore results in the activation of Auth which immediately initiates GOBACK processing.

Analysis of the semantics of RETURN and EXIT reveals important special cases which Auth must enforce. Notice that a module which receives control via EXIT must itself give up control by EXITing to another address space; it cannot RETURN because linkage to the activating address space is not saved. Also notice that a module which received control as a result of a CALL may only give up control permanently by a RETURN. Of course, it may lose control temporarily by issuing a CALL request; this is perfectly acceptable. But it MUST eventually RETURN to its caller; neither it nor any of the address spaces it calls are permitted to EXIT. If this were to happen, the linkage stack would contain saveareas for system modules that can never be reactivated, indicative of pending work that cannot be completed. EXIT signifies a major functional shift in system operation is about to take place, and this must not happen until the current functional component has completed.

Relationship Among System Address Spaces

Fortunately, these special cases can easily be detected by Auth. The linkage stack must be empty when an EXIT is issued, otherwise Auth has detected incomplete work (somewhat akin to "dead code"). Also, if RETURN is issued and the linkage stack is empty, Auth has detected a module invoked via EXIT trying to return to its (unknown) caller. If either situation is detected, Auth terminates system operation.

Finally, GOSUB, GOBACK and GOTO can be precisely specified: 1) A CALL SVC invokes Auth GOSUB processing. GOSUB pushes the caller's STO, PSW key, PSW instruction (resume) address and general purpose registers on the stack. (If the stack is full before the push, Auth terminates the system.) Then Auth activates the target address space using the information extracted from the system address space table. 2) A RETURN SVC invokes Auth GOBACK processing. If the stack is empty, Auth terminates the system. Otherwise Auth pops the top stack entry, restores the general registers from the popped entry and reactivates the address space specified by the popped STO with the popped PSW key at the popped resume virtual address. 3) A EXIT SVC invokes Auth GOTO processing. If the stack is not empty, Auth terminates the system. Otherwise Auth activates the target address space using the information extracted from the system address space table.

Notice that the final phase of GOSUB and GOBACK processing is really the GOTO function. Auth takes advantage of this and implements GOSUB as a push-GOTO combination, and GOBACK is implemented as a pop-GOTO sequence.

Relationship Among System Address Spaces

2.2.3 HOW ONE ADDRESS SPACE PASSES DATA TO ANOTHER: CALLXMIT AND RETXMIT

Given a system that consists of functions walled-off from one another, and that inevitably one such function will need to call another and pass it parameters, and that a called function will need to return a result, the system clearly must provide a means to facilitate the movement of data between address spaces. The introduction of this paper addresses this issue and introduces a system address space known as the Transmitter Module (XMIT) which does just this. Like Auth, XMIT is a highly-specialized and somewhat privileged control program module. It can move one page of data from one address space to another. Before describing how XMIT works, the storage layout of an address space must be described.

The section on virtual storage points out that most system address spaces consist of two pages. Every system address space consists of at least two pages and maybe more depending on the storage required to implement its function. Of the two required pages, one contains the module's code and data. This page is attached at virtual address X'10000', i.e. the load origin of every module in a system address space is X'10000' virtual. The second required page is known as the parameter page, and it is located at virtual address X'0', or page 0 of each address space. Every address space except XMIT has a page 0 parm page. The parameter page is initially empty and its use is entirely free-format.

When one address space wants to call another and pass it data, it moves the data to its parameter page. Then, as in CALL processing, it places

the name of the target address space in R0 and executes a CALLXMIT (CALL with Transmit) SVC. Similarly, when a called address space wants to return a result, it moves the data to the parameter page and executes a RETXMIT (RETURN with Transmit) SVC. When Auth receives control it decodes the SVC request and determines that in each of these cases it must invoke XMIT to move data before it can complete the CALL or RETURN request.

At this point, Auth is faced with a dilemma. Realize that no system address space is authorized to call Auth; it just happens to gain control when an SVC is issued. Also notice that Auth never saves any linkage to itself on the stack, it is driven entirely by the modules of the system. Now Auth finds itself in a situation where it needs to call XMIT and have XMIT return to it. The most natural way to place return linkage to Auth on the stack is to have Auth call itself recursively. Simply, Auth loads R0 with 'XMIT' and issues a CALL SVC. The recursive invocation of Auth handles the CALL only slightly differently. Auth is sensitive to XMIT as a target. If the target of a CALL is discovered to be XMIT, Auth places the sender address space STO (the one issuing the CALLXMIT) in R0 and the receiver address space's STO in R1 just before it activates XMIT. This is purely a system design convention: When XMIT receives control, it must have the sender STO in R0 and the receiver STO in R1. Having done this, Auth performs normal GOSUB processing which activates XMIT.

When XMIT receives control, it uses the sender and receiver STOs to locate the sender's and receiver's page 0 page table entry. It copies the sender's page 0 page table entry to its own page 0 page table entry,

and copies the receiver's page 0 page table entry to its own page 1 page table entry (remember that XMIT has no page 0 parameter page of its own.) At this instant, XMIT has pages of other address spaces mapped into its virtual storage. It then copies its virtual page 0 to its virtual page 1, which effects the data transfer from sender to receiver. Finally, it detaches the sender and receiver parameter pages from its address space, and returns to Auth. Auth returns to itself to the point following the recursive call, where it performs GOSUB processing to complete the original call request. When the target address space is activated, its parameter page is a copy of the caller's parameter page. Although parameter page format is undefined, the callee must be aware of the caller's parameter page usage conventions.

2.3 OPERATING SYSTEM STRUCTURE

2.3.1 OVERVIEW OF PRINCIPAL COMPONENTS

The operating system constructed for this project consists of a written collection of program modules organized functionally as follows:

1. Modules Not Housed in any Address Space:
    a. First-Level Interruption Handlers (FLIHs):
        1) External FLIH
        2) I/O FLIH
        3) Program FLIH

4) Supervisor Call (SVC) FLIH

b. Data Areas/Structures:

1) Real Page Zero

2) DAT Tables

2. Modules Housed in Separate Address Spaces:

a. Second-Level Interruption Handlers (SLIH):

1) External SLIH

2) I/O SLIH

3) Program SLIH

4) SVC SLIH (Authentication Routine)

b. User SVC Services

1) SLEEP

2) Console READ and WRITE

c. Transmitter Module

d. System Trace Table

e. List Address Spaces

1) Run List

2) Sleep List

3) Terminate List

f. Scheduler

g. Dispatcher

h. User Address Spaces

In this functional representation of the system, each function is implemented by one or more address spaces. In this project, each program source module is assigned its own address space. Related functions are contained in the same source module (hence address space) when such a grouping appears natural. For example, in the case of linked lists one

source module provides all list manipulation operations as separate entry points, e.g. enqueue and dequeue. Each entry point starts at a different virtual address within the address space and is assigned a unique name. A table entry for each entry point is placed in Auth's table of system address spaces. The table entry makes the entry point callable: it specifies the symbolic name, the STO for the address space which contains the entry point, and the virtual address of the entry point within the address space.

Appendix A contains a module directory and Appendix B contains descriptions of each source module.

2.3.2 PROCESSOR ALLOCATION

The operating system exists to run user work. If we looked at processor allocation over time, we would see CPU ownership toggling between system and user code. The system gives control of the processor to a user address space and regains its use as a result of an interruption. The system module that allocates the processor to a particular user is the dispatcher. The system is guaranteed to regain control of the CPU at some point in time after dispatching a user address space. Dispatcher processing includes setting the CPU timer with a value known as the dispatch quantum. The timer steadily decrements the quantum while user code executes and generates an interruption condition when the quantum diminishes to zero. User code can neither alter the remaining quantum nor disable the processor for interruptions, and since the user is dispatched with the CPU enabled for timer interruptions the system external (timer) FLIH must eventually gain control.

Other sources of interruption while running a user include: 1) An I/O interrupt from the console, 2) A SVC interruption generated when user code executes an SVC instruction, presumably to request a control program service, 3) A program interruption generated for a variety of reasons, each indicative of a severe programming error in user code, and 4) An external interruption from the clock comparator. The clock comparator is used to implement the SLEEP user service. The user address space runs enabled for all of these interruptions and does not have the capability to mask them. (In System/370, the SVC interruption and nearly all forms of program interruption conditions are not maskable. Those program interrupts that can be inhibited by non-privileged code are related to arithmetic operations which, if suppressed pose no threat to the health of the system.)

Clearly, system code will eventually regain control of the processor after dispatching a user address space. This transition if effected by the interruption mechanism.

2.3.3 INTERRUPTION PROCESSING

A first-level interrupt handler is a program module designed to occupy one page of real storage. Of the six System/370 interruption classes, two are not used by this project: machine check and restart interruptions. The remaining interruption classes each have their own FLIH (I/O, external, program and SVC).

A FLIH is activated by a hardware-initiated PSW swap. More specifically, the swap consists of saving the current PSW and then replacing the current PSW with a new PSW image. The storage addresses containing the pre- and post-interruption PSWs are defined by System/370 architecture and are assigned permanent locations in page 0 of processor storage. Processor interruption processing stores the current PSW at the location assigned to the particular class of interruption. The post-interruption PSW images (known as "interrupt new PSWs") must be supplied by operating system software. Saving interrupt new PSWs in page 0 is one of the first things control program software must do. The instruction address portion of an interrupt new PSW is set to point at the first instruction of the FLIH for that class. The interrupt new PSWs in this project specify EC mode, DAT off. Thus, a FLIH does not run in translate mode.

Within this project, interruption conditions may be generated when the system is running a user or when it is running itself. In both cases, the first thing a FLIH must do is save the contents of the general purpose registers (and floating-point registers if the system was running a user). In System/370, all storage accesses require the use of a base register to designate an operand address. When an interruption occurs, the register contents must be saved in main storage as System/370 does not have two sets of registers (pre- and post-interruption register files). One store instruction can save all the registers, but it must designate a base (destination) address. But in the case of interrupt processing, if a base register is established for the store instruction before saving that register, its pre-interruption contents are lost. This circular problem is solved

by a System/370 addressing feature which enables a store instruction to save all of the registers starting at any offset into real page 0 without using a base register. Having saved the registers in page 0, the FLIH is free to use them as is wishes.

Real page 0 belongs to the program FLIH. The motivation for establishing the ownership of this page will be discussed in a section on protection. What is important to realize here is that the only module which can write into page 0 is the program FLIH. This means the external, SVC and I/O FLIHs will not be able to save the general registers. Somehow each of these FLIHs must transfer control to program FLIH where the registers can be saved in page 0. The technique used to transfer control to the program FLIH is simply the generation of a program interrupt.

To the program FLIH implemented for this project, nearly every program interrupt spells trouble, with one exception. The System/370 Monitor Call (MC) instruction purposely generates a program interruption. The programmer specifies the non-privileged MC with an (arbitrary) 12-bit operand. This operand value is stored at an assigned storage location in real page zero as part of the interruption process. As an added convenience, MC needs no base register which makes it ideally suited as the means for a FLIH to transfer control to the program FLIH without disturbing any of the general registers. Hence, the first instruction of external, I/O and SVC FLIHs is Monitor Call, specifying a code which identifies this as a SAVEREGS request. MC causes program FLIH to receive control immediately, and the first thing it does is save the registers in a page zero save area. It then determines this inter-

ruption was caused by a Monitor Call. Further, it examines the MC code in page zero identifying this as a SAVEREGS request. As the registers have been saved as requested, it returns immediately to the calling FLIH by replacing the current PSW with the program old PSW.

This double interruption condition is necessary to provide a high degree of protection for page zero. It ensures there is only one module in the system that can write to that page. Once the registers have been saved, each FLIH proceeds with duties specific to its class of interrupt. Eventually a FLIH completes its function. What happens next depends on which FLIH is running and what code was running when the original interrupt was raised. Before describing the different scenarios, a discussion on the privileged vs. non-privileged mode of operation is in order.

Of all the modules in this operating system, the program FLIH is the only one that executes in supervisor state. (System startup code is also privileged, but it executes only once and thereafter is not considered part of the control program.) Naturally, certain operating system functions require the issuing of privileged instructions. Privileged operations are requested by non-privileged control program routines through the Monitor Call instruction. The Monitor Call code is set to indicate which privileged operation is required. The program FLIH immediately gains control and examines the code. If the address space requesting the operation is authorized, program FLIH performs the privileged operation requested by the code (e.g. Start I/O might be code 11). (The authorization technique is covered later in a section on security.) After the privileged operation has been performed, program

FLIH returns to its caller. Note that the use of Monitor Call by a user address space is undefined and is treated as a user program interruption.

Now back to FLIH wrap-up processing. Both the external and I/O FLIHs complete by EXITing to their respective second-level interrupt handlers. If the program FLIH was running because a running user suffered a program check, then it EXITs to an address space which terminates the running user. If program FLIH is running because the system suffered a program check that was not generated by a Monitor Call instruction, program FLIH instantly shuts down the failing system. Otherwise program FLIH performs the Monitor Call request, and may or may not return to the caller depending on the semantics of the request. Finally, the SVC FLIH completes its processing by "activating" the Authentication Routine, which can be considered the SVC SLIH. The activation of Auth requires several privileged instructions, so SVC FLIH loads some general registers with the operands for those instructions and issues a "ACTIVATE AUTH" Monitor Call. Program FLIH extracts the operands from the registers, which consist of Auth's STO, its PSW protect key value and the virtual address of Auth's first instruction. Then it constructs a PSW image for Auth using the supplied PSW key and instruction address, and specifying problem state, EC mode and DAT on. Finally, it executes a privileged instruction to load Auth's STO into control register 1, and then issues a privileged Load PSW instruction which replaces the current PSW with Auth's. The processor is now allocated to the Authentication Routine's address space. Finally, notice that ACTIVATE AUTH is an example of a Monitor Call that does not result in program FLIH returning to its caller.

Operating System Structure

Remember that EXIT is an SVC. When the FLIHs EXIT to another address space, an SVC interruption is generated. This results in the activation of SVC FLIH which ends with the scenario just described, as Auth implements CALL, RETURN and EXIT requests.

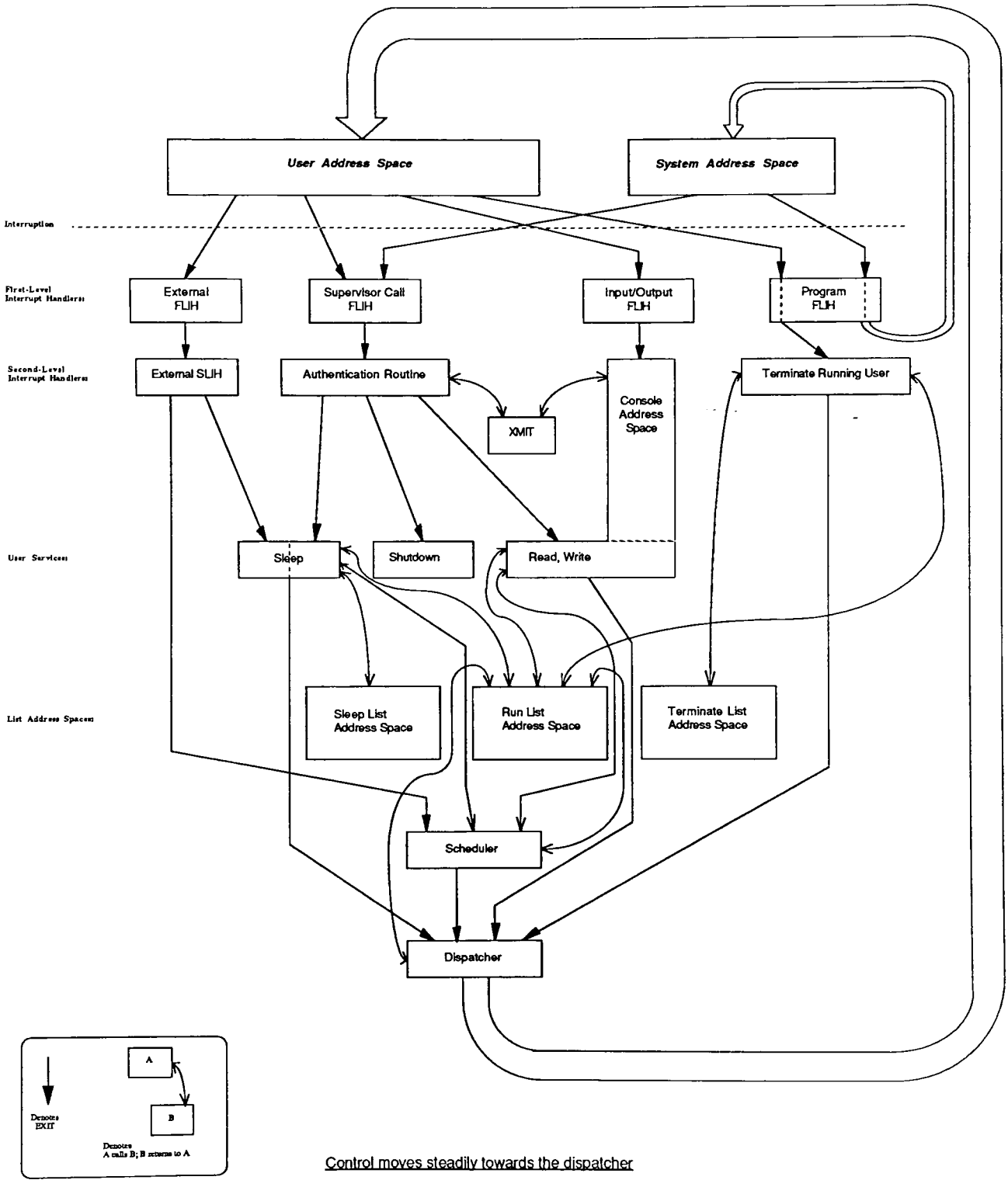
2.3.4 GENERAL FLOW OF CONTROL

We have seen how this system gives control to and receives control from a user address space. What remains to be discussed is the general flow of control through the system once it has been re-activated by an interrupt.

Healthy system operation consists of a sequence of CALL, CALLXMIT, RETURN, RETXMIT and EXIT requests as control passes from one component to another. Control always flows towards the dispatcher. Dispatcher is always the ultimate destination within the system. It is here that the system either turns the processor over to a runnable user, or places the machine into a wait state in the absence of any work. Figure 4 on page 56 includes nearly every module of the system and illustrates how control flows within it.

While the dispatcher is the ultimate destination, Auth is perhaps the center of attention. Auth facilitates every major transfer of control within the system. It also initiates system services requested by user SVCs. Auth's mode of operation depends on the state of the system at the time of the SVC interruption that triggered its activation. Suppose

Apparent Flow of Control



Control moves steadily towards the dispatcher

Figure 4.

a running user issued an SVC. The SVC instruction has a dual role defined by the type of code that issued it. When a user issues an SVC, Auth realizes the system was running a user, so this SVC must be a request for a system service. Consequently, Auth searches its table of user SVCs, looking for the requested function. On the other hand, if the system itself was running and it issued an SVC, Auth verifies the SVC was one of CALL/XMIT, RETURN/XMIT or EXIT. These are the only SVCs defined to the running system. Assuming the SVC is defined, Auth 1) performs a RETURN function if requested, or 2) searches its table of system address spaces looking for the target of the CALL or EXIT that must have been issued.

If Auth locates 1) a user table entry for the user-issued SVC request or 2) the target of a CALL or EXIT request in its table of system address spaces, Auth activates the target address space identified by the table entry. If a table search fails, Auth terminates the user or system depending on its caller.

Auth activates a target address space via a DISPATCH monitor call (not to be confused with the dispatcher address space). Program FLIH handles DISPATCH similarly to the way it handles ACTIVATE AUTH. Auth supplies a target address space STO, PSW key and virtual entry point address, and program FLIH activates the target address space.

Continuing in this way, control flows from system address space to system address space, moving steadily towards the dispatcher. Important events are logged in the trace table of system activity. The system trace routine consists of a two page trace table and code to

manage it. When a module wants to identify a particular event it calls the trace routine, passing data describing the event. The trace data is installed in the table in wrap-around fashion, i.e. the first entry in the table is replaced by the new entry if the table is full. The contents of the trace table are sometimes useful when analyzing system operation or a system failure.

2.3.5 REPRESENTATION OF USER WORK: USER ADDRESS SPACE AND USER CONTROL BLOCK

Eventually control reaches the dispatcher address space. It is here that steps are taken to allocate the processor to user code. In this system, a user is housed in an address space just like system code. User code consists of any legitimate System/370 instruction stream. This code is assigned to an address space identical in structure to a system address space: It consists of a page or pages of user-written code and data with a load origin of virtual X'10000' and a page 0 parameter page. The parameter page is used to supply data to certain of the system service routines.

The users in this project are statically defined. They exist when the system begins operation and are neither created nor destroyed during its operation, i.e. this is an experimental system. The existence of each user address space is represented by a data structure known as a user control block (UCB). The UCB contains:

1. chain pointers
2. general register images

3. floating-point register images
4. dispatch PSW
5. STO
6. dispatch quantum
7. symbolic user name
8. scheduler priority

The dispatcher has a list of every executable UCB available to it. This list is called the Run List. Dispatcher's job is simple: Allocate the processor to the UCB at the head of the Run List. It does this using a technique similar to the one used by Auth to activate a system address space. More precisely, it extracts the user's PSW, dispatch quantum and STO from the UCB and places these values in several of the general registers. Then it issues a SET DISPATCH PARAMETERS Monitor Call which activates program FLIH. Program FLIH moves the dispatch parameters contained in the general registers to page 0 holding locations, then returns to the dispatcher. Dispatcher loads the floating-point and general registers from the UCB (in that order) and issues a DISPATCH monitor call. Program FLIH automatically saves the general registers as a result of the interruption. Then it loads control register 1 with the STO supplied from the SET DISPATCH PARAMETERS. Program FLIH does not disturb the floating-point registers during its operation, hence they are intact. It reloads the general purpose registers from the page 0 savearea, and it sets the CPU timer with the dispatch quantum supplied from the SET DISPATCH PARAMETERS. Finally, it issues a privileged Load PSW instruction which replaces the current PSW with the user PSW image supplied by SET DISPATCH PARAMETERS. As the user PSW specifies EC mode with DAT on, the user address space is effectively dispatched in

translate mode. Further, the user PSW specifies that the machine is enabled for I/O and external interruptions. The user address space executes until an interruption occurs. Figure 3 on page 42 depicts the structure of a user address space.

2.3.6 AVAILABLE USER SERVICES: SLEEP, READ AND WRITE

A running user is free to request services from the control program. This project provides three user services: SLEEP, console READ and console WRITE. Each is requested by a executing a numbered SVC instruction.

If a user needs to suspend its execution for a period of time, it can block itself by executing a SLEEP SVC. Before doing so, the user places a doubleword value in R0 and R1 indicating the number of microseconds it wishes to sleep. The Sleep address space receives control at the SLEEP entry point as a result of the SVC. SLEEP removes the user's UCB from the Run List and computes its wakeup time-of-day. It saves this value in the UCB and inserts the UCB into a list of sleeping address spaces. This list is known as the Sleep List and is maintained in sorted order by wakeup time. The System/370 clock comparator is set with the wakeup time for the user at the head of the Sleep List.

Clock comparator circuitry generates an external interrupt when the current time-of-day clock matches the wakeup time-of-day. This interrupt results in the activation of the Sleep address space WAKEUP entry point. WAKEUP simply dequeues the UCB at the head of the Sleep List and passes it to the scheduler address space. Scheduler address space

ultimately adds the UCB to the Run List (thereby unblocking it) and returns to WAKEUP. WAKEUP restarts the clock comparator for the next UCB on the Sleep List, if any, then EXITs to the dispatcher.

Any user address space may send messages to the system console. One of the user address spaces is designated the system Operator address space. The Operator is permitted to issue reads to the console to obtain system commands. The Operator address space differs only from a general user address space in that it is more privileged--it can issue certain SVCs not available to other users. For example, READ is privileged. So is the SHUTDOWN SVC, which Operator issues in response to a human operator-issued Shutdown command. SHUTDOWN terminates system operation in an orderly fashion.

READ, WRITE and console I/O interrupt processing are performed by separate entry points in the Console address space. READ and WRITE both involve removing the user from the Run List and queueing him on the Console List. A READ request is held pending until data becomes available at the console. When this condition is met, a physical read operation is initiated to the console. The data retrieved is placed in Operator's parameter page via a call to XMIT, and then the Operator is made runnable (is unblocked) by the scheduler. To write to the console, the user places the message to be written in his parameter page and puts the length of the message in R0 just before issuing the WRITE SVC. A WRITE request is queued behind other console requests, if any. When the console display is available, the Console address space calls XMIT to fetch the contents of the user's parameter page (thereby retrieving a copy of the message text). A physical write operation is

then initiated to send the user-supplied message to the display. When the physical write operation concludes, the user is dequeued from the Console List and run through the scheduler to unblock him (make him dispatchable). The Console address space concludes by EXITing to the dispatcher.

The setting of the clock comparator by the Sleep address space and the physical I/O operations initiated by the Console address space can only be effected by privileged instructions. These address spaces request these operations via the Monitor Call instruction.

2.3.7 SHARED DATA STRUCTURES: LIST ADDRESS SPACES

The operating system knows of the existence of a user by a UCB. As the system processes a user, the user's UCB is passed from component to component and it continually moves from one list to another. For example, consider a user who wants to sleep. His UCB is initially on the Run List. Dispatcher retrieves a copy of his UCB from the Run List. The user code is given control of the processor, and it executes a SLEEP SVC. The Sleep address space receives control and moves the UCB from the Run List to the Sleep list. When the sleep interval expires, the Sleep address space removes the UCB from the Sleep List and passes it to the scheduler address space. The scheduler adds the UCB to the Run List, making it a dispatch candidate once again.

Notice that the UCB moves among disjoint address spaces. It is easy enough for one address space to pass a UCB to another, but how is it that one address space is able to remove a UCB from a list managed by

code in another address space? Clearly it does not make sense that a well-defined component such as the dispatcher should be made "callable" by an address space such as Sleep when that address space needs a UCB from the dispatcher's Run List. Adding this function to dispatcher increases its complexity and broadens its scope to include function not necessarily related to its prime purpose. Obviously the UCB is a shared object. This project implements very safe sharing of UCBs through the use of a new type of address space: the List Address Space.

The List Address Space is identical in structure to other system address spaces. It has no special execution time requirements or privileges, i.e. it is treated no differently than any other system address space. It simply consists of code and data that constructs and maintains a linked list of UCBs. For example, the Run List resides not inside of the dispatcher address space, but instead resides inside the Run List address space. The Run List address space consists of multiple entry points: enqueue, dequeue and get dispatch UCB. Now any address space can easily add or remove a UCB from the Run List, and the dispatcher is provided an entry point that fetches a copy of the UCB at the head of the Run List so it can dispatch a user.

The List Address Space concept offers several distinct advantages. It facilitates the sharing of an object in a clean, centrally managed way. It moves all of the logic involving linked list maintenance to one module. This is important because it eliminates the need to repeat list management code in other modules (which necessarily reduces the complexity of those modules). Moving all list management code to one address space virtually eliminates the possibility that a list will be

improperly constructed. Since the List Address Space is highly specialized, its operations are concise and can be tested exhaustively to ensure its implementation is correct. Finally, the functions of each List Address Space entry point can be tailored specifically to the type of organization required by a particular list representation. For example, the entry point to add a UCB to a list might be priority queuing in one List address space and straight FIFO in another.

2.4 SECURITY AND PROTECTION MECHANISMS

2.4.1 ENFORCING DESIGN SPECIFICATIONS: THE AUTHORIZATION LIST

The introduction proposes that Auth validate every transfer of control from one operating system module to another. Auth does exactly this for every transfer which explicitly specifies a target address space (CALL/XMIT and EXIT). Each entry in the table of system address spaces contains the symbolic name of an address space entry point, the address space STO, its PSW key and the entry point virtual address. It also contains an authorization list. The authorization list consists of a count field followed by a (possibly empty) list of STOs that are authorized to call this entry point. If the count field is zero, the entry point is considered public. An example of a public address space is the system trace routine. Any module of the system may place an entry in the trace table. If the count is non-zero, it indicates the

number of STOs in the authorization list. Auth has access to a caller's STO. When it locates the target of a call and that address space is not public, Auth searches the authorization list looking for the caller's STO. If it finds a match, it completes the call sequence. Otherwise, Auth terminates the system.

Each table entry is constructed manually when a new address space and/or entry point is added to the system. Similarly, an existing entry is manually updated when a heretofore unauthorized CALL becomes valid. As a result, the structure of the entire system is described and enforced by Auth's table of system address spaces.

2.4.2 RESTRICTING USER ACCESS TO CONTROL PROGRAM SERVICES

The system services made available to users through SVC requests are controlled in a similar fashion. Although most user services in a system would be public, this system provides several for use by only one user, the Operator address space. Each available user SVC service is represented as an entry in Auth's table of user services. The format of each entry is identical to a system address space table entry, except that the symbolic name is replaced by the SVC number. Using the same logic, Auth searches the user table in response to an SVC request, looking for a match on SVC number. If Auth can't find the SVC it terminates the user. Otherwise it examines the authorization list and invokes the service if it is public or the caller is authorized. An unauthorized caller is terminated.

The authorization technique allows users of varying privileges to be added to the system. It is conceivable that this technique could be used to create a power structure consisting of sets of users. Each set would denote certain services, and each member would be authorized to use all of the services defined by the set. A user could belong to more than one set, thus broadening its capabilities.

2.4.3 RESTRICTING THE USE OF MONITOR CALL SERVICES

The program FLIH must guard the use of the privileged instructions it implements via the Monitor Call request. The method of enforcement is not as elaborate as that used by Auth, but it is equally effective. Very few of the control program routines make use of privileged instructions. Consequently, the authorization list method offers more power than is needed as there would be few lists, each containing only one or two entries. It seems sensible enough then to simply check the caller's identity at the start of each Monitor Call routine in program FLIH. Program FLIH knows all of the STOs authorized to use all of its services. The caller's STO is checked only against the STOs of the address spaces authorized to request this service. In effect, program FLIH does search an authorization list, but the list is replaced by simple compare and branch logic as opposed to a search. Still, this method is effective. If program FLIH detects an unauthorized caller it terminates the system.

2.4.4 PREVENTING THE DESTRUCTION OF STORAGE BY FLIHS

The protection of storage through the use of DAT has been discussed at

length. However, not all of the operating system modules execute with DAT on: the FLIHs execute with DAT off. (The reason for this is that a FLIH gains control as a result of an interruption which replaces the current PSW but does not alter the STO in control register 1. If an interrupt new PSW specified DAT on, the FLIH would have to be mapped into the interrupted address space's virtual storage. Since interruptions occur when running a user or system address space, each FLIH that could be activated while a particular address space is active would have to be mapped into that address space. The fact that an interruption can occur when the system is idle further complicates this issue, as there is no active address space in this state. This problem could be solved given an interruption mechanism that reloads control register 1. This is not a System/370 feature, so a decision has been made to allow FLIHs to execute with DAT off.) As a result, a FLIH has the ability to access every byte of processor-addressable storage.

Storage protection in the DAT off CPU mode of operation is provided through the use of storage protect keys. Each FLIH is assigned its own key, i.e. SVC FLIH runs with key 1, external FLIH is key 2, I/O FLIH is key 3 and program FLIH runs with key 4. This is sufficient to prevent one FLIH from damaging the storage of another. To protect the remainder of the system from destruction by a FLIH, every page allocated to a system address space is assigned key 5, while every page of a user address space is assigned key 6. Since program FLIH needs to modify page 0, page 0 is assigned a protect key of 4. Consequently, none of the other FLIHs can alter page 0. A module can read storage in a page assigned a key different from its PSW protect key, but it can not write to a page unless storage key and PSW key are in agreement.

Security and Protection Mechanisms

SVC, external and I/O FLIHs need to pass information to their corresponding SLIH. They do this by placing data in the SLIH's parameter page and EXITing to the SLIH. To facilitate this, the SLIH parameter page is assigned a key that matches its FLIH PSW key. Since none of the SLIHs presently need to write to their parameter page, they can execute under key 5. Therefore, a FLIH can not alter the remaining pages of its SLIH address space. Figure 5 on page 69 summarizes the use of storage keys.

Finally there is the matter of saving a user's state in his UCB as a result of an interruption. This is accomplished by the Run List address space. Program FLIH saves the user's state in page 0 when an interruption occurs. It would ideally save this information in the user's UCB, but it has no way of accessing the UCB. So it moves the user's state information to a private page 0 user savearea and continues normal system operation. Eventually (and before a new user is dispatched) control must reach the Run List address space. When this happens, Run List address space moves the saved user state data from page 0 to the most recently dispatched UCB, thereby saving the user's environment. To do this, page 0 is mapped read-only into the Run List address space at virtual page 2.

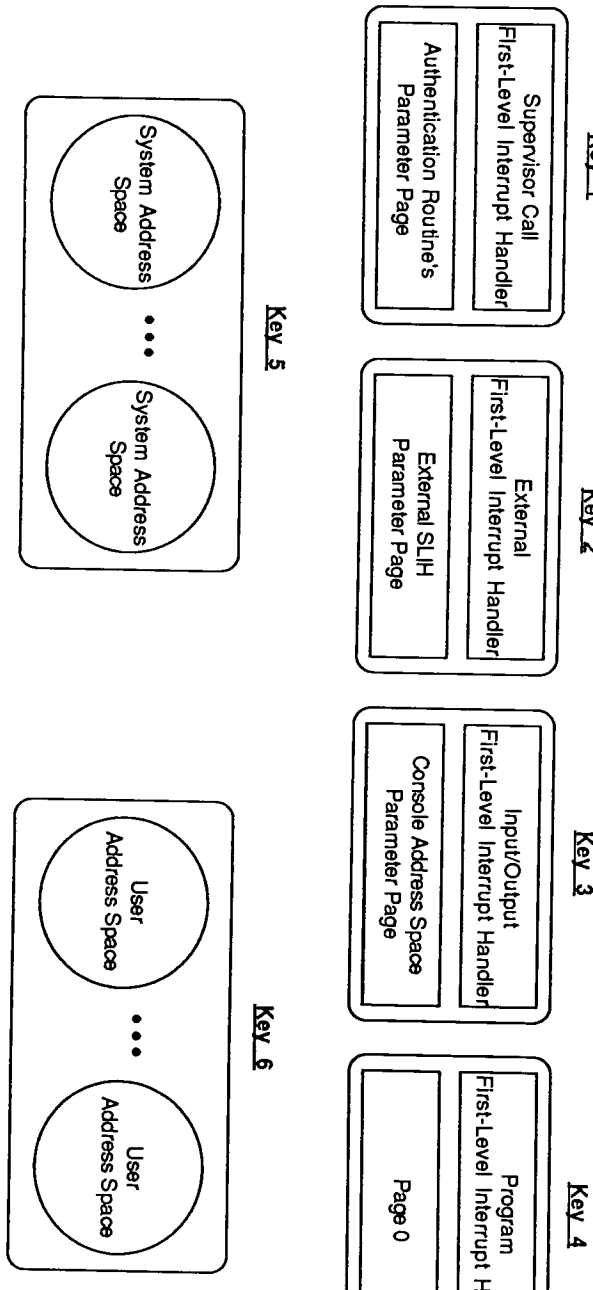

Storage Key Assignments
Key 1
Supervisor Call First-Level Interrupt Handler
Authentication Routine's Parameter Page
Key 2
External First-Level Interrupt Handler
External SLIH Parameter Page
Key 3
Input/Output First-Level Interrupt Handler
Console Address Space Parameter Page
Key 4
Program First-Level Interrupt Handler
Page 0
Key 5
System Address Space
System Address Space
Key 6
User Address Space
User Address Space

Figure 5.

2.5 ANALYSIS AND SUGGESTIONS FOR FUTURE WORK

The method of operating system organization proposed by this thesis works well. It encourages a design consisting of well-defined modules and clearly-established relationships among the components. The protection mechanisms were extremely helpful during the implementation phase. In several cases protection exceptions were raised because modules had been designed to refer to data they did not own. In these cases I had clearly strayed from my design objectives, usually by making the assumption that a module would be able to directly use an object that turned out to reside in another address space.

This control program is largely an experiment in the use of virtual storage. It does not provide a functionally rich user programming environment. Rather, it provides an operative core that stands as proof of the proposal's feasibility. Extending the system's function and usability is an obvious enhancement opportunity.

Authorization list maintenance could become unwieldy in a system consisting of many modules. It is conceivable that these lists could be constructed automatically. The structure of the system could be expressed as a directed graph G=(V,E), where V is a set of nodes, each representing a functional module and E is the set of edges linking the nodes in V. (The edge set E actually describes the module-to-module relationships ultimately encoded as authorization lists.) Ideally such a graphical representation would be implemented at a very high level, e.g. pictorially. A pictorial description would likely be the best way

to convey and represent the structure of the control program. Further, the diagram could be used as input to a procedure which generates the authorization lists implied by the graph.

The present authorization list implementation is based on an inclusion property. In cases where a module is authorized for use by all but a few routines, it would make more sense to incorporate a list reflecting exclusion. This would make it possible to code only the unauthorized address spaces in an authorization list.

Another area for improvement lies beyond the system and is concerned with the linkage editor. The system consists of source modules that are assembled to produce text files. Using VM/SP system generation utilities, an IPLable reader spool file is created which consists of an IPLable linkage editor followed by the system text files. Thus, a virtual card deck is constructed, and the system is loaded by IPLing the virtual card reader. The IPL sequence loads linkage editor bootstrap code which reads in the remaining cards of the linkage editor. Once it has established itself in main storage, the linkage editor reads each text deck of system code from the reader into memory. The editor resolves external references, relocating them relative to real location 0. As a result, an address constant embedded in a source module is relocated to reflect a real address. Unfortunately, the code that uses this real address runs in translate mode, which results in an attempt to use a real address to reference a location in its virtual storage. In other words, the editor has no provision for relocating relative to the start of the address space which contains the address constant. It should be modified to detect a text deck of an address space and

reset its relocation factor to reflect the virtual base address of the module (containing the address constant) within its address space.

This linkage editor deficiency is a very serious problem. What it really means is that source code written for an address space cannot contain address constants. This means that some type of data structures must be implemented "differently". For example, consider a stack. Ideally the source module would contain an address constant representing the last stack location so stack-full conditions could be easily detected. Unfortunately, the address constant seen by the running program, i.e. the address relocated by the linkage editor reflects the real address of the end of the stack--a completely useless value for purposes of bounds checking. An alternative implementation used in this project is to express stack-full as the maximum number of available stack entries. The stack depth is maintained as a count of the number of entries, and stack manipulation is performed by converting the count into an offset an indexing into the stack off its base address. Another problem occurs with linked lists. Statically allocated list chains cannot be established at load time because, once again, the chain pointers reflect real, not virtual, addresses. One solution is to include extra code that runs only the first time the module that maintains the list executes, and that code has the task of linking the list with pointers calculated using a similar indexing technique (i.e. using the base address of the first node and the size of a node to compute the virtual address of successive nodes). This linkage editor deficiency results in inelegant and more expensive implementations for otherwise primitive operations like stack push/pop.

Analysis and Suggestions For Future Work

Some very nice enhancements could be achieved by augmenting System/370 machine architecture. It was mentioned that an architectural change to the interrupt mechanism to facilitate control register 1 swaps would allow the first-level interrupt handlers to be placed in private address spaces. This would eliminate the need to use storage keys to prevent one component from damaging the storage of another; DAT would provide this protection in full. Storage keys could be used to partition and protect the contents of an address space, which could help protect an address space from internal damage.

System performance could be improved if the use of privileged instructions by address space was programmable and controlled by the processor microroutines. Instead of having program FLIH issue all of the privileged operations, System/370 could be enhanced to make an instruction available to certain address spaces (resulting in a new semi-privileged instruction class). This conceptually simple change could be implemented by comparing the STO in control register 1 with a firmware list of the STOs authorized to issue a particular instruction. Thus, the meaning of supervisor state now blurs, as an address space can execute a (formerly privileged) instruction so long as it is authorized, regardless of the mode of the machine. This enhancement would significantly reduce the volume of program interruptions experienced by the current design.

A third extension to the architecture would be to lift the minimum length requirements imposed on segment and page tables. The virtual storage requirements of each address space tend to be small, on the order of several pages. Yet much of the real storage allocated to

segment and page tables is wasted defining uninstalled segments/pages. Lifting minimum length requirements on these structures results in less storage waste.

In spite of these improvements this control program is very expensive to run. Almost the entire system runs in translate mode which slows everything down. DAT table fetches occur at a very high frequency, increasing main memory utilization. The presence of a translation lookaside buffer is certainly helpful, but it must be of adequate size. Certainly a good area for future work is in determining how severely this design is penalized by the DAT process, and investigating ways of reducing this penalty (either architecturally or otherwise).

The control transfer primitives are also very expensive. For example, an operating system CALL results in the activation of two first-level interrupt handlers (one of which is activated several times) and four address spaces. Changing to a different address space is expensive because several interrupts are generated, DAT is reconfigured several times and authorization must be performed at two different levels. Some speedup can be gained by moving AUTH to microcode. Additional speedup could be gained if the SVC interrupt associated with a control transfer request could be eliminated. It seems likely that a new instruction could be invented that would activate AUTH directly for a CALL, RETURN or EXIT request.

GLOSSARY

ADDRESS SPACE: 1: The set of all storage addresses accessible to the currently executing program. The entire set of addresses is either real or virtual, depending on the mode of CPU operation. 2: An entity defined by a segment table and zero or more page tables. Typically, an address space has page frames of real storage allocated to it, and these pages contain program code and/or data or both. 3: A dispatchable unit of work.

AUTHENTICATION ROUTINE: A module of the operating system that determines whether requested access for a system service or object is authorized, and that subsequently grants or denies access to the service or object.

AUTHORIZATION LIST: A list of the address spaces authorized to access a system service or object. Authorization lists are components of the authentication routine, and one authorization list exists for each system service and object.

CALL: A control transfer primitive which results in the activation of an entry point in a target address space.

CALLXMIT: A control transfer primitive which augments normal CALL processing by including an activation of the Transmitter Module to copy

the contents of the sender's parameter page to the receiver's. The transmit operation is performed before the CALL.

CONTROL BLOCK: A contiguous packet of bytes resembling a structure (i.e. consisting of one or more fields) and organized according to a programmer-defined template.

CONTROL REGISTER 0: A System/370 register accessible only in supervisor state and containing, among other things, the DAT translation format.

CONTROL REGISTER 1: A System/370 register accessible only in supervisor state and containing the length and starting address of the active segment table.

DAT MODE BIT: A field (bit 5) in an EC mode PSW which controls whether DAT is engaged.

DAT OFF: A mode of CPU operation where instruction and operand addresses are considered real and are not subject to DAT; specified by setting the PSW DAT mode bit to 0.

DAT ON: A mode of CPU operation where virtual-to-real address translation is mandatory and is specified by setting the PSW DAT mode bit to 1.

DAT TABLES: Data structures (segment and page tables) through with DAT proceeds.

DISPATCH QUANTUM: A time interval defining the maximum length of time a user address space may hold the processor.

DISPATCHER: That module of the operating system that selects an address space from a list of dispatchable candidates and subsequently relinquishes CPU control to the selected address space.

DUMP: A copy of the contents of main storage, sometimes printed but usually written to an external storage device, e.g. disk or tape.

DYNAMIC ADDRESS TRANSLATION (DAT): A System/370 CPU feature that implements virtual to real address translation, thereby providing the basis for a virtual storage environment. Given a virtual address and translation tables, and using a predefined translation algorithm, DAT yields a corresponding real address, or notifies the control program (via an interrupt) that the translation cannot be completed.

EC (EXTENDED CONTROL) MODE: A mode of CPU operation which makes DAT available and is specified by setting the PSW control mode bit (bit 12) to one.

EXIT: A control transfer primitive which results in the activation of a target address space; it differs from CALL in that no return linkage is saved.

FIRST-LEVEL INTERRUPT HANDLER (FLIH): Program code executed directly as a result of a hardware interrupt. Activation is synchronous or asynchronous, depending on the interrupt class. FLIH responsibilities

include saving the machine state and activating the appropriate second-level interrupt handler.

INTERRUPT: An immediate alteration in the normal instruction processing sequence, triggered by external events and effected by a hardware PSW swap.

I/O INTERRUPT: In System/370, the interrupt class used by channel hardware to alert the software to the occurrence of an input/output event within the channel subsystem.

IPL (INITIAL PROGRAM LOAD): A System/370 operation carried out in microcode which performs a read operation to a select device for purposes of retrieving a channel program and PSW which presumably loads a bootstrap program capable of reading in the remainder of the program being IPLed. At the conclusion of I/O activity during the IPL sequence, the IPL PSW replaces the current PSW.

KEY-CONTROLLED PROTECTION: A storage protection scheme based on the use of "keys". In System/370, each page frame of storage has an associated 4-bit storage key, set by a privileged instruction. The program status word also contains a key set by a privileged instruction. In general, each storage reference made by a running program is checked by the CPU. For store operations (and under some circumstances fetch operations), storage access is denied unless the PSW key value matches the storage key of the page frame being referenced.

MONITOR CALL: A System/370 instruction which generates a program interrupt and provides the control program with a 12-bit code which in this project identifies a service request. The numbered service requests require the execution of privileged instructions.

MONOLITHIC MONITOR: An operating system that runs completely disabled for asynchronous interrupt classes. Synchronous events, e.g. program and supervisor call interrupts are permitted in designs of this type.

PAGE FRAME: A unit of main storage organization, spanning 4096 bytes and aligned on a 4K boundary. The page frame is the unit of storage managed by a virtual storage operating system. It typically has an associated storage key and referenced/changed indicators, the latter feature being of use to storage management routines.

PAGE TABLE: A control block used by dynamic address translation. It contains entries for each page of a segment of virtual storage; each entry identifies the real page frame associated with an address in that segment, or indicates the address does not presently have a page frame assigned to it.

PFRA (PAGE FRAME REAL ADDRESS): A PTE field specifying (a portion of) the real address of the the physical page assigned to a virtual address.

PROBLEM STATE: A mode of CPU operation in which only those instructions designated non-privileged are available to the running program.

PROCESS: A dispatchable entity, consisting of a saved program counter image, a saved register image, miscellaneous saved state information and executable code and optionally data. In a virtual storage environment, the definition includes a description of the process address space, i.e. page and segment tables.

PSW (PROGRAM STATUS WORD): In System/370, a processor register containing the address of the next instruction to be executed and status bits defining the current mode(s) of CPU operation (condition code, interrupt masks, etc.)

PTE (PAGE TABLE ENTRY): An element of a page table providing a page frame real address (PFRA) and a valid/invalid indicator.

PTL (PAGE TABLE LENGTH): A STE field specifying the length of the page table corresponding to this segment.

PTO (PAGE TABLE ORIGIN): A STE field specifying (a portion of) the real address of the page table for a virtual address.

REAL ADDRESS: An address generated by a running program not subject to dynamic address translation, or the address resulting from the application of DAT to a virtual address. Each real address designates exactly one byte of physical processor storage.

RELOCATE MODE: See translate mode.

RETURN: A control transfer primitive which results the reactivation of the calling address space.

RETXMIT: A control transfer primitive which augments RETURN processing by including an activation of the Transmitter module to copy the returning module's parameter page to the caller's. The transmit is performed before RETURN processing.

RUN LIST: The list of dispatchable UCBs.

SAVE-AREA: A control block used typically for the saving of the machine registers.

SCHEDULER: An operating system module that implements a task-sequencing policy, often striving to achieve the dual goal of maximizing throughput and minimizing response time.

SECOND-LEVEL INTERRUPT HANDLER (SLIH): Code executed immediately after a FLIH whose primary responsibility is to analyze the source of the interruption and react accordingly. An example would be an I/O SLIH which determines that a device that has pending I/O requests has become available (as would be indicated by status associated with the I/O interruption). Part of SLIH operation would be to restart the device with the next queued request.

SEGMENT: A unit of virtual storage organization. A segment consists of up to sixteen 4096-byte pages. System/370 defines protection at the

segment level; if necessary a segment can be protected from store operations.

SEGMENT TABLE: A control block used by dynamic address translation. It contains an entry for each segment of an address space; each entry points at the page table which further describes the pages of the segment, or indicates the segment does not presently have any pages assigned to it.

SEGMENT TABLE REGISTER: A control register through which dynamic address translation proceeds. The segment table register points at the segment table for the active address space.

STE (SEGMENT TABLE ENTRY): An element of a segment table providing the length and real address segment's page table.

STL (SEGMENT TABLE LENGTH): A field in control register 1 which specifies the length of the active segment table in units of 16 STEs.

STO (SEGMENT TABLE ORIGIN): A field in control register 1 which specifies the real address of the active segment table.

STORAGE KEY: A 6-bit register assigned to each page frame of real storage, consisting of a four-bit key field, one bit for reference recording and one bit for change recording. The key field is used by the CPU to validate storage access requests (see key-controlled protection). The last two bits are used by the operating system page-replacement algorithm.

SUPERVISOR CALL (SVC): A System/370 instruction that, when executed results in the generation of a supervisor call interrupt.

SUPERVISOR STATE: A mode of CPU operation in which every instruction defined in the machine's instruction set, both privileged and and non-privileged, is usable.

SYSTEM/370: A machine-independent architecture developed by International Business Machines Corporation and implemented by a wide range of that company's computer systems.

TRANSLATE MODE: A mode of CPU operation in which program generated addresses are considered virtual and are subject to dynamic address translation. In System/370, translate mode is engaged by setting the DAT bit in the PSW to one.

TRANSLATION FORMAT: The specification of the structure of a virtual address, specified by a bit field in control register 0 and used by DAT to isolate the segment, page and byte indexes of a virtual address.

TRANSMITTER MODULE: A module of the operating system that facilitates inter-address space communication. It moves data from one address space to another.

USER CONTROL BLOCK (UCB): A data structure representing a user address space and containing user state data as well as a pointer to the user's segment table.

VIRTUAL ADDRESS: An address used by a program running in translate mode to refer to a byte of storage. A virtual address is subject to dynamic address translation, and may or may not map to a real main storage location at an arbitrary point in time.

VIRTUAL STORAGE: In this project, the storage that a function running in an address space would appear to own. The function logically owns a virtual storage bounded by the virtual addresses defined through its segment and page tables. These virtual addresses map to different real addresses; consequently virtual pages of the address space map to different real pages.

BIBLIOGRAPHY

1. Deitel, H.M., and Lorin, H. Operating Systems. Addison-Wesley, Reading, Mass., 1981, pp. 131-139.

2. Graham, G.S., Holt, R.C., Lazowska, E.D., and Scott, M.A. Structured Concurrent Programming with Operating Systems Applications. Addison-Wesley, Reading, Mass., 1978, pp. 8-11.

3. Holt, R.C. Concurrent Euclid, the UNIX system, and TUNIS. Addison-Wesley, Reading, Mass., 1983, ch. 3-4.

4. IBM System/370 Principles of Operation, Order No. GA22-7000, available through IBM branch offices.

5. OS/VS-DOS/VSE-VM/370 Assembler Language, Order No. GC33-4010, available through IBM branch offices.

6. Shaw, A.C. The Logical Design of Operating Systems. Prentice-Hall, Englewood Cliffs, N.J., 1974, pp. 58-59.

7. Virtual Machine/System Product (VM/SP) Introduction, Order No. GT00-1575, available through IBM branch offices.

8. VM/SP High Performance Option (HPO) System Programmer's Guide, Order No. SC19-6224, available through IBM branch offices.

9. VM/SP HPO System Logic and Problem Determination Guide - CP, Order No. LY20-0897, available through IBM branch offices.

10. VM/SP HPO Data Areas and Control Blocks - CP, Order No. LY20-0896, available through IBM branch offices.

APPENDIXES

APPENDIX A. MODULE DIRECTORY

NAME	DESCRIPTION
AUTH	Authentication Routine. Ensure the address space requesting a service is an authorized user of that service. Implement transfer of control primitives: CALL, RETURN and EXIT. Invoke the Transmitter Module when CALLXMIT or RETXMIT is issued.
CONSOLE	Console Address Space. Provide support for the system console. READ entry point: Facilitate the user-issued console READ SVC. WRITE entry point: Facilitate the user-issued console WRITE SVC. IOINT entry point: Second-level I/O interrupt handler; handle interrupts from the console.
DATABLES	DAT Tables. Contains segment and page tables for every address space (system and user) of the control program.
DSP	Dispatcher. Dispatch user address spaces or place the system into an idle wait state if it is temporarily out of work.
EXTFLIH	External First-Level Interrupt Handler. Respond directly to external interruptions. Activate the external second-level interrupt handler if the interrupt is identifiable.
EXTSLIH	External Second-Level Interrupt Handler. Handles timer-related events: Clock Comparator: Activate the WAKEUP entry point of the Sleep Address Space.

CPU Timer: Notify the scheduler of time-slice end for a user.

IOFLIH I/O First-Level Interrupt Handler.

Respond directly to an I/O interrupt by activating the IOINT entry point of the Console Address Space.

OPERATOR Operator Address Space.

Analyzes and executes human operator command requests entered at the console.

PGMFLIH Program First-Level Interrupt Handler.

Respond immediately to program interruptions:

Initiates termination processing for a user who program checked;

Shuts the system down in the event of a program check in system code;

Perform privileged operations requested via Monitor Call.

RLAS Run List Address Space.

Manage the list of dispatchable UCBs:

ENQUEUE entry point: Add a UCB to the Run List with priority queuing based on the value of the UCB priority field;

DEQUEUE entry point: Remove the UCB at the head of the Run List;

GETDUCB (Get Dispatch UCB) entry point: Fetch a copy of the first UCB on the Run List.

SCH Scheduler.

Schedule a UCB for execution:

TSEND (Time Slice End) entry point: Re-schedule a running user who consumed his dispatch quantum;

ROUSE entry point: Schedule a user whose sleep interval just expired;

IODONE entry point: Schedule a user whose console request just completed.

SHUTDOWN Shutdown Address Space

Terminates the running system in response to the operator SHUTDOWN command.

SLAS Sleep List Address Space.

Manage the list of sleeping UCBs:

ENQUEUE entry point: Add a UCB to the Sleep List using priority queuing based on wakeup time-of-day.

DEQUEUE entry point: Remove the UCB at the head of the Sleep List.

SLEEP Sleep Address Space.

Facilitate user-issued SLEEP SVC:

SLEEP entry point: Put a user to Sleep (add him to the Sleep List) for the number of microseconds he requested.

WAKEUP entry point: Driven by a timer interruption; remove the user at the head of the Sleep List and schedule him to run again. Restart the timer for the next sleeping user.

STARTUP System Startup Code.

Initialize control registers, storage keys, timers, etc. and EXIT to the Dispatcher to place the system into operation.

SVCFLIH Supervisor Call First-Level Interrupt Handler.

Respond immediately to SVC interruption by activating the Authentication Routine.

TLAS Terminate List Address Space.

Manage the list of terminated users by adding the UCB for a dead user on the Terminate List.

TRACE System Trace Routine.

Insert the trace entry supplied by the caller into the system trace table.

TRU Terminate Running User.

Remove the UCB for a user who program checked from the Run List and have his UCB added to the Terminate List.

USERN User n Address Space; 1 <= n <= 5.

Each user address space represents a unit of dispatchable work for the system.

XMIT Transmitter Module.

Copy the contents of the sender's parameter page to the receiver's.

In addition, most of these modules have an accompanying parameter page. These are included in the source listing appendix.

APPENDIX B. SOURCE LISTINGS

```
                                3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* PGM00040
                                4 *+------------------------------------------------------------------------+* PGM00050
                                5 *|                                                                        |* PGM00060
                                6 *|Module:   PGMFLIH                                                       |* PGM00070
                                7 *|                                                                        |* PGM00080
                                8 *|          Program Interrupt First-Level Interrupt Handler.             |* PGM00090
                                9 *|                                                                        |* PGM00100
                               10 *|Function: Perform early stages of processing necessary when the         |* PGM00110
                               11 *|          system is either running itself or a user and a program       |* PGM00120
                               12 *|          interrupt occurs.  First, it saves the volatile machine       |* PGM00130
                               13 *|          state which includes saving the state of a running user       |* PGM00140
                               14 *|          (if there is one).  Then it performs one of the follow-       |* PGM00150
                               15 *|          ing actions:                                                  |* PGM00160
                               16 *|                                                                        |* PGM00170
                               17 *|              1) If a FLIH issued a SAVEREGS monitor call, return       |* PGM00180
                               18 *|                 to that FLIH with SAVEREGS function complete.          |* PGM00190
                               19 *|                                                                        |* PGM00200
                               20 *|              2) If the system was running a user, that user            |* PGM00210
                               21 *|                 program checked and must be terminated.  This          |* PGM00220
                               22 *|                 module EXITs to the Terminate a Running User           |* PGM00230
                               23 *|                 address space.                                         |* PGM00240
                               24 *|                                                                        |* PGM00250
                               25 *|              3) If the system was running itself and one of its        |* PGM00260
                               26 *|                 modules suffered a program check, control program      |* PGM00270
                               27 *|                 operation must be terminated.  Load a disabled         |* PGM00280
                               28 *|                 wait PSW indicative of this fact.                      |* PGM00290
                               29 *|                                                                        |* PGM00300
                               30 *|              3) If the system issued a Monitor Call, some control      |* PGM00310
                               31 *|                 program component is in need of services that          |* PGM00320
                               32 *|                 cannot be performed in the problem state.  If the      |* PGM00330
                               33 *|                 caller is not authorized for the requested ser-        |* PGM00340
                               34 *|                 vice, terminate the control program.  Otherwise,       |* PGM00350
                               35 *|                 perform the service on the caller's behalf.            |* PGM00360
                               36 *|                                                                        |* PGM00370
                               37 *|          Each FLIH invokes PGMFLIH to save the environment in          |* PGM00380
                               38 *|          page zero.  PGMFLIH is the only module that can write         |* PGM00390
                               39 *|          to page zero; the other FLIHs have read-only access.          |* PGM00400
                               40 *|                                                                        |* PGM00410
                               41 *|          Sometimes this module returns to its caller, i.e. it          |* PGM00420
                               42 *|          performs a return from interrupt (RTI).  Other times, it      |* PGM00430
                               43 *|          activates another address space.                              |* PGM00440
                               44 *|                                                                        |* PGM00450
                               45 *|Caller:   Not called; rather, activated by a hardware PSW swap          |* PGM00460
                               46 *|          which saves the current PSW in an assigned storage            |* PGM00470
                               47 *|          location for program interrupt class and reloads the          |* PGM00480
                               48 *|          current PSW from the assigned storage location contain-       |* PGM00490
                               49 *|          ing the new PSW for program interrupt class.                  |* PGM00500
                               50 *|                                                                        |* PGM00510
                               51 *|Mode:     Supervisor state, DAT off, Key 4.                             |* PGM00520
                               52 *|                                                                        |* PGM00530
                               53 *+------------------------------------------------------------------------+* PGM00540
                               54 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* PGM00550
```

```
LOC  OBJECT CODE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                                ASM 0201 20.42 12/04/86
                                   56          PUNCH 'SPB'         Make module start on a page boundry.      PGM00570
                                   58          EXTRN AUTHST,OSPST,XMITST,SLEEPST,SLASST,CONSST,CONSPP,SHUTST PGM00590
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86
000000                              60 PGMFLIH  CSECT              Program First-Level Interrupt Handler      PGM00610

                                    62 *-------------------------------------------------------------------* PGM00630
                                    63 * A. Save registers in low core and establish addressibility:       * PGM00640
                                    64 *-------------------------------------------------------------------* PGM00650

                           00000    66          USING PSA,R0         Make low-core fields addressable          PGM00670

                                    68          CKPTREGS             Save volatile state information in lowcore PGM00690
000000 8209 0460       00460        69+         STPT  CPUTIMER       Save possible user quantum remainder      CKP00030
000004 900F 0400       00400        70+         STM   R0,R15,PSAGREGS Save general registers                  CKP00040
000008 6000 0440       00440        71+         STD   F0,PSAFREGS    Save                                      CKP00050
00000C 6020 0448       00448        72+         STO   F2,PSAFREGS+8    Floating                            CKP00060
000010 6040 0450       00450        73+         STD   F4,PSAFREGS+16     Point                             CKP00070
000014 6060 0458       00458        74+         STD   F6,PSAFREGS+24       Registers                       CKP00080
000018 8611 0494       00494        75+         STCTL 1,1,INTCREG1   Save contents of control register 1      CKP00090

00001C 58C0 006C       0006C        77          L     R12,PGMNEW+4   Get the address of this module in R12   PGM00710
                           00000    78          USING PGMFLIH,R12    Establish R12 as the overall module base  PGM00720
000020 47F0 C02C       0002C        79          8     AROUND         Move past eyecatcher                      PGM00730

000024 D7C704C603C9C840             81          0C    CL8'PGMFLIH'   Handy when poking around in storage/dumps  PGM00750

                                    83 *-------------------------------------------------------------------* PGM00770
                                    84 * 8. Move the system state at the time of the interrupt to a page   * PGM00780
                                    85 *    zero field used by SVCFLIH/AUTH.                              * PGM00790
                                    86 *-------------------------------------------------------------------* PGM00800

00002C                              88 AROUND   DS    0H             Now the real work begins:                 PGM00820

00002C D200 0490 0468 00490 00468   90          MVC   INTSTATE(1),SYSTATE Move state @ interrupt to low core  PGM00840

                                    92 *-------------------------------------------------------------------* PGM00860
                                    93 * C. Oetermine if the system was running a user when the interrupt   * PGM00870
                                    94 *    occurred.  If it was, change the state variable to indicate the * PGM00880
                                    95 *    system is now running itself before continuing on.             * PGM00890
                                    96 *-------------------------------------------------------------------* PGM00900

000032 9580 0468       00468        98          CLI   SYSTATE,RUNUSER System running a user?                 PGM00920
000036 4770 C0A0       000A0        99          8NE   CHKIDLE          No  - See if the system was idle.       PGM00930
00003A 9240 0468       00468       100          MVI   SYSTATE,RUNSYS Yes - Indicate system running itself now. PGM00940

                                   102 *-------------------------------------------------------------------* PGM00960
                                   103 * D. The system was running a user at the point of interruption.    * PGM00970
                                   104 *    We must now implement the SAVEREGS function, which moves the  * PGM00980
                                   105 *    volatile user environment to a user savearea in page 0.  Page 0 * PGM00990
                                   106 *    is shared with the Run List Address Space, and that module takes * PGM01000
                                   107 *    care of ultimately updating the running user's UC8 from the    * PGM01010
                                   108 *    saved environment data in low core.                            * PGM01020
                                   109 *                                                                   * PGM01030
                                   110 *    Save the user's general and floating-point registers and the   * PGM01040
                                   111 *    remaining dispatch quantum in the saved environment savearea.  * PGM01050
                                   112 *-------------------------------------------------------------------* PGM01060

00003E 023F 04A0 0400 004A0 00400  114          MVC   SAVGREGS(16*4),PSAGREGS Save user general registers.    PGM01080
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86

000044 D21F 04E0 0440 004E0 00440   115          MVC   SAVFREGS(4*8),PSAFREGS  Save user floating-point regs.   PGM01090
00004A D207 0500 0460 00500 00460   116          MVC   SAVTIMER,CPUTIMER       Save user dispatch quantum.      PGM01100

                                    118 *-------------------------------------------------------------------* PGM01120
                                    119 * E. Now determine if this program check we received while "running a * PGM01130
                                    120 *    user" was really from a running user or if it is a program     * PGM01140
                                    121 *    check generated by a FLIH issuing a SAVEREGS Monitor Call.  If * PGM01150
                                    122 *    it was a FLIH, the DAT bit in the program check old PSW will be * PGM01160
                                    123 *    off, because FLIHs do not run in relocate mode.  On the other  * PGM01170
                                    124 *    hand, all user work runs with DAT on; this will clearly identify * PGM01180
                                    125 *    entry into this module directly from a running user who program * PGM01190
                                    126 *    checked (as opposed to another interrupt condition vectoring us * PGM01200
                                    127 *    here to possibly checkpoint regs in low core).                 * PGM01210
                                    128 *-------------------------------------------------------------------* PGM01220

000050 9104 0028       00028        130          TM    PGMOLD,DATON  Did a running user program check?       PGM01240
000054 4780 C06A       0006A        131          BZ    FLIHSAVE      No  - A FLIH vectored here via SAVEREGS MC PGM01250

                                    133 *-------------------------------------------------------------------* PGM01270
                                    134 * F. Yes, a running user program checked.  Save his PSW from the    * PGM01280
                                    135 *    program check old PSW (by copying it into the environment save- * PGM01290
                                    136 *    area), save the program interrupt code for problem analysis, and * PGM01300
                                    137 *    terminate him.                                                 * PGM01310
                                    138 *-------------------------------------------------------------------* PGM01320

000058 D207 0498 0028 00498 00028   140          MVC   SAVPSW,PGMOLD Save user's program check PSW.          PGM01340
00005E 0200 049B 008F 0049B 0008F   141          MVC   SAVPSW+3(1),PGMC00E+1 Save interrupt code too.       PGM01350
                                    142          EXIT  TRU           Terminate the user address space.      PGM01360
000064 5800 C3D0       003D0        143+         L     R0,=CL4'TRU'       Load target address space identifier EX100040
000068 0A02                         144+         SVC   2                  Go to that address space.          EX100050

                                    146 *-------------------------------------------------------------------* PGM01380
                                    147 * G. At this point, we know the system was running a user when some  * PGM01390
                                    148 *    interrupt other than program check occurred.  We need to save  * PGM01400
                                    149 *    the user's PSW at point of interruption, but it could reside in * PGM01410
                                    150 *    any of several old PSW assigned storage locations.  The        * PGM01420
                                    151 *    technique used to identify where the original interrupt came   * PGM01430
                                    152 *    from uses the value of the storage key in the program old PSW. * PGM01440
                                    153 *    Each FLIH runs with a different key.  When a FLIH issues the   * PGM01450
                                    154 *    SAVEREGS monitor call, the current PSW is stored in the program * PGM01460
                                    155 *    old PSW assigned storage location.  We will extract the key    * PGM01470
                                    156 *    field from program check old PSW and subsequently use that key * PGM01480
                                    157 *    as an index into a branch table to fetch the address of the    * PGM01490
                                    158 *    old PSW location containing the running user's PSW.            * PGM01500
                                    159 *-------------------------------------------------------------------* PGM01510

00006A                              161 FLIHSAVE DS    0H             Here to save user PSW on behalf of FLIH: PGM01530

00006A 1755                         163          XR    R5,R5          Clear R5.                              PGM01550
00006C 4350 0029       00029        164          IC    R5,PGMOLD+1    Get 2nd byte of PSW containing storage key PGM01560
000070 8850 0004       00004        165          SRL   R5,4          Get rid of junk; key now in bits 28-31. PGM01570
000074 0650                         166          BCTR  R5,0           Subtract one because FLIH keys start at 1 PGM01580
000076 8950 0003       00003        167          SLL   R5,3           R5 = R5 * (8 bytes/instr pair)         PGM01590
00007A 47F5 C07E       0007E        168          B     PSWADDR(R5)    Get address of user old PSW in R5:     PGM01600
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 20.42 12/04/86

000D7E                             17D PSWADDR  DS    0H            PSW address assignment table:                PGM01620

00007E 4150 0020       00020        172          LA    R5,SVCOLD     User PSW in SVC old location.      *KEY=1* PGM01640
000082 47F0 C096       00096        173          B     SAVEPSW       Continue.                                  PGM01650

000086 4150 0018       00018        175          LA    R5,EXTOLD     User PSW in External old location. *KEY=2* PGM01670
00008A 47F0 C096       00096        176          B     SAVEPSW       Continue.                                  PGM01680

00008E 4150 0038       00038        178          LA    R5,IOOLD      User PSW in I/O old location.      *KEY=3* PGM01700
000092 47F0 C096       00096        179          B     SAVEPSW       Continue.                                  PGM01710

                                    181 *-------------------------------------------------------------------* PGM01730
                                    182 * H. Now we have the address of the user's PSW.  Copy that PSW to the * PGM01740
                                    183 *    environment savearea and return to the FLIH with the SAVEREGS    * PGM01750
                                    184 *    function complete.                                               * PGM01760
                                    185 *-------------------------------------------------------------------* PGM01770

000096                              187 SAVEPSW  DS    0H            Here to save user's PSW:                     PGM01790

000096 D207 0498 5000  00498 00000  189          MVC   SAVPSW,0(R5)  Copy interrupt old PSW to env. savearea.    PGM01810

0D009C 8200 0028       00028        191          LPSW  PGMOLD       Return to FLIH with regs in PSAGREGS.        PGM01830

                                    193 *-------------------------------------------------------------------* PGM01850
                                    194 * I. When we are here, it means that an interrupt occurred while the  * PGM01860
                                    195 *    system was not running a user.  The true source of interruption  * PGM01870
                                    196 *    can be determined simply by examining the system state.  If the  * PGM01880
                                    197 *    state indicates idle, then an asynchronous interrupt occurred    * PGM01890
                                    198 *    while the system was in a wait state, and we are here because    * PGM01900
                                    199 *    the corresponding FLIH issued a SAVEREGS monitor call.  Other-   * PGM01910
                                    200 *    wise, the system had to have been running itself, as this is the * PGM01920
                                    201 *    only remaining assigned system state.                            * PGM01930
                                    202 *-------------------------------------------------------------------* PGM01940

0000A0                              204 CHKIDLE  DS    0H            Here when SYSTATE ¬= RUNUSER:                PGM01960

0000A0 952D 0468       00468        206          CLI   SYSTATE,RUNIDLE Was the system idle when interrupted?    PGM01980
0000A4 4770 C0B0       000B0        207          BNE   RUNSYSTM       No  - Handle a system program check.     PGM01990

                                    209 *-------------------------------------------------------------------* PGM02010
                                    210 * J. Yes, we are here due to an asynch FLIH issuing a SAVEREGS MC.    * PGM02020
                                    211 *    Set the state to indicate system is now running itself and       * PGM02030
                                    212 *    return to the FLIH with the SAVEREGS function complete.          * PGM02040
                                    213 *-------------------------------------------------------------------* PGM02050

0000A8 9240 0468       00468        215          MVI   SYSTATE,RUNSYS Yes - Indicate system now running itself PGM02070
0000AC 8200 0028       00028        216          LPSW  PGMOLD        Return to External or I/O FLIH.             PGM02080

                                    218 *-------------------------------------------------------------------* PGM02100
                                    219 * K. We are here when the running system program checks.  Determine   * PGM02110
                                    220 *    if the program check was a failure or a monitor call.  If it is  * PGM02120
                                    221 *    not a monitor call then it is a failure.  In this case, the      * PGM02130
                                    222 *    system must be shut down.                                        * PGM02140
                                    223 *-------------------------------------------------------------------* PGM02150
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86

0000B0                            225 RUNSYSTM DS    0H            Here when Control Program program checks:  PGM02170

0000B0 9540 00BF      00BF        227          CLI   PGMCODE+1,MCEVENT Was interrupt caused by Monitor Call?  PGM02190
0000B4 4770 C2A2      002A2       228          BNE   PGMCHECK      No - Shut 'er down...                      PGM02200

                                  230 *-------------------------------------------------------------------* PGM02220
                                  231 * L. A system component issued a Monitor Call instruction.  If the    * PGM02230
                                  232 *    request is undefined, shut down the system:                      * PGM02240
                                  233 *-------------------------------------------------------------------* PGM02250

0000BB 5B10 009C      0009C       235          L     R1,MONCODE    Get Monitor Call code                      PGM02270
0000BC 5910 C104      00104       236          C     R1,MAXCODE    Is it implemented?                         PGM02280
0000C0 4720 C2A6      002A6       237          BH    BIGCODE       No - It's undefined.  Have to abend.       PGM02290

                                  239 *-------------------------------------------------------------------* PGM02310
                                  240 * M. A system component requested a defined service via Monitor Call. * PGM02320
                                  241 *    Obtain the requestor's identity and go perform the service:      * PGM02330
                                  242 *-------------------------------------------------------------------* PGM02340

0000C4 B611 C3AB      003AB       244          STCTL 1,1,CALLERID  XFER caller's identity (STD) to storage    PGM02360
0000C8 5B70 C3AB      003AB       245          L     R7,CALLERID   Get caller's STD in R7                     PGM02370

0000CC B910 0002      00002       247          SLL   R1,2           R1 = R1 * (4 bytes/branch instruction)     PGM02390
000000 47F1 C004      00004       248          B     MCTABLE(R1)   Go perform monitor service.                 PGM02400

000004                            250 MCTABLE  DS    0H            Monitor Call branch table:                 PGM02420

000004 47F0 C10B      0010B       252          B     MACTAUTH      Moncode  0: "Activate AUTH"                PGM02440
00000B 47F0 C130      00130       253          B     MSDP          Moncode  1: "Set Dispatch Paramaters"      PGM02450
00000C 47F0 C170      00170       254          B     MDSPATCH      Moncode  2: "Dispatch"                     PGM02460
0000E0 47F0 C1A4      001A4       255          B     MABEND        Moncode  3: "ABEND"                       PGM02470
0000E4 47F0 C1B0      001B0       256          B     MBIND         Moncode  4: XMIT "Bind" pages service      PGM02480
0000EB 47F0 C10C      0010C       257          B     MDETACH       Moncode  5: XMIT "Detach" pages service    PGM02490
0000EC 47F0 C202      00202       258          B     MSCK          Moncode  6: "Set Clock Comparator"         PGM02500
0000F0 47F0 C1EE      001EE       259          B     MIDLE         Moncode  7: "System No-Work Wait"          PGM02510
0000F4 47F0 C212      00212       260          B     MSAVEREG      Moncode  8: "Save registers"              PGM02520
0000FB 47F0 C216      00216       261          B     MCONSIO       Moncode  9: "Console Start I/O"           PGM02530
0000FC 47F0 C24E      0024E       262          B     MPURGECN      Moncode 10: "Purge Console Input Data"    PGM02540
000100 47F0 C2BE      002BE       263          B     MSHUTSYS      Moncode 11: "Shut Down the System"        PGM02550

                            00104 265 ENOTABLE EQU   *             $ Marks end of branch table $              PGM02570

000104 00000008                   267 MAXCODE  DC    A(((ENOTABLE-MCTABLE)/4)-1) Maximum Moncode available.  PGM02590
```

```
                                    269 *-------------------------------------------------------------------* PGM02610
                                    270 * N. Service: Activate Authentication Routine.                      * PGM02620
                                    271 *                                                                   * PGM02630
                                    272 *    Authorized Callers: SVC First-Level Interrupt Handler.         * PGM02640
                                    273 *                                                                   * PGM02650
                                    274 *    Function: -Shut system down if caller is unauthorized.         * PGM02660
                                    275 *              -Activate the Authentication Routine Address Space via * PGM02670
                                    276 *               LPSW.                                               * PGM02680
                                    277 *-------------------------------------------------------------------* PGM02690

000108                              279 MACTAUTH DS    0H            Activate Authentication Routine:         PGM02710

000108 94F0 0029      00029         281          NI    PGMOLD+1,X'F0' Turn off mode bits; leave key intact.   PGM02730
00010C 9510 0029      00029         282          CLI   PGMOLD+1,X'10' Was caller = SVC FLIH, i.e. key 1?       PGM02740
000110 4770 C2AA      002AA         283          BNE   BADAA          No  - Illegal call to ACTAUTH.  Shutdown. PGM02750

000114 B711 C3B0      003B0         285          LCTL  1,1,AUTHRTN    Yes - Load Control Register 1 w/AUTH sto PGM02770
00011B 0202 0475 C3C9 00475 003C9   286          MVC   SYSPSW+5(3),ATHVADDR Set up AUTH entry point virt addr PGM02780
00011E 940F 0471      00471         287          NI    SYSPSW+1,X'0F' Reset storage key to zeros               PGM02790
000122 D600 0471 C3C8 00471 003C8   288          OC    SYSPSW+1(1),AUTHKEY Set AUTH's PSW key                  PGM02800
000128 9240 0468      00468         289          MVI   SYSTATE,RUNSYS Indicate system is now running itself.  PGM02810
00012C 8200 0470      00470         290          LPSW  SYSPSW         Activate AUTH                            PGM02820
```

```
                                    292 *-------------------------------------------------------------------* PGM02840
                                    293 * 0. Service: Set Dispatch Parameters.                             * PGM02850
                                    294 *                                                                  * PGM02860
                                    295 *    Authorized Callers: Authentication Routine, Dispatcher.       * PGM02870
                                    296 *                                                                  * PGM02880
                                    297 *    Function: -Shut system down if caller is unauthorized.        * PGM02890
                                    298 *              -Move the dispatch parameters from the low core     * PGM02900
                                    299 *               savearea into the system or user low core variables, * PGM02910
                                    300 *               depending on who the caller is (AUTH dispatches the * PGM02920
                                    301 *               system; Dispatcher dispatches a user).            * PGM02930
                                    302 *              -RTI                                                * PGM02940
                                    303 *-------------------------------------------------------------------* PGM02950

000130                              305 MSDP     OS    OH            Set Dispatch Parameters Routine:       PGM02970
                                    306          ISIT  DSPCHER       Is caller = Dispatcher?                PGM02980
000130 5970 C3AC      003AC         307+         C     R7,OSPCHER    Is is this guy?                        IS100030
000134 4780 C15A      0015A         308          BE    OSPSDP        Yes - He is allowed.                   PGM02990
                                    309          ISIT  AUTHRTN       No  - Is it AUTH??                     PGM03000
000138 5970 C3B0      003B0         310+         C     R7,AUTHRTN    Is is this guy?                        IS100030
00013C 4770 C2AE      002AE         311          8NE   BAOSOP        No  - Shut down the system.            PGM03010

000140 0203 046C 0408 0046C 00408   313          MVC   SYSSTO,PSAGREGS+(R2*4) Yes - Move dispatch STO to locore PGM03030
000146 940F 0471      00471         314          NI    SYSPSW+1,X'0F' Set PSW key to zeros.                 PGM03040
00014A D600 0471 040C 00471 0040C   315          OC    SYSPSW+1(1),PSAGREGS+(R3*4) Set PSW key for addr space PGM03050
000150 D202 0475 0400 00475 00400   316          MVC   SYSPSW+5(3),PSAGREGS+((R3*4)+1) Set PSW entry pt vaddr PGM03060
000156 47F0 C29A      0029A         317          B     RTI           Return to AUTH.                        PGM03070

00015A                              319 DSPSDP   DS    OH            Here when Dispatcher issues a SDP:     PGM03090

00015A 0203 047C 0408 0047C 00408   321          MVC   USRSTO,PSAGREGS+(R2*4)  Move user dispatch           PGM03110
000160 D207 0480 040C 00480 0040C   322          MVC   USRPSW,PSAGREGS+(R3*4)    parameters to            PGM03120
000166 D207 0488 0414 00488 00414   323          MVC   USRTIMER,PSAGREGS+(R5*4)    low core.              PGM03130
00016C 47F0 C29A      0029A         324          B     RTI           Return to dispatcher.                  PGM03140
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86

                                    326 *-------------------------------------------------------------------* PGM03160
                                    327 * P. Service: Dispatch.                                             * PGM03170
                                    328 *                                                                   * PGM03180
                                    329 *    Authorized Callers: Authentication Routine, Dispatcher.        * PGM03190
                                    330 *                                                                   * PGM03200
                                    331 *    Function: -Shut system down if caller is unauthorized.         * PGM03210
                                    332 *              -Dispatch user or system address space, depending on * PGM03220
                                    333 *               who the caller is (see "Set Dispatch Paramaters"    * PGM03230
                                    334 *               description above).  This action includes setting the * PGM03240
                                    335 *               system state variable to indicate either running the * PGM03250
                                    336 *               system or running a user.                           * PGM03260
                                    337 *-------------------------------------------------------------------* PGM03270

000170                              339 MDSPATCH DS    0H            Dispatch Routine:                       PGM03290

                                    341          ISIT  DSPCHER       Is caller = Dispatcher?                 PGM03310
000170 5970 C3AC      003AC         342+         C     R7,DSPCHER    Is is this guy?                         IS100030
000174 4780 C190      00190         343          8E    DSPDSPCH      Yes - He is allowed.                    PGM03320
                                    344          ISIT  AUTHRTN       No  - Is caller = AUTH??                PGM03330
000178 5970 C380      00380         345+         C     R7,AUTHRTN    Is is this guy?                         IS100030
00017C 4770 C282      00282         346          8NE   8ADDSPCH      No  - Shut 'er down...                  PGM03340

000180 8711 046C      0046C         348          LCTL  1,1,SYSSTO    Yes - Load Control Register 1           PGM03360
000184 980F 0400      00400         349          LM    R0,R15,PSAGREGS Restore general registers             PGM03370
000188 9240 0468      00468         350          MVI   SYSTATE,RUNSYS Indicate system is now running itself. PGM03380
00018C 8200 0470      00470         351          LPSW  SYSPSW        Dispatch system address space.          PGM03390

000190                              353 DSPDSPCH DS    0H            Here when Dispatcher issues Dispatch:   PGM03410

000190 8711 047C      0047C         355          LCTL  1,1,USRSTO    Load Control Register 1                 PGM03430
000194 980F 0400      00400         356          LM    R0,R15,PSAGREGS Restore user's regs                   PGM03440
000198 8208 0488      00488         357          SPT   USRTIMER      Set CPU Timer dispatch quantum          PGM03450
00019C 9280 0468      00468         358          MVI   SYSTATE,RUNUSER Indicate system is running a user.  PGM03460
0001A0 8200 0480      00480         359          LPSW  USRPSW        Dispatch user address space.            PGM03470
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86
                                   361 *-------------------------------------------------------------------* PGM03490
                                   362 * Q. Service: ABEND.                                                * PGM03500
                                   363 *                                                                   * PGM03510
                                   364 *    Authorized Callers: Authentication Routine.                    * PGM03520
                                   365 *                                                                   * PGM03530
                                   366 *    Function: -Shut system down if caller is unauthorized.         * PGM03540
                                   367 *              -Shut system down indicating via wait code the fact  * PGM03550
                                   368 *               that the Authentication Routine brought the system  * PGM03560
                                   369 *               down.                                               * PGM03570
                                   370 *-------------------------------------------------------------------* PGM03580

0001A4                             372 MABEND0  DS    0H            ABEND running system:                   PGM03600

                                   374          ISIT  AUTHRTN       Is caller = AUTH?                       PGM03620
0001A4 5970 C3B0   003B0        375+         C     R7,AUTHRTN      Is Is this guy?                         IS100030
0001A8 4770 C2B6   002B6        376          BNE   BADABEND        No  - Irony is we ABEND0 anyway!       PGM03630

0001AC 8200 C310   00310        378          LPSW  ABENDPSW        Yes - "Normal" ABEND...                 PGM03650
```

```
                                    380 *-------------------------------------------------------------------* PGM03670
                                    381 * R. Service: BIND.                                                 * PGM03680
                                    382 *                                                                   * PGM03690
                                    383 *    Authorized Callers: Transmitter Module.                        * PGM03700
                                    384 *                                                                   * PGM03710
                                    385 *    Function: XMIT calls BIND with caller and receiver address space * PGM03720
                                    386 *              STOs in R0 and R1, respectively.  BIND uses these STOs * PGM03730
                                    387 *              to locate the page table entry (PTE) for page zero of  * PGM03740
                                    388 *              each address space.  It then copies the caller's page  * PGM03750
                                    389 *              0 PTE to XMIT's page 0 PTE, and moves the receiver's   * PGM03760
                                    390 *              page 0 PTE to XMIT's page 1 PTE, thereby binding these * PGM03770
                                    391 *              two pages to the XMIT address space.  The page table   * PGM03780
                                    392 *              for XMIT's segment 0 is contained within this source   * PGM03790
                                    393 *              module and is defined as an entry; XMIT segment table  * PGM03800
                                    394 *              and page table for segment 1 are defined in DATABLES.  * PGM03810
                                    395 *              BIND returns immediately to XMIT.                      * PGM03820
                                    396 *-------------------------------------------------------------------* PGM03830

0001B0                              398 MBIND    DS    0H            XMIT Bind Routine:                      PGM03850

                                    400          ISIT  XMITTER       Is caller = Transmitter Module?         PGM03870
0001B0 5970 C3B4      003B4         401+         C     R7,XMITTER    Is is this guy?                         ISI00030
0001B4 4770 C2BA      002BA         402          BNE   BADBIND       No  - Illegal caller.  Shutdown system.  PGM03880
0001B8 9856 0400      00400         403          LM    R5,R6,PSAGREGS+(R0*4) Yes - Get caller and receiver STOs PGM03890
                                    404 *                                     from saved register image.    PGM03900
0001BC 5855 0000      00000         405          L     R5,0(R5)      Get PT0 for caller's segment 0.         PGM03910
0001C0 4855 0000      00000         406          LH    R5,0(R5)      Get PTE for caller's page 0.            PGM03920
0001C4 5866 0000      00000         407          L     R6,0(R6)      Get PT0 for receiver's segment 0.       PGM03930
0001C8 4866 0000      00000         408          LH    R6,0(R6)      Get PTE for receiver's page 0.          PGM03940

0001CC                              410 COMMON   DS    0H            Here for common BIND and DETACH proc'ing: PGM03960

0001CC 4050 C368      00368         412          STH   R5,XMIT0      Bind caller page 0 to XMIT page 0.      PGM03980
0001D0 4060 C36A      0036A         413          STH   R6,XMIT0+2    Bind receiver page 0 to XMIT page 1.    PGM03990
0001D4 B20D 0000      00000         414          PTLB                Force DAT table fetches...              PGM04000
0001D8 47F0 C29A      0029A         415          B     RTI           Return to XMIT.                         PGM04010
```

```
                                417 *-------------------------------------------------------------------* PGM04030
                                418 * S. Service: DETACH.                                              * PGM04040
                                419 *                                                                   * PGM04050
                                420 *    Authorized Callers: Transmitter Module.                        * PGM04060
                                421 *                                                                   * PGM04070
                                422 *    Function: XMIT calls DETACH when it is through moving data from  * PGM04080
                                423 *              the caller's address space to the receiver's.  At this * PGM04090
                                424 *              point it no longer needs access to either parameter    * PGM04100
                                425 *              page attached to it by BIND, and for safety's sake it  * PGM04110
                                426 *              calls DETACH to detach and invalidate its page 0 and 1 * PGM04120
                                427 *              PTEs.  Page table manipulation is common code between  * PGM04130
                                428 *              BIND and DETACH.  DETACH returns immediately to XMIT.  * PGM04140
                                429 *-------------------------------------------------------------------* PGM04150

0001DC                          431 MDETACH  DS    0H            Detach Routine:                         PGM04170

                                433          ISIT  XMITTER       Is caller = Transmitter Module?         PGM04190
                                                                                                         IS100030
0001DC 5970 C3B4      003B4     434+         C     R7,XMITTER    Is is this guy?                         PGM04200
0001E0 4770 C2BE      002BE     435          BNE   BADDTACH      No  - Illegal invocation.  Shutdown.    PGM04210
0001E4 4150 0008      00008     436          LA    R5,8          Construct phony page table entries      PGM04220
0001E8 1865                     437          LR    R6,R5          indicating page invalid and PFRA = 0.  PGM04230
0001EA 47F0 C1CC      001CC     438          B     COMMON        Go invalidate XMIT page 0 and page 1 PTEs.
```

```
                                 440 *-------------------------------------------------------------------* PGM04250
                                 441 * T. Service: IOLE.                                                * PGM04260
                                 442 *                                                                   * PGM04270
                                 443 *    Authorized Caller: Dispatcher.                                 * PGM04280
                                 444 *                                                                   * PGM04290
                                 445 *    Function: DSP calls IDLE when it discovers the system has      * PGM04300
                                 446 *              nothing to do.  IDLE sets the system state variable to * PGM04310
                                 447 *              indicate the idle condition and then loads a wait PSW  * PGM04320
                                 448 *              enabled for I/O and external interrupts.             * PGM04330
                                 449 *-------------------------------------------------------------------* PGM04340
0001EE                           451 MIDLE    DS    0H                                                     PGM04360
                                 453          ISIT  DSPCHER       Is caller = Dispatcher?                  PGM04380
0001EE 5970 C3AC      003AC      454+         C     R7,DSPCHER    Is is this guy?                          IS100030
0001F2 4770 C2C2      002C2      455          BNE   BADIDLE      No  - Unauthorized invocation. Shut down.  PGM04390
0001F6 9220 0468      00468      456          MVI   SYSTATE,RUNIOLE Yes - Indicate running system going idle PGM04400
0001FA 8208 C398      00398      457          SPT   BIGTIMER      Prevent meaningless interrupts.          PGM04410
0001FE 8200 C3A0      003A0      458          LPSW  WAIT          Throw system into enabled wait...        PGM04420
```

```
                                460 *-------------------------------------------------------------------* PGM04440
                                461 * U. Service: Set Clock Comparator (SCK).                          * PGM04450
                                462 *                                                                   * PGM04460
                                463 *    Authorized Caller: Sleep Address Space.                        * PGM04470
                                464 *                                                                   * PGM04480
                                465 *    Function: Issue the privileged Set Clock Comparator instruction * PGM04490
                                466 *              on behalf of authorized but unprivileged code.  The  * PGM04500
                                467 *              NAP and ROUSE entry points of the Sleep address space * PGM04510
                                468 *              need to alter the clock comparator to faciliate user * PGM04520
                                469 *              sleep activity.  After setting the clock, SCK returns * PGM04530
                                470 *              immediately to its caller.                           * PGM04540
                                471 *-------------------------------------------------------------------* PGM04550

000202                          473 MSCK     DS    OH            Set Clock Comparator Routine:          PGM04570

                                475          ISIT  SLEEPAS       Is caller = Sleep Address Space?       PGM04590
000202 5970 C388      00388     476+         C     R7,SLEEPAS    Is is this guy?                        IS100030
000206 4770 C2C6      002C6     477          8NE   8ADSCK        No  - Unauthorized invocation. Shut down. PGM04600
00020A B206 0408      00408     478          SCKC  PSAGREGS+(R2*4) Yes - Set clock comparator.          PGM04610
00020E 47F0 C29A      0029A     479          8     RTI           Return to caller.                      PGM04620
```

```
                                481 *-------------------------------------------------------------------* PGM04640
                                482 * V. Service: SAVEREGS.                                            * PGM04650
                                483 *                                                                   * PGM04660
                                484 *    Authorized Caller: First level interrupt handlers.             * PGM04670
                                485 *                                                                   * PGM04680
                                486 *    Function: Save machine state information in low core.  This    * PGM04690
                                487 *              service is really a dummy entry point because the    * PGM04700
                                488 *              SAVEREGS function has already been performed.  We are * PGM04710
                                489 *              here because a FLIH issues a SAVEREGS.  Since the sys- * PGM04720
                                490 *              tem is not running a user or idle, control passes to * PGM04730
                                491 *              the branch table above, and it vectors us here.  This * PGM04740
                                492 *              code immediately returns to the FLIH.                * PGM04750
                                493 *                                                                   * PGM04760
                                494 *-------------------------------------------------------------------* PGM04770

000212                          496 MSAVEREG OS     0H            SAVEREGS monitor call:                PGM04790

000212 47F0 C29A      0029A     498          B      RTI           We don't have to do anything.         PGM04810
```

```
                                    500 *-------------------------------------------------------------------* PGM04830
                                    501 * W. Service: CONSIO.                                               * PGM04840
                                    502 *                                                                   * PGM04850
                                    503 *    Authorized Caller: Console Address Space.                      * PGM04860
                                    504 *                                                                   * PGM04870
                                    505 *    Function: Using the I/O command code passed in the low-order   * PGM04880
                                    506 *              byte of R0 and the length value specified in the low * PGM04890
                                    507 *              halfword of R1, modify the console channel program.  * PGM04900
                                    508 *              If this is a read operation, make sure the read CCW is * PGM04910
                                    509 *              not chained to the NOP CCW.  The residual data count * PGM04920
                                    510 *              from the read operation would be lost if it chained to * PGM04930
                                    511 *              a NOP to force concurrent CE,DE (which results in a  * PGM04940
                                    512 *              CSW residual count of 1.)                            * PGM04950
                                    513 *                                                                   * PGM04960
                                    514 *              Issue a Start I/O Fast Release (SIOF) to the console. * PGM04970
                                    515 *              If a severe error is detected shut the system down.  * PGM04980
                                    516 *              Otherwise return immediately to the caller.          * PGM04990
                                    517 *-------------------------------------------------------------------* PGM05000

000216                              519 MCONSIO  DS    0H            CONSIO monitor call:                    PGM05020

                                    521          ISIT  CONSAS        Is caller = Console Address Space?      PGM05040
000216 5970 C3C0      003C0         522+         C     R7,CONSAS     Is is this guy?                         IS100030
00021A 4770 C2CA      002CA         523          8NE   8ADCONIO      No  - Unauthorized invocation. Shut down. PGM05050
00021E D200 C388 0403 00388 00403   524          MVC   CONSCCW(1),PSAGREGS+R0+3 Yes - Move I/O cmd code to CCW PGM05060
000224 D201 C38E 0406 0038E 00406   525          MVC   CONSCCW+6(2),PSAGREGS+(R1*4)+2 Move data length to CCW PGM05070
00022A 9502 C388      00388         526          CLI   CONSCCW,X'02' Is this a read command code?       PGM05080
00022E 4770 C23A      0023A         527          8NE   CMDCHAIN      No  - It's a write.                  PGM05090
000232 9220 C38C      0038C         528          MVI   CONSCCW+4,X'20' Yes - Don't chain to NOP: Allow seperate PGM05100
                                    529 *                                      CE, DE status to trickle in. PGM05110
000236 47F0 C23E      0023E         530          8     DOIO          Go initiate the I/O.                    PGM05120

00023A                              532 CMDCHAIN DS    0H            Here when constructing write channel pgm: PGM05140

00023A 9260 C38C      0038C         534          MVI   CONSCCW+4,X'60' Chain write to NOP: Force concurrent  PGM05160
                                    535 *                                   presentation of CE, DE status.  PGM05170

00023E                              537 OOIO     DS    0H            Here to drive operator console:         PGM05190

00023E 9C01 0009      00009         539          SIOF  9             Start I/O operation to console.         PGM05210
000242 4780 C29A      0029A         540          8C    8,RTI         Return to caller on CC0.                PGM05220
000246 4710 C2D6      002D6         541          8C    1,NOCONS      Shut down system on CC3.                PGM05230
00024A 47F0 C23E      0023E         542          8     OOIO          Re-drive I/O if busy or CSW stored.     PGM05240
```

```
                                544 *-------------------------------------------------------------------* PGM05260
                                545 * X. Service: PURGECON.                                            * PGM05270
                                546 *                                                                   * PGM05280
                                547 *    Authorized Caller: Console Address Space.                      * PGM05290
                                548 *                                                                   * PGM05300
                                549 *    Function: To perform a "dummy" read operation to the console to  * PGM05310
                                550 *              purge pending input data.  The data must be purged     * PGM05320
                                551 *              because Console Address Space has no one to give it to.* PGM05330
                                552 *              The READ operation only accepts 1 byte of data and is  * PGM05340
                                553 *              chained to a NOP which forces concurrent CE, DE ending * PGM05350
                                554 *              status in the CSW.                                     * PGM05360
                                555 *                                                                     * PGM05370
                                556 *              There is no associated I/O interrupt from the read     * PGM05380
                                557 *              channel program; we spin in a Test I/O loop until the  * PGM05390
                                558 *              channel stores a CSW which clears the pending inter-   * PGM05400
                                559 *              ruption condition.                                     * PGM05410
                                560 *                                                                     * PGM05420
                                561 *-------------------------------------------------------------------* PGM05430

00024E                          563 MPURGECN DS    0H            Purge Console Input Data:                PGM05450

                                565          ISIT  CONSAS        Is caller = Console Address Space?       PGM05470
00024E 5970 C3C0      003C0     566+         C     R7,CONSAS     Is is this guy?                          IS100030
000252 4770 C2CE      002CE     567          BNE   8ADPURGE      No  - Unauthorized invocation. Shut down. PGM05480
000256 9202 C388      00388     568          MVI   CONSCCW,X'02' Move READ command code into CCW.         PGM05490
00025A 9260 C38C      0038C     569          MVI   CONSCCW+4,X'60' Command chain to NOP CCW to force CE,DE. PGM05500
00025E 9200 C38E      0038E     570          MVI   CONSCCW+6,X'00' Clear previous count setting           PGM05510
000262 9201 C38F      0038F     571          MVI   CONSCCW+7,X'01' New count: Accept one byte.            PGM05520

000266 9C01 0009      00009     573          SIOF  9             Start the read operation.                PGM05540

00026A                          575 TIOLOOP  DS    0H            Here to wait until I/O operation drains: PGM05560

00026A 9000 0009      00009     577          TIO   9                                                      PGM05580
00026E 4710 C2D6      00206     578          BC    1,NOCONS      CC3: We lost the console!                PGM05590
000272 47C0 C27A      0027A     579          BC    8+4,PURGDONE  CC0, CC1: Available (?) or CSW stored.   PGM05600
000276 47F0 C26A      0026A     580          B     TIOLOOP       CC2: Subchannel working.  Busy/wait loop. PGM05610

00027A                          582 PURGDONE DS    0H            Here when interruption condition cleared: PGM05630

00027A 9509 0088      00088     584          CLI   IOCODE+1,X'09' Interrupt from console?                 PGM05650
00027E 4770 C26A      0026A     585          BNE   TIOLOOP       No  - Ignore it and wait for console.    PGM05660
000282 950C 0044      00044     586          CLI   CSW+4,X'0C'   Yes - Channel End, Device End?           PGM05670
000286 4770 C2DA      002DA     587          BNE   8ADSTAT       No  - Abend due to unexpected status...   PGM05680
00028A 47F0 C29A      0029A     588          B     RTI           Yes - Return to caller.                  PGM05690
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86

                                    590 *-------------------------------------------------------------------* PGM05710
                                    591 * Y. Service: SHUTSYS.                                              * PGM05720
                                    592 *                                                                   * PGM05730
                                    593 *    Authorized Caller: Shutdown Address Space.                     * PGM05740
                                    594 *                                                                   * PGM05750
                                    595 *    Function: To shut the system down.  Simply load a disabled wait * PGM05760
                                    596 *              PSW to terminate the running system.                 * PGM05770
                                    597 *-------------------------------------------------------------------* PGM05780

00028E                              599 MSHUTSYS DS    0H            Purge Console Input Data:               PGM05800

                                    601          ISIT  SHUTAS        Is caller = Shutdown Address Space?     PGM05820
00028E 5970 C3C4          003C4     602+         C     R7,SHUTAS     Is is this guy?                         ISI00030
000292 4770 C202          002D2     603          8NE   8ADSHUT       No  - Abnormally shut the system down.  PGM05830
000296 8200 C360          00360     604          LPSW  SHUTDOWN     Yes - Shut down normally.               PGM05840
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM 0201 20.42 12/04/86
                                    606 *-------------------------------------------------------------* PGM05860
                                    607 * Z. Return to caller.                                        * PGM05870
                                    608 *-------------------------------------------------------------* PGM05880
00029A                              610 RTI      DS    0H           Emulate Return From Interrupt instruction: PGM05900
00029A 980F 0400      0D400         612          LM    R0,R15,PSAGREGS Restore caller's registers          PGM05920
00029E 8200 0028      00028         613          LPSW  PGMOLD       Re-activate calling address space.       PGM05930
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 20.42 12/04/86

                                    615 *-------------------------------------------------------------------* PGM05950
                                    616 * System Termination Instructions:                                  * PGM05960
                                    617 *-------------------------------------------------------------------* PGM05970

0002A2                              619 PGMCHECK OS    0H                                                     PGM05990
0002A2 8200 C2E0       002E0        620          LPSW  PROGCHK       Shutdown due to program check.          PGM06000
0002A6                              621 BIGCODE  DS    0H                                                     PGM06010
0002A6 8200 C2E8       002E8        622          LPSW  TOOBIG        Shutdown due to undefined service.      PGM06020
0002AA                              623 BADAA    DS    0H                                                     PGM06030
0002AA 8200 C2F0       002F0        624          LPSW  AABAO         Shutdown due to illegal ACTAUTH.        PGM06040
0002AE                              625 BAOSDP   DS    0H                                                     PGM06050
0002AE 8200 C2F8       002F8        626          LPSW  SDPBAD        Shutdown due to illegal Set Dispatch Parms PGM06060
0002B2                              627 BADDSPCH DS    0H                                                     PGM06070
0002B2 8200 C300       00300        628          LPSW  DSPCHBAD      Shutdown due to illegal Dispatch.       PGM06080
0002B6                              629 BADABEND DS    0H                                                     PGM06090
0002B6 8200 C308       00308        630          LPSW  ABENOBAO      Shutdown due to illegal ABEND.          PGM06100
0002BA                              631 BADBINO  DS    0H                                                     PGM06110
0002BA 8200 C318       00318        632          LPSW  BINDBAD       Shutdown due to illegal BINO.           PGM06120
0002BE                              633 BADDTACH DS    0H                                                     PGM06130
0002BE 8200 C320       00320        634          LPSW  DTACHBAD      Shutdown due to illegal DETACH.         PGM06140
0002C2                              635 BADIDLE  DS    0H                                                     PGM06150
0002C2 8200 C328       00328        636          LPSW  IDLEBAD       Shutdown due to illegal IDLE.           PGM06160
0002C6                              637 BADSCK   DS    0H                                                     PGM06170
0002C6 8200 C330       00330        638          LPSW  SCKBAD        Shutdown due to illegal SCK.            PGM06180
0002CA                              639 BADCONIO DS    0H                                                     PGM06190
0002CA 8200 C338       00338        640          LPSW  CONIOBAD      Shutdown due to illegal CONSIO.         PGM06200
0002CE                              641 BAOPURGE DS    0H                                                     PGM06210
0002CE 8200 C340       00340        642          LPSW  PURGEBAD      Shutdown due to illegal PURGECON.       PGM06220
0002D2                              643 BADSHUT  DS    0H                                                     PGM06230
0002D2 8200 C348       00348        644          LPSW  SHUTBAD       Shutdown due to illegal SHUTSYS.        PGM06240
0002D6                              645 NOCONS   DS    0H                                                     PGM06250
0002D6 8200 C350       00350        646          LPSW  CONSGONE      Shutdown due to CC3 on console SIOF.    PGM06260
00020A                              647 BAOSTAT  OS    0H                                                     PGM06270
0002DA 8200 C358       00358        648          LPSW  STATBAD       Shutdown due to bad console Unit Status PGM06280

0002E0                              650          DS    0D                                                     PGM06300
0002E0 000A000000000000             651 PROGCHK  DC    XL4'000A0000',F'0' Various                            PGM06310
0002E8 000A000000000001             652 TOOBIG   DC    XL4'000A0000',F'1'   Shutdown                         PGM06320
0002F0 000A000000000002             653 AABAD    DC    XL4'000A0000',F'2'     PSWs...                       PGM06330
0002F8 000A000000000003             654 SDPBAO   DC    XL4'000A0000',F'3'                                   PGM06340
000300 000A000000000004             655 OSPCHBAO DC    XL4'000A0000',F'4'                                   PGM06350
000308 000A000000000005             656 ABENDBAD DC    XL4'000A0000',F'5'                                   PGM06360
000310 000A000000000006             657 ABENDPSW DC    XL4'000A0000',F'6'                                   PGM06370
000318 000A000000000007             658 BINDBAD  DC    XL4'000A0000',F'7'                                   PGM06380
000320 000A000000000008             659 DTACHBAD DC    XL4'000A0000',F'8'                                   PGM06390
000328 000A000000000009             660 IDLEBAD  DC    XL4'000A0000',F'9'                                   PGM06400
000330 000A00000000000A             661 SCKBAD   DC    XL4'000A0000',F'10'                                  PGM06410
000338 000A00000000000B             662 CONIOBAD DC    XL4'000A0000',F'11'                                  PGM06420
000340 000A00000000000C             663 PURGEBAD DC    XL4'000A0000',F'12'                                  PGM06430
000348 000A00000000000D             664 SHUTBAD  DC    XL4'000A0000',F'13'                                  PGM06440
000350 000A00000000000E             665 CONSGONE DC    XL4'000A0000',F'14'                                  PGM06450
000358 000A00000000000F             666 STATBAD  DC    XL4'000A0000',F'15'                                  PGM06460
000360 000A000000777777             667 SHUTDOWN DC    XL4'000A0000',XL4'00777777'                          PGM06470
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.42 12/04/86

                                    669 *-------------------------------------------------------------------* PGM06490
                                    670 * Module constants and variables:                                  *: PGM06500
                                    671 *-------------------------------------------------------------------* PGM06510

000368                              673          DS    0D             Align constant area.                  PGM06530
                                    674          ENTRY XMITO                                                PGM06540
000368 FFFFFFFFFFFFFFFF             675 XMITO    DC    16H'-1'        Page table for XMIT segment 0.         PGM06550

                                    677          ENTRY CONSCCW                                              PGM06570
000388 0100000060000000             678 CONSCCW  CCW   1,CONSPP,X'60',*-*  Command code filled in on the fly. PGM06580
000390 0300000020000001             679 NOPCCW   CCW   3,*-*,X'20',1       Nop forces concurrent CE,DE status. PGM06590

000398 7FFFFFFFFFFFFFFF             681 BIGTIMER DC    X'7FFFFFFF',F'-1' Maximum GPU timer value              PGM06610
0003A0 030A000000000000             682 WAIT     DC    X'030A0000',F'0' Enabled wait PSW.                   PGM06620
0003A8                              683 CALLERIO DS    F              Temporary storage for Control Register 1 PGM06630
0003AC 00000000                     684 DSPCHER  DC    A(DSPST)       Dispatcher's STD                       PGM06640
0003B0 00000000                     685 AUTHRTN  DC    A(AUTHST)      Authentication Routine's STD           PGM06650
0003B4 00000000                     686 XMITTER  DC    A(XMITST)      Transmitter Module's STD               PGM06660
0003B8 00000000                     687 SLEEPAS  DC    A(SLEEPST)     Sleep Address Space STD                PGM06670
0003BC 00000000                     688 SLAS     DC    A(SLASST)      Sleep List Address Space STD           PGM06680
0003C0 00000000                     689 CONSAS   DC    A(CONSST)      Console Address Space STD              PGM06690
0003C4 00000000                     690 SHUTAS   DC    A(SHUTST)      Shutdown Address Space STD             PGM06700
0003C8 50                           691 AUTHKEY  DC    XL1'50'        AUTH PSW key                           PGM06710
0003C9 010000                       692 ATHVADDR DC    XL3'010000'    AUTH main entry point virtual address  PGM06720
                                    693          PRINT OFF                                                  PGM06730
```

POS.ID	REL.ID	FLAGS	ADDRESS
0009	0001	0C	000380
0009	0002	0C	0003AC
0009	0003	0C	000384
0009	0004	0C	0003B8
0009	0005	0C	00038C
0009	0006	0C	0003C0
0009	0007	08	000389
0009	0008	0C	0003C4

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AABAD	00004	000002F0	00653	00624
ABENDBAD	00004	00000308	00656	00630
ABENDPSW	00004	00000310	00657	00378
AROUND	00D02	0000002C	00088	00079
ATHVADDR	00003	000003C9	00692	00286
AUTHKEY	00001	000003C8	00691	00288
AUTHRTN	00004	000003B0	00685	00285 00310 00345 00375
AUTHST	00001	00000000	00058	00685
BADAA	D0002	000002AA	00623	00283
BADABEND	00002	000002B6	00629	00376
BADBIND	00002	000002BA	00631	00402
BADCONIO	00002	000002CA	00639	00523
BADDSPCH	00002	000002B2	00627	00346
BADDTACH	00002	000002BE	00633	00435
BADIDLE	0D002	000002C2	00635	00455
BADPURGE	00002	000002CE	00641	00567
BADSCK	00002	000002C6	00637	00477
BADSDP	00002	000002AE	00625	00311
BADSHUT	00002	000002D2	00643	00603
BADSTAT	00D02	0D0002DA	00647	00587
BIGCODE	00D02	0D0D02A6	00621	00237
BIGTIMER	00004	00000398	00681	00457
BINDBAD	00004	00000318	00658	00632
CALLERID	00004	000003A8	00683	00244 00245
CHKIDLE	00002	000D00A0	00204	00099
CMDCHAIN	00002	0000023A	00532	00527
COMMON	00002	000D01CC	00410	00438
CONIOBAD	00004	00000338	00662	00640
CONSAS	D0004	000003C0	00689	00522 00566
CONSCCW	00008	00000388	00678	00524 00525 00526 00528 00534 00568 00569 00570 00571 00677
CONSGONE	00D04	00000350	00665	00646
CONSPP	00001	00000000	00058	00678
CONSST	00001	000000D0	00058	00689
CPUTIMER	00008	00000460	00885	00069 00116
CSW	00008	00000040	00863	00586
DATON	00001	000D0004	00762	00130
DOIO	00002	0000023E	00537	00530 00542
DSPCHBAD	00004	00000300	00655	00628
DSPCHER	00004	000003AC	00684	00307 00342 00454
DSPDSPCH	00002	00000190	00353	00343
DSPSDP	00002	0000015A	00319	00308
DSPST	00001	00000000	00058	00684
DTACHBAD	00004	00000320	00659	00634
ENDTABLE	00001	00000104	00265	00267
EXTOLD	00008	00000018	00858	00175
FLIHSAVE	00002	0000006A	00161	00131
F0	00001	00000000	00728	00071
F2	00001	00000002	00729	00072
F4	00001	00000004	00730	00073
F6	00001	000D0006	00731	00074
IDLEBAD	00D04	00000328	00660	00636
INTCREG1	00004	00000494	00894	00075
INTSTATE	00004	00000490	00893	00090
IOCODE	00002	000000BA	00880	00584
IOOLD	00008	00000038	00862	00178

SYMBOL	LEN	VALUE	DEFN	REFERENCES
MABEND	00002	000001A4	00372	00255
MACTAUTH	00002	00000108	00279	00252
MAXCODE	00004	00000104	00267	00236
MBIND	00002	000001B0	00398	00256
MCEVENT	00001	00000040	00736	00227
MCONSIO	00002	00000216	00519	00261
MCTABLE	00002	000000D4	00250	00248 00267
MDETACH	00002	000001DC	00431	00257
MDSPATCH	00002	0D000170	00339	00254
MIDLE	00002	000001EE	00451	00259
MONCODE	00002	0000009C	00878	00235
MPURGECN	00002	0000024E	00563	00262
MSAVEREG	00002	00000212	00496	00260
MSCK	00002	00000202	00473	00258
MSDP	00002	00000130	00305	00253
MSHUTSYS	00002	0000028E	00599	00263
NOCONS	00D02	000002D6	00645	00541 00578
PGMCHECK	00002	000002A2	00619	00228
PGMCODE	00002	0000008E	00876	00141 00227
PGMFLIH	00001	00000000	00060	00078
PGMNEW	00008	00000068	00868	00077
PGMOLD	00008	00000028	00860	00130 00140 00164 00191 00216 00281 00282 00613
PROGCHK	00004	000002E0	00651	00620
PSA	00001	00000000	00856	00066 00857 00865 00871 00873 00875 00877 00879 00882
PSAFREGS	00008	00000440	00884	00071 00072 00073 00074 00115
PSAGREGS	00004	00000400	00883	00070 00114 00313 00315 00316 00321 00322 00323 00349 00356 00403 00478 00524 00525 00612
PSWADDR	00002	0000007E	00170	00168
PURGDONE	00002	0000027A	00582	00579
PURGEBAD	00004	00000340	00663	00642
RTI	00002	0000029A	00610	00317 00324 00415 00479 00498 00540 00588
RUNIDLE	00001	00000020	00758	00206 00456
RUNSYS	00001	00000040	00757	00100 00215 00289 00350
RUNSYSTM	00002	000000B0	00225	00207
RUNUSER	00001	00000080	00756	00098 00358
R0	00001	00000000	00708	00066 00070 00143 00349 00356 00403 00524 00612
R1	00001	000D0001	00709	00235 00236 00247 00248 00525
R12	00001	0000000C	00720	00077 00078
R15	00001	0000000F	00723	00070 00349 00356 00612
R2	00001	00D00002	00710	00313 00321 00478
R3	00001	00000003	00711	00315 00316 00322
R5	00001	00000005	00713	00163 00163 00164 00165 00166 00167 00168 00172 00175 00178 00189 00323 00403 00405 00405 00406 00406 00412 00436 00437
R6	00001	00000006	00714	00403 00407 00407 00408 00408 00413 00437
R7	00001	00000007	00715	00245 00307 00310 00342 00345 00375 00401 00434 00454 00476 00522 00566 00602
SAVEPSW	00002	00000096	00187	00173 00176 00179
SAVFREGS	00008	000004E0	00897	00115
SAVGREGS	00004	000004A0	00896	00114
SAVPSW	00008	00000498	00895	00140 00141 00189
SAVTIMER	00008	00000500	00898	00116
SCKBAD	00004	00000330	00661	00638
SDPBAD	00004	000002F8	00654	00626
SHUTAS	00004	000003C4	00690	00602
SHUTBAD	00004	00000348	00664	00644
SHUTDOWN	00004	00000360	00667	00604
SHUTST	00001	00000000	00058	00690

SYMBOL	LEN	VALUE	DEFN	REFERENCES
SLASST	00001	00000000	00058	00688
SLEEPAS	00004	00000388	00687	00476
SLEEPST	00001	00000000	00058	00687
STATBAD	00004	00000358	00666	00648
SVCOLD	00008	00000020	00859	00172
SYSPSW	00008	00000470	00888	00286 00287 00288 00290 00314 00315 00316 00351
SYSSTO	00004	0000046C	00887	00313 00348
SYSTATE	00004	00000468	00886	00090 00098 00100 00206 00215 00289 00350 00358 00456
TIOLOOP	00002	0000026A	00575	00580 00585
TOOBIG	00004	000002E8	00652	00622
USRPSW	00008	00000480	00891	00322 00359
USRSTO	00004	0000047C	00890	00321 00355
USRTIMER	00008	00000488	00892	00323 00357
WAIT	00004	000003A0	00682	00458
XMITST	00001	00000000	00058	00686
XMITTER	00004	000003B4	00686	00401 00434
XMITO	00002	00000368	00675	00412 00413 00674

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 20.52 12/04/86

                                       3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SVC00040
                                       4 *+----------------------------------------------------------------------+* SVC00050
                                       5 *|                                                                      |* SVC00060
                                       6 *|Module:   SVCFLIH                                                     |* SVC00070
                                       7 *|                                                                      |* SVC00080
                                       8 *|          Supervisor Call First-Level Interrupt Handler.              |* SVC00090
                                       9 *|                                                                      |* SVC00100
                                      10 *|Function: Perform early stages of processing necessary when user or|* SVC00110
                                      11 *|          system code executes a SVC instruction.  This processing |* SVC00120
                                      12 *|          entails saving the state of the machine at the point of  |* SVC00130
                                      13 *|          the interruption and results in the activation of the    |* SVC00140
                                      14 *|          SVC second-level interrupt handler, otherwise known as   |* SVC00150
                                      15 *|          the Authentication Routine.                                |* SVC00160
                                      16 *|                                                                      |* SVC00170
                                      17 *|Caller:   Not called; rather, activated by a hardware PSW swap      |* SVC00180
                                      18 *|          which saves the current PSW in an assigned storage        |* SVC00190
                                      19 *|          location for SVC interrupt class and reloads the current  |* SVC00200
                                      20 *|          PSW from the assigned storage location containing the     |* SVC00210
                                      21 *|          new PSW for SVC interrupt class.                           |* SVC00220
                                      22 *|                                                                      |* SVC00230
                                      23 *|Mode:     Problem state, OAT off, Key 1.                              |* SVC00240
                                      24 *|                                                                      |* SVC00250
                                      25 *+----------------------------------------------------------------------+* SVC00260
                                      26 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SVC00270

                                      28          PUNCH 'SLC  001000' Make module start at page 1.                  SVC00290

                                      30          EXTRN AUTHPP                                                       SVC00310
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.52 12/04/86
000000                               32 SVCFLIH  CSECT               SVC First-Level Interrupt Handler          SVC00330

                                     34 *-------------------------------------------------------------------* SVC00350
                                     35 * A. Save registers in low core and establish addressibility:      * SVC00360
                                     36 *-------------------------------------------------------------------* SVC00370

                              00000  38          USING PSA,R0        Make low-core fields addressible           SVC00390

000000 AF00 0008       00008         40          MC    SAVEREGS,0    Save volative state information in lowcore SVC00410

000004 58C0 0064       00064         42          L     R12,SVCNEW+4  Get the address of this module in R12      SVC00430
                              00000  43          USING SVCFLIH,R12   Establish R12 as the overall module base   SVC00440
000008 47F0 C014       00014         44          B     AROUND        Move past eyecatcher                       SVC00450

00000C E2E5C3C6D3C9C840              46          DC    CL8'SVCFLIH'  Handy when poking around in storage/dumps  SVC00470

                                     48 *-------------------------------------------------------------------* SVC00490
                                     49 * 8. Clear the Authentication Routine's parameter page and insert the * SVC00500
                                     50 *    system state variable @ time of interrupt into it.            * SVC00510
                                     51 *                                                                  * SVC00520
                                     52 *    *Note: PGMFLIH saved the state of the system at the point of  * SVC00530
                                     53 *           interruption in INTSTATE, then it set SYSTATE to indicate * SVC00540
                                     54 *           the system is running itself.                          * SVC00550
                                     55 *-------------------------------------------------------------------* SVC00560

000014                               57 AROUND   DS    0H            Now the real work begins:                  SVC00580

000014 9825 C060       00060         59          LM    R2,R5,CLRAUTH Load MVCL parms to clear AUTH's parm page  SVC00600
000018 0E24                          60          MVCL  R2,R4         Clear out AUTH's parameter page.           SVC00610

00001A 5820 C060       00060         62          L     R2,CLRAUTH    Reload A(AUTH's parm page) destroyed by mv SVC00630
                              00000  63          USING APP,R2        R2 will be the base reg for AUTH's parm pg SVC00640

00001E D200 2000 0490 00000 00490    65          MVC   APPSTATE,INTSTATE Copy state @ interrupt to parm page.  SVC00660

                                     67 *-------------------------------------------------------------------* SVC00680
                                     68 * C. Normally we would place the caller's STO into the AUTH parm    * SVC00690
                                     69 *    page.  We will do this for the user shortly.  Here, however, we * SVC00700
                                     70 *    must be wise to the fact that the system issued a SVC, and not  * SVC00710
                                     71 *    all parts of the system run in an address space and consequently * SVC00720
                                     72 *    do not have a STO.  The only system modules that do not have an * SVC00730
                                     73 *    address space are the first-level interrupt handlers.  A FLIH   * SVC00740
                                     74 *    STO will be represented by a zero.  The FLIHs can be detected by * SVC00750
                                     75 *    examining the SVC old PSW for DAT off, as they do not run in    * SVC00760
                                     76 *    translate mode:                                               * SVC00770
                                     77 *-------------------------------------------------------------------* SVC00780

000024                               79 RUNSYSTM DS    0H            Here when system is running itself:        SVC00800

000024 9104 0020       00020         81          TM    SVCOLD,DATON  Was FLIH running? (IF dat on THEN No)     SVC00820
000028 4710 C03A       0003A         82          BO    SAVESTO       No  - Save system/user STO.                SVC00830
00002C D200 2004 0021 00004 00021    83          MVC   APPSTO(1),SVCOLD+1 Yes - Extract PSW key & mode bits.  SVC00840
000032 94F0 2004       00004         84          NI    APPSTO,X'F0'  Turn off mode bits; leave caller's key.  SVC00850
000036 47F0 C040       00040         85          B     SAVECODE      Continue state-saving venture.             SVC00860
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.52 12/04/86

                                     87 *-------------------------------------------------------------------* SVC00880
                                     88 * D. Save user or system STO in AUTH parm page.  PGMFLIH saved the   * SVC00890
                                     89 *    contents of control register 1 in low core as part of its state  * SVC00900
                                     90 *    saving process.                                                  * SVC00910
                                     91 *-------------------------------------------------------------------* SVC00920

00003A                               93 SAVESTO  DS    0H            Here to save system/user STO:           SVC00940

00003A D203 2004 0494 00004 00494    95          MVC   APPSTO,INTCREG1 Put caller's STO in AUTH parm page.     SVC00960

                                     97 *-------------------------------------------------------------------* SVC00980
                                     98 * E. Move the SVC interrupt code, PSW key, resume virtual address and * SVC00990
                                     99 *    saved general register image to AUTH's parm page:               * SVC01000
                                    100 *-------------------------------------------------------------------* SVC01010

000040                              102 SAVECODE DS    0H            Here to copy remaining parms to AUTH ppage SVC01030

000040 D201 2002 008A 00002 0008A   104          MVC   APPCODE,SVCCODE Move SVC interrupt code to AUTH ppage   SVC01050
000046 D200 2008 0021 00008 00021   105          MVC   APPVADDR(1),SVCOLD+1 Save caller's key.               SVC01060
00004C 94F0 2008      00008         106          NI    APPVADDR,X'F0' Turn off unused bits.                  SVC01070
000050 D202 2009 0025 00009 00025   107          MVC   APPVADDR+1(3),SVCOLD+5 Copy resume vaddr to parm page SVC01080
000056 D23F 200C 0400 0000C 00400   108          MVC   APPREGS(16*4),PSAGREGS  Copy interrupt regs to parm page SVC01090

                                    110 *-------------------------------------------------------------------* SVC01110
                                    111 * F. Transfer control to the SVC second-level interrupt handler,     * SVC01120
                                    112 *    otherwise known as the Authentication Routine:                  * SVC01130
                                    113 *-------------------------------------------------------------------* SVC01140

00005C AF00 0000      00000         115          MC    ACTAUTH,0     Request privileged ACTIVATE AUTHENTICATION SVC01160
```

```
000060                             117          DS    0D      Following 4 words used by LOAD MULTIPLE:  SVC01180
000060 00000000                    118 CLRAUTH  DC    A(AUTHPP) Real address of AUTH's parameter page   SVC01190
000064 00001000                    119          DC    F'4096' It's a full page                          SVC01200
000068 0000000000000000            120          DC    2F'0'   Unused GREG & pad,length for MVCL clear SVC01210

                                   122          PRINT OFF                                               SVC01230
```

POS.ID	REL.ID	FLAGS	ADDRESS
0002	0001	0C	000060

ASM 0201 20.52 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ACTAUTH	00001	00000000	00169	00115
APP	00001	00000000	00359	00063
APPCODE	00002	00000002	00362	00104
APPREGS	00004	0000000C	00365	00108
APPSTATE	00001	00000000	00360	00065
APPSTO	00004	00000004	00363	00083 00084 00095
APPVADDR	00004	00000008	00364	00105 00106 00107
AROUND	00002	00000014	00057	00044
AUTHPP	00001	00000000	00030	00118
CLRAUTH	00004	00000060	00118	00059 00062
DATON	00001	00000004	00191	00081
INTCREG1	00004	00000494	00323	00095
INTSTATE	00004	00000490	00322	00065
PSA	00001	00000000	00285	00038 00286 00294 00300 00302 00304 00306 00308 00311
PSAGREGS	00004	00000400	00312	00108
R0	00001	00000000	00137	00038
R12	00001	0000000C	00149	00042 00043
R2	00001	00000002	00139	00059 00060 00062 00063
R4	00001	00000004	00141	00060
R5	00001	00000005	00142	00059
SAVECODE	00002	00000040	00102	00085
SAVEREGS	00001	00000008	00177	00040
SAVESTO	00002	0000003A	00093	00082
SVCCODE	00002	0000008A	00303	00104
SVCFLIH	00001	00000000	00032	00043
SVCNEW	00008	00000060	00296	00042
SVCOLD	00008	00000020	00288	00081 00083 00105 00107

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM 0201 20.39 12/04/86

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* EXT00040
                                     4 *+--------------------------------------------------------------------+* EXT00050
                                     5 *|                                                                    |* EXT00060
                                     6 *|Module:   EXTFLIH                                                   |* EXT00070
                                     7 *|                                                                    |* EXT00080
                                     8 *|          External Interrupt First-Level Interrupt Handler.         |* EXT00090
                                     9 *|                                                                    |* EXT00100
                                    10 *|Function: Perform early stages of processing necessary when the    |* EXT00110
                                    11 *|          system is either idle or running a user and an external  |* EXT00120
                                    12 *|          interrupt occurs.  It basically saves the volatile        |* EXT00130
                                    13 *|          machine state and calls the external interrupt second    |* EXT00140
                                    14 *|          level interrupt handler, passing it the identity of the  |* EXT00150
                                    15 *|          interrupt source.                                         |* EXT00160
                                    16 *|                                                                    |* EXT00170
                                    17 *|Caller:   Not called; rather, activated by a hardware PSW swap     |* EXT00180
                                    18 *|          which saves the current PSW in an assigned storage       |* EXT00190
                                    19 *|          location for external interrupt class and reloads the    |* EXT00200
                                    20 *|          current PSW from the assigned storage location contain-  |* EXT00210
                                    21 *|          ing the new PSW for external interrupt class.            |* EXT00220
                                    22 *|                                                                    |* EXT00230
                                    23 *|Mode:     Problem state, OAT off, Key 2.                            |* EXT00240
                                    24 *|                                                                    |* EXT00250
                                    25 *+--------------------------------------------------------------------+* EXT00260
                                    26 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* EXT00270

                                    28          PUNCH 'SPB'       Make module start on a page boundry.          EXT00290

                                    30          EXTRN EXTSPP                                                    EXT00310
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.39 12/04/86
000000                              32 EXTFLIH  CSECT             External First-Level Interrupt Handler     EXT00330

                                    34 *-------------------------------------------------------------------* EXT00350
                                    35 * A. Save registers in low core and establish addressibility:      * EXT00360
                                    36 *-------------------------------------------------------------------* EXT00370

                           00000    38          USING PSA,R0        Make low-core fields addressible           EXT00390

000000 AF00 0008       00008        40          MC    SAVEREGS,0    Save volative state information in lowcore EXT00410

000004 58C0 005C       0005C        42          L     R12,EXTNEW+4  Get the address of this module in R12      EXT00430
                           00000    43          USING EXTFLIH,R12   Establish R12 as the overall module base  EXT00440
000008 47F0 C014       00014        44          B     AROUND        Move past eyecatcher                       EXT00450

00000C C5E7E3C6D3C9C840             46          DC    CL8'EXTFLIH'  Handy when poking around in storage/dumps  EXT00470

                                    48 *-------------------------------------------------------------------* EXT00490
                                    49 * B. Clear the External Second-Level Interrupt Handler's parameter  * EXT00500
                                    50 *    page:                                                         * EXT00510
                                    51 *-------------------------------------------------------------------* EXT00520

000014                              53 AROUND0  DS    0H            Now the real work begins:                  EXT00540

000014 9825 C030       00030        55          LM    R2,R5,CLREXT  Load MVCL parms to clear SLIH parm page    EXT00560
000018 0E24                         56          MVCL  R2,R4         Clear out SLIH's parameter page.           EXT00570

                                    58 *-------------------------------------------------------------------* EXT00590
                                    59 * C. Place identification of interrupting source in SLIH parm page:  * EXT00600
                                    60 *-------------------------------------------------------------------* EXT00610

00001A 5820 C030       00030        62          L     R2,CLREXT     Get pointer to External SLIH parm page     EXT00630
00001E D201 2000 0086 00000 00086   63          MVC   0(2,R2),EXTCODE Move in the interruption code             EXT00640

                                    65 *-------------------------------------------------------------------* EXT00660
                                    66 * D. Transfer control to the External Second-Level Interrupt Handler: * EXT00670
                                    67 *-------------------------------------------------------------------* EXT00680

                                    69          EXIT  EXTS          Goto External SLIH.                        EXT00700
000024 5800 C040       00040        70+         L     R0,=CL4'EXTS'       Load target address space identifier EXI00040
000028 0A02                         71+         SVC   2                   Go to that address space.            EXI00050
```

```
000030                                  73           DS    0D          Following 4 words used by LOAD MULTIPLE:  EXT00720
000030 00000000                         74 CLREXT    DC    A(EXTSPP)   Real address of SLIH's parameter page     EXT00730
000034 00001000                         75           DC    F'4096'     It's a full page                          EXT00740
000038 0000000000000000                 76           DC    2F'0'       Unused GREG & pad,length for MVCL clear   EXT00750

                                        78           PRINT OFF                                                     EXT00770
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000014	00053	00044
CLREXT	00004	00000030	00074	00055 00062
EXTCODE	00002	00000086	00257	00063
EXTFLIH	00001	00000000	00032	00043
EXTNEW	00008	00000058	00251	00042
EXTSPP	00001	00000000	00030	00074
PSA	00001	00000000	00241	00038 00242 00250 00256 00258 00260 00262 00264 00267
R0	00001	00000000	00093	00038 00070
R12	00001	0000000C	00105	00042 00043
R2	00001	00000002	00095	00055 00056 00062 00063
R4	00001	00000004	00097	00056
R5	00001	00000005	00098	00055
SAVEREGS	00001	00000008	00133	00040

```
                                3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* IOF00040
                                4 *+-----------------------------------------------------------------------------+* IOF00050
                                5 *|                                                                             |* IOF00060
                                6 *|Module:    IOFLIH                                                            |* IOF00070
                                7 *|                                                                             |* IOF00080
                                8 *|           I/O First-Level Interrupt Handler.                               |* IOF00090
                                9 *|                                                                             |* IOF00100
                               10 *|Function:  Perform early stages of processing necessary when the             |* IOF00110
                               11 *|           system is either idle or running a user and an I/O                |* IOF00120
                               12 *|           interrupt occurs.  It basically saves the volatile               |* IOF00130
                               13 *|           machine state and calls the console I/O second-level             |* IOF00140
                               14 *|           interrupt handler, passing it the identity of the               |* IOF00150
                               15 *|           interruption source.                                             |* IOF00160
                               16 *|                                                                             |* IOF00170
                               17 *|Caller:    Not called; rather, activated by a hardware PSW swap             |* IOF00180
                               18 *|           which saves the current PSW in an assigned storage              |* IOF00190
                               19 *|           location for I/O interrupt class and reloads the current        |* IOF00200
                               20 *|           PSW frm the assigned storage location containing the new        |* IOF00210
                               21 *|           PSW for the I/O interrupt class.                                |* IOF00220
                               22 *|                                                                             |* IOF00230
                               23 *|Mode:      Problem state, DAT off, Key 3.                                    |* IOF00240
                               24 *|                                                                             |* IOF00250
                               25 *+-----------------------------------------------------------------------------+* IOF00260
                               26 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* IOF00270

                               28          PUNCH 'SPB'        Make module start on a page boundry.        IOF00290

                               30          EXTRN CONSPP                                                   IOF00310
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.37 12/04/86
000000                              32 IOFLIH   CSECT              I/O First-Level Interrupt Handler          IOF00330
                                    34 *-------------------------------------------------------------------* IOF00350
                                    35 * A. Save registers in low core and establish addressibility:      * IOF00360
                                    36 *-------------------------------------------------------------------* IOF00370
                              000D0 38           USING PSA,R0       Make low-core fields addressible           IOF00390
000000 AF0D 0008      00008         40           MC    SAVEREGS,0   Save volative state information in lowcore IOF00410
000004 5BC0 007C      0007C         42           L     R12,IONEW+4  Get the address of this module in R12      IOF00430
                            00000   43           USING IOFLIH,R12   Establish R12 as the overall module base  IOF00440
00D0D8 47F0 C014      0D014         44           B     AROUND       Move past eyecatcher                       IOF00450
00000C C9D6C6D3C9C84040             46           DC    CL8'IOFLIH'  Handy when poking around in storage/dumps IOF00470
                                    48 *-------------------------------------------------------------------* IOF00490
                                    49 * 8. Move the Channel Status Word and the address of the interrupting * IOF00500
                                    50 *    device into the Console Address Space's parameter page.       * IOF00510
                                    51 *-------------------------------------------------------------------* IOF00520
000014                              53 AROUND   DS    0H           Now the real work begins:                  IOF00540
000014 5820 C030      00D30         55           L     R2,CLRCONS   Get pointer to Console A. S. parm page    IOF00560
000018 D201 2FF4 00BA 00FF4 0008A   56           MVC   4084(2,R2),IOCODE Move in the device address.         IOF00570
00001E D207 2FF8 0040 00FF8 D004D   57           MVC   4088(8,R2),CSW    Move in the CSW.                    IOF00580
                                    59 *-------------------------------------------------------------------* IOF00600
                                    60 * C. Transfer control to the Console Address Space, also known as the * IOF00610
                                    61 *    I/O second-level interrupt handler.                           * IOF00620
                                    62 *-------------------------------------------------------------------* IOF00630
                                    64           EXIT  CNSI         Goto Console I/O interrupt handler.        IOF00650
000024 5800 C040      00040         65+          L     R0,=CL4'CNSI'      Load target address space identifier EX100040
000028 0A02                         66+          SVC   2                  Go to that address space.           EX100050
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.37 12/04/86
000030                                 68           DS    0D          Following 4 words used by LOAD MULTIPLE:   IOF00670
000030 00000000                        69 CLRCONS   DC    A(CONSPP)   Real address of SLIH's parameter page      IOF00680
000034 00001000                        70           DC    F'4096'     It's a full page                           IOF00690
000038 0000000000000000                71           DC    2F'0'       Unused GREG & pad,length for MVCL clear    IOF00700

                                       73           PRINT OFF                                                      IOF00720
```

POS.ID	REL.ID	FLAGS	ADDRESS
0002	0001	0C	000030

ASM 0201 20.37 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000014	00053	00044
CLRCONS	00004	00000030	00069	00055
CONSPP	00001	00000000	00030	00069
CSW	00008	00000040	00243	00057
IOCODE	00002	000000BA	00260	00056
IOFLIH	00001	00000000	00032	00043
IONEW	00008	00000078	00250	00042
PSA	00001	00000000	00236	00038 00237 00245 00251 00253 00255 00257 00259 00262
R0	00001	00000000	00088	00038 00065
R12	00001	0000000C	00100	00042 00043
R2	00001	00000002	00090	00055 00056 00057
SAVEREGS	00001	00000008	00128	00040

```
  LOC  OBJECT COOE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                           ASM 0201 15.45 11/23/86

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* DAT00030
                                     4 *+---------------------------------------------------------------------+* DAT00040
                                     5 *|                                                                     |* DAT00050
                                     6 *|Module:   DATABLES                                                   |* DAT00060
                                     7 *|                                                                     |* DAT00070
                                     8 *|          Dynamic Address Translation tables and register images.    |* DAT00080
                                     9 *|                                                                     |* DAT00090
                                    10 *|Function: This module defines segment and page tables for every     |* DAT00100
                                    11 *|          address space in the system (user and system alike).      |* 0AT00110
                                    12 *|          The segment tables are exposed via the Assembler ENTRY    |* DAT00120
                                    13 *|          statement.  This makes it possible for modules of the     |* DAT00130
                                    14 *|          system to establish the segment table origin (STO) of     |* DAT00140
                                    15 *|          of system address spaces at link-edit time.               |* 0AT00150
                                    16 *|                                                                     |* DAT00160
                                    17 *|          Graphically, the relationship between these data areas    |* DAT00170
                                    18 *|          appears as:                                               |* DAT00180
                                    19 *|                                                                     |* DAT00190
                                    20 *|          Fullword                                                   |* DAT00200
                                    21 *|         +----------+                                                |* DAT00210
                                    22 *|   STO:  |    *     |  (Not contained in this module.)              |* DAT00220
                                    23 *|         +----|-----+                                                |* DAT00230
                                    24 *|              |                                                      |* DAT00240
                                    25 *|        +-----+                                                      |* DAT00250
                                    26 *|        |                                                            |* DAT00260
                                    27 *|        |       |<------------- 16 Fullword entries ------------>|  |* DAT00270
                                    28 *|        |                                                            |* DAT00280
                                    29 *|        |       +--------+--------+---------+------/ ... /-------+  |* DAT00290
                                    30 *|        +--->>>>|        |        |         |                   |  |* DAT00300
                                    31 *| Segment Table: |   *    |   *    |   ...   |              ...  |  |* DAT00310
                                    32 *|                |   |    |   |    |         |                   |  |* DAT00320
                                    33 *|                +---|----+---|----+---------+------/ ... /-------+  |* DAT00330
                                    34 *|                    |        |                                     |* DAT00340
                                    35 *|  +-------------------------+      (etc.)                          |* DAT00350
                                    36 *|  |                 |                                              |* 0AT00360
                                    37 *|  |     +-------------+                                            |* DAT00370
                                    38 *|  |     |                                                          |* DAT00380
                                    39 *|  |     |                                                          |* DAT00390
                                    40 *|  |     |                                                          |* 0AT00400
                                    41 *|  |     |       |<------------- 16 Halfword entries ------------>|  |* DAT00410
                                    42 *|  |     |                                                          |* DAT00420
                                    43 *|  |     |       +--------+--------+---------+------/ ... /-------+  |* DAT00430
                                    44 *|  |     +---->>>>|        |        |         |                  |  |* DAT00440
                                    45 *|  | Page Table: |   *    |   *    |   ...   |              ...  |  |* DAT00450
                                    46 *|  |             |   |    |   |    |         |                   |  |* DAT00460
                                    47 *|  |             +---|----+---|----+---------+------/ ... /-------+  |* DAT00470
                                    48 *|  |                 |        |                                     |* DAT00480
                                    49 *|  |     +-------------+      |      (etc.)                          |* 0AT00490
                                    50 *|  |     |                    |                                     |* DAT00500
                                    51 *|  |     |       +----------+ |    +----------+                     |* DAT00510
                                    52 *|  |     +--->>>>| Page     | +--->>>>| Page     |                  |* DAT00520
                                    53 *|  |             | Frame:   |      | Frame:   |                     |* DAT00530
                                    54 *|  |             | 4096     |      | 4096     |                     |* 0AT00540
                                    55 *|  |             | bytes    |      | bytes    |                     |* DAT00550
```

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT ASM 0201 15.45 11/23/86

```
                                56 *|   |              +---------+          +---------+                    |* DAT00560
                                57 *|   |                                                                 |* 0AT00570
                                58 *|   |                                                                 |* DAT00580
                                59 *|   |                                                                 |* DAT00590
                                60 *|   |                                                                 |* DAT00600
                                61 *|   |              |<------------- 16 Halfword entries ------------>|  |* 0AT00610
                                62 *|   |                                                                 |* DAT00620
                                63 *|   |              +--------+--------+---------+------/ ... /-------+  |* 0AT00630
                                64 *|   +--------->>>>|        |        |         |                       |  |* 0AT00640
                                65 *|     Page Table:  |   *    |   *    |   ...   |                  ...  |  |* DAT00650
                                66 *|                  |   |    |   |    |         |                       |  |* 0AT00660
                                67 *|                  +---|----+---|----+---------+------/ ... /-------+  |* DAT00670
                                68 *|                      |         |                                       |* DAT00680
                                69 *|          +-----------+         |       (etc.)                          |* 0AT00690
                                70 *|          |                     |                                       |* 0AT00700
                                71 *|          |        +---------+  |        +---------+                    |* 0AT00710
                                72 *|          +--->>>>| Page    |  +--->>>>| Page    |                    |* DAT00720
                                73 *|                   | Frame:  |          | Frame:  |                    |* DAT00730
                                74 *|                   | 4096    |          | 4096    |                    |* DAT00740
                                75 *|                   | bytes   |          | bytes   |                    |* DAT00750
                                76 *|                   +---------+          +---------+                    |* DAT00760
                                77 *|                                                                       |* DAT00770
                                78 *|                                                                       |* DAT00780
                                79 *|          This is only a sketch of the relationship between the       |* 0AT00790
                                80 *|          various data structures.  Refer to "System/370 Prin-        |* DAT00800
                                81 *|          ciples of Operation", Chapter 3: Storage for a complete     |* DAT00810
                                82 *|          explanation of the dynamic address translation process      |* DAT00820
                                83 *|          and a formal description of these structures.               |* DAT00830
                                84 *|                                                                       |* DAT00840
                                85 *|Caller:   Not called.  This module contains no executable code.       |* DAT00850
                                86 *|          Further, this module is not an address space, so all       |* DAT00860
                                87 *|          address constants resolved by the linkage editor reflect   |* 0AT00870
                                88 *|          real addresses.                                             |* DAT00880
                                89 *|                                                                       |* DAT00890
                                90 *|Mode:     Key 0.  No modules alter this CSECT.                       |* 0AT00900
                                91 *|                                                                       |* 0AT00910
                                92 *+-----------------------------------------------------------------------+* DAT00920
                                93 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* 0AT00930

                                95          PUNCH 'SPB'        Make module start on a page boundry.       DAT00950

                                97          EXTRN SLEEPPP,SLEEP,XMIT,XMITO,AUTHPP,AUTH,EXTSPP,EXTSLIH      DAT00970
                                98          EXTRN TRUPP,TRU,TRACEPP,TRACE,DSPPP,OSP,SCHPP,SCH,SHUTDOWN     0AT00980
                                99          EXTRN SLASPP,SLAS,RLASPP,RLAS,TLASPP,TLAS,CONSPP,CONSOLE        DAT00990

                               101          EXTRN USER1PP,USER1,USER2PP,USER2,USER3PP,USER3                 0AT01010
                               102          EXTRN USER4PP,USER4,USER5PP,USER5,OPPP,OPERATOR                 DAT01020

000000                         104 0ATABLES CSECT              DAT Tables Module.  Contains no code.      0AT01040

                               106 *-------------------------------------------------------------------* DAT01060
                               107 * A. Oefine the segment table for every address space in the system.  * DAT01070
                               108 *    Segment tables are 16 words in length; each word is referred to  * 0AT01080
                               109 *    as a segment table entry (STE) and contains a pointer to the     * 0AT01090
                               110 *    page table associated with the segment.  Each such pointer is    * DAT01100
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.45 11/23/86
                                    111 *    known as the page table origin (PTO).                                  * DAT01110
                                    112 *                                                                           * DAT01120
                                    113 *    Boundry alignment is critical; each table must start on a 64-          * DAT01130
                                    114 *    byte boundry or a translation exception will be generated when         * DAT01140
                                    115 *    DAT attempts to fetch a STE.                                           * DAT01150
                                    116 *                                                                           * DAT01160
                                    117 *    Each control program module has components in segments zero and        * DAT01170
                                    118 *    one.  Segment zero is contains the module's parameter page, and        * DAT01180
                                    119 *    segment one contains module code and other data areas.  Segments       * DAT01190
                                    120 *    2 through 15 are unused but are defined as invalid entries just        * DAT01200
                                    121 *    to force alignment for the next segment table.                         * DAT01210
                                    122 *--------------------------------------------------------------------------* DAT01220
                                    124          ENTRY SLEEPST,XMITST,AUTHST,EXTSST,TRUST,TRACEST,DSPST,SCHST      DAT01240
                                    125          ENTRY SLASST,RLASST,TLASST,CONSST,SHUTST                          DAT01250
000000 000004C0000004E0             127 AUTHST   DC    A(AUTH0),A(AUTH1),14F'-1'                                   DAT01270
000040 0000050000000520             128 CONSST   DC    A(CONS0),A(CONS1),14F'-1'                                   DAT01280
000080 0000054000000560             129 DSPST    DC    A(DSP0),A(DSP1),14F'-1'                                     DAT01290
0000C0 00000580000005A0             130 EXTSST   DC    A(EXTS0),A(EXTS1),14F'-1'                                   DAT01300
000100 200005C0000005E0             131 RLASST   DC    XL1'20',AL3(RLAS0),A(RLAS1),14F'-1'                         DAT01310
000140 0000060000000620             132 SCHST    DC    A(SCH0),A(SCH1),14F'-1'                                     DAT01320
000180 FFFFFFFF00000640             133 SHUTST   DC    F'-1',A(SHUT1),14F'-1' No pages in segment 0.               DAT01330
0001C0 0000066000000680             134 SLASST   DC    A(SLAS0),A(SLAS1),14F'-1'                                   DAT01340
000200 000006A0000006C0             135 SLEEPST  DC    A(SLEEP0),A(SLEEP1),14F'-1'                                 DAT01350
000240 000006E000000700             136 TLASST   DC    A(TLAS0),A(TLAS1),14F'-1'                                   DAT01360
000280 0000072020000740             137 TRACEST  DC    A(TRACE0),XL1'20',AL3(TRACE1),14F'-1' 3 pgs in segment 0    DAT01370
0002C0 0000076000000780             138 TRUST    DC    A(TRU0),A(TRU1),14F'-1'                                     DAT01380
000300 10000000000007A0             139 XMITST   DC    XL1'10',AL3(XMIT0),A(XMIT1),14F'-1' 2 pages in segment 0    DAT01390
                                    141          ENTRY USER1ST,USER2ST,USER3ST,USER4ST,USER5ST,OPST                DAT01410
000340 000007C0000007E0             143 USER1ST  DC    A(USER10),A(USER11),14F'-1'                                 DAT01430
000380 0000080000000820             144 USER2ST  DC    A(USER20),A(USER21),14F'-1'                                 DAT01440
0003C0 0000084000000860             145 USER3ST  DC    A(USER30),A(USER31),14F'-1'                                 DAT01450
000400 00000880000008A0             146 USER4ST  DC    A(USER40),A(USER41),14F'-1'                                 DAT01460
000440 000008C0000008E0             147 USER5ST  DC    A(USER50),A(USER51),14F'-1'                                 DAT01470
000480 0000090000000920             148 OPST     DC    A(OP0),A(OP1),14F'-1'                                       DAT01480
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.45 11/23/86
                                    150 *-------------------------------------------------------------------* DAT01500
                                    151 * B. Define page tables for each valid segment of an address space.   * DAT01510
                                    152 *    Page tables consist of 16 halfword page table entries (PTEs).    * DAT01520
                                    153 *    Each PTE contains a page frame real address (PFRA) referred to   * DAT01530
                                    154 *    here as a frame prefix.  The frame prefix concatenated with      * DAT01540
                                    155 *    three zeros on the right forms the real address of a page frame. * DAT01550
                                    156 *                                                                     * DAT01560
                                    157 *    Boundry alignment is critical; each table must start on a 32-    * DAT01570
                                    158 *    byte boundry or a translation exception will be generated when   * DAT01580
                                    159 *    DAT attempts to fetch a PTE.                                     * DAT01590
                                    160 *                                                                     * DAT01600
                                    161 *    Each control program module has at least one page in segments 0  * DAT01610
                                    162 *    and 1.  Unused PTEs are defined as invalid entries just to       * DAT01620
                                    163 *    maintain boundry alignment.                                      * DAT01630
                                    164 *-------------------------------------------------------------------* DAT01640

0004C0 000000FFFFFFFFFF             166 AUTH0    DC    AL3(AUTHPP),XL1'FF',14H'-1'                            DAT01660
0004E0 000000FFFFFFFFFF             167 AUTH1    DC    AL3(AUTH),XL1'FF',14H'-1'                              DAT01670

000500 000000FFFFFFFFFF             169 CONS0    DC    AL3(CONSPP),XL1'FF',14H'-1'                            DAT01690
000520 000000FFFFFFFFFF             170 CONS1    DC    AL3(CONSOLE),XL1'FF',14H'-1'                           DAT01700

000540 000000FFFFFFFFFF             172 DSP0     DC    AL3(DSPPP),XL1'FF',14H'-1'                             DAT01720
000560 000000FFFFFFFFFF             173 DSP1     DC    AL3(DSP),XL1'FF',14H'-1'                               DAT01730

000580 000000FFFFFFFFFF             175 EXTS0    DC    AL3(EXTSPP),XL1'FF',14H'-1'                            DAT01750
0005A0 000000FFFFFFFFFF             176 EXTS1    DC    AL3(EXTSLIH),XL1'FF',14H'-1'                           DAT01760

0005C0 000000FF                     178 RLAS0    DC    AL3(RLASPP),X'FF'                                      DAT01780
0005C4                      005C4   179          ORG   RLAS0+4                                                DAT01790
0005C4 0000FFFFFFFFFFFF             180          DC    XL2'0000',13H'-1'                                      DAT01800
0005E0                      005E0   181          ORG                                                          DAT01810
0005E0 000000FFFFFFFFFF             182 RLAS1    DC    AL3(RLAS),XL1'FF',14H'-1'                              DAT01820

000600 000000FFFFFFFFFF             184 SCH0     DC    AL3(SCHPP),XL1'FF',14H'-1'                             DAT01840
000620 000000FFFFFFFFFF             185 SCH1     DC    AL3(SCH),XL1'FF',14H'-1'                               DAT01850

000640 000000FFFFFFFFFF             187 SHUT1    DC    AL3(SHUTDOWN),XL1'FF',14H'-1'                          DAT01870

000660 000000FFFFFFFFFF             189 SLAS0    DC    AL3(SLASPP),XL1'FF',14H'-1'                            DAT01890
000680 000000FFFFFFFFFF             190 SLAS1    DC    AL3(SLAS),XL1'FF',14H'-1'                              DAT01900

0006A0 000000FFFFFFFFFF             192 SLEEP0   DC    AL3(SLEEPPP),XL1'FF',14H'-1'                           DAT01920
0006C0 000000FFFFFFFFFF             193 SLEEP1   DC    AL3(SLEEP),XL1'FF',14H'-1'                             DAT01930

0006E0 000000FFFFFFFFFF             195 TLAS0    DC    AL3(TLASPP),XL1'FF',14H'-1'                            DAT01950
000700 000000FFFFFFFFFF             196 TLAS1    DC    AL3(TLAS),XL1'FF',14H'-1'                              DAT01960

000720 000000FFFFFFFFFF             198 TRACE0   DC    AL3(TRACEPP),XL1'FF',14H'-1'                           DAT01980
000740 000000                       199 TRACE1   DC    AL3(TRACE)                                             DAT01990
000743                      00742   200          ORG   TRACE1+2                                               DAT02000
000742 001000                       201          DC    AL3(TRACE+4096)                                        DAT02010
000745                      00744   202          ORG   TRACE1+4                                               DAT02020
000744 002000FFFFFFFFFF             203          DC    AL3(TRACE+8192),XL1'FF',12H'-1'                        DAT02030
000760                      00760   204          ORG                                                          DAT02040
```

```
  LOC  OBJECT CODE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                    ASM 0201 15.45 11/23/86
000760 000000FFFFFFFFFF             206 TRU0     OC    AL3(TRUPP),XL1'FF',14H'-1'                          0AT02060
000780 000000FFFFFFFFFF             207 TRU1     OC    AL3(TRU),XL1'FF',14H'-1'                            0AT02070

0007A0 000000FFFFFFFFFF             209 XMIT1    OC    AL3(XMIT),XL1'FF',14H'-1'                           0AT02090

0007C0 000000FFFFFFFFFF             211 USER10   OC    AL3(USER1PP),XL1'FF',14H'-1'                        0AT02110
0007E0 000000FFFFFFFFFF             212 USER11   OC    AL3(USER1),XL1'FF',14H'-1'                          0AT02120

000800 000000FFFFFFFFFF             214 USER20   OC    AL3(USER2PP),XL1'FF',14H'-1'                        0AT02140
000820 000000FFFFFFFFFF             215 USER21   OC    AL3(USER2),XL1'FF',14H'-1'                          0AT02150

000840 000000FFFFFFFFFF             217 USER30   OC    AL3(USER3PP),XL1'FF',14H'-1'                        0AT02170
000860 000000FFFFFFFFFF             218 USER31   OC    AL3(USER3),XL1'FF',14H'-1'                          0AT02180

000880 000000FFFFFFFFFF             220 USER40   OC    AL3(USER4PP),XL1'FF',14H'-1'                        0AT02200
0008A0 000000FFFFFFFFFF             221 USER41   OC    AL3(USER4),XL1'FF',14H'-1'                          0AT02210

0008C0 000000FFFFFFFFFF             223 USER50   OC    AL3(USER5PP),XL1'FF',14H'-1'                        0AT02230
0008E0 000000FFFFFFFFFF             224 USER51   OC    AL3(USER5),XL1'FF',14H'-1'                          0AT02240

000900 000000FFFFFFFFFF             226 OP0      OC    AL3(OPPP),XL1'FF',14H'-1'                           0AT02260
000920 000000FFFFFFFFFF             227 OP1      OC    AL3(OPERATOR),XL1'FF',14H'-1'                       0AT02270
                                    228          ENO                                                       0AT02280
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AUTH	00001	00000000	00097	00167
AUTHPP	00001	00000000	00097	00166
AUTHST	00004	00000000	00127	00124
AUTH0	00003	000004C0	00166	00127
AUTH1	00003	000004E0	00167	00127
CONSOLE	00001	00000000	00099	00170
CONSPP	00001	00000000	00099	00169
CONSST	00004	00000040	00128	00125
CONS0	00003	00000500	00169	00128
CONS1	00003	00000520	00170	00128
OSP	00001	00000000	00098	00173
DSPPP	00001	00000000	00098	00172
DSPST	00004	00000080	00129	00124
OSP0	00003	00000540	00172	00129
OSP1	00003	00000560	00173	00129
EXTSLIH	00001	00000000	00097	00176
EXTSPP	00001	00000000	00097	00175
EXTSST	00004	000000C0	00130	00124
EXTS0	00003	00000580	00175	00130
EXTS1	00003	000005A0	00176	00130
OPERATOR	00001	00000000	00102	00227
OPPP	00001	00000000	00102	00226
OPST	00004	00000480	00148	00141
OP0	00003	00000900	00226	00148
OP1	00003	00000920	00227	00148
RLAS	00001	00000000	00099	00182
RLASPP	00001	00000000	00099	00178
RLASST	00001	00000100	00131	00125
RLAS0	00003	000005C0	00178	00131 00179
RLAS1	00003	000005E0	00182	00131
SCH	00001	00000000	00098	00185
SCHPP	00001	00000000	00098	00184
SCHST	00004	00000140	00132	00124
SCH0	00003	00000600	00184	00132
SCH1	00003	00000620	00185	00132
SHUTDOWN	00001	00000000	00098	00187
SHUTST	00004	00000180	00133	00125
SHUT1	00003	00000640	00187	00133
SLAS	00001	00000000	00099	00190
SLASPP	00001	00000000	00099	00189
SLASST	00004	000001C0	00134	00125
SLAS0	00003	00000660	00189	00134
SLAS1	00003	00000680	00190	00134
SLEEP	00001	00000000	00097	00193
SLEEPPP	00001	00000000	00097	00192
SLEEPST	00004	00000200	00135	00124
SLEEP0	00003	000006A0	00192	00135
SLEEP1	00003	000006C0	00193	00135
TLAS	00001	00000000	00099	00196
TLASPP	00001	00000000	00099	00195
TLASST	00004	00000240	00136	00125
TLAS0	00003	000006E0	00195	00136
TLAS1	00003	00000700	00196	00136
TRACE	00001	00000000	00098	00199 00201 00203
TRACEPP	00001	00000000	00098	00198

SYMBOL	LEN	VALUE	DEFN	REFERENCES
TRACEST	00004	00000280	00137	00124
TRACE0	00003	00000720	00198	00137
TRACE1	00003	00000740	00199	00137 00200 00202
TRU	00001	00000000	00098	00207
TRUPP	00001	00000000	00098	00206
TRUST	00004	000002C0	00138	00124
TRU0	00003	00000760	00206	00138
TRU1	00003	00000780	00207	00138
USER1	00001	00000000	00101	00212
USER1PP	00001	00000000	00101	00211
USER1ST	00004	00000340	00143	00141
USER10	00003	000007C0	00211	00143
USER11	00003	000007E0	00212	00143
USER2	00001	00000000	00101	00215
USER2PP	00001	00000000	00101	00214
USER2ST	00004	00000380	00144	00141
USER20	00003	00000800	00214	00144
USER21	00003	00000820	00215	00144
USER3	00001	00000000	00101	00218
USER3PP	00001	00000000	00101	00217
USER3ST	00004	000003C0	00145	00141
USER30	00003	00000840	00217	00145
USER31	00003	00000860	00218	00145
USER4	00001	00000000	00102	00221
USER4PP	00001	00000000	00102	00220
USER4ST	00004	00000400	00146	00141
USER40	00003	00000880	00220	00146
USER41	00003	000008A0	00221	00146
USER5	00001	00000000	00102	00224
USER5PP	00001	00000000	00102	00223
USER5ST	00004	00000440	00147	00141
USER50	00003	000008C0	00223	00147
USER51	00003	000008E0	00224	00147
XMIT	00001	00000000	00097	00209
XMITST	00001	00000300	00139	00124
XMIT0	00001	00000000	00097	00139
XMIT1	00003	000007A0	00209	00139

```
                                3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* AUT00030
                                4 *+------------------------------------------------------------------------+* AUT00040
                                5 *|                                                                        |* AUT00050
                                6 *|Module:   AUTH                                                          |* AUT00060
                                7 *|                                                                        |* AUT00070
                                8 *|          Authentication Routine.                                       |* AUT00080
                                9 *|                                                                        |* AUT00090
                               10 *|Function: Ensure a requestor is authorized to request a specific        |* AUT00100
                               11 *|          service.                                                      |* AUT00110
                               12 *|                                                                        |* AUT00120
                               13 *|          Occasionally, user code will request the system to per-      |* AUT00130
                               14 *|          form a service on its behalf.  Similarly, modules of the      |* AUT00140
                               15 *|          control program need to transfer control to other con-        |* AUT00150
                               16 *|          trol program modules, or request a specific system            |* AUT00160
                               17 *|          module to perform a utility function.  Each of these          |* AUT00170
                               18 *|          requests takes the form of an SVC instruction.  Users         |* AUT00180
                               19 *|          request numbered services via an SVC, while the system        |* AUT00190
                               20 *|          activates different parts of itself by CALLs, RETURNs or      |* AUT00200
                               21 *|          EXITs, each of which results in a transfer of control to      |* AUT00210
                               22 *|          another part of the system.                                   |* AUT00220
                               23 *|                                                                        |* AUT00230
                               24 *|          The Authentication Routine ensures each transfer of con-      |* AUT00240
                               25 *|          trol or request for service is authorized.  It does this      |* AUT00250
                               26 *|          by determining if the caller (source) is authorized to        |* AUT00260
                               27 *|          invoke the callee (target).  In order to do this, it          |* AUT00270
                               28 *|          must perform several steps.                                   |* AUT00280
                               29 *|                                                                        |* AUT00290
                               30 *|          First, the caller's identity must be authenticated. Each      |* AUT00300
                               31 *|          caller is identified by a ticket it does not supply and      |* AUT00310
                               32 *|          therefore consequently cannot forge.  In this system,        |* AUT00320
                               33 *|          the ticket is the address of the caller's segment table      |* AUT00330
                               34 *|          in real memory.  This segment table origin (STO) value       |* AUT00340
                               35 *|          is not available to the address space it defines, but        |* AUT00350
                               36 *|          each STO is known by AUTH.                                    |* AUT00360
                               37 *|                                                                        |* AUT00370
                               38 *|          Having obtained the caller's identity, AUTH examines a      |* AUT00380
                               39 *|          table of system modules or user services as appropriate      |* AUT00390
                               40 *|          looking for the requested function.  When located, it        |* AUT00400
                               41 *|          searches an authorization list associated with that          |* AUT00410
                               42 *|          function, looking for the caller's STO.  If found, AUTH      |* AUT00420
                               43 *|          activates (dispatches) the target address space imple-       |* AUT00430
                               44 *|          menting that function.  If not found, the caller is not      |* AUT00440
                               45 *|          authorized, and the user or system is terminated depend-     |* AUT00450
                               46 *|          ing on the caller.                                            |* AUT00460
                               47 *|                                                                        |* AUT00470
                               48 *|Caller:   Indirectly invoked by user and system code alike when        |* AUT00480
                               49 *|          either executes an SVC instruction.  AUTH is really          |* AUT00490
                               50 *|          called when the SVC first-level interrupt handler            |* AUT00500
                               51 *|          (module SVCFLIH) executes a Monitor Call instruction         |* AUT00510
                               52 *|          for the Activate Auth (ACTAUTH) service.  The MC instr-       |* AUT00520
                               53 *|          uction generates a program interrupt trapped by the          |* AUT00530
                               54 *|          program first-level interrupt handler (module PGMFLIH).      |* AUT00540
                               55 *|          PGMFLIH activates the Authentication Routine address         |* AUT00550
```

```
                                    56 *|          space by loading control register 1 with AUTH's segment  |* AUT00560
                                    57 *|          table origin and issuing a Load PSW (LPSW) instruction   |* AUT00570
                                    58 *|          specifying the virtual address of the AUTH main entry    |* AUT00580
                                    59 *|          point.                                                   |* AUT00590
                                    60 *|                                                                  |* AUT00600
                                    61 *|          *Note: AUTH can be considered the SVC second-level      |* AUT00610
                                    62 *|                 interrupt handler.                               |* AUT00620
                                    63 *|                                                                  |* AUT00630
                                    64 *|Mode:     Problem state, OAT on, Key 5.                           |* AUT00640
                                    65 *|                                                                  |* AUT00650
                                    66 *+------------------------------------------------------------------+* AUT00660
                                    67 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* AUT00670

                                    69          EXTRN SLEEPST,XMITST,AUTHST,EXTSST,TRUST,TRACEST             AUT00690
                                    70          EXTRN OSPST,SCHST,SLASST,RLASST,TLASST,CONSST,OPST,SHUTST     AUT00700
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.24 11/23/86

                                   72          CSECTION AUTH       Authentication Routine.                  AUT00720

                                   74+         PUNCH 'SPB'              Make module start on a page boundry.  CSE00040

000000                             76+AUTH     CSECT                     Establish R12 as module overall base: CSE00060

000000 05C0                        78+         BALR  R12,0               Give R12 a temporary base value;      CSE00080
                            00002  79+         USING *,R12               R12 now points @ SL instruction       CSE00090
000002 5FC0 C1A6    001A8          80+         SL    R12,=A(*-AUTH)      R12 = R12 - offset from CSECT start   CSE00100
                                   81+         DROP  R12                 R12 doesn't point at SL anymore;      CSE00110

                            00000  83+         USING AUTH,R12            Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012    00012          85+         B     AROUND              Jump past symbolic module identifier. CSE00150
00000A C1E4E3C840404040            86+MODID    DC    CL8'AUTH'           Handy when poking around in storage.  CSE00160
000012                             87+AROUND   DS    0H                  End of CSECTION expansion.             CSE00170

                                   89 *-------------------------------------------------------------------* AUT00740
                                   90 * A. Determine if this is a system or user request:                 * AUT00750
                                   91 *-------------------------------------------------------------------* AUT00760

                            00000  93          USING APP,R0        Make parameter page fields available.     AUT00780

000012 4810 0002    00002          95          LH    R1,APPCODE    Get the SVC code issued.                  AUT00800

000016 9580 0000    00000          97          CLI   APPSTATE,RUNUSER Is the system running a user?          AUT00820
00001A 4770 C030    00030          98          BNE   RUNSYSTM      No - System is running itself.            AUT00830

                                  100 *-------------------------------------------------------------------* AUT00850
                                  101 * 8. A running user issued an SVC.  Do the following:             * AUT00860
                                  102 *                                                                 * AUT00870
                                  103 *      IF (The user is authorized for the service) THEN           * AUT00880
                                  104 *                                                                 * AUT00890
                                  105 *         Activate the service;                                   * AUT00900
                                  106 *                                                                 * AUT00910
                                  107 *      ELSE                                                       * AUT00920
                                  108 *                                                                 * AUT00930
                                  109 *         Terminate the user;                                     * AUT00940
                                  110 *                                                                 * AUT00950
                                  111 *-------------------------------------------------------------------* AUT00960

00001E 4120 C760    00760         113          LA    R2,USRTABLE   Point @ table of user SVC services.       AUT00980
000022 45E0 C112    00112         114          BAL   R14,VALIDATE  Determine if this user is authorized.     AUT00990
000026 4780 C0E2    000E2         115          BZ    GOTO          If so, GOTO the target address space.     AUT01000
                                  116          EXIT  TRU           If not, terminate the user.             AUT01010
00002A 5800 C1AC    001AC         117+         L     R0,=CL4'TRU'       Load target address space identifier EXI00040
00002E 0A02                       118+         SVC   2                  Go to that address space.          EXI00050

                                  120 *-------------------------------------------------------------------* AUT01030
                                  121 * C. The system issued an SVC to itself.                           * AUT01040
                                  122 *                                                                 * AUT01050
                                  123 *      IF (The SVC is defined) THEN                               * AUT01060
                                  124 *                                                                 * AUT01070
                                  125 *         Start performing the service;                           * AUT01080
                                  126 *                                                                 * AUT01090
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.24 11/23/86
                                    127 *      ELSE                                                      * AUT01100
                                    128 *                                                                * AUT01110
                                    129 *         Terminate the system;                                  * AUT01120
                                    130 *                                                                * AUT01130
                                    131 *--------------------------------------------------------------* AUT01140

000030                              133 RUNSYSTM DS    0H            Here when system issues an SVC to itself:  AUT01160

000030 5910 C060      00060         135          C     R1,MAXCODE    Is the SVC defined?                        AUT01180
000034 4720 C0F2      000F2         136          BH    BADSVC        No  - It's too big.                        AUT01190
000038 1831                         137          LR    R3,R1         Yes - Copy SVC code to R3.                 AUT01200
00003A 5810 000C      0000C         138          L     R1,APPREGS+R0 We really want R1 to contain target name.  AUT01210
00003E 4120 C7B0      007B0         139          LA    R2,SYSTABLE   Point @ table of system address spaces.   AUT01220
000042 8930 0002      00002         140          SLL   R3,2          R3 = SVC code * (4 bytes/branch instr)     AUT01230
000046 47F3 C04A      0004A         141          B     SVCTABLE(R3)  Start performing the requested service.    AUT01240

00004A                              143 SVCTA8LE DS    0H            System SVC service branch table:           AUT01260

00004A 47F0 C07E      0007E         145          B     CALL          "CALL":     Authenticate caller.           AUT01280
00004E 47F0 C064      00064         146          B     CALLXMIT      "CALLXMIT": Authenticate caller.           AUT01290
000052 47F0 C0D0      000D0         147          B     EXIT          "EXIT":     Authenticate caller.           AUT01300
000056 47F0 C0B2      000B2         148          B     RETURN        "RETURN":   Begin RETURN processing.       AUT01310
00005A 47F0 C0A0      000A0         149          B     RETXMIT       "RETXMIT":  Begin RETXMIT processing.      AUT01320

                            0005E   151 ENDTA8LE EQU   *             $ Marks end of branch table $              AUT01340

00005E 0000
000060 00000004                     153 MAXCODE  DC    A(((ENDTABLE-SVCTABLE)/4)-1) Biggest system SVC code       AUT01360
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.24 11/23/86

                                  155 *-------------------------------------------------------------------* AUT01380
                                  156 * D. Implement CALLXMIT request:                                     * AUT01390
                                  157 *                                                                   * AUT01400
                                  158 *      Determine if caller is authorized;                           * AUT01410
                                  159 *                                                                   * AUT01420
                                  160 *      IF (Access rights confirmed) THEN DO;                        * AUT01430
                                  161 *                                                                   * AUT01440
                                  162 *         PUSH original caller's registers onto savearea stack;     * AUT01450
                                  163 *                                                                   * AUT01460
                                  164 *         Prepare for a recursive call to AUTH by placing parameters* AUT01470
                                  165 *         needed by XMIT in R2 and R4:                              * AUT01480
                                  166 *                                                                   * AUT01490
                                  167 *             -R2 specifies receiver's STO                          * AUT01500
                                  168 *             -R4 specifies caller's STO                            * AUT01510
                                  169 *                                                                   * AUT01520
                                  170 *         CALL the Transmitter Module to move parameters;           * AUT01530
                                  171 *                                                                   * AUT01540
                                  172 *         GOTO the target address space;                            * AUT01550
                                  173 *                                                                   * AUT01560
                                  174 *         END;                                                      * AUT01570
                                  175 *                                                                   * AUT01580
                                  176 *      ELSE                                                         * AUT01590
                                  177 *                                                                   * AUT01600
                                  178 *         Terminate the system;                                     * AUT01610
                                  179 *                                                                   * AUT01620
                                  180 *-------------------------------------------------------------------* AUT01630

000064                            182 CALLXMIT DS    0H            Here to implement CALLXMIT request:      AUT01650

000064 45E0 C112      00112       184          BAL   R14,VALIDATE  Confirm or deny access rights:           AUT01670
000068 4770 C0EA      000EA       185          BNZ   ABENDSYS     Access denied; Shut down the system.     AUT01680

00006C 45E0 C164      00164       187          BAL   R14,PUSH      Access permitted; Stack caller's savearea. AUT01700

000070 5840 0004      00004       189          L     R4,APPSTO     Get caller's STO for recursive call      AUT01720

                                  191          CALL  XMIT          Call the Transmitter Module (*Recursion*). AUT01740
000074 5800 C1B0      001B0       192+         L     R0,=CL4'XMIT'       Load target address space identifier CAL00040
000078 0A00                       193+         SVC   0                   Go to that address space.          CAL00050

00007A 47F0 C0E2      000E2       195          B     GOTO          When XMIT returns, GOTO the target A.S.  AUT01760
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.24 11/23/86

                                    197 *-------------------------------------------------------------------* AUT01780
                                    198 * E. Implement CALL request:                                        * AUT01790
                                    199 *                                                                   * AUT01800
                                    200 *      Determine if caller is authorized;                           * AUT01810
                                    201 *                                                                   * AUT01820
                                    202 *      IF (Access rights confirmed) THEN DO;                        * AUT01830
                                    203 *                                                                   * AUT01840
                                    204 *         IF (Target address space is Transmitter Module) THEN      * AUT01850
                                    205 *                                                                   * AUT01860
                                    206 *            /* This is a recursive activation of AUTH */           * AUT01870
                                    207 *                                                                   * AUT01880
                                    208 *            Put caller and receiver STOs in R0 & R1, respectively; * AUT01890
                                    209 *                                                                   * AUT01900
                                    210 *         GOSUB to the target address space (described later);      * AUT01910
                                    211 *                                                                   * AUT01920
                                    212 *         END;                                                      * AUT01930
                                    213 *                                                                   * AUT01940
                                    214 *      ELSE                                                         * AUT01950
                                    215 *                                                                   * AUT01960
                                    216 *         Terminate the system;                                     * AUT01970
                                    217 *                                                                   * AUT01980
                                    218 *-------------------------------------------------------------------* AUT01990

00007E                              220 CALL     DS    0H           Here to implement CALL request:          AUT02010

00007E 45E0 C112      00112         222          BAL   R14,VALIDATE Confirm or deny access rights:           AUT02030
000082 4770 C0EA      000EA         223          BNZ   ABENDSYS     Access denied; Shut down the system.     AUT02040

000086 0503 000C C1B0 0000C 001B0   225          CLC   APPREGS(4),=CL4'XMIT' Is target Transmitter Module?  AUT02060
00008C 4770 C098      00098         226          BNE   GOSUB        No  - Continue on with the CALL...    AUT02070
000090 5800 001C      0001C         227          L     R0,APPREGS+(R4*4) Yes - Get caller's STO (used by XMIT) AUT02080
000094 5810 0014      00014         228          L     R1,APPREGS+(R2*4) Get receiver's STO (also used by XMIT) AUT02090

                                    230 *-------------------------------------------------------------------* AUT02110
                                    231 * F. GOSUB: Activate a target address space like a subroutine.  Save * AUT02120
                                    232 *          return registers and GOTO the target address space:      * AUT02130
                                    233 *-------------------------------------------------------------------* AUT02140

000098                              235 GOSUB    DS    0H           Here to perform GOSUB function:          AUT02160

000098 45E0 C164      00164         237          BAL   R14,PUSH     Push caller's savearea onto stack        AUT02180
00009C 47F0 C0E2      000E2         238          B     GOTO         Activate target address space.           AUT02190
```

```
                                240 *-------------------------------------------------------------------* AUT02210
                                241 * G. Implement RETXMIT request:                                     * AUT02220
                                242 *                                                                   * AUT02230
                                243 *      Prepare for a recursive call to AUTH by placing parameters   * AUT02240
                                244 *      needed by XMIT in R2 and R4:                                 * AUT02250
                                245 *                                                                   * AUT02260
                                246 *           -R2 specifies receiver's STO                            * AUT02270
                                247 *           -R4 specifies caller's STO                              * AUT02280
                                248 *                                                                   * AUT02290
                                249 *      CALL the Transmitter Module;                                 * AUT02300
                                250 *                                                                   * AUT02310
                                251 *      Join normal RETURN processing;                               * AUT02320
                                252 *                                                                   * AUT02330
                                253 *-------------------------------------------------------------------* AUT02340

0000A0                          255 RETXMIT  DS    0H            Here to implement RETXMIT processing:     AUT02360

0000A0 45E0 C18A      0018A     257          BAL   R14,TOPENTRY Get address of top of stack in R11.       AUT02380
0000A4 582B 0000      00000     258          L     R2,0(R11)     Fetch receiver's STO from top of stack.   AUT02390
0000A8 5840 0004      00004     259          L     R4,APPSTO     Get caller's STO (guy doing the RETURN).  AUT02400

                                261          CALL  XMIT          Call the Transmitter Module (*Recursion*). AUT02420
0000AC 5800 C1B0      00180     262+         L     R0,=CL4'XMIT'      Load target address space identifier  CAL00040
0000B0 0A00                     263+         SVC   0                  Go to that address space.           CAL00050
```

```
                                    265 *-------------------------------------------------------------------* AUT02440
                                    266 * H. Implement RETURN request (otherwise known as GOBACK function):  * AUT02450
                                    267 *                                                                   * AUT02460
                                    268 *      Set up dispatch parameters from stacked savearea.            * AUT02470
                                    269 *                                                                   * AUT02480
                                    270 *      Restore caller's registers by popping them off the savearea  * AUT02490
                                    271 *        stack.                                                     * AUT02500
                                    272 *                                                                   * AUT02510
                                    273 *      Return to the caller via Dispatch.                           * AUT02520
                                    274 *                                                                   * AUT02530
                                    275 *-------------------------------------------------------------------* AUT02540

0000B2                              277 RETURN   DS    0H            Here to implement RETURN request:        AUT02560
                        000B2       278 GOBACK   EQU   *             (Otherwise known as GOBACK function)     AUT02570

0000B2 45E0 C18A        0018A       280          BAL   R14,TOPENTRY  Get address of top of stack in R11.     AUT02590
0000B6 4820 C75C        0075C       281          LH    R2,STKDEPTH   Get current stack depth                 AUT02600
0000BA 0620                         282          BCTR  R2,0          Decrement by one                       AUT02610
0000BC 4020 C75C        0075C       283          STH   R2,STKDEPTH   Update popped stack depth.             AUT02620
0000C0 9823 B000        00000       284          LM    R2,R3,0(R11)  Get STO & resume virtual address for   AUT02630
                                    285 *                                 the guy we are returning to.      AUT02640

0000C4 AF00 0001        00001       287          MC    SDP,0         Set dispatch parameters in low core.     AUT02660

0000C8 980F B008        00008       289          LM    R0,R15,8(R11) Restore caller's registers from stack    AUT02680

0000CC AF00 0002        00002       291          MC    DISPATCH,0    Return to caller.                        AUT02700
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.24 11/23/86
                                    293 *-------------------------------------------------------------------* AUT02720
                                    294 * I. Implement EXIT request:                                        * AUT02730
                                    295 *                                                                   * AUT02740
                                    296 *      IF (The stack is empty) THEN                                 * AUT02750
                                    297 *                                                                   * AUT02760
                                    298 *         ABEND the system;                                         * AUT02770
                                    299 *                                                                   * AUT02780
                                    300 *      Determine if the caller is authorized;                       * AUT02790
                                    301 *                                                                   * AUT02800
                                    302 *      IF (Access rights confirmed) THEN                            * AUT02810
                                    303 *                                                                   * AUT02820
                                    304 *         GOTO the target address space (described later);          * AUT02830
                                    305 *                                                                   * AUT02840
                                    306 *      ELSE                                                         * AUT02850
                                    307 *                                                                   * AUT02860
                                    308 *         ABEND the system;                                         * AUT02870
                                    309 *                                                                   * AUT02880
                                    310 *-------------------------------------------------------------------* AUT02890

000000                              312 EXIT     OS    0H            Here to implement EXIT request:          AUT02910

000000 0401 C75C C75C 0075C 0075C   314          NC    STKDEPTH,STKDEPTH Is stack empty?                      AUT02930
000006 4770 C0FA      000FA         315          BNZ   NONEMPTY      No  - Must EXIT with empty stack...      AUT02940

00000A 45E0 C112      00112         317          BAL   R14,VALIDATE  Yes - Confirm or deny access rights:    AUT02960
00000E 4770 C0EA      000EA         318          BNZ   ABENDSYS      Access denied                            AUT02970

                                    320 *-------------------------------------------------------------------* AUT02990
                                    321 * J. GOTO: Set up dispatch parameters in low core and dispatch the   * AUT03000
                                    322 *          target address space:                                    * AUT03010
                                    323 *-------------------------------------------------------------------* AUT03020

0000E2                              325 GOTO     OS    0H            Here when access to an address space is  AUT03040
                                    326 *                            permitted:                             AUT03050

0000E2 AF00 0001      00001         328          MC    SDP,0         Dispatch parameters in R2 and R3;        AUT03070
0000E6 AF00 0002      00002         329          MC    DISPATCH,0    GOTO that target address space...        AUT03080
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM 0201 15.24 11/23/B6
                                    331 *-------------------------------------------------------------------* AUT03100
                                    332 * K. General ABEND section.  Shut down the failing system after     * AUT03110
                                    333 *    placing an ABEND code in register 0:                           * AUT03120
                                    334 *-------------------------------------------------------------------* AUT03130
0000EA                              336 ABENDSYS DS    0H          Here when system integrity is questionable AUT03150
0000EA 4100 0000    00000           337          LA    R0,0        RD ABEND code for unauthorized call        AUT03160
0000EE 47F0 C10E    0010E           338          B     SHUTDOWN                                               AUT03170

0000F2                              340 BADSVC   DS    0H          Here when system issues undefined SVC:     AUT03190
0000F2 4100 0001    00001           341          LA    R0,1        R0 ABEND code for undefined SVC            AUT03200
0000F6 47F0 C10E    0010E           342          B     SHUTDOWN                                               AUT03210

0000FA                              344 NONEMPTY DS    0H          Here if stack not empty when EXITing:      AUT03230
0000FA 410D 0002    00002           345          LA    R0,2        R0 ABEND code for non-empty stack          AUT03240
0000FE 47F0 C10E    D010E           346          B     SHUTDOWN                                               AUT03250

000102                              348 STAKFULL DS    0H          Here if stack overflows during CALL:       AUT03270
000102 4100 0003    00003           349          LA    R0,3        R0 ABEND code for full stack               AUT032B0
000106 47F0 C10E    0010E           350          B     SHUTDOWN                                               AUT03290

00010A                              352 STKEMPTY DS    0H          Here if stack empty during RETURN:         AUT03310
00010A 4100 0004    00004           353          LA    R0,4        R0 ABEND code for empty stack              AUT03320

00010E                              355 SHUTDOWN DS    0H          End of the world:                          AUT03340

00010E AF00 0003    00003           357          MC    ABEND,0     Shut 'er down...                           AUT03360
```

```
                                        359 *-------------------------------------------------------------------* AUT03380
                                        360 * Subroutine: VALIOATE                                              * AUT03390
                                        361 *                                                                   * AUT03400
                                        362 * Function:   Look for a requested service in the system or user    * AUT03410
                                        363 *             service table.  If the service is found, search its   * AUT03420
                                        364 *             associated authorization list (if present) for the    * AUT03430
                                        365 *             caller's identity (STO).                              * AUT03440
                                        366 *                                                                   * AUT03450
                                        367 *             If the service is not found, set a non-zero condition * AUT03460
                                        368 *             code and return.                                      * AUT03470
                                        369 *                                                                   * AUT03480
                                        370 *             If the service is found but the caller is not author- * AUT03490
                                        371 *             ized, set a non-zero condition code and return.       * AUT03500
                                        372 *                                                                   * AUT03510
                                        373 *             If the service is found and the requestor is author-  * AUT03520
                                        374 *             ized, return with a zero condition code after placing * AUT03530
                                        375 *             the target address space STO in R2 and the entry point* AUT03540
                                        376 *             virtual address in R3.                                * AUT03550
                                        377 *                                                                   * AUT03560
                                        378 * Inputs:     R1 = Request - Either user SVC code or system address * AUT03570
                                        379 *                            space name.                            * AUT03580
                                        380 *                                                                   * AUT03590
                                        381 *             R2 = Address of table to be searched.                 * AUT03600
                                        382 *                                                                   * AUT03610
                                        383 * Outputs:    R2 = Real address of STO for target address space.    * AUT03620
                                        384 *                                                                   * AUT03630
                                        385 *             R3 = PSW key in byte 0, virtual address of entry point* AUT03640
                                        386 *                  in bytes 1-3.                                    * AUT03650
                                        387 *                                                                   * AUT03660
                                        388 *-------------------------------------------------------------------* AUT03670

000112                                  390 VALIOATE OS    OH            Confirm or deny access privileges:      AUT03690

                              00000     392          USING SERV,R2       OSECT for table entries                 AUT03710

000112                                  394 FINOSERV OS    OH            Search for the requested service:       AUT03730

000112 5910 2000       00000            396          C     R1,SERVIO     Is this the requested service?          AUT03750
000116 4780 C134       00134            397          8E    FOUNOSRV      Yes - Now on to authorization routine.  AUT03760
00011A 0503 2000 C758  00000 00758      398          CLC   SERVIO(4),OELIM No - Oid we search the entire table?  AUT03770
000120 4780 C158       00158            399          8E    OENY          Yes - Service doesn't exist...          AUT03780
000124 5850 200C       0000C            400          L     R5,SERVLCNT   No  - Get # of STOs in authorization list AUT03790
000128 8950 0002       00002            401          SLL   R5,2          R5 = (# of STOs) * (4 bytes/STO)        AUT03800
00012C 4125 2010       00010            402          LA    R2,SERVLIST-SERV(R5,R2) Point @ next table entry      AUT03810
000130 47F0 C112       00112            403          8     FINOSERV      Examine next table entry...             AUT03820

000134                                  405 FOUNOSRV OS    OH            Here when service is located in table:  AUT03840

000134 0403 200C 200C  0000C 0000C      407          NC    SERVLCNT,SERVLCNT Is this a "public" service?         AUT03860
00013A 4780 C15C       0015C            408          8Z    CONFIRM       Yes - Good news for requestor.          AUT03870
00013E 5840 200C       0000C            409          L     R4,SERVLCNT   No  - Get # of STOs in authorization list AUT03880
000142 1755                             410          XR    R5,R5         Get offset to first list element        AUT03890

000144                                  412 SRCHLIST OS    OH            Search authorization list 4 caller's STO: AUT03910
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM 0201 15.24 11/23/86
000144 5865 2010      00010         414          L      R6,SERVLIST-SERV(R5,R2) Get Authorized STO      AUT03930
000148 5960 0004      00004         415          C      R6,APPSTO       Is this the requestor?          AUT03940
00014C 4780 C15C      0015C         416          BE     CONFIRM         Yes - The requestor is authorized.AUT03950
000150 4155 0004      00004         417          LA     R5,4(R5)        No  - Get offset of next authorized STOAUT03960
000154 4640 C144      00144         418          BCT    R4,SRCHLIST     Examine next STO in authorization list...AUT03970

000158                              420 DENY     DS     0H              Here when access must be denied:AUT03990

000158 12CC                         422          LTR    R12,R12         Set non-zero condition code,    AUT04010
00015A 07FE                         423          BR     R14             Return to caller.               AUT04020

00015C                              425 CONFIRM  DS     0H              Here when access is permitted:  AUT04040

00015C 9823 2004      00004         427          LM     R2,R3,SERVASTO STO now in R2; PSW key and entry point AUT04060
                                    428 *                                virtual address in R3.        AUT04070
000160 17FF                         429          XR     R15,R15         Force condition code 0          AUT04080
000162 07FE                         430          BR     R14             Return to caller.               AUT04090
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM 0201 15.24 11/23/86
                                    432 *-------------------------------------------------------------* AUT04110
                                    433 * Subroutine: PUSH                                            * AUT04120
                                    434 *                                                             * AUT04130
                                    435 * Function:   Push caller's savearea onto savearea stack.  The * AUT04140
                                    436 *             caller's savearea is really stored in this module's * AUT04150
                                    437 *             parameter page.                                 * AUT04160
                                    438 *                                                             * AUT04170
                                    439 *             If the stack is full, ABEND the system.         * AUT04180
                                    440 *                                                             * AUT04190
                                    441 *-------------------------------------------------------------* AUT04200

000164                              443 PUSH     DS    0H            Push caller's savearea onto stack:  AUT04220

000164 4870 C75C       0075C        445          LH    R7,STKDEPTH   Get current stack depth            AUT04240
000168 4970 C75E       0075E        446          CH    R7,MAXDEPTH   Is the stack full?                 AUT04250
00016C 4780 C102       00102        447          BE    STAKFULL      Yes - ABEND the system.           AUT04260
000170 4197 0001       00001        448          LA    R9,1(R7)      No  - Increment stack depth       AUT04270
000174 4090 C75C       0075C        449          STH   R9,STKDEPTH   Update in storage.               AUT04280
000178 4180 0048       00048        450          LA    R8,SENTRYSZ   Size of a stack entry (bytes)    AUT04290
00017C 1C68                         451          MR    R6,R8         R7 now contains offset to new stack entry  AUT04300
00017E 41B7 C1B8       00188        452          LA    R11,STACK(R7) Compute top-of-stack address.     AUT04310
000182 D247 B000 0004 00000 00004   453          MVC   0(SENTRYSZ,R11),APPSTO Push caller's STO, resume virtual AUT04320
                                    454 *                                      address and registers.   AUT04330
000188 07FE                         455          BR    R14           Return to caller.                 AUT04340

                                    457 *-------------------------------------------------------------* AUT04360
                                    458 * Subroutine: TOPENTRY                                         * AUT04370
                                    459 *                                                             * AUT04380
                                    460 * Function:   Returns the address of the current top of stack entry. * AUT04390
                                    461 *                                                             * AUT04400
                                    462 *             If the stack is empty, ABEND the system.        * AUT04410
                                    463 *                                                             * AUT04420
                                    464 *-------------------------------------------------------------* AUT04430

00018A                              466 TOPENTRY DS    0H            Get address of top stack entry:   AUT04450

00018A D401 C75C C75C 0075C 0075C   468          NC    STKDEPTH,STKDEPTH Is stack empty?               AUT04470
000190 4780 C10A       0010A        469          BZ    STKEMPTY      Yes - Fatal error...              AUT04480
000194 4830 C75C       0075C        470          LH    R3,STKDEPTH   No  - Get current stack depth     AUT04490
000198 0630                         471          BCTR  R3,0          Minus 1 for offset calculations...  AUT04500
00019A 4140 0048       00048        472          LA    R4,SENTRYSZ   Get size of a stack entry         AUT04510
00019E 1C24                         473          MR    R2,R4         Compute offset to top of stack in R3. AUT04520
0001A0 41B3 C1B8       001B8        474          LA    R11,STACK(R3) Compute top-of-stack address.     AUT04530
0001A4 07FE                         475          BR    R14           Return with answer in R11.        AUT04540
```

```
                                    477 *-------------------------------------------------------------------* AUT04560
                                    478 * Module constants, variables and data areas:                      * AUT04570
                                    479 *-------------------------------------------------------------------* AUT04580

0001A8                              481          LTORG                                                        AUT04600
0001A8 00000002                     482                =A(*-AUTH)
0001AC E3D9E440                     483                =CL4'TRU'
0001B0 E7D4C9E3                     484                =CL4'XMIT'
0001B8                              485          DS    0D                                                     AUT04610
                         00048      486 SENTRYSZ EQU   4+4+64        # of bytes in stack entry: STO+vaddr+regs AUT04620
                         00014      487 SENTRIES EQU   20            # of stack entries                      AUT04630
0001B8                              488 STACK    DS    (SENTRIES)XL(SENTRYSZ) Savearea stack                AUT04640
000758 FFFFFFFF                     489 DELIM    DC    F'-1'          Marks end of user/system service table AUT04650
00075C 0000                         490 STKDEPTH DC    H'0'           Current number of stack entries        AUT04660
00075E 0014                         491 MAXDEPTH DC    AL2(*-*+SENTRIES) Maximum # of stack entries          AUT04670
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 15.24 11/23/86
                                    493 *-------------------------------------------------------------------* AUT04690
                                    494 * User service table:  This table defines those system services      * AUT04700
                                    495 *   available to general users.  The services are numbered, and are  * AUT04710
                                    496 *   invoked by executing an SVC instruction specifying the number of * AUT04720
                                    497 *   the desired service.  The services defined by this table are NOT * AUT04730
                                    498 *   available to the control program.  Each table entry has a format * AUT04740
                                    499 *   described by the SERV DSECT (SERV copy file).                    * AUT04750
                                    5D0 *-------------------------------------------------------------------* AUT04760

000760                              502 USRTABLE DS    0D                                                     AUT04780

000760 00000003                     504 READ     DC    F'3'          Console READ service                     AUT04800
000764 00000000                     505          DC    A(CONSST)     Console Address Space STO                AUT04810
000768 30010200                     506          DC    XL4'30010200' Virtual address of READ entry point      AUT04820
00076C 00000001                     507          DC    A((READLEND-READLIST)/4) # people in authorization list AUT04830
000770 00000000                     508 READLIST DC    A(OPST)       System Operator Address Space            AUT04840
                          00774     509 READLEND EQU   *                                                    AUT04850

000774 00000004                     511 WRITE    DC    F'4'          Console WRITE service                    AUT04870
000778 00000000                     512          DC    A(CONSST)     Console Address Space STO                AUT04880
00077C 30010100                     513          DC    XL4'30010100' Virtual address of WRITE entry point     AUT04890
000780 00000000                     514          DC    F'0'          WRITE is public; no authorization list   AUT04900

000784 00000005                     516 SLEEP    DC    F'5'          SLEEP service                            AUT04920
000788 00000000                     517          DC    A(SLEEPST)    SLEEP Address Space STO                  AUT04930
00078C 50010100                     518          DC    XL4'50010100' Virtual address of NAP entry point       AUT04940
000790 00000000                     519          DC    F'0'          SLEEP is public; no authorization list   AUT04950

000794 00000006                     521 SHUTOWN  DC    F'6'          System SHUTDOWN service                  AUT04970
000798 00000000                     522          DC    A(SHUTST)     Console Address Space STO                AUT04980
00079C 50010000                     523          DC    XL4'50010000' Virtual address of READ entry point      AUT04990
0007A0 00000001                     524          DC    A((SHUTLEND-SHUTLIST)/4) # people in authorization list AUT05000
0007A4 00000000                     525 SHUTLIST DC    A(OPST)       System Operator Address Space            AUT05010
                          007A8     526 SHUTLEND EQU   *                                                    AUT05020

0007A8 FFFFFFFF                     528          DC    F'-1'     $-$-$ End of User Service Table $-$-$      AUT05040
```

```
                                    530 *-------------------------------------------------------------------* AUT05060
                                    531 * System Service Table:  This table defines every callable function  * AUT05070
                                    532 *   of the Control Program.  Each table entry lists the name of the  * AUT05080
                                    533 *   function, the address of the STO for the address space imple-    * AUT05090
                                    534 *   menting the function, the PSW key that the address space will run * AUT05100
                                    535 *   with, the virtual address of the entry point of                 * AUT05110
                                    536 *   the function within the address space, and the function's       * AUT05120
                                    537 *   authorization list.  Like the User Service Table, each entry in  * AUT05130
                                    538 *   this table is described by the SERV DSECT (SERV copy file).      * AUT05140
                                    539 *                                                                   * AUT05150
                                    540 *   These functions are not directly available to users, although    * AUT05160
                                    541 *   some may be executed on a user's behalf as part of a request for * AUT05170
                                    542 *   a system service.                                               * AUT05180
                                    543 *-------------------------------------------------------------------* AUT05190

000780                              545 SYSTA8LE DS    0D            System Service Table:                    AUT05210

                         00780      547 SYSWAKE  EQU   *             Sleep Address Space WAKEUP function:     AUT05230
000780 C1E6C102                     548          DC    CL4'AWAK'     Name                                     AUT05240
000784 00000000                     549          DC    A(SLEEPST)    Sleep Address Space STO                  AUT05250
000788 50010200                     550          DC    XL4'50010200' Virtual address of WAKEUP entry point    AUT05260
00078C 00000001                     551          DC    A((AWAKLEND-AWAKLIST)/4) # people in authorization list AUT05270
0007C0 00000000                     552 AWAKLIST DC    A(EXTSST)     External interrupt SLIH                  AUT05280
                         007C4      553 AWAKLEN0 EQU   *                                                    AUT05290

                         007C4      555 SYSCNSI  EQU   *             Console Addr. Space I/O Interrupt function AUT05310
0007C4 C3D5E2C9                     556          DC    CL4'CNSI'     Name                                     AUT05320
0007C8 00000000                     557          DC    A(CONSST)     Console Address Space STO                AUT05330
0007CC 30010300                     558          DC    XL4'30010300' Virtual address of IOINT entry point     AUT05340
0007D0 00000001                     559          DC    A((CNSILEND-CNSILIST)/4) # people in authorization list AUT05350
0007D4 30000000                     560 CNSILIST DC    XL4'30000000' IOFLIH protect key serves as its STO     AUT05360
                         00708      561 CNSILEN0 EQU   *                                                    AUT05370

                         007D8      563 SYSOSP   EQU   *             Dispatcher:                              AUT05390
000708 C4E2D740                     564          DC    CL4'DSP'      Name                                     AUT05400
0007DC 00000000                     565          DC    A(DSPST)      DSP STO                                  AUT05410
0007E0 50010000                     566          DC    XL4'50010000' Virtual address of DSP main entry point  AUT05420
0007E4 00000005                     567          DC    A((DSPLEND-DSPLIST)/4) # of people in authorization list AUT05430
0007E8 0000000000000000             568 DSPLIST  0C    F'0',A(SCHST,SLEEPST,TRUST,CONSST) 0 = Startup code key AUT05440
                         007FC      569 DSPLEND  EQU   *                                                    AUT05450

                         007FC      571 SYSEXTS  EQU   *             External Interrupt SLIH:                 AUT05470
0007FC C5E7E3E2                     572          DC    CL4'EXTS'     Name                                     AUT05480
000800 00000000                     573          DC    A(EXTSST)     EXTS STO                                 AUT05490
000804 50010000                     574          DC    XL4'50010000' Virtual address of EXT SLIH main entry pnt AUT05500
000808 00000001                     575          DC    A((EXTSLEN0-EXTSLIST)/4) # people in authorization list AUT05510
00080C 20000000                     576 EXTSLIST DC    XL4'20000000' EXT FLIH protect key serves as its STO   AUT05520
                         00810      577 EXTSLEN0 EQU   *                                                    AUT05530

                         00810      579 SYSRLDQ  EQU   *             Run List Address Space - "Dequeue" entry: AUT05550
000810 D9D3C4D8                     580          DC    CL4'RLDQ'     Name                                     AUT05560
000814 00000000                     581          DC    A(RLASST)     RLAS STO                                 AUT05570
000818 50010200                     582          DC    XL4'50010200' Virtual address of DEQUEUE entry point   AUT05580
00081C 00000004                     583          DC    A((RLDQLEND-RLDQLIST)/4) # people in authorization list AUT05590
000820 0000000000000000             584 RLDQLIST 0C    A(SCHST,SLEEPST,TRUST,CONSST)                          AUT05600
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 15.24 11/23/86
                                   00830    585 RLDQLEND EQU   *                                                      AUT05610

                                   00830    587 SYSRLEQ  EQU   *            Run List Address Space - "Enqueue" entry: AUT05630
000830 D9D3C5D8                           588          DC    CL4'RLEQ'     Name                                     AUT05640
000834 00000000                           589          DC    A(RLASST)     RLAS STO                                 AUT05650
000838 50010100                           590          DC    XL4'50010100' Virtual address of ENQUEUE entry point   AUT05660
00083C 00000002                           591          DC    A((RLEQLEND-RLEQLIST)/4) # people in authorization list AUT05670
000840 0000000000000000                   592 RLEQLIST DC    A(SCHST,CONSST)                                          AUT05680
                                   00848   593 RLEQLEND EQU   *                                                      AUT05690

                                   00848   595 SYSRLGD  EQU   *            Run List Address Space - "Get UCB" entry: AUT05710
000848 D9D3C7C4                           596          DC    CL4'RLGD'     Name                                     AUT05720
00084C 00000000                           597          DC    A(RLASST)     RLAS STO                                 AUT05730
000850 50010300                           598          DC    XL4'50010300' Virtual address of GETDUCB entry point   AUT05740
000854 00000001                           599          DC    A((RLGDLEND-RLGDLIST)/4) # people in authorization list AUT05750
000858 00000000                           600 RLGDLIST DC    A(DSPST)      Only caller is dispatcher                AUT05760
                                   0085C   601 RLGDLEND EQU   *                                                      AUT05770

                                   0085C   603 SYSSIOD  EQU   *            Scheduler - "I/O Done" entry point:      AUT05790
00085C E2C9D6C4                           604          DC    CL4'SIOO'     Name                                     AUT05800
000860 00000000                           605          DC    A(SCHST)      SCH STO                                  AUT05810
000864 50010300                           606          DC    XL4'50010300' Virtual address of IODONE entry point    AUT05820
000868 00000001                           607          DC    A((SIODLEND-SIODLIST)/4) # people in authorization list AUT05830
00086C 00000000                           608 SIODLIST DC    A(CONSST)     Only caller is Console Address Space.    AUT05840
                                   00870   609 SIODLEND EQU   *                                                      AUT05850

                                   00870   611 SYSSROU  EQU   *            Scheduler - "Rouse" entry point:         AUT05870
000870 E2D9D6E4                           612          OC    CL4'SROU'     Name                                     AUT05880
000874 00000000                           613          OC    A(SCHST)      SCH STO                                  AUT05890
000878 50010200                           614          DC    XL4'50010200' Virtual address of ROUSE entry point     AUT05900
00087C 00000001                           615          DC    A((SROULEND-SROULIST)/4) # people in authorization list AUT05910
000880 00000000                           616 SROULIST DC    A(SLEEPST)    Only caller is Sleep address space.      AUT05920
                                   00884   617 SROULEND EQU   *                                                      AUT05930

                                   00884   619 SYSSTSE  EQU   *            Scheduler - "Time Slice End" entry point: AUT05950
000884 E2E3E2C5                           620          DC    CL4'STSE'     Name                                     AUT05960
000888 00000000                           621          DC    A(SCHST)      SCH STO                                  AUT05970
00088C 50010100                           622          DC    XL4'50010100' Virtual address of TSE entry point       AUT05980
000890 00000001                           623          DC    A((STSELEND-STSELIST)/4) # people in authorization list AUT05990
000894 00000000                           624 STSELIST DC    A(EXTSST)     Only caller is External Interrupt SLIH.  AUT06000
                                   00898   625 STSELEND EQU   *                                                      AUT06010

                                   00898   627 SYSSLDQ  EQU   *            Sleep List Address Space - "DEQUEUE" e.p. AUT06030
000898 E2D3C4D8                           628          DC    CL4'SLDQ'     Name                                     AUT06040
00089C 00000000                           629          DC    A(SLASST)     SCH STO                                  AUT06050
0008A0 50010200                           630          DC    XL4'50010200' Virtual address of DEQUEUE entry point   AUT06060
0008A4 00000001                           631          DC    A((SLDQLEND-SLDQLIST)/4) # people in authorization list AUT06070
0008A8 00000000                           632 SLDQLIST DC    A(SLEEPST)    Only caller is Sleep address space.      AUT06080
                                   008AC   633 SLDQLEND EQU   *                                                      AUT06090

                                   008AC   635 SYSSLEQ  EQU   *            Sleep List Address Space - "ENQUEUE" e.p. AUT06110
0008AC E2D3C5D8                           636          DC    CL4'SLEQ'     Name                                     AUT06120
0008B0 00000000                           637          DC    A(SLASST)     SLAS STO                                 AUT06130
0008B4 50010100                           638          DC    XL4'50010100' Virtual address of ENQUEUE entry point   AUT06140
0008B8 00000001                           639          DC    A((SLEQLEND-SLEQLIST)/4) # people in authorization list AUT06150
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 15.24 11/23/86
0008BC 00000000                      640 SLEQLIST DC    A(SLEEPST)    Only caller is Sleep address space.        AUT06160
                                008C0 641 SLEQLEND EQU   *                                                        AUT06170

                                008C0 643 SYSTLAS  EQU   *             Terminate List Address Space:              AUT06190
0008C0 E303C1E2                      644           DC    CL4'TLAS'     Name                                       AUT06200
0008C4 00000000                      645           DC    A(TLASST)     TLAS STO                                   AUT06210
0008C8 50010000                      646           DC    XL4'50010000' Virtual address of TLAS main entry point   AUT06220
0008CC 00000001                      647           DC    A((TLASLEND-TLASLIST)/4) # people in authorization list  AUT06230
0008D0 00000000                      648 TLASLIST DC    A(TRUST)      Only caller is Terminate Running User.     AUT06240
                                008D4 649 TLASLEND EQU   *                                                        AUT06250

                                008D4 651 SYSTRACE EQU   *             System Internal Trace Table Routine:       AUT06270
0008D4 E3D9C1C3                      652           DC    CL4'TRAC'     Name                                       AUT06280
0008D8 00000000                      653           DC    A(TRACEST)    TRACE STO                                  AUT06290
0008DC 50010000                      654           DC    XL4'50010000' Virtual address of TRAC main entry point   AUT06300
0008E0 00000000                      655           DC    F'0'          No authorization list; anyone can call it. AUT06310

                                008E4 657 SYSTRU   EQU   *             Terminate a Running User:                  AUT06330
0008E4 E3D9E440                      658           DC    CL4'TRU'      Name                                       AUT06340
0008E8 00000000                      659           DC    A(TRUST)      TRU STO                                    AUT06350
0008EC 50010000                      660           DC    XL4'50010000' Virtual address of TRU main entry point    AUT06360
0008F0 00000002                      661           DC    A((TRULEND-TRULIST)/4) # of people in authorization list AUT06370
0008F4 4000000000000000              662 TRULIST  DC    XL4'40000000',A(AUTHST) PGM FLIH protect key is its STO   AUT06380
                                008FC 663 TRULEND  EQU   *                                                        AUT06390

                                008FC 665 SYSXMIT  EQU   *             Transmitter Module (XMIT):                 AUT06410
0008FC E7D4C9E3                      666           OC    CL4'XMIT'     Name                                       AUT06420
000900 00000000                      667           DC    A(XMITST)     XMIT STO                                   AUT06430
000904 00010000                      668           DC    XL4'00010000' Virtual address of XMIT main entry point   AUT06440
000908 00000002                      669           DC    A((XMITLEND-XMITLIST)/4) # people in authorization list  AUT06450
00090C 0000000000000000              670 XMITLIST DC    A(AUTHST,CONSST)                                           AUT06460
                                00914 671 XMITLEND EQU   *                                                        AUT06470

000914 FFFFFFFF                      673           DC    F'-1'      $-$-$ End of System Service Table $-$-$         AUT06490
                                     674           PRINT OFF                                                        AUT06500
```

POS.ID	REL.ID	FLAGS	ADDRESS
000F	0001	0C	000788
000F	0001	0C	000784
000F	0001	0C	0007F0
000F	0001	0C	000824
000F	0001	0C	000880
000F	0001	0C	0008A8
000F	0001	0C	00088C
000F	0002	0C	000900
000F	0003	0C	0008F8
000F	0003	0C	00090C
000F	0004	0C	0007C0
000F	0004	0C	000800
000F	0004	0C	000894
000F	0005	0C	0007F4
000F	0005	0C	000828
000F	0005	0C	000800
000F	0005	0C	0008E8
000F	0006	0C	000808
000F	0007	0C	00070C
000F	0007	0C	000858
000F	0008	0C	0007EC
000F	0008	0C	000820
000F	0008	0C	000840
000F	0008	0C	000860
000F	0008	0C	000874
000F	0008	0C	000888
000F	0009	0C	00089C
000F	0009	0C	000880
000F	000A	0C	000814
000F	000A	0C	000834
000F	000A	0C	00084C
000F	000B	0C	0008C4
000F	000C	0C	000764
000F	000C	0C	000778
000F	000C	0C	0007C8
000F	000C	0C	0007F8
000F	000C	0C	00082C
000F	000C	0C	000844
000F	000C	0C	00086C
000F	000C	0C	000910
000F	0000	0C	000770
000F	000D	0C	0007A4
000F	000E	0C	000798

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ABEND	00001	00000003	00724	00357
ABENOSYS	00002	000000EA	00336	00185 00223 00318
APP	00001	00000000	00775	00093
APPCODE	00002	00000002	00778	00095
APPREGS	00004	0000000C	00781	00138 00225 00227 00228
APPSTATE	00001	00000000	00776	00097
APPSTO	00004	00000004	00779	00189 00259 00415 00453
AROUND	00002	00000012	00087	00085
AUTH	00001	00000000	00076	00083 00482
AUTHST	00001	00000000	00069	00662 00670
AWAKLEND	00001	000007C4	00553	00551
AWAKLIST	00004	000007C0	00552	00551
BAOSVC	00002	000000F2	00340	00136
CALL	00002	0000007E	00220	00145
CALLXMIT	00002	00000064	00182	00146
CNSILEND	00001	00000708	00561	00559
CNSILIST	00004	00000704	00560	00559
CONFIRM	00002	0000015C	00425	00408 00416
CONSST	00001	00000000	00070	00505 00512 00557 00568 00584 00592 00608 00670
DELIM	00004	00000758	00489	00398
DENY	00002	00000158	00420	00399
DISPATCH	00001	00000002	00723	00291 00329
DSPLEND	00001	000007FC	00569	00567
DSPLIST	00004	000007E8	00568	00567
DSPST	00001	00000000	00070	00565 00600
ENOTABLE	00001	0000005E	00151	00153
EXIT	00002	000000D0	00312	00147
EXTSLEND	00001	00000810	00577	00575
EXTSLIST	00004	0000080C	00576	00575
EXTSST	00001	00000000	00069	00552 00573 00624
FINOSERV	00002	00000112	00394	00403
FOUNOSRV	00002	00000134	00405	00397
GOSUB	00002	00000098	00235	00226
GOTO	00002	000000E2	00325	00115 00195 00238
MAXCODE	00004	00000060	00153	00135
MAXDEPTH	00002	0000075E	00491	00446
NONEMPTY	00002	000000FA	00344	00315
OPST	00001	00000000	00070	00508 00525
PUSH	00002	00000164	00443	00187 00237
READLEND	00001	00000774	00509	00507
READLIST	00004	00000770	00508	00507
RETURN	00002	000000B2	00277	00148
RETXMIT	00002	000000A0	00255	00149
RLASST	00001	00000000	00070	00581 00589 00597
RLOQLEND	00001	00000830	00585	00583
RLOQLIST	00004	00000820	00584	00583
RLEQLEND	00001	00000848	00593	00591
RLEQLIST	00004	00000840	00592	00591
RLGOLEND	00001	0000085C	00601	00599
RLGOLIST	00004	00000858	00600	00599
RUNSYSTM	00002	00000030	00133	00098
RUNUSER	00001	00000080	00737	00097
R0	00001	00000000	00689	00093 00117 00138 00192 00227 00262 00289 00337 00341 00345 00349 00353
R1	00001	00000001	00690	00095 00135 00137 00138 00228 00396
R11	00001	0000000B	00700	00258 00284 00289 00452 00453 00474

```
SYMBOL     LEN   VALUE    DEFN    REFERENCES
R12        00001 0000000C 00701   00078 00079 00080 00081 00083 00422 00422
R14        00001 0000000E 00703   00114 00184 00187 00222 00237 00257 00280 00317 00423 00430 00455 00475
R15        00001 0000000F 00704   00289 00429 00429
R2         00001 00000002 00691   00113 00139 00228 00258 00281 00282 00283 00284 00392 00402 00402 00414 00427 00473
R3         00001 00000003 00692   00137 00140 00141 00284 00427 00470 00471 00474
R4         00001 00000004 00693   00189 00227 00259 00409 00418 00472 00473
R5         00001 00000005 00694   00400 00401 00402 00410 00410 00414 00417 00417
R6         00001 00000006 00695   00414 00415 00451
R7         00001 00000007 00696   00445 00446 00448 00452
R8         00001 00000008 00697   00450 00451
R9         00001 00000009 00698   00448 00449
SCHST      00001 00000000 00070   00568 00584 00592 00605 00613 00621
SDP        00001 00000001 00722   00287 00328
SENTRIES   00001 00000014 00487   00488 00491
SENTRYSZ   00001 00000048 00486   00450 00453 00472 00488
SERV       00001 00000000 00813   00392 00402 00414
SERVASTO   00004 00000004 00815   00427
SERVID     00004 00000000 00814   00396 00398
SERVLCNT   00004 0000000C 00818   00400 00407 00407 00409
SERVLIST   00004 00000010 00819   00402 00414
SHUTDDWN   00002 0000010E 00355   00338 00342 00346 00350
SHUTLEND   00001 000007A8 00526   00524
SHUTLIST   00004 000007A4 00525   00524
SHUTST     00001 00000000 00070   00522
SIODLEND   00001 00000870 00609   00607
SIODLIST   00004 0000086C 00608   00607
SLASST     00001 00000000 00070   00629 00637
SLDQLEND   00001 000008AC 00633   00631
SLDQLIST   00004 000008A8 00632   00631
SLEEPST    00001 00000000 00069   00517 00549 00568 00584 00616 00632 00640
SLEQLEND   00001 000008C0 00641   00639
SLEQLIST   00004 0000088C 00640   00639
SRCHLIST   00002 00000144 00412   00418
SROULEND   00001 00000884 00617   00615
SROULIST   00004 00000880 00616   00615
STACK      00072 00000188 00488   00452 00474
STAKFULL   00002 00000102 00348   00447
STKDEPTH   00002 0000075C 00490   00281 00283 00314 00314 00445 00449 00468 00468 00470
STKEMPTY   00002 0000010A 00352   00469
STSELEND   00001 00000898 00625   00623
STSELIST   00004 00000894 00624   00623
SVCTABLE   00002 0000004A 00143   00141 00153
SYSTA8LE   00008 00000780 00545   00139
TLASLEND   00001 000008D4 00649   00647
TLASLIST   00004 000008D0 00648   00647
TLASST     00001 00000000 00070   00645
TOPENTRY   00002 0000018A 00466   00257 00280
TRACEST    00001 00000000 00069   00653
TRULEND    00001 000008FC 00663   00661
TRULIST    00004 000008F4 00662   00661
TRUST      00001 00000000 00069   00568 00584 00648 00659
USRTABLE   00008 00000760 00502   00113
VALIDATE   00002 00000112 00390   00114 00184 00222 00317
XMITLEND   00001 00000914 00671   00669
XMITLIST   00004 0000090C 00670   00669
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
XMITST	00001	00000000	00069	00667

```
  LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                  ASM 0201 20.40 12/04/86

                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* EXT00040
                                    4 *+-------------------------------------------------------------------+* EXT00050
                                    5 *|                                                                   |* EXT00060
                                    6 *|Module:    EXTSLIH                                                 |* EXT00070
                                    7 *|                                                                   |* EXT00080
                                    8 *|           External Interrupt Second-Level Interrupt Handler.      |* EXT00090
                                    9 *|                                                                   |* EXT00100
                                   10 *|Function:  Analyze the source of the external interrupt and        |* EXT00110
                                   11 *|           dispatch the appropriate module tailored specifically   |* EXT00120
                                   12 *|           to that interrupt class.                                |* EXT00130
                                   13 *|                                                                   |* EXT00140
                                   14 *|           The only external interrupts expected by this control   |* EXT00150
                                   15 *|           program come from the CPU Timer and the Time-of-Oay     |* EXT00160
                                   16 *|           Clock Comparator.  The CPU Timer is used to govern the  |* EXT00170
                                   17 *|           dispatch quantum for a user, while the Clock Comparator |* EXT00180
                                   18 *|           is used to facilitate user Sleep requests.  An external |* EXT00190
                                   19 *|           interrupt from any other source e.g. interval timer     |* EXT00200
                                   20 *|           will not be recognized and the system will consequently |* EXT00210
                                   21 *|           terminate itself.                                       |* EXT00220
                                   22 *|                                                                   |* EXT00230
                                   23 *|           Module size is small; refer to the source listing for   |* EXT00240
                                   24 *|           additional details.                                     |* EXT00250
                                   25 *|                                                                   |* EXT00260
                                   26 *|Caller:    Called exclusively by EXTFLIH.                          |* EXT00270
                                   27 *|                                                                   |* EXT00280
                                   28 *|Mode:      Problem state, OAT on, Key 5.                           |* EXT00290
                                   29 *|                                                                   |* EXT00300
                                   30 *+-------------------------------------------------------------------+* EXT00310
                                   31 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* EXT00320

                                   33          CSECTION EXTSLIH    External Interrupt SLIH.               EXT00340

                                   35+         PUNCH 'SPB'              Make module start on a page boundry.  CSE00040

000000                             37+EXTSLIH  CSECT                    Establish R12 as module overall base: CSE00060

000000 05C0                        39+         8ALR  R12,0              Give R12 a temporary base value;      CSE00080
                             00002 40+         USING *,R12              R12 now points @ SL instruction       CSE00090
000002 5FC0 C046       00048       41+         SL    R12,=A(*-EXTSLIH)  R12 = R12 - offset from CSECT start  CSE00100
                                   42+         OROP  R12                R12 doesn't point at SL anymore;      CSE00110

                             00000 44+         USING EXTSLIH,R12        Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012       00012       46+         B     AROUNO             Jump past symbolic module identifier. CSE00150
00000A C5E7E3E203C9C840            47+MODIO    OC    CL8'EXTSLIH'       Handy when poking around in storage.  CSE00160
000012                             48+AROUNO   OS    0H                 End of CSECTION expansion.            CSE00170

000012 0501 C040 0000 00040 00000  50          CLC   CLKCMPAR,0(RO) Is interrupt from the clock comparator?   EXT00360
000018 4770 C022       00022       51          8NE   CHKTIMER       No  - See if it was CPU timer.            EXT00370
                                   52          EXIT  AWAK           Yes - Awaken sleeping user.               EXT00380
00001C 5800 C04C       0004C       53+         L     RO,=CL4'AWAK'      Load target address space identifier  EXI00040
000020 0A02                        54+         SVC   2                  Go to that address space.             EXI00050
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.40 12/04/86

000022                               56 CHKTIMER DS    0H            Re-examine interrupt source:          EXT00400

000022 D501 C042 0000 00042 00000    58          CLC   CPUTIMER,0(R0) Did CPU timer pop?                   EXT00420
000028 4770 C032      00032          59          BNE   ITSOVER       No  - What could it have been???       EXT00430
                                     60          EXIT  STSE          Yes - Go to Scheduler, Time Slice End EXT00440
00002C 5800 C050      00050          61+         L     R0,=CL4'STSE'      Load target address space identifier EXI00040
000030 0A02                          62+         SVC   2                  Go to that address space.       EXI00050
                                     63 *                                  entry point.                  EXT00450

000032                               65 ITSOVER  DS    0H            Here to terminate system:             EXT00470

                                     67          CRASH               Crash due to unrecognized ext interrupt. EXT00490
000032 005CC3D9C1E2C85C              68+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher. CRA00030

000040                               70          DS    0D            Alignment for constants:              EXT00510
000040 1004                          71 CLKCMPAR DC    XL2'1004'     Clock Comparator external interrupt code. EXT00520
000042 1005                          72 CPUTIMER DC    XL2'1005'     CPU Timer external interrupt code.    EXT00530
000048                               73          LTORG                                                     EXT00540
000048 00000002                      74                =A(*-EXTSLIH)
00004C C1E6C1D2                      75                =CL4'AWAK'
000050 E2E3E2C5                      76                =CL4'STSE'
                                     77          PRINT OFF                                               EXT00550
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00048	00046
CLKTIMER	00002	00000022	00056	00051
CLKCMPAR	00002	00000040	00071	00050
CPUTIMER	00002	00000042	00072	00058
EXTSLIH	00001	00000000	00037	00044 00074
ITSOVER	00002	00000032	00065	00059
R0	00001	00000000	00092	00050 00053 00058 00061
R12	00001	0000000C	00104	00039 00040 00041 00042 00044

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM 0201 12.42 11/15/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRU00030
                                      4 *+------------------------------------------------------------------+* TRU00040
                                      5 *|                                                                  |* TRU00050
                                      6 *|Module:   TRU                                                     |* TRU00060
                                      7 *|                                                                  |* TRU00070
                                      8 *|          Terminate a Running User.                               |* TRU00080
                                      9 *|                                                                  |* TRU00090
                                     10 *|Function: Prevent a user from ever running again by moving his    |* TRU00100
                                     11 *|          UC8 to the list of terminated user UC8s.                |* TRU00110
                                     12 *|                                                                  |* TRU00120
                                     13 *|          Under various circumstances the system may have to      |* TRU00130
                                     14 *|          terminate a user address space.  Reasons include illegal|* TRU00140
                                     15 *|          (unauthorized) request for a service and out-right      |* TRU00150
                                     16 *|          program check, among others.                            |* TRU00160
                                     17 *|                                                                  |* TRU00170
                                     18 *|Caller:   Invoked by several modules; see AUTH for the complete   |* TRU00180
                                     19 *|          list of callers.  No parameters are passed to TRU.      |* TRU00190
                                     20 *|                                                                  |* TRU00200
                                     21 *|          *Note: TRU can be considered a program interrupt        |* TRU00210
                                     22 *|                 second-level interrupt handler.                  |* TRU00220
                                     23 *|                                                                  |* TRU00230
                                     24 *|Mode:     Problem state, DAT on, Key 5.                           |* TRU00240
                                     25 *|                                                                  |* TRU00250
                                     26 *+------------------------------------------------------------------+* TRU00260
                                     27 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRU00270

                                     29          CSECTION TRU        Terminate Running User.                  TRU00290

                                     31+         PUNCH 'SP8'             Make module start on a page boundry. CSE00040

000000                               33+TRU      CSECT                    Establish R12 as module overall base: CSE00060

000000 05C0                          35+         BALR  R12,0              Give R12 a temporary base value;      CSE00080
                              00002  36+         USING *,R12              R12 now points @ SL instruction       CSE00090
000002 5FC0 C066       00068         37+         SL    R12,=A(*-TRU)      R12 = R12 - offset from CSECT start   CSE00100
                                     38+         DROP  R12                R12 doesn't point at SL anymore;      CSE00110

                              00000  40+         USING TRU,R12            Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012       00012         42+         B     AROUND             Jump past symbolic module identifier. CSE00150
00000A E3D9E44040404040              43+MODID    DC    CL8'TRU'           Handy when poking around in storage.  CSE00160
000012                               44+AROUND   DS    0H                 End of CSECTION expansion.             CSE00170

                                     46          CALL RLDQ           Dequeue running user from run list.      TRU00310
000012 5800 C06C       0006C         47+         L     R0,=CL4'RLDQ'      Load target address space identifier  CAL00040
000016 0A00                          48+         SVC   0                  Go to that address space.             CAL00050

                              00000  50          USING UC8,R0        Parameter page now holds a UC8.          TRU00330

000018 D21F C048 0000 00048 00000    52          MVC   SAVEAREA,UC8  Copy portion of UC8 destroyed by trace.  TRU00350
00001E 0717 0000 0000 00000 00000    53          XC    0(6*4,R0),0(R0) Clear out the trace entry.           TRU00360
000024 0207 0000 0078 00000 00078    54          MVC   0(8,R0),UC8ID Move terminated UC8 ident to trace entry. TRU00370
                                     55          TRACE               Trace Terminate function.            TRU00380
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 12.42 11/15/86

00002A 0207 0018 C00A 00018 0000A    56+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
000030 5800 C070      00070          57+         L     R0,=CL4'TRAC'      Load target address space identifier  CAL00040
000034 0A01                          58+         SVC   1                  Go to that address space.             CAL00050

000036 021F 0000 C048 00000 00048    60          MVC   UCB(L'SAVEAREA),SAVEAREA Restore partially destroyed UCB TRU00400

                                     62          CALLXMIT TLAS      Move user to terminate list.              TRU00420
00003C 5800 C074      00074          63+         L     R0,=CL4'TLAS'      Load target address space identifier  CAL00040
000040 0A01                          64+         SVC   1                  Go to that address space.             CAL00050

                                     66          EXIT OSP           Goto Oispatcher.                          TRU00440
000042 5800 C078      00078          67+         L     R0,=CL4'OSP'       Load target address space identifier  EXI00040
000046 0A02                          68+         SVC   2                  Go to that address space.             EXI00050

000048                               70          DS    0D           Force alignment for constants.            TRU00460
000048                               71 SAVEAREA DS    XL(8*4)      Savearea for portion of UCB               TRU00470
000068                               72          LTORG                                                       TRU00480
000068 00000002                      73                =A(*-TRU)
00006C D903C408                      74                =CL4'RLDQ'
000070 E3D9C1C3                      75                =CL4'TRAC'
000074 E3D3C1E2                      76                =CL4'TLAS'
000078 C4E20740                      77                =CL4'OSP'
                                     78          PRINT OFF                                                   TRU00490
```

ASM 0201 12.42 11/15/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00044	00042
MODID	00008	0000000A	00043	00056
R0	00001	00000000	00093	00047 00050 00053 00053 00054 00056 00057 00063 00067
R12	00001	0000000C	00105	00035 00036 00037 00038 00040
SAVEAREA	00032	00000048	00071	00052 00060 00060
TRU	00001	00000000	00033	00040 00073
UCB	00001	00000000	00186	00050 00052 00060 00199
UCBID	00008	00000078	00193	00054

```
                                                                              ASM 0201 20.53 12/04/86
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TLA00040
                                     4 *+--------------------------------------------------------------------+* TLA00050
                                     5 *|                                                                    |* TLA00060
                                     6 *|Module:   TLAS                                                      |* TLA00070
                                     7 *|                                                                    |* TLA00080
                                     8 *|          Terminate List Address Space.                             |* TLA00090
                                     9 *|                                                                    |* TLA00100
                                    10 *|Function: Enqueue a UCB on the Terminate List.                      |* TLA00110
                                    11 *|                                                                    |* TLA00120
                                    12 *|          The module maintains a pool of space organized into 10    |* TLA00130
                                    13 *|          slots.  Each slot is large enough to contain a UCB.       |* TLA00140
                                    14 *|          The Terminate List consists of UCBs allocated sequen-     |* TLA00150
                                    15 *|          tially in the pool.  It serves as a "graveyard" repos-    |* TLA00160
                                    16 *|          itory for "dead" users.  Any user that has ABENDED due    |* TLA00170
                                    17 *|          to a programming error or has been terminated by the      |* TLA00180
                                    18 *|          system is added to this list.                             |* TLA00190
                                    19 *|                                                                    |* TLA00200
                                    20 *|          List members are not chained; rather, they are added to   |* TLA00210
                                    21 *|          the next sequentially-numbered slot.                      |* TLA00220
                                    22 *|                                                                    |* TLA00230
                                    23 *|Caller:   Called by several modules.  See its authorization list   |* TLA00240
                                    24 *|          in AUTH.                                                  |* TLA00250
                                    25 *|                                                                    |* TLA00260
                                    26 *|Mode:     Problem state, DAT on, Key 5.                             |* TLA00270
                                    27 *|                                                                    |* TLA00280
                                    28 *+--------------------------------------------------------------------+* TLA00290
                                    29 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TLA00300

                                    31          CSECTION TLAS      Terminate List Address Space.             TLA00320

                                    33+         PUNCH 'SPB'             Make module start on a page boundry.  CSE00040

000000                              35+TLAS     CSECT                    Establish R12 as module overall base: CSE00060

000000 05C0                         37+         BALR  R12,0              Give R12 a temporary base value;      CSE00080
                            00002   38+         USING *,R12              R12 now points @ SL instruction       CSE00090
000002 5FC0 C036      00038         39+         SL    R12,=A(*-TLAS)     R12 = R12 - offset from CSECT start   CSE00100
                                    40+         DROP  R12                R12 doesn't point at SL anymore;      CSE00110

                            00000   42+         USING TLAS,R12           Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012         44+         B     AROUND             Jump past symbolic module identifier. CSE00150
00000A E3D3C1E240404040             45+MODID    DC    CL8'TLAS'          Handy when poking around in storage.  CSE00160
000012                              46+AROUND   DS    0H                 End of CSECTION expansion.            CSE00170

                                    48 *---------------------------------------------------------------------* TLA00340
                                    49 * A. Search for a free UCB slot in the Terminate List.                * TLA00350
                                    50 *---------------------------------------------------------------------* TLA00360

000012 4110 C040      00040         52          LA    R1,UCBAREA   Point at UCB slot pool.                   TLA00380
                            00000   53          USING UCB,R1       Access UCB DSECT.                         TLA00390

000016                              55 FINDUCB  DS    0H           Search for free UCB:                      TLA00410
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.53 12/04/86
000016 D503 C030 1000 00030 00000    57          CLC   FREE,UCB      Is this one free?                    TLA00430
00001C 4780 C028      D0028          58          BE    FOUNDUCB      Yes - This'll do.                    TLA00440
000020 4110 1090      00090          59          LA    R1,SLOTSIZE(,R1) No  - Move to next,               TLA00450
000024 47F0 C016      00016          60          B     FINDUCB       And check it out.                    TLA00460

                                     62 *-------------------------------------------------------------------* TLA00480
                                     63 * B. Move UCB from parameter page into newly-acquired Terminate List * TLA00490
                                     64 *    slot.                                                       * TLA00500
                                     65 *-------------------------------------------------------------------* TLA00510

000028                               67 FOUNDUCB DS    0H            Here with available UCB space @ R1:  TLA00530

000028 D284 1000 0000 00000 00000    69          MVC   UCB(UCBSIZE),0(R0) Install new UCB in table.       TLA00550

                                     71          RETURN            Return to caller.                    TLA00570
00002E 0A03                          72+         SVC   3                                                  RET00030
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.53 12/04/86
                                     74 *-------------------------------------------------------------------* TLA00590
                                     75 * Constants and variables:                                         * TLA00600
                                     76 *-------------------------------------------------------------------* TLA00610

000030                               78          DS    0D            Align constant area.                   TLA00630
000030 C6D9C5C5                      79 FREE     DC    CL4'FREE'     Number of UCBs on sleep list.          TLA00640
000038                               80          LTORG                                                      TLA00650
000038 00000002                      81                =A(*-TLAS)

                                     83 *-------------------------------------------------------------------* TLA00670
                                     84 * UCB pool / Terminate List:                                       * TLA00680
                                     85 *-------------------------------------------------------------------* TLA00690

000040                               87          OS    0D            Force alignment.                       TLA00710

                         00090       89 SLOTSIZE EQU   X'90'         Slightly larger than a UCB.            TLA00730

000040                               91 UCBAREA  DS    (10)XL(SLOTSIZE) Define enough space for 10 UCBs.      TLA00750
0005E0                   00040       92          ORG   UCBAREA                                               TLA00760
000040 C609C5C5                      93          DC    CL4'FREE'                                            TLA00770
000044                   000D0       94          ORG   UCBAREA+SLOTSIZE                                     TLA00780
0000D0 C6D9C5C5                      95          DC    CL4'FREE'                                            TLA00790
0000D4                   00160       96          ORG   UCBAREA+(SLOTSIZE*2)                                 TLA00800
000160 C6D9C5C5                      97          DC    CL4'FREE'                                            TLA00810
000164                   001F0       98          ORG   UCBAREA+(SLOTSIZE*3)                                 TLA00820
0001F0 C6D9C5C5                      99          DC    CL4'FREE'                                            TLA00830
0001F4                   00280      100          ORG   UCBAREA+(SLOTSIZE*4)                                 TLA00840
000280 C6D9C5C5                     101          DC    CL4'FREE'                                            TLA00850
000284                   00310      102          ORG   UCBAREA+(SLOTSIZE*5)                                 TLA00860
000310 C6D9C5C5                     103          DC    CL4'FREE'                                            TLA00870
000314                   003A0      104          ORG   UCBAREA+(SLOTSIZE*6)                                 TLA00880
0003A0 C6D9C5C5                     105          DC    CL4'FREE'                                            TLA00890
0003A4                   00430      106          ORG   UCBAREA+(SLOTSIZE*7)                                 TLA00900
000430 C6D9C5C5                     107          DC    CL4'FREE'                                            TLA00910
000434                   004C0      108          ORG   UCBAREA+(SLOTSIZE*8)                                 TLA00920
0004C0 C6D9C5C5                     109          DC    CL4'FREE'                                            TLA00930
0004C4                   00550      110          ORG   UCBAREA+(SLOTSIZE*9)                                 TLA00940
000550 C6D9C5C5                     111          DC    CL4'FREE'                                            TLA00950

                                    113          PRINT OFF                                                  TLA00970
```

ASM 0201 20.53 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00046	00044
FINDUC8	00002	00000016	00055	00060
FOUNDUC8	00002	00000028	00067	00058
FREE	00004	00000030	00079	00057
R0	00001	00000000	00128	00069
R1	00001	00000001	00129	00052 00053 00059 00059
R12	00001	0000000C	00140	00037 00038 00039 00040 00042
SLOTSIZE	00001	00000090	00089	00059 00091 00094 00096 00098 00100 00102 00104 00106 00108 00110
TLAS	00001	00000000	00035	00042 00081
UC8	00001	00000000	00223	00053 00057 00069 00237
UC8AREA	00144	00000040	00091	00052 00092 00094 00096 00098 00100 00102 00104 00106 00108 00110
UC8SIZE	00001	00000085	00237	00069

```
                                3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* XMI00030
                                4 *+-----------------------------------------------------------------+* XMI00040
                                5 *|                                                                 |* XMI00050
                                6 *|Module:   XMIT                                                   |* XMI00060
                                7 *|                                                                 |* XMI00070
                                8 *|          Transmitter Module.                                     |* XMI00080
                                9 *|                                                                 |* XMI00090
                               10 *|Function: Implement inter-address space communication.           |* XMI00100
                               11 *|                                                                 |* XMI00110
                               12 *|          XMIT is invoked during normal system operation when:   |* XMI00120
                               13 *|                                                                 |* XMI00130
                               14 *|            1) One address space CALLs another and needs to pass |* XMI00140
                               15 *|               parameters (CALL with transmit: CALLXMIT),        |* XMI00150
                               16 *|                                                                 |* XMI00160
                               17 *|            2) An address space RETURNs to its caller, and needs |* XMI00170
                               18 *|               to pass back results (RETURN w/transmit: RETXMIT). |* XMI00180
                               19 *|                                                                 |* XMI00190
                               20 *|          Every operating system address space (except XMIT and   |* XMI00200
                               21 *|          AUTH) has a paramater page attached to it at virtual   |* XMI00210
                               22 *|          address zero.  When an address space wants to transmit |* XMI00220
                               23 *|          data to another address space, it places that data in  |* XMI00230
                               24 *|          the parameter page and invokes either CALL or RETURN   |* XMI00240
                               25 *|          with the XMIT option.  XMIT will move the entire       |* XMI00250
                               26 *|          parameter page to the appropriate destination page as  |* XMI00260
                               27 *|          part of the transfer-of-control process.  Thus, address |* XMI00270
                               28 *|          spaces may communicate up to 4096 bytes of data among  |* XMI00280
                               29 *|          themselves.                                            |* XMI00290
                               30 *|                                                                 |* XMI00300
                               31 *|          To move data between address spaces, XMIT attaches the |* XMI00310
                               32 *|          sender's page 0 to its page 0 and attaches the         |* XMI00320
                               33 *|          receiver's page 0 to its page 1.  Then It simply copies |* XMI00330
                               34 *|          page 0 to page 1.                                      |* XMI00340
                               35 *|                                                                 |* XMI00350
                               36 *|          Since XMIT will be called upon to move data between    |* XMI00360
                               37 *|          pages with keys differing from its own it must run with |* XMI00370
                               38 *|          key 0.  This is not viewed as a potential integrity    |* XMI00380
                               39 *|          exposure because the module is Isolated in an address  |* XMI00390
                               40 *|          space alone with itself.                               |* XMI00400
                               41 *|                                                                 |* XMI00410
                               42 *|Caller:   Invoked by AUTH on behalf of any system address space  |* XMI00420
                               43 *|          needing to transmit data to another during CALL or     |* XMI00430
                               44 *|          RETURN processing.  Also called by Console Address Space |* XMI00440
                               45 *|          to move read and write data between itself and user     |* XMI00450
                               46 *|          parameter pages.                                        |* XMI00460
                               47 *|                                                                 |* XMI00470
                               48 *|          AUTH dispatches XMIT with the following conventions:   |* XMI00480
                               49 *|             -R0 always contains the sender's STO,               |* XMI00490
                               50 *|             -R1 always contains the receiver's STO.             |* XMI00500
                               51 *|                                                                 |* XMI00510
                               52 *|Mode:     Problem state, DAT on, Key 0.                          |* XMI00520
                               53 *|                                                                 |* XMI00530
                               54 *+-----------------------------------------------------------------+* XMI00540
                               55 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* XMI00550
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM 0201 16.47 11/22/86
                                    57           CSECTION XMIT       Transmitter Module:                        XMI00570
                                    59+          PUNCH 'SPB'             Make module start on a page boundry.   CSE00040
000000                              61+XMIT      CSECT                    Establish R12 as module overall base: CSE00060
000000 05C0                         63+          BALR  R12,0              Give R12 a temporary base value;      CSE00080
                              00002 64+          USING *,R12              R12 now points @ SL instruction       CSE00090
000002 5FC0 C036      00038         65+          SL    R12,=A(*-XMIT)     R12 = R12 - offset from CSECT start   CSE00100
                                    66+          DROP  R12                R12 doesn't point at SL anymore;      CSE00110
                              00000 68+          USING XMIT,R12           Now R12 has A(This CSECT) in it.      CSE00130
000006 47F0 C012      00012         70+          B     AROUND             Jump past symbolic module identifier. CSE00150
00000A E7D4C9E340404040             71+MODID     DC    CL8'XMIT'          Handy when poking around in storage.  CSE00160
000012                              72+AROUND    DS    0H                 End of CSECTION expansion.             CSE00170
000012 AF00 0004      00004         74           MC    BIND,0       Attach caller and receiver parameter pages XMI00590
                                    75 *                            to pages 0 and 1 of this address space.  XMI00600
000016 9825 C028      00028         77           LM    R2,R5,XMITCTOR Get caller-to-receiver move parameters  XMI00620
00001A 0E42                         78           MVCL  R4,R2        Move caller's parameter page to receiver's XMI00630
00001C AF00 0005      00005         80           MC    DETACH,0     Detach caller and receiver parameter pages XMI00650
                                    82           RETURN             Go back to Authentication Routine.       XMI00670
000020 0A03                         83+          SVC   3                                                     RET00030
000028                              85           DS    0D           Force doubleword alignment for constants: XMI00690
000028 00000000                     87 XMITCTOR  DC    A(X'00000000') Virtual addresss of caller's parm page. XMI00710
00002C 00001000                     88           DC    F'4096'         Page size is 4K .                      XMI00720
000030 00001000                     89           DC    A(X'00001000') Virtual address of receiver's parm page . XMI00730
000034 00001000                     90           DC    F'4096'         Page size is 4K .                      XMI00740
                                    92           PRINT OFF                                                   XMI00760
```

ASM 0201 16.47 11/22/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00072	00070
BIND	00001	00000004	00143	00074
DETACH	00001	00000005	00144	00080
R12	00001	0000000C	00119	00063 00064 00065 00066 00068
R2	00001	00000002	00109	00077 00078
R4	00001	00000004	00111	00078
R5	00001	00000005	00112	00077
XMIT	00001	00000000	00061	00068 00163
XMITCTOR	00004	00000028	00087	00077

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRA00030
                                    4 *+-------------------------------------------------------------------+* TRA00040
                                    5 *|                                                                   |* TRA00050
                                    6 *|Module:   TRACE                                                    |* TRA00060
                                    7 *|                                                                   |* TRA00070
                                    8 *|          System Internal Trace Table.                             |* TRA00080
                                    9 *|                                                                   |* TRA00090
                                   10 *|Function: Install a fixed-length trace entry in the trace table.   |* TRA00100
                                   11 *|          Each entry has the following format:                     |* TRA00110
                                   12 *|                                                                   |* TRA00120
                                   13 *|                                (fullword)                         |* TRA00130
                                   14 *|                   0     1     2     3     4     5     6     7      |* TRA00140
                                   15 *|               +-----+-----+-----+-----+-----+-----+-----+-----+   |* TRA00150
                                   16 *|               | Mod Ident | ********************************* |   |* TRA00160
                                   17 *|               +-----+-----+-----+-----+-----+-----+-----+-----+   |* TRA00170
                                   18 *|                                                                   |* TRA00180
                                   19 *|          Words 2-7 are free-format.  Words 0-1 include a symbolic  |* TRA00190
                                   20 *|          identification (eye-catcher) of the module logging the   |* TRA00200
                                   21 *|          entry.  The caller's id is supplied automatically by the |* TRA00210
                                   22 *|          TRACE macro.                                             |* TRA00220
                                   23 *|                                                                   |* TRA00230
                                   24 *|Caller:   Callable by any operating system address space except    |* TRA00240
                                   25 *|          the Authentication Routine and the Transmitter Module.   |* TRA00250
                                   26 *|          It is not necessary for either of these components to    |* TRA00260
                                   27 *|          log trace entries.  Furthermore, neither AUTH nor XMIT   |* TRA00270
                                   28 *|          have parameter pages in the conventional sense, so they  |* TRA00280
                                   29 *|          cannot effectively pass a trace entry to TRACE.          |* TRA00290
                                   30 *|                                                                   |* TRA00300
                                   31 *|          The requirement that a caller be an address space        |* TRA00310
                                   32 *|          precludes the use of the trace facility by first-level   |* TRA00320
                                   33 *|          interrupt handlers, who are not address spaces.          |* TRA00330
                                   34 *|                                                                   |* TRA00340
                                   35 *|          To pass an entry to Trace, the caller builds an image of |* TRA00350
                                   36 *|          the entry in its parameter page, words 0-5, then calls   |* TRA00360
                                   37 *|          trace via the TRACE macro.                               |* TRA00370
                                   38 *|                                                                   |* TRA00380
                                   39 *|Mode:     Problem state, DAT on, Key 5.                            |* TRA00390
                                   40 *|                                                                   |* TRA00400
                                   41 *+-------------------------------------------------------------------+* TRA00410
                                   42 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRA00420

                                   44          CSECTION TRACE      System internal trace table routine:      TRA00440

                                   46+         PUNCH 'SP8'              Make module start on a page boundry. CSE00040

000000                             48+TRACE    CSECT                     Establish R12 as module overall base: CSE00060

000000 05C0                        50+         BALR  R12,0               Give R12 a temporary base value;      CSE00080
                         00002     51+         USING *,R12               R12 now points @ SL instruction       CSE00090
000002 5FC0 C05E         00060     52+         SL    R12,=A(*-TRACE)     R12 = R12 - offset from CSECT start   CSE00100
                                   53+         OROP  R12                 R12 doesn't point at SL anymore;      CSE00110

                         00000     55+         USING TRACE,R12           Now R12 has A(This CSECT) in it.      CSE00130
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 12.42 11/15/86

000006 47F0 C012      00012          57+         B     AROUND           Jump past symbolic module identifier. CSE00150
00000A E3D9C1C3C5404040              58+MODID    DC    CL8'TRACE'       Handy when poking around in storage.  CSE00160
000012                               59+AROUND   DS    0H               End of CSECTION expansion.             CSE00170

                                      61 *-------------------------------------------------------------------* TRA00460
                                      62 * A. Set up trace table base register.  The table begins on the next  * TRA00470
                                      63 *    page boundry.                                                     * TRA00480
                                      64 *-------------------------------------------------------------------* TRA00490

000012 18DC                           66          LR    R13,R12          This module uses two base registers, so TRA00510
000014 41D0 0FFF      00FFF           67          LA    R13,4095(R13)      set up R13 as trace table base register: TRA00520
000018 41DD 0001      00001           68          LA    R13,1(R13)       R13 = R12 + 4096;                       TRA00530

                            01000     70          USING TRACE+4096,R13 Tell Assembler about second base.       TRA00550

                                      72 *-------------------------------------------------------------------* TRA00570
                                      73 * B. Update the count of entries in the table, then calculate the     * TRA00580
                                      74 *    address of the new entry.                                         * TRA00590
                                      75 *-------------------------------------------------------------------* TRA00600

00001C 5870 C054      00054           77          L     R7,COUNT         Get # of entries currently in table.   TRA00620
000020 41A7 0001      00001           78          LA    R10,1(R7)        Increment that count.                  TRA00630

000024 4180 0020      00020           80          LA    R8,TENTRYSZ      Get size of a trace entry.             TRA00650
000028 1C68                           81          MR    R6,R8            Compute offset to next free entry.     TRA00660
00002A 41B7 D000      01000           82          LA    R11,TABLE(R7)    Get address of next free entry.        TRA00670

                                      84 *-------------------------------------------------------------------* TRA00690
                                      85 * C. Install the new trace entry.                                    * TRA00700
                                      86 *-------------------------------------------------------------------* TRA00710

00002E D207 B000 0018 00000 00018     88          MVC   0(8,R11),TENTRYSZ-8(R0) Move module ident to trace entry TRA00730
000034 0217 8008 0000 00008 00000     89          MVC   8(TENTRYSZ-8,R11),0(R0) Move trace data to trace entry  TRA00740

                                      91 *-------------------------------------------------------------------* TRA00760
                                      92 * D. Force wrap-around for next entry if table is full.              * TRA00770
                                      93 *-------------------------------------------------------------------* TRA00780

00003A 59A0 C050      00050           95          C     R10,SIZE         Is table full now?                     TRA00800
00003E 4770 C044      00044           96          8NE   UPDTSIZE         No  - Still has room for more entries. TRA00810
000042 17AA                           97          XR    R10,R10          Yes - Wrap NEXT -> FIRST.              TRA00820

000044                                99 UPDTSIZE DS    0H               Here when trace entry has been installed: TRA00840

000044 50A0 C054      00054          101          ST    R10,COUNT        Update count of table entries.         TRA00860
000048 50B0 C058      00058          102          ST    R11,CURRENT      Save virtual address of current entry for TRA00870
                                     103 *                                  use during debugging.              TRA00880

                                     105          RETURN                 Return to caller.                      TRA00900
00004C 0A03                          106+         SVC   3                                                       RET00030

                            00002    108 NUMPAGES EQU   2                Two page trace table.                  TRA00920
                            00020    109 TENTRYSZ EQU   32               Size of a trace table entry (bytes).   TRA00930

000050                               111          DS    0F                                                      TRAC0950
```

```
  LOC  OBJECT CODE    AODR1 AODR2  STMT   SOURCE STATEMENT                                    ASM 0201 12.42 11/15/86

000050 00000100                    112 SIZE     DC    A(*-*+((NUMPAGES*4096)/TENTRYSZ)) Table capacity        TRA00960
000054 00000000                    113 COUNT    DC    F'0'          Current number of entries in table.       TRA00970
000058 00000000                    114 CURRENT  DC    F'0'          Virtual address of current entry.         TRA00980
000060                             115          LTORG                                                         TRA00990
000060 0D000002                    116               =A(*-TRACE)

000064                 01000       118          ORG   TRACE+4096                                              TRA01010

001000                             120 TABLE    DS    XL4096        Page 1 of trace table.                    TRA01030
002000                             121          DS    XL4096        Page 2 of trace table.                    TRA01040

                                   123          PRINT OFF                                                     TRA01060
```

ASM 0201 12.42 11/15/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00059	00057
COUNT	00004	0D000054	00113	00077 00101
CURRENT	00004	00000058	00114	00102
NUMPAGES	00001	00000002	00108	00112
R0	00001	00000000	00138	00088 00089
R10	00001	0000000A	00148	00078 00095 00097 00097 00101
R11	00001	0000000B	00149	00082 00088 00089 00102
R12	00001	0000000C	00150	00050 00051 00052 00053 00055 00066
R13	00001	0000000D	00151	00066 00067 00067 00068 00068 00070
R6	00001	00000006	00144	00081
R7	00001	00000007	00145	00077 00078 00082
R8	00001	00000008	00146	00080 00081
SIZE	00004	00000050	00112	00095
TABLE	04096	00001000	00120	00082
TENTRYSZ	00001	00000020	00109	00080 00088 00089 00112
TRACE	00001	00000000	00048	00055 00070 00116 00118
UPDTSIZE	00002	00000044	00099	00096

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                               ASM 0201 13.15 11/22/86

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLE00030
                                     4 *+-----------------------------------------------------------------+* SLE00040
                                     5 *|                                                                 |* SLE00050
                                     6 *|Module:   SLEEP                                                  |* SLE00060
                                     7 *|                                                                 |* SLE00070
                                     8 *|          Sleep Address Space.                                   |* SLE00080
                                     9 *|                                                                 |* SLE00090
                                    10 *|Function: Put users to sleep and awaken them from naps.          |* SLE00100
                                    11 *|                                                                 |* SLE00110
                                    12 *|          The module has two entry points: NAP and WAKEUP.  User |* SLE00120
                                    13 *|          sleep requests are processed by NAP, which suspends    |* SLE00130
                                    14 *|          execution of that user's address space for the time    |* SLE00140
                                    15 *|          interval requested (in microseconds).  WAKEUP handles  |* SLE00150
                                    16 *|          the expiration of a user sleep interval, and causes that |* SLE00160
                                    17 *|          user to resume running.                                |* SLE00170
                                    18 *|                                                                 |* SLE00180
                                    19 *|          Sleep uses the services of the Sleep List Address Space. |* SLE00190
                                    20 *|                                                                 |* SLE00200
                                    21 *|          See the NAP and WAKEUP entry points for further details. |* SLE00210
                                    22 *|                                                                 |* SLE00220
                                    23 *|Caller:    NAP is a public entry point called by any user needing |* SLE00230
                                    24 *|          to SLEEP.  To SLEEP, a user executes the SLEEP SVC.    |* SLE00240
                                    25 *|                                                                 |* SLE00250
                                    26 *|          The WAKEUP entry point is called exclusively by EXTSLIH. |* SLE00260
                                    27 *|                                                                 |* SLE00270
                                    28 *|Mode:      Problem state, DAT on, Key 5.                          |* SLE00280
                                    29 *|                                                                 |* SLE00290
                                    30 *+-----------------------------------------------------------------+* SLE00300
                                    31 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLE00310

                                    33          CSECTION SLEEP      Sleep Address Space:                    SLE00330

                                    35+         PUNCH 'SP8'            Make module start on a page boundry.  CSE00040

000000                              37+SLEEP    CSECT                  Establish R12 as module overall base: CSE00060

000000 05C0                         39+         BALR  R12,0            Give R12 a temporary base value;      CSE00080
                            00002   40+         USING *,R12            R12 now points @ SL instruction       CSE00090
000002 5FC0 C2A6       002A8        41+         SL    R12,=A(*-SLEEP)  R12 = R12 - offset from CSECT start   CSE00100
                                    42+         DROP  R12              R12 doesn't point at SL anymore;      CSE00110

                            00000   44+         USING SLEEP,R12        Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012       00012        46+         B     AROUND           Jump past symbolic module identifier. CSE00150
00000A E2D3C5C5D7404040             47+MODID    DC    CL8'SLEEP'       Handy when poking around in storage.  CSE00160
000012                              48+AROUND   DS    0H               End of CSECTION expansion.            CSE00170

                                    50          CRASH               Non-executable entry point.             SLE00350
000012 005CC3D9C1E2C85C             51+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.   CRA00030
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 13.15 11/22/86
                                     53 *-------------------------------------------------------------------* SLE00370
                                     54 * A. NAP entry point.  Begin by getting the UCB of the user that   * SLE00380
                                     55 *    issued the SLEEP request.                                      * SLE00390
                                     56 *-------------------------------------------------------------------* SLE00400

                                     58          ENTRYPT NAP,100     Entry point for user-issued SLEEP SVC:  SLE00420

00001A                         00100 60+         ORG   SLEEP+X'100'                                          ENT00040

000100                               62+NAP      DS    0H            Define entry point name.                  ENT00060
                                     63+         PUSH  USING         Save overall USING environment.           ENT00070
000100 05C0                          64+         BALR  R12,0         Establish temporary base.                 ENT00080
                               00102 65+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000102 5FC0 C1AA       002AC         66+         SL    R12,=A(*-SLEEP)   R12 now points @ CSECT start.        ENT00100
                                     67+         POP   USING         Restore overall USING environment.        ENT00110

                                     69          CALL  RLDQ          Get UCB of new sleeper.                   SLE00440
000106 5800 C2B0       002B0         70+         L     R0,=CL4'RLDQ'       Load target address space identifier CAL00040
00010A 0A00                          71+         SVC   0                   Go to that address space.         CAL00050

                               00000 73          USING UCB,R0        UCB image now in parameter page.          SLE00460

                                     75 *-------------------------------------------------------------------* SLE00480
                                     76 * B. Construct a NAP trace entry.                                   * SLE00490
                                     77 *-------------------------------------------------------------------* SLE00500

00010C D21F C278 0000 0027B 00000    79          MVC   SAVEAREA,UCB  Save trace entry portion of parameter page SLE00520

000112 D207 0000 C298 00000 00298    81          MVC   0(8,R0),=CL8'*NAP*' Put entry name in trace entry.      SLE00540
000118 D207 0008 0008 00008 00008    82          MVC   8(8,R0),UCBGREGS+R0 Put nap time usec in trace entry.   SLE00550
00011E D207 0010 0078 00010 00078    83          MVC   16(8,R0),UCBID      Put userid in trace entry.          SLE00560
                                     B4          TRACE                     Create the trace entry.             SLE00570
000124 D207 0018 C00A 00018 0000A    85+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00012A 5800 C2B4       002B4         86+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
00012E 0A01                          87+         SVC   1                   Go to that address space.         CAL00050

000130 D21F 0000 C278 00000 00278    89          MVC   UCB(L'SAVEAREA),SAVEAREA Restore partially-destroyed UCB SLE00590

                                     91 *-------------------------------------------------------------------* SLE00610
                                     92 * C. Convert the user's specified nap time into a TOD clock format.  * SLE00620
                                     93 *    He told us how many microseconds he wants to sleep; that value  * SLE00630
                                     94 *    is his saved R0 and R1.  Compute the user's wakeup time as the   * SLE00640
                                     95 *    sum of his nap time and the current time-of-day.                * SLE00650
                                     96 *-------------------------------------------------------------------* SLE00660

000136 D407 0008 0008 00008 00008    98          NC    UCBGREGS+R0(8),UCBGREGS+R0 See if nap time is zero.     SLE00680
00013C 4780 C17A       0017A         99          BZ    NAPEXIT       It was: Pointless to let him sleep.       SLE00690

000140 B205 C260       00260        101          STCK  TOD           Get present TOD clock value.              SLE00710
000144 9823 C260       00260        102          LM    R2,R3,TOD     Get time in register pair.                SLE00720
000148 9845 0008       00008        103          LM    R4,R5,UCBGREGS+R0 Get user's nap time (usec) from R0,R1.  SLE00730
00014C 8D40 000C       0000C        104          SLDL  R4,12         Convert user usec to TOD clock format.    SLE00740
000150 1E35                         105          ALR   R3,R5         Add low-order TOD words.                  SLE00750
000152 47C0 C15C       0015C        106          BC    8+4,ADDHIGH   No carry-out; sum high-order TOD words.  SLE00760
000156 4110 0001       00001        107          LA    R1,1          Carry = 1.                                SLE00770
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 13.15 11/22/86
00015A 1E21                        108           ALR   R2,R1         Add carry-out to high-order TOD word.      SLE00780

00015C                             110 ADDHIGH   DS    0H            Here to sum high-order TOD words:          SLE00800

00015C 1E24                        112           ALR   R2,R4         Now clock comparator TOD in R2,R3.         SLE00820
00015E 9023 0008       00008       113           STM   R2,R3,UCBGREGS+R0 User's R0, R1 used by enqueue logic.  SLE00830
000162 9023 C268       00268       114           STM   R2,R3,NEWSLEEP Save new sleeper's wakeup TOD.            SLE00840

                                   116 *-------------------------------------------------------------------*  SLE00860
                                   117 * D. Put the user on the Sleep List.                                *  SLE00870
                                   118 *-------------------------------------------------------------------*  SLE00880

                                   120           CALLXMIT SLEQ       Add UCB to sleep list.                   SLE00900
000166 5800 C288       002B8       121+          L     R0,=CL4'SLEQ'       Load target address space identifier CAL00040
00016A 0A01                        122+          SVC   1                   Go to that address space.          CAL00050

                                   124 *-------------------------------------------------------------------*  SLE00920
                                   125 * E. SLEQ will return the wakeup TOD for the first user on the sleep  *  SLE00930
                                   126 *    list in words 0 & 1 of the parameter page.  If the returned TOD  *  SLE00940
                                   127 *    is greater than TOD just calculated for the new sleeper, the     *  SLE00950
                                   128 *    clock comparator has to be reset with the new sleeper's wakeup   *  SLE00960
                                   129 *    TOD because he will awaken before anyone else on the list.       *  SLE00970
                                   130 *-------------------------------------------------------------------*  SLE00980

00016C D507 C268 0000  00268 00000  132           CLC   NEWSLEEP,0(R0) Returned TOD : New sleeper's TOD       SLE01000
000172 4720 C17A       0017A       133           BH    NAPEXIT       <=: Guy @ head of list wakes up first.     SLE01010

000176 AF00 0006       00006       135           MC    SCK,0         >: Reset clock comparator for new sleeper. SLE01030
                                   136 *===>                         *Note: R2, R3 still contain new user's TOD SLE01040

00017A                             138 NAPEXIT   DS    0H            Exit with user sleeping soundly:           SLE01060

                                   140           EXIT  OSP           Go to dispatcher.                          SLE01080
00017A 5800 C2BC       002BC       141+          L     R0,=CL4'OSP'        Load target address space identifier EXI00040
00017E 0A02                        142+          SVC   2                   Go to that address space.          EXI00050
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 13.15 11/22/86
                                   144 *-------------------------------------------------------------------* SLE01100
                                   145 * A'. WAKEUP entry point.  Begin by getting the UCB for the user we   * SLE01110
                                   146 *     must wake up.                                                   * SLE01120
                                   147 *-------------------------------------------------------------------* SLE01130

                                   149          ENTRYPT WAKEUP,200   Awaken sleeping user:                  SLE01150

000180                      00200  151+         ORG   SLEEP+X'200'                                         ENT00040

000200                             153+WAKEUP   DS    0H            Define entry point name.                ENT00060
                                   154+         PUSH  USING         Save overall USING environment.         ENT00070
000200 05C0                        155+         BALR  R12,0         Establish temporary base.               ENT00080
                            00202  156+         USING *,R12         R12 now points @ next SL instruction:   ENT00090
000202 5FC0 C0BE      002C0        157+         SL    R12,=A(*-SLEEP)   R12 now points @ CSECT start.      ENT00100
                                   158+         POP   USING         Restore overall USING environment.      ENT00110

                                   160          CALL  SLDQ          Get UCB for waking user.                SLE01170
000206 5800 C2C4      002C4        161+         L     R0,=CL4'SLDQ'       Load target address space identifier CAL00040
00020A 0A00                        162+         SVC   0                   Go to that address space.         CAL00050

                                   164 *-------------------------------------------------------------------* SLE01190
                                   165 * B'. Construct a WAKEUP trace entry.                               * SLE01200
                                   166 *-------------------------------------------------------------------* SLE01210

00020C D21F C278 0000 00278 00000  168          MVC   SAVEAREA,UCB  Save trace portion of parameter page.   SLE01230

000212 D207 0000 C2A0 00000 002A0  170          MVC   0(8,R0),=CL8'*WAKEUP*' Move entry name to trace entry. SLE01250
000218 D207 0008 0078 00008 00078  171          MVC   8(8,R0),UCBID         Move userid to trace entry.     SLE01260
00021E D707 0010 0010 00D10 00010  172          XC    16(8,R0),16(R0)       Clear remainder of entry.       SLE01270
                                   173          TRACE                 Create trace entry.                 SLE01280
000224 D207 0018 C00A 00018 0000A  174+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00022A 5800 C2B4      002B4        175+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
00022E 0A01                        176+         SVC   1                   Go to that address space.         CAL00050

000230 D21F 0000 C278 00000 00278  178          MVC   UCB(L'SAVEAREA),SAVEAREA Restore partially-destroyed UCB SLE01300

                                   180 *-------------------------------------------------------------------* SLE01320
                                   181 * C'. Make the user runnable: Reschedule him.                       * SLE01330
                                   182 *-------------------------------------------------------------------* SLE01340

                                   184          CALLXMIT SROU       Reschedule awakening user.             SLE01360
000236 5800 C2C8      002C8        185+         L     R0,=CL4'SROU'       Load target address space identifier CAL00040
00023A 0A01                        186+         SVC   1                   Go to that address space.         CAL00050

                                   188 *-------------------------------------------------------------------* SLE01380
                                   189 * D'. The Sleep List Address Space returned to us a doubleword in the * SLE01390
                                   190 *     UCB forward and backward pointers that is either zero if no     * SLE01400
                                   191 *     other users are sleeping or the wakeup TOD for the next (new    * SLE01410
                                   192 *     first) user on the sleep list.  If no users are sleeping, we    * SLE01420
                                   193 *     must set the clock as high as we can to prevent meaningless     * SLE01430
                                   194 *     interruptions from it.  Otherwise we set the clock from the new * SLE01440
                                   195 *     wakeup TOD.  The exit to Dispatcher.                            * SLE01450
                                   196 *-------------------------------------------------------------------* SLE01460

00023C D407 0000 0000 00000 00000  198          NC    0(8,R0),0(R0) Is there another sleeping user?         SLE01480
```

```
LOC    OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT
000242 4780 C24E       0024E        199            BZ    MAXCLOCK      No  - Must "turn off" clock comparator.   SLE01490
000246 9823 0000       00000        200            LM    R2,R3,0(R0)   Yes - Get wakeup TOD for next user.    SLE01500
00024A 47F0 C252       00252        201            B     RESETCC       Go reset the timer.                   SLE01510

00024E                              203 MAXCLOCK   DS    0H            Here to turn off clock comparator:    SLE01530

00024E 9823 C270       00270        205            LM    R2,R3,MAXTOD  Get largest possible TOD.             SLE01550

000252                              207 RESETCC    DS    0H            Here to set clock comparator:         SLE01570

000252 AF00 0006       00006        209            MC    SCK,0         Set Clock Comparator.                 SLE01590

                                    211            EXIT  DSP           Go to dispatcher.                     SLE01610
000256 5800 C2BC       002BC        212+           L     R0,=CL4'DSP'      Load target address space identifier EXI00040
00025A 0A02                         213+           SVC   2               Go to that address space.           EXI00050
```

```
  LOC  OBJECT CODE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                ASM 0201 13.15 11/22/86
000260                               215           OS    00            Align constant area.                  SLEG1630
000260                               216 TOO       OS    0             Current TOO.                          SLE01640
000268                               217 NEWSLEEP  OS    0             Wakeup TOO for the new sleeper.       SLE01650
000270 FFFFFFFFFFFFFFFF              218 MAXTOO    OC    2F'-1'        Maximum TOO clock value.              SLE01660
000278                               219 SAVEAREA  OS    XL(8*4)       8 word savearea.                      SLE01670
000298                               220           LTORG                                                     SLE01680
000298 5C05C1075C404040              221                 =CL8'*NAP*'
0002A0 5CE6C1D2C5E4D75C              222                 =CL8'*WAKEUP*'
0002A8 00000002                      223                 =A(*-SLEEP)
0002AC 00000102                      224                 =A(*-SLEEP)
0002B0 0903C408                      225                 =CL4'RLOQ'
0002B4 E309C1C3                      226                 =CL4'TRAC'
0002B8 E203C508                      227                 =CL4'SLEQ'
0002BC C4E20740                      228                 =CL4'OSP'
0002C0 00000202                      229                 =A(*-SLEEP)
0002C4 E203C408                      230                 =CL4'SLOQ'
0002C8 E20906E4                      231                 =CL4'SROU'
                                     232           PRINT OFF                                                 SLE01690
```

SYMBOL LEN VALUE DEFN REFERENCES ASM 0201 13.15 11/22/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ADDHIGH	00002	0000015C	00110	00106
AROUND	0D002	00000012	00048	00046
MAXCLOCK	00002	0000024E	00203	00199
MAXTOD	D0004	00000270	00218	00205
MODID	00008	0000000A	00047	00085 00174
NAPEXIT	00002	0000017A	00138	00099 00133
NEWSLEEP	00008	00000268	00217	00114 00132
RESETCC	00002	00000252	00207	00201
R0	00001	00000000	00247	00070 00073 00081 00082 00082 00083 00085 00086 00098 00098 00103 00113 00121 00132 00141 00161 00170 00171 00172 00172 00174 00175 00185 00198 00198 00200 00212
R1	00001	00000001	00248	00107 00108
R12	00001	0000000C	00259	00039 00040 00041 00042 00044 00064 00065 00066 00155 00156 00157
R2	00001	D0000002	00249	00102 00108 00112 00113 00114 00200 00205
R3	00001	00000003	00250	00102 00105 00113 00114 00200 00205
R4	00001	00000004	00251	00103 00104 00112
R5	00001	00000005	00252	00103 00105
SAVEAREA	00032	00000278	00219	00079 00089 00089 00168 00178 00178
SCK	00001	00000006	00285	00135 00209
SLEEP	00001	00000000	00037	00044 00060 00151 00223 00224 00229
TOD	00008	00000260	00216	00101 00102
UCB	00001	00000000	00341	00073 00079 00089 00168 00178 00355
UCBGREGS	00004	00000008	00344	00082 00098 00098 00103 00113
UCBID	00008	00000078	00348	00083 00171

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLA00040
                                    4 *+----------------------------------------------------------------------------------------------+* SLA00050
                                    5 *|                                                                                              |* SLA00060
                                    6 *|Module:   SLAS                                                                                |* SLA00070
                                    7 *|                                                                                              |* SLA00080
                                    8 *|          Sleep List Address Space.                                                           |* SLA00090
                                    9 *|                                                                                              |* SLA00100
                                   10 *|Function: Perform enqueue and dequeue functions on the Sleep List.                            |* SLA00110
                                   11 *|                                                                                              |* SLA00120
                                   12 *|          The module maintains a pool of space organized into 10                           |* SLA00130
                                   13 *|          slots.  Each slot is large enough to contain a UCB.                              |* SLA00140
                                   14 *|          The Sleep List consists of UCBs allocated in this pool,                          |* SLA00150
                                   15 *|          chained together in a one-way list and sorted by ascend-                          |* SLA00160
                                   16 *|          ing wakeup time.  Space for a new list element is                                 |* SLA00170
                                   17 *|          acquired by stepping sequentially through the pool in                            |* SLA00180
                                   18 *|          slot-size increments until a free slot is located.                               |* SLA00190
                                   19 *|                                                                                              |* SLA00200
                                   20 *|          List members are chained by virtual address via the UCB                          |* SLA00210
                                   21 *|          forward pointer field.                                                           |* SLA00220
                                   22 *|                                                                                              |* SLA00230
                                   23 *|          See the Enqueue and Oequeue entry points for further                             |* SLA00240
                                   24 *|          details.                                                                         |* SLA00250
                                   25 *|                                                                                              |* SLA00260
                                   26 *|Caller:   Called exclusively by the Sleep Address Space.                                   |* SLA00270
                                   27 *|                                                                                              |* SLA00280
                                   28 *|Mode:     Problem state, OAT on, Key 5.                                                    |* SLA00290
                                   29 *|                                                                                              |* SLA00300
                                   30 *+----------------------------------------------------------------------------------------------+* SLA00310
                                   31 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLA00320

                                   33          CSECTION SLAS       Sleep List Address Space:                                        SLA00340

                                   35+         PUNCH 'SPB'             Make module start on a page boundry.                         CSE00040

000000                             37+SLAS     CSECT                    Establish R12 as module overall base:                       CSE00060

000000 05C0                        39+         BALR  R12,0              Give R12 a temporary base value;                            CSE00080
                         00002     40+         USING *,R12              R12 now points @ SL instruction                             CSE00090
000002 5FC0 C206         00208     41+         SL    R12,=A(*-SLAS)     R12 = R12 - offset from CSECT start                         CSE00100
                                   42+         DROP  R12                R12 doesn't point at SL anymore;                            CSE00110

                         00000     44+         USING SLAS,R12           Now R12 has A(This CSECT) in it.                            CSE00130

000006 47F0 C012         00012     46+         B     AROUND0            Jump past symbolic module identifier.                       CSE00150
00000A E203C1E240404040            47+MODID    DC    CL8'SLAS'          Handy when poking around in storage.                        CSE00160
000012                             48+AROUND   DS    0H                 End of CSECTION expansion.                                  CSE00170

                                   50          CRASH               Not an executable entry point...                               SLA00360
000012 005CC309C1E2C85C            51+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.                          CRA00030
```

```
  LOC  OBJECT CODE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                    ASM 0201 20.48 12/04/86
                                    53 *-------------------------------------------------------------------* SLA00380
                                    54 * A. ENQUEUE entry: Start things off by constructing a trace entry   * SLA00390
                                    55 *                   for the UCB about to be enqueued.                * SLA00400
                                    56 *-------------------------------------------------------------------* SLA00410

                                    58          ENTRYPT ENQUEUE,100 Enqueue entry point:                    SLA00430

00001A                       00100  60+         ORG   SLAS+X'100'                                           ENT00040

000100                              62+ENQUEUE  OS    0H            Oefine entry point name.                  ENT00060
                                    63+         PUSH  USING         Save overall USING environment.           ENT00070
000100 05C0                         64+         BALR  R12,0         Establish temporary base.                 ENT00080
                             00102  65+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000102 5FC0 C10A       0020C        66+         SL    R12,=A(*-SLAS)    R12 now points @ CSECT start.       ENT00100
                                    67+         POP   USING         Restore overall USING environment.        ENT00110

000106 021F C2A4 0000 002A4 00000   69          MVC   SAVEAREA,0(R0) Copy trace portion of parameter page     SLA00450

00010C 0207 0000 C2C8 00000 002C8   71          MVC   0(8,R0),=CL8'*ENQ*'   Move entryid to trace           SLA00470
000112 0207 0008 0078 00008 00078   72          MVC   8(8,R0),UCBIO-UCB(R0) Move name of guy we're enqueueing SLA00480
000118 0707 0010 0010 00010 00010   73          XC    16(8,R0),16(R0)       Clear unused portion of entry   SLA00490
                                    74          TRACE               Create trace entry.                     SLA00500
00011E 0207 0018 C00A 00018 0000A   75+         MVC   24(8,R0),MODIO Move symbolic module name to pp words 6-7 TRA00030
000124 5800 C2E0       002E0        76+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
000128 0A01                         77+         SVC   1                   Go to that address space.         CAL00050

00012A 021F 0000 C2A4 00000 002A4   79          MVC   0(L'SAVEAREA,R0),SAVEAREA Restore destroyed UCB for ENQ SLA00520

                                    81 *-------------------------------------------------------------------* SLA00540
                                    82 * B. Find a free slot for the new UCB.                             * SLA00550
                                    83 *-------------------------------------------------------------------* SLA00560

000130 4110 C300       00300        85          LA    R1,UCBAREA    Point @ UCB area.                       SLA00580

                             00000  87          USING UCB,R1        Access UCB OSECT                        SLA00600

000134                              89 FINOUCB  OS    0H            Search for free UCB:                    SLA00620

000134 0503 C288 1000 00288 00000   91          CLC   FREE,UCB      Is this one free?                       SLA00640
00013A 4780 C146       00146        92          BE    FOUNOUCB      Yes - This'll do.                       SLA00650
00013E 4110 1090       00090        93          LA    R1,SLOTSIZE(,R1) No  - Move to next,                  SLA00660
000142 47F0 C134       00134        94          B     FINOUCB       And check it out.                       SLA00670

                                    96 *-------------------------------------------------------------------* SLA00690
                                    97 * C. Move UCB from parameter page into newly-acquired Sleep List     * SLA00700
                                    98 *    slot, and set its forward pointer to null.                    * SLA00710
                                    99 *-------------------------------------------------------------------* SLA00720

000146                             101 FOUNOUCB OS    0H            Here with available UCB space @ R1:     SLA00740

000146 0284 1000 0000 00000 00000  103          MVC   UCB(UCBSIZE),0(R0) Install new UCB in table.         SLA00760

00014C 0203 1000 C28C 00000 0028C  105          MVC   UCBFWO,NULL   Give new UCB a null forward pointer.    SLA00780

                                   107          OROP  R1            But retain address in R1...             SLA00800
```

```
 LDC  08JECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.48 12/04/86
                                    109 *-------------------------------------------------------------------* SLA00820
                                    110 * D. Search for a point of insertion along the Sleep List.  This list * SLA00830
                                    111 *    is maintained in sorted order on the user's wakeup TOD in R0 and * SLA00840
                                    112 *    R1.  The list sort key is strictly increasing.                   * SLA00850
                                    113 *-------------------------------------------------------------------* SLA00860

000152 5850 C290       00290        115          L     R5,FIRST      0 if list = empty ( A(first UC8 on list) ) SLA00880
000156 1766                         116          XR    R6,R6         Clear pointer to previous.                 SLA00890

                              00000 118          USING UC8,R5                                                   SLA00910

000158                              120 SORT     DS    0H            Insert new UC8 sorted by TOD:              SLA00930

000158 1255                         122          LTR   R5,R5         Search over?                               SLA00950
00015A 47D0 C172       00172        123          8NP   INSERT        Yes - Either empty or End-of-list.         SLA00960

00015E D507 1008 5008 00008 00008   125          CLC   UC8GREGS+R0-UC8(8,R1),UC8GREGS+R0 No  - New:Member    SLA00980
000164 4740 C172       00172        126          8L    INSERT        New < Member; Insert new.                  SLA00990
000168 1865                         127          LR    R6,R5         New > Member; Save pointer to current.     SLA01000
00016A 5850 50D0       00000        128          L     R5,UC8FWD     Get pointer to next.                       SLA01010
00016E 47F0 C158       00158        129          8     SORT          Keep looking.                              SLA01020

                                    131 *-------------------------------------------------------------------* SLA01040
                                    132 * E. Insert the new UC8 into the Sleep List.  The insertion technique * SLA01050
                                    133 *    of course has multiple cases: 1) insertion into an empty list,   * SLA01060
                                    134 *    2) insertion at the end of a non-empty list, 3) insertion into   * SLA01070
                                    135 *    the middle of the list, and 4) a special case of 3 where the new * SLA01080
                                    136 *    UC8 becomes the new first UC8 on the list.  The list is singly-  * SLA01090
                                    137 *    linked through the forward pointer field.                        * SLA01100
                                    138 *-------------------------------------------------------------------* SLA01110

000172                              140 INSERT   DS    0H            Found point of insertion:                  SLA01130

000172 1255                         142          LTR   R5,R5         Beginning, end, or in between?             SLA01150
000174 4780 C18A       0018A        143          8Z    BEGOREND      0 = Beginning or end.                      SLA01160

000178 5050 1000       00000        145          ST    R5,UC8FWD-UC8(,R1) > 0 = Middle; Forward(new) <-- next.  SLA01180
00017C 1266                         146          LTR   R6,R6         Are we inserting a new first UC8?          SLA01190
00017E 4780 C190       00190        147          8Z    NEWFIRST      Yes - Then we have to update the anchor.   SLA01200
000182 5010 6000       00000        148          ST    R1,UC8FWD-UC8(,R6) No  - Forward(prev) <-- new.          SLA01210
000186 47F0 C19C       0019C        149          8     INSERTED      Done with insert.                          SLA01220

                                    151          DROP  R5                                                       SLA01240

00018A                              153 BEGOREND DS    0H            Here to determine if new first or last:    SLA01260

00018A 1266                         155          LTR   R6,R6         Is there a pointer to a previous UCB?      SLA01280
00018C 4770 C198       00198        156          8NZ   NEWLAST       Yes - Add new UC8 to the end of the list.  SLA01290

000190                              158 NEWFIRST DS    0H            Here to make anchor point at first UC8:    SLA01310

000190 5010 C290       00290        160          ST    R1,FIRST      Point @ first UC8 on sleep list.           SLA01330
000194 47F0 C19C       0019C        161          8     INSERTED      Done with insert.                          SLA01340

000198                              163 NEWLAST  DS    0H            Here for insertion @ end of list:          SLA01360
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.48 12/04/86
000198 5010 6000       00000         165           ST    R1,UCBFWD-UCB(,R6) Next(last) --> new last.          SLA01380

                                     167 *-------------------------------------------------------------------* SLA01400
                                     168 * F. Get the wakeup TOD for the first user on the Sleep List and send * SLA01410
                                     169 *    it back to the caller in parameter page words 0 & 1.             * SLA01420
                                     170 *-------------------------------------------------------------------* SLA01430

00019C                               172 INSERTED OS    0H            Here when UCB is linked:                SLA01450

00019C 5810 C290       00290         174           L     R1,FIRST      Point @ first user on Sleep List.       SLA01470

                              00000  176           USING UCB,R1        Map UCB references through R1.          SLA01490

0001A0 0207 0000 1008  00000 00008   178           MVC   0(8,R0),UCBGREGS+R0 Move his wakeup TOD to parm page.  SLA01510

                                     180           RETXMIT           Send TOD back to caller.                SLA01530
0001A6 0A04                          181+          SVC   4                                                   RET00030

                                     183           DROP  R1                                                  SLA01550
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.48 12/04/86
                                   185 *-------------------------------------------------------------------* SLA01570
                                   186 * A'. Dequeue entry.  Start by ensuring we are not trying to dequeue  * SLA01580
                                   187 *                     from an empty list.  ABEND if this is the case. * SLA01590
                                   188 *-------------------------------------------------------------------* SLA01600
                                   190          ENTRYPT DEQUEUE,200 Dequeue entry point:                     SLA01620
0001A8                      00200  192+         ORG   SLAS+X'200'                                            ENT00040
000200                             194+DEQUEUE  DS    0H            Define entry point name.                  ENT00060
                                   195+         PUSH  USING         Save overall USING environment.           ENT00070
000200 05C0                        196+         BALR  R12,0         Establish temporary base.                 ENT00080
                            00202  197+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000202 5FC0 C0E2       002E4       198+         SL    R12,=A(*-SLAS)    R12 now points @ CSECT start.       ENT00100
                                   199+         POP   USING         Restore overall USING environment.        ENT00110
000206 D403 C290 C290 00290 00290  201          NC    FIRST,FIRST   Is list empty?                           SLA01640
00020C 4770 C218       00218       202          BNZ   DEQ           No  - We're okay.                       SLA01650
                                   203          CRASH               Yes - Can't dequeue empty list.           SLA01660
000210 005CC309C1E2C85C            204+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.     CRA00030
                                   206 *-------------------------------------------------------------------* SLA01680
                                   207 * B'. Clear the parameter page.                                       * SLA01690
                                   208 *-------------------------------------------------------------------* SLA01700
000218                             210 DEQ      DS    0D            Start dequeue process:                    SLA01720
000218 9825 C294       00294       212          LM    R2,R5,CLEARPP Get MVCL clear parameters.                SLA01740
00021C 0E24                        213          MVCL  R2,R4         Clear the parameter page.                 SLA01750
                                   215 *-------------------------------------------------------------------* SLA01770
                                   216 * C'. Move the UCB at the head of the list to the parameter page.     * SLA01780
                                   217 *-------------------------------------------------------------------* SLA01790
00021E 5810 C290       00290       219          L     R1,FIRST      Get pointer to first UCB.                 SLA01810
                            00000  221          USING UCB,R1        Access DSECT fields.                      SLA01830
000222 D284 0000 1000 00000 00000  223          MVC   0(UCBSIZE,R0),UCB Move UCB to parm page.                SLA01850
                                   225 *-------------------------------------------------------------------* SLA01870
                                   226 * D'. Construct a trace entry showing which UCB we just dequeued.     * SLA01880
                                   227 *-------------------------------------------------------------------* SLA01890
000228 D21F C2A4 0000 002A4 00000  229          MVC   SAVEAREA,0(R0) Copy trace portion of parameter page     SLA01910
00022E D207 0000 C2D0 00000 002D0  231          MVC   0(8,R0),=CL8'*DEQ*'   Move entryid to trace            SLA01930
000234 D207 0008 0078 00008 00078  232          MVC   8(8,R0),UCBID-UCB(R0) Move name of guy we're dequeueing SLA01940
00023A D707 0010 0010 00010 00010  233          XC    16(8,R0),16(R0)       Clear unused portion of entry   SLA01950
                                   234          TRACE               Create trace entry.                       SLA01960
000240 D207 0018 C00A 00018 0000A  235+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
000246 5800 C2E0       002E0       236+         L     R0,=CL4'TRAC'       Load target address space identifier  CAL00040
00024A 0A01                        237+         SVC   1                   Go to that address space.           CAL00050
00024C D21F 0000 C2A4 00000 002A4  239          MVC   0(L'SAVEAREA,R0),SAVEAREA Restore destroyed DEQ'd UCB.  SLA01980
```

```
                                    241 *-------------------------------------------------------------------* SLA02000
                                    242 * E'. Adjust the Sleep List anchor to point at the next (new) head   * SLA02010
                                    243 *     UCB, if any.                                                  * SLA02020
                                    244 *-------------------------------------------------------------------* SLA02030

000252 D203 C290 1000 00290 00000   246          MVC   FIRST,UCBFWD  Update anchor with new first.           SLA02050

                                    248 *-------------------------------------------------------------------* SLA02070
                                    249 * F'. Free up the space held by the dequeued UCB.                   * SLA02080
                                    250 *-------------------------------------------------------------------* SLA02090

000258                              252 FRETUCB  DS    0H             Here to release UCB space:             SLA02110

000258 D78F 1000 1000 00000 00000   254          XC    UCB(SLOTSIZE),UCB Clear out the space it occupied.     SLA02130

00025E 0203 1000 C288 00D00 00288   256          MVC   UCBFWD,FREE   Release UCB space: Tag it with eye-catcher SLA02150

                                    258 *-------------------------------------------------------------------* SLA02170
                                    259 * G'. If there are any sleepers, the timer must be reset to wake up  * SLA02180
                                    260 *     the next sleeping user.  This is not done here.  Instead, we  * SLA02190
                                    261 *     return to the caller the clock comparator TOD for the new     * SLA02200
                                    262 *     head-of-list user (in UCB forward, backward pair).  He will   * SLA02210
                                    263 *     restart the timer.  If there are no more sleeping users, send * SLA02220
                                    264 *     back a zero clock comparator TOD.                             * SLA02230
                                    265 *-------------------------------------------------------------------* SLA02240

000264 D403 C290 C290 00290 00290   267          NC    FIRST,FIRST   Are there more sleepers?                SLA02260
00026A 4780 C27C      0027C         268          BZ    SENDZERO      No  - Return zero.                      SLA02270

00026E 5810 C290      00290         270          L     R1,FIRST      Yes - Restore pointer to first for USING. SLA02290
000272 D207 0000 1008 00000 000D8   271          MVC   0(8,R0),UCBGREGS+R0 Send back his clock comparator TOD. SLA02300
000278 47F0 C282      00282         272          B     DEQDONE       That's all.                             SLA02310

00027C                              274 SENDZERO DS    0H             No other sleepers in the system.       SLA02330

00027C D707 0000 0000 00000 00000   276          XC    0(8,R0),0(R0) Send back zeros in place of clk comparator SLA02350

                                    278 *-------------------------------------------------------------------* SLA02370
                                    279 * H'. The new Clock Comparator TOD is now in the parameter page along * SLA02380
                                    280 *     with the the dequeued UCB.  Return them both to the caller.   * SLA02390
                                    281 *-------------------------------------------------------------------* SLA02400

000282                              283 OEQDONE  DS    0H             Dequeue processing is complete.        SLA02420

                                    285          RETXMIT             Return to caller.                      SLA02440
000282 0A04                         286+         SVC   4                                                    RET00030
```

```
                                 288 *-------------------------------------------------------------------* SLA02460
                                 289 * Constants and variables:                                         * SLA02470
                                 290 *-------------------------------------------------------------------* SLA02480

000288                           292          DS    0D            Align constant area.                   SLA02500
000288 C6D9C5C5                  293 FREE     DC    CL4'FREE'     Eye-catcher indicating UCB space is free. SLA02510
00028C 00000000                  294 NULL     DC    F'0'          Null pointer.                          SLA02520
000290 00000000                  295 FIRST    DC    F'0'          Pointer to first UCB on sleep list.    SLA02530
000294 0000000000001000          296 CLEARPP  DC    F'0',F'4096',2F'0' Used by MVCL to clear parameter page. SLA02540
0002A4                           297 SAVEAREA DS    XL(8*4)       Holds piece of parm page while tracing. SLA02550
0002C8                           298          LTORG                                                      SLA02560
0002C8 5CC5D5D85C404040          299                =CL8'*ENQ*'
0002D0 5CC4C5D85C404040          300                =CL8'*DEQ*'
0002D8 00000002                  301                =A(*-SLAS)
0002DC 00000102                  302                =A(*-SLAS)
0002E0 E3D9C1C3                  303                =CL4'TRAC'
0002E4 00000202                  304                =A(*-SLAS)

                                 306 *-------------------------------------------------------------------* SLA02580
                                 307 * UC8 pool / Sleep List:                                           * SLA02590
                                 308 *-------------------------------------------------------------------* SLA02600

0002E8                   00300   310          ORG   SLAS+X'300'                                           SLA02620

                         00090   312 SLOTSIZE EQU   X'90'         Slightly larger than a UC8.            SLA02640

000300                           314 UC8AREA  OS    (10)XL(SLOTSIZE) Define enough space for 10 UC8s.     SLA02660
0008A0                   00300   315          ORG   UC8AREA                                               SLA02670
000300 C6D9C5C5                  316          DC    CL4'FREE'                                             SLA02680
000304                   00390   317          ORG   UC8AREA+SLOTSIZE                                      SLA02690
000390 C6D9C5C5                  318          DC    CL4'FREE'                                             SLA02700
000394                   00420   319          ORG   UCBAREA+(SLOTSIZE*2)                                  SLA02710
000420 C6D9C5C5                  320          DC    CL4'FREE'                                             SLA02720
000424                   00480   321          ORG   UC8AREA+(SLOTSIZE*3)                                 SLA02730
000480 C6D9C5C5                  322          DC    CL4'FREE'                                             SLA02740
000484                   00540   323          ORG   UC8AREA+(SLOTSIZE*4)                                 SLA02750
000540 C6D9C5C5                  324          DC    CL4'FREE'                                             SLA02760
000544                   005D0   325          ORG   UC8AREA+(SLOTSIZE*5)                                 SLA02770
0005D0 C6D9C5C5                  326          DC    CL4'FREE'                                             SLA02780
0005D4                   00660   327          ORG   UC8AREA+(SLOTSIZE*6)                                 SLA02790
000660 C6D9C5C5                  328          DC    CL4'FREE'                                             SLA02800
000664                   006F0   329          ORG   UC8AREA+(SLOTSIZE*7)                                 SLA02810
0006F0 C6D9C5C5                  330          DC    CL4'FREE'                                             SLA02820
0006F4                   00780   331          ORG   UC8AREA+(SLOTSIZE*8)                                 SLA02830
000780 C6D9C5C5                  332          DC    CL4'FREE'                                             SLA02840
000784                   00810   333          ORG   UC8AREA+(SLOTSIZE*9)                                 SLA02850
000810 C6D9C5C5                  334          DC    CL4'FREE'                                             SLA02860

                                 336          PRINT OFF                                                   SLA02880
```

SYMBOL LEN VALUE DEFN REFERENCES ASM 0201 20.48 12/04/86

```
AROUND   00002 00000012 00048  00046
BEGOREND 00002 0000018A 00153  00143
CLEARPP  00004 00000294 00296  00212
DEQ      00008 00000218 00210  00202
DEQDONE  00002 00000282 00283  00272
FINDUCB  00002 00000134 00089  00094
FIRST    00004 00000290 00295  00115 00160 00174 00201 00201 00219 00246 00267 00267 00270
FOUNDUCB 00002 00000146 00101  00092
FREE     00004 00000288 00293  00091 00256
INSERT   00002 00000172 00140  00123 00126
INSERTED 00002 0000019C 00172  00149 00161
MODID    00008 0000000A 00047  00075 00235
NEWFIRST 00002 00000190 00158  00147
NEWLAST  00002 00000198 00163  00156
NULL     00004 0000028C 00294  00105
R0       00001 00000000 00351  00069 00071 00072 00072 00073 00073 00075 00076 00079 00103 00125 00125 00178 00178 00223
                               00229 00231 00232 00232 00233 00233 00235 00236 00239 00271 00271 00276 00276
R1       00001 00000001 00352  00085 00087 00093 00093 00107 00125 00145 00148 00160 00165 00174 00176 00183 00219 00221
                               00270
R12      00001 0000000C 00363  00039 00040 00041 00042 00044 00064 00065 00066 00196 00197 00198
R2       00001 00000002 00353  00212 00213
R4       00001 00000004 00355  00213
R5       00001 00000005 00356  00115 00118 00122 00122 00127 00128 00142 00142 00145 00151 00212
R6       00001 00000006 00357  00116 00116 00127 00146 00146 00148 00155 00155 00165
SAVEAREA 00032 000002A4 00297  00069 00079 00079 00229 00239 00239
SENOZERO 00002 0000027C 00274  00268
SLAS     00001 00000000 00037  00044 00060 00192 00301 00302 00304 00310
SLOTSIZE 00001 00000090 00312  00093 00254 00314 00317 00319 00321 00323 00325 00327 00329 00331 00333
SORT     00002 00000158 00120  00129
UCB      00001 00000000 00446  00072 00087 00091 00103 00118 00125 00145 00148 00165 00176 00221 00223 00232 00254 00254
                               00460
UCBAREA  00144 00000300 00314  00085 00315 00317 00319 00321 00323 00325 00327 00329 00331 00333
UCBFWD   00004 00000000 00447  00105 00128 00145 00148 00165 00246 00256
UCBGREGS 00004 00000008 00449  00125 00125 00178 00271
UCBID    00008 00000078 00453  00072 00232
UCBSIZE  00001 00000085 00460  00103 00223
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
=CL8'*ENQ*'	00008	000002C8	00299	00071
=CL8'*DEQ*'	00008	000002D0	00300	00231
=A(*-SLAS)	00004	000002D8	00301	00041
=A(*-SLAS)	00004	000002DC	00302	00066
=CL4'TRAC'	00004	000002E0	00303	00076 00236
=A(*-SLAS)	00004	000002E4	00304	00198

```
 LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                   ASM 0201 22.08 11/22/86

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* CON00030
                                     4 *+--------------------------------------------------------------------+* CON00040
                                     5 *|                                                                    |* CON00050
                                     6 *|Module:   CONSOLE                                                   |* CON00060
                                     7 *|                                                                    |* CON00070
                                     8 *|          Console Address Space.                                    |* CON00080
                                     9 *|                                                                    |* CON00090
                                    10 *|Function: Handle user-issued console REAO and WRITE requests, and   |* CON00100
                                    11 *|          process I/O interrupts from the console.                  |* CON00110
                                    12 *|                                                                    |* CON00120
                                    13 *|          Provides access to the system console.  Each WRITE re-    |* CON00130
                                    14 *|          quest is queued.  If the display is not in use, the       |* CON00140
                                    15 *|          user's message is written to it, otherwise we exit to     |* CON00150
                                    16 *|          dispatcher.  The user's message text is in his parameter  |* CON00160
                                    17 *|          page.  He supplies its length in his RO.  The message     |* CON00170
                                    18 *|          text is copied to our parameter page through which all    |* CON00180
                                    19 *|          input/output operations take place.  The write is         |* CON00190
                                    20 *|          initiated and we exit to dispatcher.                      |* CON00200
                                    21 *|                                                                    |* CON00210
                                    22 *|          The only address space permitted to REAO is the system    |* CON00220
                                    23 *|          Operator.  When it issues a REAO request, we chain him    |* CON00230
                                    24 *|          into this address space as an active reader.  When an     |* CON00240
                                    25 *|          attention interrupt is received from the console we       |* CON00250
                                    26 *|          queue him into the console queue.  If there is an active  |* CON00260
                                    27 *|          writer, we queue the Operator immediately behind him.     |* CON00270
                                    28 *|          When the console is free we issue a REAO to it.  This     |* CON00280
                                    29 *|          module's parameter page is the input buffer.  When the    |* CON00290
                                    30 *|          read completes we copy our parameter page to the          |* CON00300
                                    31 *|          reader's parameter page thereby supplying him with the    |* CON00310
                                    32 *|          data.  Then we schedule Operator to run again, restart    |* CON00320
                                    33 *|          the console if necessary and exit to dispatcher.          |* CON00330
                                    34 *|                                                                    |* CON00340
                                    35 *|          The interrupt entry point ensures an interrupt is legal.  |* CON00350
                                    36 *|          If it signifies the end of a write operation we schedule  |* CON00360
                                    37 *|          the writer to run again, restart the console if there     |* CON00370
                                    38 *|          are queued requests and exit to dispatcher.  If the       |* CON00380
                                    39 *|          interrupt signified conclusion of a read operation, we    |* CON00390
                                    40 *|          act as described towards the end of the preceeding        |* CON00400
                                    41 *|          paragraph on REAO.                                        |* CON00410
                                    42 *|                                                                    |* CON00420
                                    43 *|          Consult the entry point in question for further details.  |* CON00430
                                    44 *|                                                                    |* CON00440
                                    45 *|Caller:   Invoked by the system on behalf of a user address space.  |* CON00450
                                    46 *|          Also called by I/O FLIH.                                  |* CON00460
                                    47 *|                                                                    |* CON00470
                                    48 *|          *Note: CONSOLE can be considered an I/O SLIH.             |* CON00480
                                    49 *|                                                                    |* CON00490
                                    50 *|Mode:     Problem state, OAT on, Key 3.                             |* CON00500
                                    51 *|                                                                    |* CON00510
                                    52 *+--------------------------------------------------------------------+* CON00520
                                    53 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* CON00530

                                    55          EXTRN CONSST                                                  CON00550
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86

                                     57           CSECTION CONSOLE    Console Address Space:                  CON00570

                                     59+          PUNCH 'SPB'              Make module start on a page boundry.  CSE00040

000000                               61+CONSOLE   CSECT                     Establish R12 as module overall base: CSE00060

000000 05C0                          63+          BALR  R12,0               Give R12 a temporary base value;      CSE00080
                              00002  64+          USING *,R12               R12 now points @ SL instruction       CSE00090
000002 5FC0 C55E      00560          65+          SL    R12,=A(*-CONSOLE)   R12 = R12 - offset from CSECT start  CSE00100
                                     66+          OROP  R12                 R12 doesn't point at SL anymore;      CSE00110

                              00000  68+          USING CONSOLE,R12         Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012          70+          B     AROUND              Jump past symbolic module identifier. CSE00150
00000A C3D6D5E2D6D3C540              71+MODID     DC    CL8'CONSOLE'        Handy when poking around in storage.  CSE00160
000012                               72+AROUND    DS    0H                  End of CSECTION expansion.             CSE00170

                                     74           CRASH               Non-executable entry point.            CON00590
000012 005CC3D9C1E2C85C              75+          DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.    CRA00030
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86
                                    77 *-------------------------------------------------------------------* CON00610
                                    78 * A. WRITE entry point.  Start by getting the writer's UCB from the  * CON00620
                                    79 *    run list and marking him as a writer (in his priority field).   * CON00630
                                    80 *-------------------------------------------------------------------* CON00640

                                    82          ENTRYPT WRITE,100   Entry point for user-issued WRITE SVC:    CON00660

00001A                        00100 84+         ORG   CONSOLE+X'100'                                         ENT00040

000100                              86+WRITE    OS    0H            Define entry point name.                  ENT00060
                                    87+         PUSH  USING         Save overall USING environment.           ENT00070
000100 05C0                         88+         BALR  R12,0         Establish temporary base.                 ENT00080
                              00102 89+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000102 5FC0 C462        00564       90+         SL    R12,=A(*-CONSOLE) R12 now points @ CSECT start.       ENT00100
                                    91+         POP   USING         Restore overall USING environment.        ENT00110

                                    93          CALL  RLDQ          Get UCB of writer.                        CON00680
000106 5800 C568        00568       94+         L     R0,=CL4'RLDQ'       Load target address space identifier CAL00040
00010A 0A00                         95+         SVC   0                   Go to that address space.           CAL00050

                              00000 97          USING UCB,R0        UCB instance now in parameter page.       CON00700

00010C D403 0008 0008 00008 00008   99          NC    UCBGREGS+R0(4),UCBGREGS+R0 Did user supply non-0 length? CON00720
000112 4770 C120        00120      100          BNZ   MOVECMD        Yes - Continue.                           CON00730
                                   101          CALLXMIT RLEQ       No  - Put user back on Run List.          CON00740
000116 5800 C56C        0056C      102+         L     R0,=CL4'RLEQ'       Load target address space identifier CAL00040
00011A 0A01                        103+         SVC   1                   Go to that address space.           CAL00050
00011C 47F0 C180        00180      104          B     WRITEXIT      Dummy out write processing for this user. CON00750

000120                             106 MOVECMD  DS    0H            Here to make UCB a writer:                CON00770

000120 9201 0084        00084      108          MVI   UCBPRI,X'01'  Move WRITE I/O command code to user prty  CON00790

                                   110 *-------------------------------------------------------------------* CON00810
                                   111 * B. Install writer's UCB in the UCB table.  FINDSLOT returns address * CON00820
                                   112 *    of installed UCB in R1.                                          * CON00830
                                   113 *-------------------------------------------------------------------* CON00840

000124 45E0 C462        00462      115          BAL   R14,FINDSLOT  Subroutine installs parameter page UCB.  CON00860

                                   117 *-------------------------------------------------------------------* CON00880
                                   118 * C. Construct a WRITE trace entry.                                  * CON00890
                                   119 *-------------------------------------------------------------------* CON00900

000128 0207 0000 C548 00000 00548  121          MVC   0(8,R0),=CL8'*WRITE*' Put entry name in trace entry.   CON00920
00012E D207 0008 0078 00008 00078  122          MVC   8(8,R0),UCBID   Put userid in trace entry.             CON00930
000134 D707 0010 0010 00010 00010  123          XC    16(8,R0),16(R0) Clear unused portion of trace entry.   CON00940
                                   124          TRACE               Create the trace entry.               CON00950
00013A D207 0018 C00A 00018 0000A  125+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
000140 5800 C570        00570      126+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
000144 0A01                        127+         SVC   1                   Go to that address space.           CAL00050

                                   129          DROP  R0            Disassociate R0 from UCB DSECT.       CON00970

                                   131 *-------------------------------------------------------------------* CON00990
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 22.08 11/22/86
                                    132 * D. Enqueue the new UCB on the Console List:                                  * CON01000
                                    133 *--------------------------------------------------------------------------* CON01010

000146 D403 C51C C51C 0051C 0051C   135          NC    FIRST,FIRST   Is there anybody on the list?                    CON01030
00014C 4770 C15C      0015C         136          BNZ   MUSTWAIT      Yes - Put this guy @ the end of the queue.    CON01040
000150 5010 C51C      0051C         137          ST    R1,FIRST      No  - The new UCB is the first on the Q       CON01050
000154 5010 C520      00520         138          ST    R1,LAST       He's also the last UCB on the queue.          CON01060
000158 47F0 C168      00168         139          B     INCOUNT       Enqueue basically complete.                   CON01070

00015C                              141 MUSTWAIT DS    0H            Here to put new UCB @ end of queue:           CON01090

00015C 5820 C520      00520         143          L     R2,LAST       Get pointer to last UCB on list.              CON01110
000160 5012 0000      00000         144          ST    R1,UCBFWD-UCB(R2) Forward(old last) <- A(new)               CON01120
000164 5010 C520      00520         145          ST    R1,LAST       LAST <- A(new)                                CON01130

                                    147 *--------------------------------------------------------------------------* CON01150
                                    148 * E. Increment the count of UCBs on the Console List.                       * CON01160
                                    149 *--------------------------------------------------------------------------* CON01170

000168                              151 INCOUNT  DS    0H            Here after enqueue has been performed:        CON01190

000168 5810 C518      00518         153          L     R1,COUNT      Get number of UCBs on Console List            CON01210
00016C 4120 0001      00001         154          LA    R2,1          Comparand and increment value                 CON01220
000170 1E12                         155          ALR   R1,R2         Increment count                               CON01230
000172 5010 C518      00518         156          ST    R1,COUNT      Update in storage.                            CON01240

                                    158 *--------------------------------------------------------------------------* CON01260
                                    159 * F. If the new guy is the only one on the queue, restart the idle    *    CON01270
                                    160 *    console.                                                         *    CON01280
                                    161 *--------------------------------------------------------------------------* CON01290

000176 1912                         163          CR    R1,R2         Is COUNT = 1?                                 CON01310
000178 4770 C180      00180         164          BNE   WRITEXIT      No  - He'll have to wait for his turn.        CON01320
00017C 45E0 C486      00486         165          BAL   R14,RESTART   Yes - Restart the idle console.              CON01330

000180                              167 WRITEXIT DS    0H            Here when WRITE processing is complete:       CON01350

                                    169          EXIT  DSP           Go to dispatcher.                             CON01370
000180 5800 C574      00574         170+         L     R0,=CL4'DSP'      Load target address space identifier    EXI00040
000184 0A02                         171+         SVC   2                 Go to that address space.               EXI00050
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86
                                     173 *-------------------------------------------------------------------* CON01390
                                     174 * A'. READ entry point.  Begin by getting the reader's UCB from the  * CON01400
                                     175 *     run list and marking him as a reader (in his priority field).  * CON01410
                                     176 *-------------------------------------------------------------------* CON01420

                                     178          ENTRYPT READ,200    Queue a console read request for Operator: CON0144D

000186                         00200 180+         ORG   CONSOLE+X'200'                                        ENT00040

000200                               182+READ     DS    0H            Define entry point name.                ENT00060
                                     183+         PUSH  USING         Save overall USING environment.         ENT00070
000200 05C0                          184+         BALR  R12,0         Establish temporary base.               ENT00080
                               00202 185+         USING *,R12         R12 now points @ next SL instruction:   ENT00090
000202 5FC0 C376       0D578         186+         SL    R12,=A(*-CONSOLE) R12 now points @ CSECT start.       ENT00100
                                     187+         POP   USING         Restore overall USING environment.      ENT00110

                                     189          CALL  RLDQ          Get reader's UCB.                       CON01460
000206 5800 C568       00568         190+         L     R0,=CL4'RLDQ'       Load target address space identifier CAL00040
00020A 0A00                          191+         SVC   0                   Go to that address space.         CAL00050

                               00000 193          USING UCB,R0        Parameter page contains a UCB instance. CON01480

00020C 92D2 0084       0D084         195          MVI   UCBPRI,X'D2'  Move READ I/O command code to priority.  CON01500

                                     197 *-------------------------------------------------------------------* CON01520
                                     198 * B'. Install reader's UCB in the UCB table. FINDSLOT returns address * CON01530
                                     199 *     of installed UCB in R1.                                       * CON01540
                                     200 *-------------------------------------------------------------------* CON01550

000210 45E0 C462       00462         202          BAL   R14,FINDSLOT  Subroutine installs parameter page UCB.  CON01570

                                     204 *-------------------------------------------------------------------* CON01590
                                     205 * C'. Construct a READ trace entry.                                 * CON01600
                                     206 *-------------------------------------------------------------------* CON01610

000214 D207 0000 C550 00000 00550    208          MVC   0(8,R0),=CL8'*READ*' Move entry name to trace entry.   CON01630
00021A D207 0008 0078 D00D8 00078    209          MVC   8(8,R0),UCBID       Move userid to trace entry.       CON01640
000220 D707 0010 0010 00010 00010    210          XC    16(8,R0),16(R0)     Clear remainder of entry.         CON01650
                                     211          TRACE                     Create trace entry.               CON01660
000226 D207 0018 C0DA 00018 0000A    212+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00022C 5800 C570       00570         213+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
000230 0A01                          214+         SVC   1                   Go to that address space.         CAL00050

                                     216 *-------------------------------------------------------------------* CON01680
                                     217 * D'. Set ACTIVE READER pointer to the newly-installed UCB and exit.  * CON01690
                                     218 *-------------------------------------------------------------------* CON01700

000232 5010 C524       00524         220          ST    R1,ACTIVRDR   We now have an active reader.           CON01720
                                     221          EXIT OSP            Go to dispatcher.                     CON01730
000236 5800 C574       00574         222+         L     R0,=CL4'DSP'        Load target address space identifier EXI00040
00023A 0A02                          223+         SVC   2                   Go to that address space.         EXI00050

                                     225          DROP  R0            Disassociate R0 from UCB DSECT.       CON01750
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86
                                    227 *-------------------------------------------------------------------* CON01770
                                    228 * A". IOINT entry point.  Start by verifying the interrupt is in fact * CON01780
                                    229 *     from the console and that the status associated with this event * CON01790
                                    23D *     is "clean", i.e. not indicative of any outboard errors.       * CON01800
                                    231 *-------------------------------------------------------------------* CON01810

                                    233          ENTRYPT IOINT,300   Console I/O interrupt entry point:      CON01830

00023C                      00300   235+         ORG   CONSOLE+X'300'                                        ENT00040

000300                              237+IOINT    DS    0H            Define entry point name.                ENT00060
                                    238+         PUSH  USING         Save overall USING environment.         ENT00070
000300 05C0                         239+         BALR  R12,0         Establish temporary base.               ENT00080
                            00302   240+         USING *,R12         R12 now points @ next SL instruction:   ENT00090
000302 5FC0 C27A      0057C         241+         SL    R12,=A(*-CONSOLE) R12 now points @ CSECT start.       ENT00100
                                    242+         POP   USING         Restore overall USING environment.      ENT00110

                            00000   244          USING UCB,R1        For the duration of the source file.    CON01850

000306 9509 0FF5      00FF5         246          CLI   4085(R0),X'09' Is the interrupt from the console?     CON01870
00030A 4780 C316      00316         247          BE    CNSIOINT      Yes - Now examine status.               CON01880
                                    248          CRASH               Interrupt from alien device; shut down. CON01890
00030E 005CC309C1E2C85C             249+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.   CRA00030

000316                              251 CNSIOINT DS    0H            Here to examine Device Status:          CON01910

000316 9173 0FFC      00FFC         253          TM    4092(R0),B'01110011' Bad Unit Status?                 CON01930
00031A 4780 C326      00326         254          BZ    CKCHSTAT      No  - Check for channel problems.       CON01940
                                    255          CRASH               Yes - Severe console error(s). Shut dwn. CON01950
00031E 005CC3D9C1E2C85C             256+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.   CRA00030

000326                              258 CKCHSTAT DS    0H            Here to examine Channel Status:         CON01970

000326 9500 0FFD      00FFD         260          CLI   4093(R0),0    Any sign of trouble?                    CON01990
00032A 4780 C336      00336         261          BZ    CLEAN         No  - Clean status.                     CON02000
                                    262          CRASH               Yes - Severe I/O error(s). Shut down.   CON02010
00032E 005CC3D9C1E2C85C             263+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.   CRA00030

                                    265 *-------------------------------------------------------------------* CON02030
                                    266 * B". If ending status in the CSW indicates ATTENTION from the con-   * CON02040
                                    267 *     sole, we may have to perform a read operation.  If we have an   * CON02050
                                    268 *     active reader, initiate read processing.  Otherwise, purge the  * CON02060
                                    269 *     pending console input, as we don't have a reader for it.        * CON02070
                                    270 *-------------------------------------------------------------------* CON02080

000336                              272 CLEAN    DS    0H            Here on receipt of valid console interrupt CON02100

000336 9580 0FFC      00FFC         274          CLI   4092(R0),X'80' Is this an ATTENTION interrupt?       CON02120
00033A 4770 C39E      0039E         275          BNE   CHKCEDE       No  - Maybe it was ending status.       CON02130
00033E D403 C524 C524 00524 00524   276          NC    ACTIVRDR,ACTIVRDR Yes - Is there an active reader?  CON02140
000344 4770 C350      00350         277          BNZ   DOREAD        Yes - Perform a read operation.         CON02150

000348 AF00 000A      0000A         279          MC    PURGECON,0    No  - Purge useless pending console input. CON02170

00034C 47F0 C45C      0045C         281          B     INTDONE       Exit to dispatcher.                     CON02190
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86
                                    283 *-------------------------------------------------------------------* CON02210
                                    284 * C". We have an active reader ready for console input.  If the      * CON02220
                                    285 *     Console List is empty, move the reader to the List, start the  * CON02230
                                    286 *     read operation and exit to the dispatcher.  Otherwise queue the * CON02240
                                    287 *     reader immediately behind the first UCB on the list and exit to * CON02250
                                    288 *     the dispatcher.                                                 * CON02260
                                    289 *-------------------------------------------------------------------* CON02270

000350                              291 OOREAO   OS    OH            Here on ATTENTION interrupt w/active rdr:  CON02290

000350 5810 C524      00524         293          L     R1,ACTIVROR   Get pointer to active reader.             CON02310
000354 0703 C524 C524 00524 00524   294          XC    ACTIVROR,ACTIVROR Reset active reader pointer in storage CON02320

00035A 5820 C518      00518         296          L     R2,COUNT      Get # of people in Console List           CON02340
00035E 4130 0001      00001         297          LA    R3,1          Comparand and increment value             CON02350
000362 1E23                         298          ALR   R2,R3         Increment count of people in Console List. CON02360
000364 5020 C518      00518         299          ST    R2,COUNT      Update count in storage.                  CON02370

000368 1923                         301          CR    R2,R3         Is he the only I/O user in the system?    CON02390
00036A 4770 C37E      0037E         302          BNE   Q2NO          No  - Queue him behind the leader.        CON02400
00036E 5010 C51C      0051C         303          ST    R1,FIRST      Yes - Make him the first on Console List  CON02410
000372 5010 C520      00520         304          ST    R1,LAST       He's also the last on the list.           CON02420
000376 45E0 C486      00486         305          BAL   R14,RESTART   Go perform the console read operation.    CON02430
00037A 47F0 C45C      0045C         306          B     INTOONE       That's the end of read processing.        CON02440

00037E                              308 Q2NO     OS    OH            Here to queue reader second in list:      CON02460

00037E 5850 C51C      0051C         310          L     R5,FIRST      Get pointer to first UCB on list          CON02480
000382 0203 1000 5000 00000 00000   311          MVC   UCBFWO,0(R5)  Make new 2nd point at original 2nd       CON02490
000388 5015 0000      00000         312          ST    R1,UCBFWO-UCB(R5) Make first point at new               CON02500
00038C 0503 C51C C520 0051C 00520   313          CLC   FIRST,LAST    Was there originally only one UCB on list? CON02510
000392 4770 C45C      0045C         314          BNE   INTOONE       No  - Then pointer to last doesn't change. CON02520
000396 5010 C520      00520         315          ST    R1,LAST       Yes - Then new UCB is now last on the list CON02530
00039A 47F0 C45C      0045C         316          B     INTOONE       Leave with new UCB enqueued.              CON02540

                                    318 *-------------------------------------------------------------------* CON02560
                                    319 * O". We are here when the CSW status was other than ATTENTION.  The * CON02570
                                    320 *     status is either one of: Channel End, Oevice End, or both pre- * CON02580
                                    321 *     sented together.  A CE, OE pair indicates the conclusion of a  * CON02590
                                    322 *     write operation.  CE indicates partial conclusion of a read op- * CON02600
                                    323 *     eration and supplies (albeit indirectly) the number of bytes    * CON02610
                                    324 *     read.  It will be followed shortly by a solitary OE interrupt   * CON02620
                                    325 *     indicating completion of the read.  CE, OE come back together   * CON02630
                                    326 *     concurrently for a write because the write CCW is command-      * CON02640
                                    327 *     chained to a NOP.  For a read operation, the read channel pro-  * CON02650
                                    328 *     gram consists of a singular read CCW which terminates slowly by * CON02660
                                    329 *     component, each component announcing its completion by a        * CON02670
                                    330 *     distinct I/O interrupt presenting the ending status of the op-  * CON02680
                                    331 *     eration within that component.                                  * CON02690
                                    332 *-------------------------------------------------------------------* CON02700

00039E                              334 CHKCEOE  OS    OH            Here when interrupt was not ATTENTION:    CON02720

00039E 950C OFFC      00FFC         336          CLI   4092(R0),X'0C' Is it channel end, device end combo?    CON02740
0003A2 4780 C3B8      003B8         337          BE    OEQUCB          Yes - Write I/O operation has completed. CON02750
```

```
  LOC  OBJECT CODE    AODR1 AODR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86

0003A6 9508 0FFC      00FFC          338          CLI   4092(R0),X'08' No  - Channel end for a read?            CON02760
0003AA 4770 C3B8      003B8          339          BNE   DEQUCB         No  - DE means read I/O operation done.  CON02770
0003AE D201 C540 0FFE 00540 00FFE    340          MVC   RESID,4094(R0) Yes - Residual count comes back with CE. CON02780
000384 47F0 C45C      0045C          341          B     INTDONE        Go to dispatcher and wait for solitary DE. CON02790

                                     343 *-------------------------------------------------------------------* CON02810
                                     344 * E". When we are here we know an I/O operation has completed. If the * CON02820
                                     345 *     operation was a write, continue at step F".  For conclusion   * CON02830
                                     346 *     of a read, compute the number of bytes read and save it in the * CON02840
                                     347 *     reader's R0.  Then copy our parameter page to the reader's to  * CON02850
                                     348 *     transmit the input data to his address space.                 * CON02860
                                     349 *-------------------------------------------------------------------* CON02870

0003B8                               351 DEQUCB   OS    0H             Here after CE, DE to DEQ active I/O UCB:   CON02890

0003B8 D403 C518 C518 00518 00518    353          NC    COUNT,COUNT    Is the Console List empty??                CON02910
0003BE 4770 C3CA      003CA          354          BNZ   WEREOKAY       No  - Good.                               CON02920
                                     355          CRASH                How did that happen???                     CON02930
0003C2 005CC3D9C1E2C85C              356+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.      CRA00030

0003CA                               358 WEREOKAY OS    0H             Here when we know I/O is not unsolicited:  CON02950

0003CA 5810 C51C      0051C          360          L     R1,FIRST       Get pointer to first UCB on Console List.  CON02970

0003CE 9502 1084      00084          362          CLI   UCBPRI,X'02'   Is this the conclusion of a read operation CON02990
0003D2 4770 C3F2      003F2          363          BNE   IOEN0          No  - I/O operation was a write.           CON03000
000306 5820 C53C      0053C          364          L     R2,F132        Yes - Get buffer size (size of parm page)  CON03010
0003DA 4830 C540      00540          365          LH    R3,RESID       Get residual count from CE CSW             CON03020
0003DE 1F23                          366          SLR   R2,R3          Buffer size - residual count = bytes read. CON03030
0003E0 5020 1008      00008          367          ST    R2,UCBGREGS+R0 Place bytes read count in user's R0.       CON03040

0003E4 5820 1080      00080          369          L     R2,UC8STO      Get user's STO                             CON03060
0003E8 5840 C528      00528          370          L     R4,OURSTO      Get Console Address Space STO              CON03070
                                     371          CALL  XMIT           Copy our parameter page to his, thereby    CON03080
0003EC 5800 C580      00580          372+         L     R0,=CL4'XMIT'        Load target address space identifier CAL00040
0003F0 0A00                          373+         SVC   0                    Go to that address space.          CAL00050
                                     374 *                               passing the input read data to him.  CON03090

                                     376 *-------------------------------------------------------------------* CON03110
                                     377 * E". Now we are ready to dequeue the active UCB from the head of the * CON03120
                                     378 *     Console List.  But first, trace the interrupt.  Then, copy the * CON03130
                                     379 *     UCB to the parameter page and finally carefully unchain the UCB.* CON03140
                                     380 *-------------------------------------------------------------------* CON03150

0003F2                               382 IOEND    OS    0H             Here to dequeue UCB for completed I/O op:  CON03170

0003F2 0207 0000 C558 00000 00558    384          MVC   0(8,R0),=CL8'*IOINT*' Move entry name to trace entry.  CON03190
0003F8 020B 0008 0FF4 00008 00FF4    385          MVC   8(12,R0),4084(R0) Move device address to trace entry. CON03200
0003FE D703 0014 0014 00014 00014    386          XC    20(4,R0),20(R0) Clear unused portion of trace entry.   CON03210
                                     387          TRACE                Create trace entry of this interrupt.  CON03220
000404 D207 0018 C00A 00018 0000A    388+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00040A 5800 C570      00570          389+         L     R0,=CL4'TRAC'        Load target address space identifier CAL00040
00040E 0A01                          390+         SVC   1                    Go to that address space.          CAL00050

000410 D284 0000 1000 00000 00000    392          MVC   0(UCBSIZE,R0),UCB Copy UCB to parameter page              CON03240
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86
000416 5800 C518       00518          393           L     RO,COUNT      Get Console List length                  CON03250
00041A 0600                           394           BCTR  RO,0          Decrement                                CON03260
00041C 5000 C518       00518          395           ST    RO,COUNT      Update count in storage.                 CON03270
000420 1200                           396           LTR   RO,RO         Was he the only UCB on the list?         CON03280
000422 4780 C430       00430          397           BZ    ONLYONE       Yes - Dequeue him correctly.             CON03290
000426 D203 C51C 1000 0051C 00000     398           MVC   FIRST,UCBFWD  No  - FIRST <- A(Next UCB)             CON03300
00042C 47F0 C43C       0043C          399           B     DEQDONE       He has now been removed from the list.   CON03310

000430                                401 ONLYONE   DS    0H            Here when dequeuing the only UCB on list: CON03330

000430 D703 C51C C51C 0051C 0051C     403           XC    FIRST,FIRST   Zero first pointer,                     CON03350
000436 D703 C520 C520 00520 00520     404           XC    LAST,LAST     Zero last pointer.                       CON03360

                                      406 *-------------------------------------------------------------------* CON03380
                                      407 * F". Release the space held by the UCB in the Console List, and then * CON03390
                                      408 *     call the scheduler to make this UCB runnable again.  The call   * CON03400
                                      409 *     to scheduler passes the UCB image in the parameter page.        * CON03410
                                      410 *-------------------------------------------------------------------* CON03420

00043C                                412 DEQDONE   DS    0H            Here when UCB has been dequeued:         CON03440

00043C D78F 1000 1000 00000 00000     414           XC    UCB(SLOTSIZE),UCB Clear out storage held by UCB.       CON03460
000442 D203 1000 C514 00000 00514     415           MVC   UCBFWD,FREE   Mark the slot free.                      CON03470

                                      417           CALLXMIT SIOD       Schedule the user to run.                CON03490
000448 5800 C584       00584          418+          L     RO,=CL4'SIOD'       Load target address space identifier CAL00040
00044C 0A01                           419+          SVC   1                   Go to that address space.        CAL00050

                                      421 *-------------------------------------------------------------------* CON03510
                                      422 * G". If there are more queued console requests, start the next.     * CON03520
                                      423 *     Then exit to dispatcher.                                        * CON03530
                                      424 *-------------------------------------------------------------------* CON03540

00044E D403 C518 C518 00518 00518     426           NC    COUNT,COUNT   Are there queued requests?              CON03560
000454 4780 C45C       0045C          427           BZ    INTDONE       No  - That's a wrap.                     CON03570
000458 45E0 C486       00486          428           BAL   R14,RESTART   Yes - Restart console for next q'd request CON03580

00045C                                430 INTDONE   DS    0H            Here when interrupt processing is complete CON03600

                                      432           EXIT DSP            Go to dispatcher.                        CON03620
00045C 5800 C574       00574          433+          L     RO,=CL4'DSP'        Load target address space identifier EXI00040
000460 0A02                           434+          SVC   2                   Go to that address space.        EXI00050
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86

                                      436 *-------------------------------------------------------------------* CON03640
                                      437 * Subroutine: FINDSLOT                                               * CON03650
                                      438 *                                                                   * CON03660
                                      439 * Called by:  WRITE and READ entry points.                          * CON03670
                                      440 *                                                                   * CON03680
                                      441 * Function:   Search the UCB table for a free slot.  Move the param- * CON03690
                                      442 *             eter page UCB instance to the slot.  Then, set the UCB  * CON03700
                                      443 *             forward pointer to null and return to he caller.      * CON03710
                                      444 *-------------------------------------------------------------------* CON03720

000462                                446 FINDSLOT DS    OH            Called to locate free UCB slot:           CON03740

000462 4110 C600       00600          448          LA    R1,UCBAREA    Point at UCB slot pool.                   CON03760
000466                                449 FINDUCB  DS    OH            Search for free UCB:                      CON03770

000466 D503 C514 1000 00514 00000     451          CLC   FREE,UCB      Is this one free?                         CON03790
00046C 4780 C478       00478          452          BE    FOUNDUCB      Yes - This'll do.                         CON03800
000470 4110 1090       00090          453          LA    R1,SLOTSIZE(,R1) No  - Move to next,                    CON03810
000474 47F0 C466       00466          454          B     FINDUCB       And check it out.                         CON03820

000478                                456 FOUNDUCB DS    OH            Here when free UCB slot has been located: CON03840

000478 D284 1000 0000 00000 00000     458          MVC   UCB(UCBSIZE),0(R0) Install new UCB in table.          CON03860
00047E D203 1000 C510 00000 00510     459          MVC   UCBFWD,NULL   Give new node a null forward pointer.    CON03870

000484 07FE                           461          BR    R14           Return to caller with slot address in R1. CON03890
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86

                                   463 *-------------------------------------------------------------------* CON03910
                                   464 * Subroutine: RESTART                                               * CON03920
                                   465 *                                                                   * CON03930
                                   466 * Called by:  WRITE and IOINT entry points.                         * CON03940
                                   467 *                                                                   * CON03950
                                   468 * Function:   Start an I/O operation to the console on behalf of the * CON03960
                                   469 *             first user on the Console List.  The return to the    * CON03970
                                   470 *             caller.                                               * CON03980
                                   471 *                                                                   * CON03990
                                   472 *             A read operation is simple.  Merely clear the parameter * CON04000
                                   473 *             page which doubles as a read buffer and start the read  * CON04010
                                   474 *             channel program in PGMFLIH.                           * CON04020
                                   475 *                                                                   * CON04030
                                   476 *             A write operation is described below.                 * CON04040
                                   477 *-------------------------------------------------------------------* CON04050

000486                             479 RESTART  DS    0H            Called to locate free UCB slot:          CON04070

000486 5810 C51C      0051C        481          L     R1,FIRST      Get pointer to UCB for whom we will do I/O CON04090

00048A 9502 1084      00084        483          CLI   UCBPRI,X'02'  Is this guy a reader?                    CON04110
00048E 4770 C4A0      004A0        484          BNE   WRITER       No  - Process a writer.                   CON04120
000492 9825 C52C      0052C        485          LM    R2,R5,CLRPP   Yes - Get MVCL clear parameters.         CON04130
000496 0E24                        486          MVCL  R2,R4         Clear parameter page (READ input buffer). CON04140
000498 5850 C53C      0053C        487          L     R5,F132       Get READ buffer size                      CON04150
00049C 47F0 C4E0      004E0        488          B     GETIOCMD      Proceed on towards SIO...                 CON04160

                                   490 *-------------------------------------------------------------------* CON04180
                                   491 * For a writer, copy his parameter page to ours.  This gives us a    * CON04190
                                   492 * copy of the message to be written in our parameter page.  Slide the * CON04200
                                   493 * user's message ten spaces to the right in our parameter page and   * CON04210
                                   494 * insert his userid as a prefix to the message in the first ten pos- * CON04220
                                   495 * itions of the text.  Then start the write I/O operation.          * CON04230
                                   496 *-------------------------------------------------------------------* CON04240

0004A0                             498 WRITER   DS    0H            Here to set up for WRITE I/O operation:  CON04260

0004A0 5820 C528      00528        500          L     R2,OURSTO     Get Console Address Space STO            CON04280
0004A4 5840 1080      00080        501          L     R4,UCBSTO     Get User's STO                           CON04290
                                   502          CALL  XMIT          Move user's write data to our parm page. CON04300
0004A8 5800 C580      00580        503+         L     R0,=CL4'XMIT'       Load target address space identifier CAL00040
0004AC 0A00                        504+         SVC   0                   Go to that address space.          CAL00050

0004AE 5830 1008      00008        506          L     R3,UCBGREGS+R0 Get user-supplied message length from R0. CON04320
0004B2 4120 0046      00046        507          LA    R2,70         Get maximum length of a message          CON04330
0004B6 1932                        508          CR    R3,R2         Is user's message length <= 70 bytes?    CON04340
0004B8 47D0 C4BE      004BE        509          BNH   LENGTHOK      Yes - Use user-supplied length.          CON04350
0004BC 1832                        510          LR    R3,R2         No  - Write only first 70 bytes of message CON04360

0004BE                             512 LENGTHOK DS    0H            Here with length of message in R3:       CON04380

0004BE 1853                        514          LR    R5,R3         Get length of message in R5              CON04400
0004C0 1823                        515          LR    R2,R3         Get length of message in R2              CON04410
0004C2 0620                        516          BCTR  R2,0          Subtract 1                               CON04420
```

```
0004C4                                  518 SHIFT    DS    OH             Slide message 10 spaces to the right:      CON04440

0004C4 D200 3009 2000 00009 00000       520          MVC   9(1,R3),0(R2) Move one byte at a time.                   CON04460
0004CA 0620                             521          BCTR  R2,0          Back up source to the previous byte of msg CON04470
0004CC 4630 C4C4      004C4             522          BCT   R3,SHIFT      Back up destination by one byte.           CON04480

0004D0 D207 0000 1078 00000 00078       524          MVC   0(8,R0),UCBIO Move userid to message buffer               CON04500
0004D6 D201 0008 C588 00008 00588       525          MVC   8(2,R0),=C': ' Move field separator to message buffer    CON04510

0004DC 4155 000A      0000A             527          LA    R5,10(R5)     R5 = Length of user's message + 10.        CON04530

0004E0                                  529 GETIOCMD OS    OH             Here with message length in R5:            CON04550

0004E0 4300 1084      00084             531          IC    R0,UCBPRI     Get I/O operation command code in R0       CON04570
0004E4 1815                             532          LR    R1,R5         R1 <- msg length per CONSIO convention.    CON04580

0004E6 AF00 0009      00009             534          MC    CONSIO,0      Do Start I/O...                            CON04600

0004EA 07FE                             536          BR    R14           Return to caller.                          CON04620
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 22.08 11/22/86

                                    538 *-------------------------------------------------------------------* CON04640
                                    539 * Module constants and data areas:                                  * CON04650
                                    540 *-------------------------------------------------------------------* CON04660
0004F0                              541          DS    0D             Align constant area.                  CON04670
0004F0                              542 SAVEAREA DS    XL(8*4)        8 word savearea.                      CON04680
000510 FFFFFFFF                     543 NULL     DC    F'-1'          Null UCB forward pointer.             CON04690
000514 C609C5C5                     544 FREE     DC    CL4'FREE'      Comparand for free UCB slot search.   CON04700
000518 00000000                     545 COUNT    DC    F'0'           Number of UC8s installed in the table. CON04710
00051C 00000000                     546 FIRST    DC    F'0'           Pointer to first queued UC8.          CON04720
000520 00000000                     547 LAST     DC    F'0'           Pointer to last  queued UC8.          CON04730
000524 00000000                     548 ACTIVRDR DC    F'0'           Pointer to active reader UCB.         CON04740
000528 00000000                     549 OURSTO   DC    A(CONSST)      Console Address Space STO.            CON04750
00052C 00000000                     550 CLRPP    DC    F'0'           Used by MVCL to clear parameter page. CON04760
000530 00001000                     551 F4096    DC    F'4096'        USed by MVCL parameter page clear length. CON04770
000534 0000000000000000             552          DC    2F'0'          Used by MVCL parameter page clear.    CON04780
00053C 00000084                     553 F132     DC    F'132'         Smaller read buffer size.             CON04790
000540                              554 RESID    DS    H              Holds CSW residual data count from read. CON04800
000548                              555          LTORG                                                      CON04810
000548 5CE609C9E3C55C40             556                =CL8'*WRITE*'
000550 5C09C5C1C45C4040             557                =CL8'*READ*'
000558 5CC9D6C9D5E35C40             558                =CL8'*IOINT*'
000560 00000002                     559                =A(*-CONSOLE)
000564 00000102                     560                =A(*-CONSOLE)
000568 D9D3C4D8                     561                =CL4'RLDQ'
00056C D9D3C5D8                     562                =CL4'RLEQ'
000570 E3D9C1C3                     563                =CL4'TRAC'
000574 C4E2D740                     564                =CL4'DSP'
000578 00000202                     565                =A(*-CONSOLE)
00057C 00000302                     566                =A(*-CONSOLE)
000580 E7D4C9E3                     567                =CL4'XMIT'
000584 E2C9D6C4                     568                =CL4'SIOD'
000588 7A40                         569                =C': '
                                    570 *-------------------------------------------------------------------* CON04820
                                    571 * UCB pool / Console List:                                         * CON04830
                                    572 *-------------------------------------------------------------------* CON04840

00058A                    00600     574          ORG   CONSOLE+X'600'                                      CON04860

                          00090     576 SLOTSIZE EQU   X'90'           Slightly larger than a UCB.         CON04880

000600                              578 UC8AREA  DS    (10)XL(SLOTSIZE) Define enough space for 10 UCBs.    CON04900
000BA0                    00600     579          ORG   UCBAREA                                             CON04910
000600 C6D9C5C5                     580          DC    CL4'FREE'                                           CON04920
000604                    00690     581          ORG   UCBAREA+SLOTSIZE                                    CON04930
000690 C6D9C5C5                     582          DC    CL4'FREE'                                           CON04940
000694                    00720     583          ORG   UCBAREA+(SLOTSIZE*2)                              CON04950
000720 C6D9C5C5                     584          DC    CL4'FREE'                                           CON04960
000724                    007B0     585          ORG   UC8AREA+(SLOTSIZE*3)                              CON04970
0007B0 C6D9C5C5                     586          DC    CL4'FREE'                                           CON04980
0007B4                    00840     587          ORG   UCBAREA+(SLOTSIZE*4)                              CON04990
000840 C6D9C5C5                     588          DC    CL4'FREE'                                           CON05000
000844                    008D0     589          ORG   UCBAREA+(SLOTSIZE*5)                              CON05010
0008D0 C6D9C5C5                     590          DC    CL4'FREE'                                           CON05020
0008D4                    00960     591          ORG   UCBAREA+(SLOTSIZE*6)                              CON05030
000960 C6D9C5C5                     592          DC    CL4'FREE'                                           CON05040
```

```
  LOC  OBJECT CODE     ADDR1 AODR2  STMT   SOURCE STATEMENT                    ASM 0201 22.08 11/22/86
000964                       009F0   593          ORG   UCBAREA+(SLOTSIZE*7)                    CON05050
0009F0 C6D9C5C5                      594          DC    CL4'FREE'               CON05060
0009F4                       00A80   595          ORG   UCBAREA+(SLOTSIZE*8)                    CON05070
000A80 C6D9C5C5                      596          DC    CL4'FREE'               CON05080
000A84                       00B10   597          ORG   UCBAREA+(SLOTSIZE*9)                    CON05090
000B10 C609C5C5                      598          DC    CL4'FREE'               CON05100

                                     600          PRINT OFF                     CON05120
```

POS.ID	REL.ID	FLAGS	ADDRESS
0002	0001	0C	000528

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ACTIVRDR	00004	00000524	00548	00220 00276 00276 00293 00294 00294
AROUND	00002	00000012	00072	00070
CHKCEDE	00002	0000039E	00334	00275
CKCHSTAT	00002	00000326	00258	00254
CLEAN	00002	00000336	00272	00261
CLRPP	00004	0000052C	00550	00485
CNSIOINT	00002	00000316	00251	00247
CONSIO	00001	00000009	00656	00534
CONSOLE	00001	00000000	00061	00068 00084 00180 00235 00559 00560 00565 00566 00574
CONSST	00001	00000000	00055	00549
COUNT	00004	00000518	00545	00153 00156 00296 00299 00353 00353 00393 00395 00426 00426
DEQDONE	00002	0000043C	00412	00399
DEQUCB	00002	000003B8	00351	00337 00339
DOREAD	00002	00000350	00291	00277
FINDSLOT	00002	00000462	00446	00115 00202
FINDUCB	00002	00000466	00449	00454
FIRST	00004	0000051C	00546	00135 00135 00137 00303 00310 00313 00360 00398 00403 00403 00481
FOUNDUCB	00002	00000478	00456	00452
FREE	00004	00000514	00544	00415 00451
F132	00004	0000053C	00553	00364 00487
GETIOCMD	00002	000004E0	00529	00488
INCOUNT	00002	00000168	00151	00139
INTDONE	00002	0000045C	00430	00281 00306 00314 00316 00341 00427
IOEND	00002	000003F2	00382	00363
LAST	00004	00000520	00547	00138 00143 00145 00304 00313 00315 00404 00404
LENGTHOK	00002	000004BE	00512	00509
MODID	00008	0000000A	00071	00125 00212 00388
MOVECMD	00002	00000120	00106	00100
MUSTWAIT	00002	0000015C	00141	00136
NULL	00004	00000510	00543	00459
ONLYONE	00002	00000430	00401	00397
OURSTO	00004	00000528	00549	00370 00500
PURGECON	00001	0000000A	00657	00279
Q2ND	00002	0000037E	00308	00302
RESID	00002	00000540	00554	00340 00365
RESTART	00002	00000486	00479	00165 00305 00428
R0	00001	00000000	00615	00094 00097 00099 00099 00102 00121 00122 00123 00123 00125 00126 00129 00170 00190 00193 00208 00209 00210 00210 00212 00213 00222 00225 00246 00253 00260 00274 00336 00338 00340 00367 00372 00384 00385 00385 00386 00386 00388 00389 00392 00393 00394 00395 00396 00396 00418 00433 00458 00503 00506 00524 00525 00531
R1	00001	00000001	00616	00137 00138 00144 00145 00153 00155 00156 00163 00220 00244 00293 00303 00304 00312 00315 00360 00448 00453 00453 00481 00532
R12	00001	0000000C	00627	00063 00064 00065 00066 00068 00088 00089 00090 00184 00185 00186 00239 00240 00241
R14	00001	0000000E	00629	00115 00165 00202 00305 00428 00461 00536
R2	00001	00000002	00617	00143 00144 00154 00155 00163 00296 00298 00299 00301 00364 00366 00367 00369 00485 00486 00500 00507 00508 00510 00515 00516 00520 00521
R3	00001	00000003	00618	00297 00298 00301 00365 00366 00506 00508 00510 00514 00515 00520 00522
R4	00001	00000004	00619	00370 00486 00501
R5	00001	00000005	00620	00310 00311 00312 00485 00487 00514 00527 00527 00532
SHIFT	00002	000004C4	00518	00522
SLOTSIZE	00001	00000090	00576	00414 00453 00578 00581 00583 00585 00587 00589 00591 00593 00595 00597
UCB	00001	00000000	00710	00097 00144 00193 00244 00312 00392 00414 00414 00451 00458 00724
UCBAREA	00144	00000600	00578	00448 00579 00581 00583 00585 00587 00589 00591 00593 00595 00597
UCBFWD	00004	00000000	00711	00144 00311 00312 00398 00415 00459
UCBGREGS	00004	00000008	00713	00099 00099 00367 00506

SYMBOL	LEN	VALUE	DEFN	REFERENCES
UCBID	00008	00000078	00717	00122 00209 00524
UCBPRI	00001	00000084	00719	00108 00195 00362 00483 00531
UCBSIZE	00001	00000085	00724	00392 00458
UCBSTO	00004	00000080	00718	00369 00501
WEREOKAY	00002	000003CA	00358	00354
WRITER	00002	000004A0	00498	00484
WRITEXIT	00002	00000180	00167	00104 00164

```
  LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                                ASM 0201 21.09 11/18/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SCH00030
                                      4 *+---------------------------------------------------------------------+* SCH00040
                                      5 *|                                                                     |* SCH00050
                                      6 *|Module:   SCH                                                        |* SCH00060
                                      7 *|                                                                     |* SCH00070
                                      8 *|          Scheduler.                                                 |* SCH00080
                                      9 *|                                                                     |* SCH00090
                                     10 *|Function: Schedule the execution of a user address space based on    |* SCH00100
                                     11 *|          a history of his execution characteristics.                |* SCH00110
                                     12 *|                                                                     |* SCH00120
                                     13 *|          The Scheduler has a unique entry point for each type of    |* SCH00130
                                     14 *|          user.  For example, the Time Slice End entry point         |* SCH00140
                                     15 *|          is invoked for a user who has consumed his alloted dis-    |* SCH00150
                                     16 *|          patch quantum.  This could be indicative of a compute-     |* SCH00160
                                     17 *|          bound task, and should be treated as such i.e. less        |* SCH00170
                                     18 *|          favorably than other user classes (e.g. I/O bound or       |* SCH00180
                                     19 *|          interactive).  Other entry points are described below;     |* SCH00190
                                     20 *|          for further details consult the entry point in question.  |* SCH00200
                                     21 *|                                                                     |* SCH00210
                                     22 *|Caller:   Called by several modules, refer to the SCH                |* SCH00220
                                     23 *|          authorization list in AUTH.                                |* SCH00230
                                     24 *|                                                                     |* SCH00240
                                     25 *|Mode:     Problem state, OAT on, Key 5.                              |* SCH00250
                                     26 *|                                                                     |* SCH00260
                                     27 *+---------------------------------------------------------------------+* SCH00270
                                     28 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SCH00280

                                     30          CSECTION SCH       Scheduler:                                 SCH00300

                                     32+         PUNCH 'SPB'            Make module start on a page boundry.  CSE00040

000000                               34+SCH      CSECT                   Establish R12 as module overall base: CSE00060

000000 05C0                          36+         8ALR  R12,0             Give R12 a temporary base value;      CSE00080
                           00002     37+         USING *,R12             R12 now points @ SL instruction       CSE00090
000002 5FC0 C376      00378          38+         SL    R12,=A(*-SCH)     R12 = R12 - offset from CSECT start   CSE00100
                                     39+         OROP  R12               R12 doesn't point at SL anymore;      CSE00110

                           00000     41+         USING SCH,R12           Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012          43+         B     AROUNO            Jump past symbolic module identifier. CSE00150
00000A E2C3C84040404040              44+M0010    OC    CL8'SCH'          Handy when poking around in storage.  CSE00160
000012                               45+AROUNO   OS    OH                End of CSECTION expansion.            CSE00170

                                     47          CRASH              Main entry point is non-exectuable...      SCH00320
000012 005CC309C1E2C85C              48+         OC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.     CRA00030
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 21.09 11/18/86
                                     50 *-------------------------------------------------------------------* SCH00340
                                     51 * A. Time Slice End entry point.                                     * SCH00350
                                     52 *                                                                    * SCH00360
                                     53 *      -Trace the invocation of this entry point.                    * SCH00370
                                     54 *      -Get the UCB of the guy who consumed all his time.  He is the * SCH00380
                                     55 *        first user on the run list.                                 * SCH00390
                                     56 *      -Set his UCB priority to indicate he consumed his quantum.    * SCH00400
                                     57 *      -Give him a full quantum for the next round.                  * SCH00410
                                     58 *      -Call Run List Address Space ENQUEUE entry point to place this * SCH00420
                                     59 *        UCB back on the run list.                                   * SCH00430
                                     60 *      -Go to dispatcher.                                            * SCH00440
                                     61 *-------------------------------------------------------------------* SCH00450

                                     63          ENTRYPT TSEN0,100   Time Slice End entry point:             SCH00470

00001A                        00100  65+         ORG   SCH+X'100'                                            ENT00040

000100                               67+TSEN0    OS    0H            Define entry point name.                ENT00060
                                     68+         PUSH  USING         Save overall USING environment.         ENT00070
000100 05C0                          69+         BALR  R12,0         Establish temporary base.               ENT00080
                              00102  70+         USING *,R12         R12 now points @ next SL instruction:   ENT00090
000102 5FC0 C27A        0037C        71+         SL    R12,=A(*-SCH)     R12 now points @ CSECT start.     ENT00100
                                     72+         POP   USING         Restore overall USING environment.      ENT00110

                                     74          CALL  RLOQ          Dequeue active UCB from the runlist.    SCH00490
000106 5800 C380        00380        75+         L     R0,=CL4'RLOQ'       Load target address space identifier CAL00040
00010A 0A00                          76+         SVC   0                   Go to that address space.         CAL00050

                              00000  78          USING UCB,R0        Activate the UCB DSECT.                 SCH00510

00010C D21F C338 0000 00338 00000    80          MVC   SAVEAREA,UCB  Copy trace portion of parameter page.   SCH00530

000112 0207 0000 C360 00000 00360    82          MVC   0(8,R0),=CL8'*TSEN0*' Put entry name in trace entry.  SCH00550
000118 0207 0008 0078 00008 00078    83          MVC   8(8,R0),UCBID Put symbolic user name in trace entry.  SCH00560
                                     84          TRACE               Trace invocation of this entry point.   SCH00570
00011E D207 0018 C00A 00018 0000A    85+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
000124 5800 C384        00384        86+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
000128 0A01                          87+         SVC   1                   Go to that address space.         CAL00050

00012A 021F 0000 C338 00000 00338    89          MVC   UCB(L'SAVEAREA),SAVEAREA Restore UCB destroyed by trace. SCH00590

000130 9220 0084        00084        91          MVI   UCBPRI,UCBPRITE Indicate time slice end for runlist enq. SCH00610
000134 0207 0070 C358 00070 00358    92          MVC   UCBTIMER,NEWSLICE Replenish user's dispatch quantum.   SCH00620

                                     94          CALLXMIT RLEQ       Re-queue runnable UCB to dispatch list. SCH00640
00013A 5800 C388        00388        95+         L     R0,=CL4'RLEQ'       Load target address space identifier CAL00040
00013E 0A01                          96+         SVC   1                   Go to that address space.         CAL00050

                                     98          EXIT OSP            Go to dispatcher.                       SCH00660
000140 5800 C38C        0038C        99+         L     R0,=CL4'OSP'        Load target address space identifier EXI00040
000144 0A02                         100+         SVC   2                   Go to that address space.         EXI00050
```

```
                                    102 *-------------------------------------------------------------------* SCH00680
                                    103 * B. Rouse entry point.                                            * SCH00690
                                    104 *                                                                   * SCH00700
                                    105 *      * Passed a UCB dequeued from the Sleep List.               * SCH00710
                                    106 *                                                                   * SCH00720
                                    107 *      -Trace the invocation of this entry point.                 * SCH00730
                                    108 *      -Set his UCB priority to indicate he is waking up.         * SCH00740
                                    109 *      -Give him a full time slice for his next dispatch.         * SCH00750
                                    110 *      -Add UCB to run list.                                      * SCH00760
                                    111 *      -Return to caller.                                         * SCH00770
                                    112 *-------------------------------------------------------------------* SCH00780

                                    114          ENTRYPT ROUSE,200   Rouse sleeping user entry point:       SCH00800

000146                       00200  116+         ORG   SCH+X'200'                                          ENT00040

000200                              118+ROUSE    DS    0H            Define entry point name.               ENT00060
                                    119+         PUSH  USING         Save overall USING environment.        ENT00070
000200 05C0                         120+         BALR  R12,0         Establish temporary base.              ENT00080
                             00202  121+         USING *,R12         R12 now points @ next SL instruction:  ENT00090
000202 5FC0 C18E       00390        122+         SL    R12,=A(*-SCH)     R12 now points @ CSECT start.    ENT00100
                                    123+         POP   USING         Restore overall USING environment.     ENT00110

000206 D21F C338 0000 00338 00000   125          MVC   SAVEAREA,UCB    Copy UCB data because trace destroys it. SCH00820

00020C D207 0000 C368 00000 00368   127          MVC   0(8,R0),=CL8'*ROUSE*' Move entry point name to trc entry SCH00840
000212 D207 0008 0078 00008 00078   128          MVC   8(8,R0),UCBID Move user's name to trace entry.       SCH00850
                                    129          TRACE               Trace invocation of this entry point. SCH00860
000218 D207 0018 C00A 00018 0000A   130+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00021E 5800 C384       00384        131+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
000222 0A01                         132+         SVC   1                   Go to that address space.        CAL00050

000224 D21F 0000 C338 00000 00338   134          MVC   UCB(L'SAVEAREA),SAVEAREA Restore partially-destroyed UCB SCH00880

00022A 9280 0084       00084        136          MVI   UCBPRI,UCBPRIWK   Indicate user is waking up.       SCH00900
00022E D207 0070 C358 00070 00358   137          MVC   UCBTIMER,NEWSLICE Replenish user's dispatch quantum. SCH00910

                                    139          CALLXMIT RLEQ       Add runnable UCB to dispatch list.     SCH00930
000234 5800 C388       00388        140+         L     R0,=CL4'RLEQ'       Load target address space identifier CAL00040
000238 0A01                         141+         SVC   1                   Go to that address space.        CAL00050

                                    143          RETURN              Return to caller.                      SCH00950
00023A 0A03                         144+         SVC   3                                                    RET00030
```

```
                                        146 *-------------------------------------------------------------------* SCH00970
                                        147 * C. IODONE entry point.                                            * SCH00980
                                        148 *                                                                   * SCH00990
                                        149 *       * Passed a UCB dequeued from the Console Wait List.         * SCH01000
                                        150 *                                                                   * SCH01010
                                        151 *       -Trace the invocation of this entry point.                  * SCH01020
                                        152 *       -Set his UCB priority to indicate he is coming back from an * SCH01030
                                        153 *        I/O operation.                                             * SCH01040
                                        154 *       -Add UCB to run list.                                       * SCH01050
                                        155 *       -Return to caller.                                          * SCH01060
                                        156 *-------------------------------------------------------------------* SCH01070

                                        158           ENTRYPT IODONE,300  Console I/O completed entry point:    SCH01090

00023C                          00300   160+          ORG   SCH+X'300'                                          ENT00040

000300                                  162+IODONE    OS    0H            Define entry point name.            ENT00060
                                        163+          PUSH  USING         Save overall USING environment.     ENT00070
000300 05C0                             164+          BALR  R12,0         Establish temporary base.           ENT00080
                                00302   165+          USING *,R12         R12 now points @ next SL instruction: ENT00090
000302 5FC0 C092        00394           166+          SL    R12,=A(*-SCH)     R12 now points @ CSECT start.   ENT00100
                                        167+          POP   USING         Restore overall USING environment.  ENT00110

000306 D21F C338 0000 00338 00000       169           MVC   SAVEAREA,UCB    Copy UCB data because trace destroys it. SCH01110

00030C D207 0000 C370 00000 00370       171           MVC   0(8,R0),=CL8'*IODONE*' Mov entry point name to trc entry SCH01130
000312 D207 0008 0078 00008 00078       172           MVC   8(8,R0),UCBID Move user's name to trace entry.      SCH01140
                                        173           TRACE               Trace invocation of this entry point. SCH01150
000318 D207 0018 C00A 00018 0000A       174+          MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00031E 5800 C384        00384           175+          L     R0,=CL4'TRAC'      Load target address space identifier CAL00040
000322 0A01                             176+          SVC   1                  Go to that address space.        CAL00050

000324 D21F 0000 C338 00000 00338       178           MVC   UCB(L'SAVEAREA),SAVEAREA Restore partially-destroyed UCB SCH01170

00032A 9240 0084        00084           180           MVI   UCBPRI,UCBPRIIO  Indicate user is coming back from I/O. SCH01190

                                        182           CALLXMIT RLEQ       Add runnable UCB to dispatch list.  SCH01210
00032E 5800 C388        00388           183+          L     R0,=CL4'RLEQ'      Load target address space identifier CAL00040
000332 0A01                             184+          SVC   1                  Go to that address space.        CAL00050

                                        186           RETURN              Return to caller.                   SCH01230
000334 0A03                             187+          SVC   3                                                  RET00030
```

```
000338                                 189          DS    0D            Force alignment for constants.      SCH01250
000338                                 190 SAVEAREA DS    XL(8*4)       Holds part of UCB while tracing.    SCH01260
000358 0000000100000000                191 NEWSLICE DC    XL8'0000000100000000' Dispatch quantum.           SCH01270
000360                                 192          LTORG                                                   SCH01280
000360 5CE3E2C5D5C45C40                193                =CL8'*TSEND*'
000368 5CD9D6E4E2C55C40                194                =CL8'*ROUSE*'
000370 5CC9D6C4D6D5C55C                195                =CL8'*IODONE*'
000378 00000002                        196                =A(*-SCH)
00037C 00000102                        197                =A(*-SCH)
000380 D9D3C4D8                        198                =CL4'RLDQ'
000384 E3D9C1C3                        199                =CL4'TRAC'
000388 D9D3C5D8                        200                =CL4'RLEQ'
00038C C4E2D740                        201                =CL4'DSP'
000390 00000202                        202                =A(*-SCH)
000394 00000302                        203                =A(*-SCH)
                                       204          PRINT OFF                                               SCH01290
```

ASM 0201 21.09 11/18/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00045	00043
MODID	00008	0000000A	00044	00085 00130 00174
NEWSLICE	00008	00000358	00191	00092 00137
R0	00001	00000000	00219	00075 00078 00082 00083 00085 00086 00095 00099 00127 00128 00130 00131 00140 00171 00172 00174 00175 00183
R12	00001	0000000C	00231	00036 00037 00038 00039 00041 00069 00070 00071 00120 00121 00122 00164 00165 00166
SAVEAREA	00032	00000338	00190	00080 00089 00089 00125 00134 00134 00169 00178 00178
SCH	00001	00000000	00034	00041 00065 00116 00160 00196 00197 00202 00203
UCB	00001	00000000	00312	00078 00080 00089 00125 00134 00169 00178 00326
UCBID	00008	00000078	00319	00083 00128 00172
UCBPRI	00001	00000084	00321	00091 00136 00180
UCBPRIIO	00001	00000040	00324	00180
UCBPRITE	00001	00000020	00325	00091
UCBPRIWK	00001	00000080	00323	00136
UCBTIMER	00008	00000070	00318	00092 00137

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* DSP00030
                                    4 *+------------------------------------------------------------------------+* DSP00040
                                    5 *|                                                                        |* DSP00050
                                    6 *|Module:   DSP                                                           |* DSP00060
                                    7 *|                                                                        |* DSP00070
                                    8 *|          Dispatcher.                                                   |* DSP00080
                                    9 *|                                                                        |* DSP00090
                                   10 *|Function: Dispatch a user address space.                                |* DSP00100
                                   11 *|                                                                        |* DSP00110
                                   12 *|          DSP dispatches any runnable user or places the system         |* DSP00120
                                   13 *|          in an idle enabled wait state if there is no runnable         |* DSP00130
                                   14 *|          work.  It requests an image of the next runnable UCB          |* DSP00140
                                   15 *|          on the run list.  If there is none, the system has            |* DSP00150
                                   16 *|          nothing to do.  Otherwise, the dispatcher extracts key        |* DSP00160
                                   17 *|          fields frmom the UCB and sets low-core dispatch para-         |* DSP00170
                                   18 *|          meters from them.  Finally, it restores the user's            |* DSP00180
                                   19 *|          floating point and general registers and issues a             |* DSP00190
                                   20 *|          DISPATCH Monitor Call to have PGMFLIH perform the final       |* DSP00200
                                   21 *|          steps of dispatch processing.                                 |* DSP00210
                                   22 *|                                                                        |* DSP00220
                                   23 *|Caller:   DSP has many callers.  Consult its authorization list in      |* DSP00230
                                   24 *|          AUTH.                                                         |* DSP00240
                                   25 *|                                                                        |* DSP00250
                                   26 *|Mode:     Problem state, DAT on, Key 5.                                 |* DSP00260
                                   27 *|                                                                        |* DSP00270
                                   28 *+------------------------------------------------------------------------+* DSP00280
                                   29 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* DSP00290

                                   31          CSECTION DSP        Dispatcher.                                  DSP00310

                                   33+         PUNCH 'SPB'             Make module start on a page boundry.     CSE00040

000000                             35+DSP      CSECT                    Establish R12 as module overall base:   CSE00060

000000 05C0                        37+         BALR  R12,0              Give R12 a temporary base value;        CSE00080
                          00002    38+         USING *,R12              R12 now points @ SL instruction         CSE00090
000002 5FC0 C0B6  000B8            39+         SL    R12,=A(*-DSP)      R12 = R12 - offset from CSECT start     CSE00100
                                   40+         DROP  R12                R12 doesn't point at SL anymore;        CSE00110

                          00000    42+         USING DSP,R12            Now R12 has A(This CSECT) in it.        CSE00130

000006 47F0 C012  00012            44+         B     AROUND             Jump past symbolic module identifier.   CSE00150
00000A C4E2D74040404040            45+MODID    DC    CL8'DSP'           Handy when poking around in storage.    CSE00160
000012                             46+AROUND   DS    0H                 End of CSECTION expansion.               CSE00170

                                   48          CALL RLGD           Get next dispatchable user from run list.   DSP00330
000012 5800 C0BC  000BC            49+         L     R0,=CL4'RLGD'      Load target address space identifier    CAL00040
000016 0A00                        50+         SVC   0                  Go to that address space.               CAL00050

                          00000    52          USING UCB,R0        Address UCB image in parameter page.        DSP00350

000018 D503 C070 0000 00070 00000  54          CLC   NOWORK,UCB    Did Run List address space have a UCB?      DSP00370
00001E 4770 C032  00032            55          BNE   PREPARE       Yes - Start dispatching it.                 DSP00380
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 13.44 11/17/86

                                      56          TRACE               No  - Trace no work available condition.   DSP00390
000022 D207 0018 C00A 00018 0000A     57+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
000028 5800 C0C0      000C0           58+         L     R0,=CL4'TRAC'      Load target address space identifier  CAL00040
00002C 0A01                           59+         SVC   1                  Go to that address space.             CAL00050

00002E AF00 0007      00007           61          MC    IDLE,0         Enter no-work idle wait state.            DSP00410

000032                                63 PREPARE  DS    0H             Here with dispatchable candidate UCB:     DSP00430

000032 5820 0080      00080           65          L     R2,UCBSTO      Get user's STO.                           DSP00450
000036 9836 0068      00068           66          LM    R3,R6,UCBPSW   Get user's PSW (R3,4) and quantum (R5,6). DSP00460
00003A AF00 0001      00001           67          MC    SDP,0          Set Dispatch Parameters in low core.      DSP00470

00003E D23F C074 0008 00074 00008     69          MVC   USERREGS,UCBGREGS Save reg image destroyed by tracing:    DSP00490
000044 9026 0000      00000           70          STM   R2,R6,0(R0)    Construct trace entry for user dispatch.  DSP00500
                                      71          TRACE               Insert trace entry into table.            DSP00510
000048 D207 0018 C00A 00018 0000A     72+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00004E 5800 C0C0      000C0           73+         L     R0,=CL4'TRAC'      Load target address space identifier  CAL00040
000052 0A01                           74+         SVC   1                  Go to that address space.             CAL00050

000054 7800 0048      00048           76          LE    F0,UCBFREGS    Restore                                   DSP00530
000058 7820 0050      00050           77          LE    F2,UCBFREGS+8    user's                                  DSP00540
00005C 7840 0058      00058           78          LE    F4,UCBFREGS+16     floating point                        DSP00550
000060 7860 0060      00060           79          LE    F6,UCBFREGS+24       registers.                          DSP00560

000064 980F C074      00074           81          LM    R0,R15,USERREGS Restore user's general-purpose registers DSP00580

000068 AF00 0002      00002           83          MC    DISPATCH,0     Signal DISPATCH.                          DSP00600

000070                                85          DS    0D             Align constant area.                      DSP00620
000070 D5D6D5C5                       86 NOWORK   DC    CL4'NONE'      Indicates no dispatchable UCBs.           DSP00630
000074                                87 USERREGS DS    XL(16*4)       Temporary storage for user's general regs. DSP00640
0000B8                                88          LTORG                                                        DSP00650
0000B8 00000002                       89                =A(*-DSP)
0000BC D9D3C7C4                       90                =CL4'RLGD'
0000C0 E3D9C1C3                       91                =CL4'TRAC'
                                      92          PRINT OFF                                                    DSP00660
```

ASM 0201 13.44 11/17/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00046	00044
DISPATCH	00001	00000002	00141	00083
OSP	00001	00000000	00035	00042 00089
F0	00001	00000000	00127	00076
F2	00001	00000002	00128	00077
F4	00001	00000004	00129	00078
F6	00001	00000006	00130	00079
IOLC	00001	00000007	00146	00061
MODIO	00008	0000000A	00045	00057 00072
NOWORK	00004	00000070	00086	00054
PREPARE	00002	00000032	00063	00055
R0	00001	00000000	00107	00049 00052 00057 00058 00070 00072 00073 00081
R12	00001	0000000C	00119	00037 00038 00039 00040 00042
R15	00001	0000000F	00122	00081
R2	00001	00000002	00109	00065 00070
R3	00001	00000003	00110	00066
R6	00001	00000006	00113	00066 00070
SOP	00001	00000001	00140	00067
UCB	00001	00000000	00199	00052 00054 00212
UCBFREGS	00004	00000048	00203	00076 00077 00078 00079
UCBGREGS	00004	00000008	00202	00069
UCBPSW	00008	00000068	00204	00066
UCBSTO	00004	00000080	00207	00065
USERREGS	00064	00000074	00087	00069 00081

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                              ASM 0201 14.58 11/23/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* STA00030
                                      4 *+-------------------------------------------------------------------------+* STA00040
                                      5 *|                                                                          |* STA00050
                                      6 *|Module:    STARTUP                                                        |* STA00060
                                      7 *|                                                                          |* STA00070
                                      8 *|           System Startup code.                                           |* STA00080
                                      9 *|                                                                          |* STA00090
                                     10 *|Function:  Initializes data needed to get the system running.             |* STA00100
                                     11 *|                                                                          |* STA00110
                                     12 *|Caller:    Not called.  Receives control from the loader via LDT          |* STA00120
                                     13 *|           card.                                                          |* STA00130
                                     14 *|                                                                          |* STA00140
                                     15 *|Mode:      Supervisor state, DAT off, Key 0.                              |* STA00150
                                     16 *|                                                                          |* STA00160
                                     17 *+-------------------------------------------------------------------------+* STA00170
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* STA00180

                                     20          EXTRN SVCFLIH,EXTFLIH,PGMFLIH,DATABLES,AUTH,USER1PP,TLAS         STA00200
                                     21          EXTRN EXTSPP,AUTHPP,IOFLIH,CONSPP,CONSCCW,CONSOLE                STA00210

                                     23          CSECTION STARTUP    System startup code:                        STA00230

                                     25+         PUNCH 'SPB'               Make module start on a page boundry.  CSE00040

000000                               27+STARTUP  CSECT                     Establish R12 as module overall base: CSE00060

000000 05C0                          29+         8ALR  R12,0               Give R12 a temporary base value;      CSE00080
                            00002    30+         USING *,R12               R12 now points @ SL instruction       CSE00090
0000D2 5FC0 C13E       00140         31+         SL    R12,=A(*-STARTUP)   R12 = R12 - offset from CSECT start   CSE00100
                                     32+         DRDP  R12                 R12 doesn't point at SL anymore;      CSE00110

                            00000    34+         USING STARTUP,R12         Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012       00012         36+         B     AROUND              Jump past symbolic module identifier. CSE00150
000D0A E2E3C1D9E3E4D740              37+MODID    DC    CL8'STARTUP'        Handy when poking around in storage.  CSE00160
000012                               38+ARDUND   DS    0H                  End of CSECTION expansion.            CSE00170

                            00000    40          USING PSA,R0        Address low core fields.                    STA00250

000012 8200 C0C8       000C8         42          LPSW  ECMODE        First thing to do is turn on EC mode.       STA00270

000016                               44 NEXTINST DS    0H            Preceeding LPSW vectors us here.            STA00290

000016 9220 0468       00468         46          MVI   SYSTATE,RUNIDLE System is initially idle.                 STA00310

00001A D207 0470 C0D0 00470 000D0    48          MVC   SYSPSW,SSYSPSW Initialize system Addr Space dispatch PSW STA00330

000020 D207 0060 C0D8 00060 000D8    50          MVC   SVCNEW,SSVCNEW Initialize                                 STA00350
000026 D207 0058 C0E0 00058 000E0    51          MVC   EXTNEW,SEXTNEW   interrupt                                STA00360
00D02C D207 0078 C0E8 00078 000E8    52          MVC   IONEW,SIONEW       new                                    STA00370
000032 D207 0068 C0F0 00068 000F0    53          MVC   PGMNEW,SPGMNEW       PSWs.                                STA00380

000038 B700 C110       00110         55          LCTL  0,0,CREG0     Set                                         STA00400
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 14.58 11/23/86

00003C B722 C114      00114          56          LCTL  2,2,CREG2       control                                  STA00410
000040 B788 C118      00118          57          LCTL  8,8,CREG8         registers 0, 2 & 8.                      STA00420

                                     59 *-------------------------------------------------------------------* STA00440
                                     60 * Set special storage keys for individually protected pages of the   * STA00450
                                     61 * system.                                                            * STA00460
                                     62 *-------------------------------------------------------------------* STA00470

000044 5810 C144      00144          64          L     R1,=A(DATABLES) DAT Tables are key 0.                    STA00490
000048 1744                          65          XR    R4,R4           Zero key register                        STA00500
00004A 0841                          66          SSK   R4,R1           Set key                                  STA00510
00004C 5810 C148      00148          67          L     R1,=A(STARTUP) Startup code is key 0.                    STA00520
000050 0841                          68          SSK   R4,R1           Set key                                  STA00530

000052 5810 C14C      0014C          70          L     R1,=A(SVCFLIH) SVC FLIH is key 1.                        STA00550
000056 4340 C13C      0013C          71          IC    R4,KEY1         Get key                                  STA00560
00005A 0841                          72          SSK   R4,R1           Set key                                  STA00570
00005C 5810 C150      00150          73          L     R1,=A(AUTHPP) AUTH parameter page is also key 1.         STA00580
000060 0841                          74          SSK   R4,R1           Set key                                  STA00590

000062 5810 C154      00154          76          L     R1,=A(EXTFLIH) External FLIH is key 2.                   STA00610
000066 4340 C13D      00130          77          IC    R4,KEY2         Get key                                  STA00620
00006A 0841                          78          SSK   R4,R1           Set key                                  STA00630
00006C 5810 C158      00158          79          L     R1,=A(EXTSPP) External SLIH parameter page is also key 2 STA00640
000070 0841                          80          SSK   R4,R1           Set key                                  STA00650

000072 5810 C15C      0015C          82          L     R1,=A(IOFLIH) I/O FLIH is key 3.                         STA00670
000076 4340 C13E      0013E          83          IC    R4,KEY3         Get key                                  STA00680
00007A 0841                          84          SSK   R4,R1           Set key                                  STA00690
00007C 5810 C160      00160          85          L     R1,=A(CONSPP) Console Address Space parameter page key 3 STA00700
000080 0841                          86          SSK   R4,R1           Set key                                  STA00710
000082 5810 C164      00164          87          L     R1,=A(CONSOLE) COnsole Address Space is key 3.           STA00720
000086 0841                          88          SSK   R4,R1           Set key                                  STA00730

000088 5810 C168      00168          90          L     R1,=A(PGMFLIH) Program FLIH is key 4.                    STA00750
00008C 4340 C13F      0013F          91          IC    R4,KEY4         Get key                                  STA00760
000090 0841                          92          SSK   R4,R1           Set key                                  STA00770
000092 1711                          93          XR    R1,R1           Point @ page zero                        STA00780
000094 0841                          94          SSK   R4,R1           Page zero is key 4.                      STA00790

                                     96 *-------------------------------------------------------------------* STA00810
                                     97 * Set remaining pages of system code/data to key 5.                  * STA00820
                                     98 *-------------------------------------------------------------------* STA00830

000096 9814 C11C      0011C         100          LM    R1,R4,SYSKPARM Get parameters to set system storage keys STA00850
00009A                              101 SETSYSK  DS    OH             Loop to set storage keys for system A. S. STA00860
00009A 0841                         102          SSK   R4,R1           Set key for page of system address space. STA00870
00009C 8712 C09A      0009A         103          BXLE  R1,R2,SETSYSK Process every page                         STA00880

                                    105 *-------------------------------------------------------------------* STA00900
                                    106 * Set remaining pages of user code/data to key 6.                   * STA00910
                                    107 *-------------------------------------------------------------------* STA00920

0000A0 9814 C12C      0012C         109          LM    R1,R4,USRKPARM Get parameters to set user storage keys   STA00940
0000A4                              110 SETUSRK  DS    OH             Loop to set storage keys for user addr sp STA00950
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 14.58 11/23/86
0000A4 0841                        111          SSK   R4,R1          Set key for page of user address space.    STA00960
0000A6 8712 C0A4       000A4       112          BXLE  R1,R2,SETUSRK Process every page                         STA00970

0000AA D203 0048 C16C 00048 0016C  114          MVC   CAW,=A(CONSCCW) Channel Address Word points @ chan pgm    STA00990
0000B0 9230 0048       00048       115          MVI   CAW,X'30'      Move CONSPP key to CAW.                    STA01000

0000B4 B204 C0F8       000F8       117          SCK   TSUB0          Initialize TOD clock.                      STA01020
0000B8 B206 C100       00100       118          SCKC  BIGCLOCK       Initialize clock comparator.               STA01030
0000BC B20B C108       00108       119          SPT   BIGTIMER       Initialize CPU timer.                      STA01040

                                   121          EXIT DSP           Start the system.                          STA01060
0000C0 5800 C170       00170       122+         L     R0,=CL4'DSP'       Load target address space identifier EXI00040
0000C4 0A02                        123+         SVC   2                  Go to that address space.            EXI00050

0000C8                             125          DS    0D           Align constant area.                       STA01080
0000C8 0008000000000016            126 ECMODE   DC    XL4'00080000',A(NEXTINST)                               STA01090
0000D0 0459000000000000            127 SSYSPSW  DC    XL4'04590000',F'0'                                      STA01100
0000D8 0019000000000000            128 SSVCNEW  DC    XL4'00190000',A(SVCFLIH)                                STA01110
0000E0 0029000000000000            129 SEXTNEW  DC    XL4'00290000',A(EXTFLIH)                                STA01120
0000E8 0039000000000000            130 SIONEW   DC    XL4'00390000',A(IOFLIH)                                 STA01130
0000F0 0048000000000000            131 SPGMNEW  DC    XL4'00480000',A(PGMFLIH)                                STA01140
0000F8 0000000000000000            132 TSUB0    DC    2F'0'                                                   STA01150
000100 FFFFFFFFFFFFFFFF            133 BIGCLOCK DC    2F'-1'                                                  STA01160
000108 7FFFFFFFFFFFFFFF            134 BIGTIMER DC    XL4'7FFFFFFF',F'-1'                                     STA01170
000110 01800C00                    135 CREG0    DC    XL4'01800C00'                                           STA01180
000114 80000000                    136 CREG2    DC    XL4'80000000'                                           STA01190
000118 00008000                    137 CREG8    DC    XL4'00008000'                                           STA01200
00011C 0000000000001000            138 SYSKPARM DC    A(AUTH),F'4096',A(TLAS),XL4'00000050'                   STA01210
00012C 0000000000001000            139 USRKPARM DC    A(USER1PP),F'4096',A(STARTUP-1000),XL4'00000060'        STA01220
00013C 10                          140 KEY1     DC    X'10'                                                   STA01230
00013D 20                          141 KEY2     DC    X'20'                                                   STA01240
00013E 30                          142 KEY3     DC    X'30'                                                   STA01250
00013F 40                          143 KEY4     DC    X'40'                                                   STA01260
000140                             144          LTORG                                                         STA01270
000140 00000002                    145                =A(*-STARTUP)
000144 00000000                    146                =A(DATABLES)
000148 00000000                    147                =A(STARTUP)
00014C 00000000                    148                =A(SVCFLIH)
000150 00000000                    149                =A(AUTHPP)
000154 00000000                    150                =A(EXTFLIH)
000158 00000000                    151                =A(EXTSPP)
00015C 00000000                    152                =A(IOFLIH)
000160 00000000                    153                =A(CONSPP)
000164 00000000                    154                =A(CONSOLE)
000168 00000000                    155                =A(PGMFLIH)
00016C 00000000                    156                =A(CONSCCW)
000170 C4E2D740                    157                =CL4'DSP'
                                   158          PRINT OFF                                                     STA01280
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00038	00036
AUTH	00001	00000000	00020	00138
AUTHPP	00001	00000000	00021	00149
BIGCLOCK	00004	00000100	00133	00118
BIGTIMER	00004	00000108	00134	00119
CAW	00004	00000048	00329	00114 00115
CONSCCW	00001	00000000	00021	00156
CONSOLE	00001	00000000	00021	00154
CONSPP	00001	00000000	00021	00153
CREG0	00004	00000110	00135	00055
CREG2	00004	00000114	00136	00056
CREG8	00004	00000118	00137	00057
DATABLES	00001	00000000	00020	00146
ECMODE	00004	000000C8	00126	00042
EXTFLIH	00001	00000000	00020	00129 00150
EXTNEW	00008	00000058	00331	00051
EXTSPP	00001	00000000	00021	00151
IOFLIH	00001	00000000	00021	00130 00152
IONEW	00008	00000078	00335	00052
KEY1	00001	0000013C	00140	00071
KEY2	00001	00000130	00141	00077
KEY3	00001	0000013E	00142	00083
KEY4	00001	0000013F	00143	00091
NEXTINST	00002	00000016	00044	00126
PGMFLIH	00001	00000000	00020	00131 00155
PGMNEW	00008	00000068	00333	00053
PSA	00001	00000000	00321	00040 00322 00330 00336 00338 00340 00342 00344 00347
RUNIDLE	00001	00000020	00223	00046
R0	00001	00000000	00173	00040 00122
R1	00001	00000001	00174	00064 00066 00067 00068 00070 00072 00073 00074 00076 00078 00079 00080 00082 00084 00085 00086 00087 00088 00090 00092 00093 00093 00094 00100 00102 00103 00109 00111 00112
R12	00001	0000000C	00185	00029 00030 00031 00032 00034
R2	00001	00000002	00175	00103 00112
R4	00001	00000004	00177	00065 00065 00066 00068 00071 00072 00074 00077 00078 00080 00083 00084 00086 00088 00091 00092 00094 00100 00102 00109 00111
SETSYSK	00002	0000009A	00101	00103
SETUSRK	00002	000000A4	00110	00112
SEXTNEW	00004	000000E0	00129	00051
SIONEW	00004	000000E8	00130	00052
SPGMNEW	00004	000000F0	00131	00053
SSVCNEW	00004	000000D8	00128	00050
SSYSPSW	00004	000000D0	00127	00048
STARTUP	00001	00000000	00027	00034 00139 00145 00147
SVCFLIH	00001	00000000	00020	00128 00148
SVCNEW	00008	00000060	00332	00050
SYSKPARM	00004	0000011C	00138	00100
SYSPSW	00008	00000470	00353	00048
SYSTATE	00004	00000468	00351	00046
TLAS	00001	00000000	00020	00138
TSUB0	00004	000000F8	00132	00117
USER1PP	00001	00000000	00020	00139
USRKPARM	00004	0000012C	00139	00109

```
  LOC  OBJECT CODE    AODR1 ADDR2  STMT   SOURCE STATEMENT                                         ASM 0201 20.48 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SHU00040
                                      4 *+---------------------------------------------------------------------------+* SHU00050
                                      5 *|                                                                           |* SHU00060
                                      6 *|Module:   SHUTDOWN                                                          |* SHU00070
                                      7 *|                                                                           |* SHU00080
                                      8 *|          Shut down the running system.                                     |* SHU00090
                                      9 *|                                                                           |* SHU00100
                                     10 *|Function: Terminate system operation merely by issuing a SHUTSYS          |* SHU00110
                                     11 *|          monitor call.                                                     |* SHU00120
                                     12 *|                                                                           |* SHU00130
                                     13 *|Caller:   Invoked on behalf of the Operator Address Space when it         |* SHU00140
                                     14 *|          issues an SVC 6 to kill the system.                               |* SHU00150
                                     15 *|                                                                           |* SHU00160
                                     16 *|Mode:     Problem state, OAT on, Key 5.                                     |* SHU00170
                                     17 *|                                                                           |* SHU00180
                                     18 *+---------------------------------------------------------------------------+* SHU00190
                                     19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SHU00200

                                     21          CSECTION SHUTDOWN   CSECT invoked for Operator shutdown reqst. SHU00220

                                     23+         PUNCH 'SPB'              Make module start on a page boundry.  CSE00040

000000                               25+SHUTDOWN CSECT                    Establish R12 as module overall base: CSE00060

000000 05C0                          27+         BALR  R12,0              Give R12 a temporary base value;       CSE00080
                           00002     28+         USING *,R12              R12 now points @ SL instruction        CSE00090
000002 5FC0 C016          00018      29+         SL    R12,=A(*-SHUTDOWN) R12 = R12 - offset from CSECT start    CSE00100
                                     30+         DROP  R12                R12 doesn't point at SL anymore;       CSE00110

                           00000     32+         USING SHUTOOWN,R12       Now R12 has A(This CSECT) in it.       CSE00130

000006 47F0 C012          00012      34+         B     AROUND             Jump past symbolic module identifier. CSE00150
00000A E2C8E4E3C406E6D5              35+MOOID    DC    CL8'SHUTDOWN'      Handy when poking around in storage.  CSE00160
000012                               36+AROUND   DS    0H                 End of CSECTION expansion.             CSE00170

000012 AF00 000B          0000B      38          MC    SHUTSYS,0      Signal PGMFLIH to shut the system down.    SHU00240

000018                               40          LTORG                                                           SHU00260
000018 00000002                      41                =A(*-SHUTDOWN)
                                     42          PRINT OFF                                                       SHU00270
```

ASM 0201 20.48 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00036	00034
R12	00001	0000000C	00069	00027 00028 00029 00030 00032
SHUTDOWN	00001	00000000	00025	00032 00041
SHUTSYS	00001	0000000B	00100	00038

```
  LOC  OBJECT CODE    ADDR1 AOOR2  STMT   SOURCE STATEMENT                               ASM 0201 20.43 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* RLA00030
                                      4 *+----------------------------------------------------------------------+* RLA00040
                                      5 *|                                                                      |* RLA00050
                                      6 *|Module:   RLAS                                                        |* RLA00060
                                      7 *|                                                                      |* RLA00070
                                      8 *|          Run List Address Space.                                     |* RLA00080
                                      9 *|                                                                      |* RLA00090
                                     10 *|Function: Perform enqueue and dequeue functions on the Run List,      |* RLA00100
                                     11 *|          as well as saving the user's environment which consists     |* RLA00110
                                     12 *|          of his general and floating-point registers, CPU timer      |* RLA00120
                                     13 *|          and PSW.                                                    |* RLA00130
                                     14 *|                                                                      |* RLA00140
                                     15 *|          The module maintains a pool of space organized into 10      |* RLA00150
                                     16 *|          slots.  Each slot is large enough to contain a UCB.         |* RLA00160
                                     17 *|          The Run List consists of UCBs allocated in this pool,       |* RLA00170
                                     18 *|          chained together in a one-way list and sorted by            |* RLA00180
                                     19 *|          descending priority.  Space for a new list element is       |* RLA00190
                                     20 *|          acquired by stepping sequentially through the pool in       |* RLA00200
                                     21 *|          slot-size increments until a free slot is located.          |* RLA00210
                                     22 *|                                                                      |* RLA00220
                                     23 *|          List members are chained by virtual address via the UCB     |* RLA00230
                                     24 *|          forward pointer field.                                      |* RLA00240
                                     25 *|                                                                      |* RLA00250
                                     26 *|          See the Enqueue and Dequeue entry points for further        |* RLA00260
                                     27 *|          details.                                                    |* RLA00270
                                     28 *|                                                                      |* RLA00280
                                     29 *|          There is a third entry point known as Get Dispatch UCB.     |* RLA00290
                                     30 *|          This entry point is called exclusively by the dispatcher    |* RLA00300
                                     31 *|          when it wants to dispatch a UCB.  The entry point passes    |* RLA00310
                                     32 *|          back either an image of the first UCB on the Run List or    |* RLA00320
                                     33 *|          an out-of-work indicator if there aren't any runnable       |* RLA00330
                                     34 *|          user address spaces.                                        |* RLA00340
                                     35 *|                                                                      |* RLA00350
                                     36 *|Caller:   Several callers; refer to RLAS authorization list in        |* RLA00360
                                     37 *|          AUTH for the exact list of callers.                         |* RLA00370
                                     38 *|                                                                      |* RLA00380
                                     39 *|Mode:     Problem state, DAT on, Key 5.                               |* RLA00390
                                     40 *|                                                                      |* RLA00400
                                     41 *+----------------------------------------------------------------------+* RLA00410
                                     42 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* RLA00420

                                     44          EXTRN USER1ST,USER2ST,USER3ST,USER4ST,USER5ST,OPST                 RLA00440

                                     46          CSECTION RLAS       Run List Address Space:                        RLA00460

                                     48+         PUNCH 'SPB'              Make module start on a page boundry.   CSE00040

000000                               50+RLAS     CSECT                     Establish R12 as module overall base: CSE00060

000000 05C0                          52+         BALR  R12,0               Give R12 a temporary base value;      CSE00080
                           00002     53+         USING *,R12               R12 now points @ SL instruction       CSE00090
000002 5FC0 C436        00438        54+         SL    R12,=A(*-RLAS)      R12 = R12 - offset from CSECT start   CSE00100
                                     55+         DROP  R12                 R12 doesn't point at SL anymore;      CSE00110
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86

                                    57+          USING RLAS,R12          Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012       00012       59+          B     AROUND            Jump past symbolic module identifier. CSE00150
00000A D9D3C1E240404040            60+MODID     DC    CL8'RLAS'         Handy when poking around in storage.  CSE00160
000012                             61+AROUND    DS    0H                End of CSECTION expansion.             CSE00170

                                   63           CRASH                 Not an executable entry point...      RLA00480
000012 005CC3D9C1E2C85C            64+          DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.     CRA00030
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86
                                     66 *-------------------------------------------------------------------* RLA00500
                                     67 * A. ENQUEUE entry: Start things off by constructing a trace entry  * RLA00510
                                     68 *                   reflecting the UCB about to be enqueued.  (That * RLA00520
                                     69 *                   UCB resides in the parameter page.)             * RLA00530
                                     70 *-------------------------------------------------------------------* RLA00540

                                     72          ENTRYPT ENQUEUE,100 Enqueue entry point:                    RLA00560

00001A                        00100  74+         ORG   RLAS+X'100'                                           ENT00040

000100                               76+ENQUEUE  OS    0H            Define entry point name.                  ENT00060
                                     77+         PUSH  USING         Save overall USING environment.           ENT00070
000100 05C0                          78+         BALR  R12,0         Establish temporary base.                 ENT00080
                              00102  79+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000102 5FC0 C33A        0043C        80+         SL    R12,=A(*-RLAS)    R12 now points @ CSECT start.       ENT00100
                                     81+         POP   USING         Restore overall USING environment.        ENT00110

000106 021F C3F8 0000 003F8 00000    83          MVC   SAVEAREA,0(R0) Copy trace portion of parameter page     RLA00580

00010C 0207 0000 C420 00000 00420    85          MVC   0(8,R0),=CL8'*ENQ*'   Move entryid to trace           RLA00600
000112 0207 0008 0078 00008 00078    86          MVC   8(8,R0),UCBID-UCB(R0) Move name of guy we're enqueueing RLA00610
000118 0707 0010 0010 00010 00010    87          XC    16(8,R0),16(R0)       Clear unused portion of entry   RLA00620
                                     88          TRACE               Create trace entry.                     RLA00630
00011E 0207 0018 C00A 00018 0000A    89+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
000124 5800 C440        00440        90+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
000128 0A01                          91+         SVC   1                   Go to that address space.           CAL00050

00012A 021F 0000 C3F8 00000 003F8    93          MVC   0(L'SAVEAREA,R0),SAVEAREA Restore destroyed UCB for ENQ RLA00650

                                     95 *-------------------------------------------------------------------* RLA00670
                                     96 * C. Find a free slot for the new UCB.                              * RLA00680
                                     97 *-------------------------------------------------------------------* RLA00690

000130 4110 C500        00500        99          LA    R1,UCBAREA    Point @ UCB area.                       RLA00710

                              00000 101          USING UCB,R1        Access UCB DSECT                        RLA00730

000134                               103 FINDUCB  DS    0H            Search for free UCB:                    RLA00750

000134 0503 C300 1000 00300 00000    105         CLC   FREE,UCB       Is this one free?                       RLA00770
00013A 4780 C146        00146        106         BE    FOUNDUCB      Yes - This'll do.                       RLA00780
00013E 4110 1090        00090        107         LA    R1,SLOTSIZE(,R1) No  - Move to next,                  RLA00790
000142 47F0 C134        00134        108         B     FINDUCB       And check it out.                       RLA00800

                                     110 *-------------------------------------------------------------------* RLA00820
                                     111 * D. Move UCB from parameter page into newly-acquired Run List slot.  * RLA00830
                                     112 *-------------------------------------------------------------------* RLA00840

000146                               114 FOUNDUCB DS    0H            Here with available UCB space @ R1:     RLA00860

000146 0284 1000 0000 00000 00000    116         MVC   UCB(UCBSIZE),0(R0) Install new UCB in table.          RLA00880

                                     118 *-------------------------------------------------------------------* RLA00900
                                     119 * E. Search for a point of insertion along the Run List.  This list  * RLA00910
                                     120 *    is maintained in sorted order on the user's priority assigned by * RLA00920
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                    ASM 0201 20.43 12/04/86
                                    121 *    the scheduler.  This sort key is strictly decreasing.  In the    * RLA00930
                                    122 *    case of a tie (tie list), enqueue the UCB at the end of the tie  * RLA00940
                                    123 *    list.                                                            * RLA00950
                                    124 *-------------------------------------------------------------------* RLA00960

00014C 1799                         126          XR    R9,R9          Clear R9.                              RLA00980
00014E 4390 1084      00084         127          IC    R9,UCBPRI      Get user's dispatch priority.          RLA00990

                                    129          DROP  R1             But retain address of new UCB in R1...  RLA01010

000152 5850 C3DC      003DC         131          L     R5,FIRST       0 if list = empty ( A(first UCB on list) ) RLA01030
000156 1766                         132          XR    R6,R6          Clear pointer to previous.              RLA01040

                            00000   134          USING UCB,R5                                                  RLA01060

000158                              136 SORT     DS    0H             Insert new UCB sorted by dispatch priority RLA01080

000158 1255                         138          LTR   R5,R5          Search over?                            RLA01100
00015A 47D0 C170      00170         139          BNP   INSERT         Yes - Either empty or EOL.              RLA01110

00015E BD91 5084      00084         141          CLM   R9,B'0001',UCBPRI No  - New:Member                    RLA01130
000162 4720 C170      00170         142          BH    INSERT         New > Member; Insert new.               RLA01140
000166 1865                         143          LR    R6,R5          New <= Member; Save pointer to current. RLA01150
000168 5850 5000      00000         144          L     R5,UCBFWD      Get pointer to next.                    RLA01160
00016C 47F0 C158      00158         145          B     SORT           Keep looking.                           RLA01170

                                    147 *-------------------------------------------------------------------* RLA01190
                                    148 * F. Insert the new UCB into the Run List.  The insertion technique   * RLA01200
                                    149 *    of course has multiple cases: 1) insertion into an empty list,   * RLA01210
                                    150 *    2) insertion at the end of a non-empty list, 3) insertion into   * RLA01220
                                    151 *    the middle of the list and 4) a special case of case 3 where the * RLA01230
                                    152 *    inserted node becomes becomes first on the list.  The list is    * RLA01240
                                    153 *    singly-linked along the forward pointer field.                   * RLA01250
                                    154 *-------------------------------------------------------------------* RLA01260

000170                              156 INSERT   DS    0H             Found point of insertion:               RLA01280

000170 1255                         158          LTR   R5,R5          Beginning, middle or end?               RLA01300
000172 4780 C18C      0018C         159          BZ    BEGIN          0 = List empty => Beginning.            RLA01310
000176 4740 C19A      0019A         160          BM    END            -1 = Insert after last (in R6).  (End.) RLA01320

00017A 5050 1000      00000         162          ST    R5,UCBFWD-UCB(,R1) >0 = Middle.  Forward(new) <-- next. RLA01340
00017E 1266                         163          LTR   R6,R6          Is there a previous UCB?                RLA01350
000180 4780 C192      00192         164          BZ    NEWFIRST       No  - we have a new UCB at head of list. RLA01360
000184 5010 6000      00000         165          ST    R1,UCBFWD-UCB(,R6) Yes - Forward(prev) <-- new.         RLA01370
000188 47F0 C1A4      001A4         166          B     INSERTED       Done with insert.                       RLA01380

                                    168          DROP  R5                                                      RLA01400

                            00000   170          USING UCB,R1                                                  RLA01420

00018C                              172 BEGIN    DS    0H             Here for insertion into empty list:     RLA01440

00018C D203 1000 C3D4 00000 003D4   174          MVC   UCBFWD,NULL    No forward pointer.                     RLA01460
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86

000192                            176 NEWFIRST DS    0H           Here to set anchor to first UCB on list:   RLA01480

000192 5010 C3DC      003DC       178          ST    R1,FIRST     Set anchor.                                RLA01500
000196 47F0 C1A4      001A4       179          B     INSERTED     Done with insert.                          RLA01510

00019A                            181 END      DS    0H           Here for insertion @ end of list:          RLA01530

00019A 5010 6000      00000       183          ST    R1,UCBFWD-UCB(,R6) Next(last) --> new last.            RLA01550
00019E D203 1000 C3D4 00000 003D4 184          MVC   UCBFWD,NULL  Next(new) = null.                        RLA01560

                                  186          DROP  R1                                                      RLA01580

                                  188 *-------------------------------------------------------------------* RLA01600
                                  189 * G. Increment count of number of UCBs in the Run List, then return. * RLA01610
                                  190 *-------------------------------------------------------------------* RLA01620

0001A4                            192 INSERTED DS    0H           Here when UCB is linked:                   RLA01640

0001A4 5810 C3D8      003D8       194          L     R1,COUNT     Fetch count.                               RLA01660
0001A8 4111 0001      00001       195          LA    R1,1(R1)     Increment.                                 RLA01670
0001AC 5010 C3D8      003D8       196          ST    R1,COUNT     Update in storage.                         RLA01680

                                  198          RETURN             Return to caller.                        RLA01700
0001B0 0A03                       199+         SVC   3                                                       RET00030
```

```
  LOC  OBJECT CODE     ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86
                                     201 *-------------------------------------------------------------------* RLA01720
                                     202 * A'. Dequeue entry.  Start by ensuring we are not trying to dequeue  * RLA01730
                                     203 *                     from an empty list.  ABEND if this is the case. * RLA01740
                                     204 *-------------------------------------------------------------------* RLA01750

                                     206          ENTRYPT DEQUEUE,200 Dequeue entry point:                     RLA01770

0001B2                         00200 208+         ORG   RLAS+X'200'                                            ENT00040

000200                               210+DEQUEUE  DS    0H            Define entry point name.                  ENT00060
                                     211+         PUSH  USING         Save overall USING environment.           ENT00070
000200 05C0                          212+         BALR  R12,0         Establish temporary base.                 ENT00080
                               00202 213+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000202 5FC0 C242         00444       214+         SL    R12,=A(*-RLAS)    R12 now points @ CSECT start.       ENT00100
                                     215+         POP   USING         Restore overall USING environment.        ENT00110

000206 D403 C308 C308 003D8 003D8    217          NC    COUNT,COUNT   Is list empty?                           RLA01790
00020C 4770 C218         00218       218          BNZ   SAVEUSER      No  - We're okay.                        RLA01800
                                     219          CRASH               Yes - Can't dequeue empty list.           RLA01810
000210 005CC3D9C1E2C85C              220+         DC    XL1'00',CL7'*CRASH*' Undefined opcode, eye-catcher.     CRA00030

                                     222 *-------------------------------------------------------------------* RLA01830
                                     223 * B'. Save the state of the last user to run from the saved environ-  * RLA01840
                                     224 *     ment data in low core.                                        * RLA01850
                                     225 *-------------------------------------------------------------------* RLA01860

000218                               227 SAVEUSER DS    0D            Checkpoint the last running user's state. RLA01880

000218 45E0 C390         00390       229          BAL   R14,CKPTUSER  Do it now.                               RLA01900

                                     231 *-------------------------------------------------------------------* RLA01920
                                     232 * C'. Clear the parameter page.                                      * RLA01930
                                     233 *-------------------------------------------------------------------* RLA01940

00021C 9825 C3E4         003E4       235          LM    R2,R5,CLEARPP Get MVCL clear parameters.                RLA01960
000220 0E24                          236          MVCL  R2,R4         Clear the parameter page.                 RLA01970

                                     238 *-------------------------------------------------------------------* RLA01990
                                     239 * D'. Move the UCB at the head of the list to the parameter page.    * RLA02000
                                     240 *-------------------------------------------------------------------* RLA02010

000222 5810 C3DC         003DC       242          L     R1,FIRST      Get pointer to first UCB.                 RLA02030

                                00000 244          USING UCB,R1        Access DSECT fields.                    RLA02050

000226 D284 0000 1000 00000 00000    246          MVC   0(UCBSIZE,R0),UCB Move UCB to parm page.                RLA02070

                                     248 *-------------------------------------------------------------------* RLA02090
                                     249 * E'. Construct a trace entry showing which UCB we just dequeued.    * RLA02100
                                     250 *-------------------------------------------------------------------* RLA02110

00022C D21F C3F8 0000 003F8 00000    252          MVC   SAVEAREA,0(R0) Copy trace portion of parameter page     RLA02130

000232 D207 0000 C428 00000 00428    254          MVC   0(8,R0),=CL8'*DEQ*'   Move entryid to trace            RLA02150
000238 D207 0008 0078 00008 00078    255          MVC   8(8,R0),UCBID-UCB(R0) Move name of guy we're dequeueing RLA02160
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT
00023E 0707 0010 0010 00010 00010   256          XC    16(8,R0),16(R0)       Clear unused portion of entry      RLA02170
                                    257          TRACE                       Create trace entry.                RLA02180
000244 0207 0018 C00A 00018 0000A   258+         MVC   24(8,R0),MODID Move symbolic module name to pp words 6-7 TRA00030
00024A 5800 C440      00440         259+         L     R0,=CL4'TRAC'         Load target address space identifier CAL00040
00024E 0A01                         260+         SVC   1                     Go to that address space.          CAL00050

000250 021F 0000 C3F8 00000 003F8   262          MVC   0(L'SAVEAREA,R0),SAVEAREA Restore destroyed DEQ'd UCB.   RLA02200

                                    264 *-------------------------------------------------------------------* RLA02220
                                    265 * F'. Adjust the Run List anchor to point at the next (new) head UCB, * RLA02230
                                    266 *     if any.                                                       * RLA02240
                                    267 *-------------------------------------------------------------------* RLA02250

000256 0503 1000 C304 00000 00304   269          CLC   UCBFWD,NULL   Was this the only UCB in the list?        RLA02270
00025C 4770 C26A      0026A         270          BNE   MOREUCBS      No  - There are others.                   RLA02280
000260 0703 C30C C30C 0030C 0030C   271          XC    FIRST,FIRST   Yes - The set anchor to zero.             RLA02290
000266 47F0 C270      00270         272          B     FRETUCB       Free up this UCB's space.                 RLA02300

00026A                              274 MOREUCBS DS    0H            Here if UCB list is not empty:            RLA02320

00026A 0203 C30C 1000 0030C 00000   276          MVC   FIRST,UCBFWD  Update anchor with new first.           RLA02340

                                    278 *-------------------------------------------------------------------* RLA02360
                                    279 * G'. Free up the space held by the dequeued UCB.                   * RLA02370
                                    280 *-------------------------------------------------------------------* RLA02380

000270                              282 FRETUCB  DS    0H            Here to release UCB space:                RLA02400

000270 078F 1000 1000 00000 00000   284          XC    UCB(SLOTSIZE),UCB Clear out storage held by UCB.        RLA02420
000276 0203 1000 C300 00000 00300   285          MVC   UCBFWD,FREE   Release UCB space: Tag it with eye-catcher RLA02430

                                    287 *-------------------------------------------------------------------* RLA02450
                                    288 * H'. Decrement the count of dispatchable UCBs.                    * RLA02460
                                    289 *-------------------------------------------------------------------* RLA02470

00027C 5810 C308      00308         291          L     R1,COUNT      Get length of list.                       RLA02490
000280 0610                         292          BCTR  R1,0          Minus 1.                                  RLA02500
000282 5010 C308      00308         293          ST    R1,COUNT      Update in storage.                        RLA02510

                                    295 *-------------------------------------------------------------------* RLA02530
                                    296 * I'. The dequeued UCB is now in the parameter page.  Return it to    * RLA02540
                                    297 *     the caller.                                                   * RLA02550
                                    298 *-------------------------------------------------------------------* RLA02560

000286                              300 DEQDONE  DS    0H            Dequeue processing is complete.           RLA02580

                                    302          RETXMIT             Return to caller.                         RLA02600
000286 0A04                         303+         SVC   4                                                       RET00030

                                    305          DROP  R1                                                      RLA02620
```

```
                                    307 *-------------------------------------------------------------------* RLA02640
                                    308 * A". GETDUC8 entry.  If the system is just coming up we have to     * RLA02650
                                    309 *                     establish the Run List chain.  Initialization * RLA02660
                                    310 *                     code must do this because the linkage editor  * RLA02670
                                    311 *                     cannot relocate relative to the start of an   * RLA02680
                                    312 *                     address space.  This means that the source    * RLA02690
                                    313 *                     module cannot establish chain links based on  * RLA02700
                                    314 *                     address constants because the relocated adcons* RLA02710
                                    315 *                     would reflect real addresses.  Having this    * RLA02720
                                    316 *                     address space construct the list during startup* RLA02730
                                    317 *                     ensures the chain link pointers are virtual   * RLA02740
                                    318 *                     addresses defined with respect to this address* RLA02750
                                    319 *                     space.                                        * RLA02760
                                    320 *-------------------------------------------------------------------* RLA02770

                                    322          ENTRYPT GETDUC8,300 Get Dispatch UC8 entry point:            RLA02790

000288                      00300   324+         ORG   RLAS+X'300'                                            ENT00040

000300                              326+GETDUC8  DS    0H            Define entry point name.                  ENT00060
                                    327+         PUSH  USING         Save overall USING environment.           ENT00070
000300 05C0                         328+         BALR  R12,0         Establish temporary base.                 ENT00080
                            00302   329+         USING *,R12         R12 now points @ next SL instruction:     ENT00090
000302 5FC0 C146            00448   330+         SL    R12,=A(*-RLAS)    R12 now points @ CSECT start.       ENT00100
                                    331+         POP   USING         Restore overall USING environment.        ENT00110

000306 9501 C41C            0041C   333          CLI   STARTUP,YES  Is the system in startup mode?            RLA02810
00030A 4770 C32E            0032E   334          BNE   SAVENV       No  - We may have a running user to ckpt.  RLA02820

00030E 9200 C41C            0041C   336          MVI   STARTUP,NO   Yes - Indicate from now on we have to ckpt. RLA02840
000312 4110 C500            00500   337          LA    R1,UC8AREA   Get virtual address of 1st UC8 on run list RLA02850
000316 5010 C3DC            003DC   338          ST    R1,FIRST     Set anchor: Can't have loader do this       RLA02860
00031A 5850 C3D8            003D8   339          L     R5,COUNT     Get number of UC8s in this system.          RLA02870
00031E 0650                         340          BCTR  R5,0         Minus 1 for n-1 forward pointers.           RLA02880

                            00000   342          USING UC8,R6       Associate DSECT with R6                     RLA02900

000320                              344 LINKIT   DS    0H           Establish the UC8 chain:                    RLA02920

000320 1861                         346          LR    R6,R1        R6 points @ current UC8                     RLA02940
000322 4111 0090            00090   347          LA    R1,SLOTSIZE(R1) R1 points @ next UC8                     RLA02950
000326 5010 6000            00000   348          ST    R1,UC8FWD    Make current point @ next                   RLA02960
00032A 4650 C320            00320   349          BCT   R5,LINKIT    Process n-1 UC8 forward pointers.           RLA02970

                                    351          DROP  R6           R6 is not UCB address anymore.              RLA02990

                                    353 *-------------------------------------------------------------------* RLA03010
                                    354 * 8". We may have to checkpoint the environment for a running user.  * RLA03020
                                    355 *     Let the checkpoint subroutine make that decision and perform  * RLA03030
                                    356 *     the action if warranted.                                      * RLA03040
                                    357 *-------------------------------------------------------------------* RLA03050

00032E                              359 SAVENV   OS    0H           Here to save dispatch user environment:     RLA03070

00032E 45E0 C390            00390   361          BAL   R14,CKPTUSER Yes - Save user's state data now.           RLA03090
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86

                                    363 *-------------------------------------------------------------------* RLA03110
                                    364 * C". Clear the parameter page.                                      * RLA03120
                                    365 *-------------------------------------------------------------------* RLA03130

000332 9825 C3E4       003E4        367          LM    R2,R5,CLEARPP Get MVCL clear parameters.               RLA03150
000336 0E24                         368          MVCL  R2,R4         Clear the parameter page.                RLA03160

                                    370 *-------------------------------------------------------------------* RLA03180
                                    371 * D". See if there is any dispatchable work in the system.  If there * RLA03190
                                    372 *     is, copy the first UCB on the Run List to the parameter page,   * RLA03200
                                    373 *     and issue a Set Oispatch UCB Real Address Monitor Call with the * RLA03210
                                    374 *     virtual address of the UCB in R1.                               * RLA03220
                                    375 *-------------------------------------------------------------------* RLA03230

000338 0403 C3D8 C3D8 003D8 00308   377          NC    COUNT,COUNT   Is list empty?                          RLA03250
00033E 4780 C354       00354        378          BZ    NOOSP         Yes - Nobody to dispatch.                RLA03260

000342 5810 C30C       0030C        380          L     R1,FIRST      No  - Get pointer to first UCB on list.  RLA03280
000346 5010 C3E0       003E0        381          ST    R1,CHECKPTR   Set pointer to dispatch UCB for ckpt oprtn RLA03290

                              00000 383          USING UCB,R1        Access OSECT fields.                     RLA03310

00034A D284 0000 1000 00000 00000   385          MVC   O(UCBSIZE,R0),UCB Move dispatch UCB to parm page.     RLA03330

000350 47F0 C35A       0035A        387          B     TRACEIT       That's all.                              RLA03350

                                    389 *-------------------------------------------------------------------* RLA03370
                                    390 * E". There are no dispatchable UCBs in the system at present.  Send  * RLA03380
                                    391 *     back a "no work" indicator to the dispatcher so he can put the  * RLA03390
                                    392 *     system into a wait state.                                       * RLA03400
                                    393 *-------------------------------------------------------------------* RLA03410

000354                              395 NODSP    OS    OH            Here when no runnable work exists:       RLA03430

000354 0203 0000 C3F4 00000 003F4   397          MVC   O(L'NOWORK,R0),NOWORK Move no work indicator to parm pg RLA03450

                                    399 *-------------------------------------------------------------------* RLA03470
                                    400 * F". Trace the dispatch UCB or nowork indicator.                    * RLA03480
                                    401 *-------------------------------------------------------------------* RLA03490

00035A                              403 TRACEIT  OS    OH            Here to trace what we found:             RLA03510

00035A D21F C3F8 0000 003F8 00000   405          MVC   SAVEAREA,O(R0) Save trace portion of parameter page.   RLA03530

000360 0207 0000 C430 00000 00430   407          MVC   O(8,R0),=CL8'*GOUCB*' Identify the entry point.      RLA03550
000366 070F 0008 0008 00008 00008   408          XC    8(16,R0),8(R0) Clear remainder of trace entry        RLA03560
00036C 0403 C308 C308 003D8 00308   409          NC    COUNT,COUNT   Are we out of work?                    RLA03570
000372 4780 C37C       0037C        410          BZ    OOTRACE       Yes, empty trace entry reflects that.    RLA03580
000376 0207 0008 0078 00008 00078   411          MVC   8(8,R0),UCBID-UCB(R0) No  - Trace dispatch UCB id.    RLA03590

00037C                              413 OOTRACE  OS    OH                                                     RLA03610

                                    415          TRACE               Construct trace entry.                 RLA03630
00037C 0207 0018 C00A 00018 0000A   416+         MVC   24(8,R0),MOOIO Move symbolic module name to pp words 6-7 TRA00030
000382 5800 C440       00440        417+         L     R0,=CL4'TRAC'       Load target address space identifier CAL00040
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 20.43 12/04/86

0003B6 0A01                        418+         SVC   1                  Go to that address space.          CAL00050
000388 D21F 0000 C3F8 00000 003F8  419          MVC   0(L'SAVEAREA,R0),SAVEAREA Restore destroyed UCB image.  RLA03640

                                   421 *-------------------------------------------------------------------* RLA03660
                                   422 * G". Transmit the results back to the caller.                      * RLA03670
                                   423 *-------------------------------------------------------------------* RLA03680

                                   425          RETXMIT                                                     RLA03700
00038E 0A04                        426+         SVC   4                                                     RET00030
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86
                                    428 *-------------------------------------------------------------------* RLA03720
                                    429 * Subroutine: CKPTUSER                                              * RLA03730
                                    430 *                                                                   * RLA03740
                                    431 * Function:   Save the state of the last user to be dispatched.     * RLA03750
                                    432 *             It is not necessary to save user state every time this* RLA03760
                                    433 *             subroutine is called; in fact, to do so under some    * RLA03770
                                    434 *             circumstances would be disastrous.  For example, both * RLA03780
                                    435 *             the DEQUEUE and Get Dispatch UCB entry points call    * RLA03790
                                    436 *             CHECKPOINT.  Sometimes both entry points are called   * RLA03800
                                    437 *             within one pass through the supervisor.  The call to  * RLA03810
                                    43B *             DEQUEUE should checkpoint the state of the last user  * RLA03820
                                    439 *             dispatched.  Eventually dispatcher will call GDUCB    * RLA03830
                                    440 *             looking for work.  If we were to blindly checkpoint   * RLA03840
                                    441 *             the user at the head of the dispatch list (who normally* RLA03850
                                    442 *             is the last person to have been dispatched), we would * RLA03860
                                    443 *             save the PREVIOUS user's environment in the NEXT dis- * RLA03870
                                    444 *             patch user's UCB.  This scenerio is avoided by ensuring* RLA03880
                                    445 *             we only checkpoint a user that is known to have been  * RLA03B90
                                    446 *             dispatched.  We are guaranteed that a call to GDUCB   * RLA03900
                                    447 *             will necessitate a checkpoint operation the next time * RLA03910
                                    448 *             this module is invoked because dispatcher is the only * RLA03920
                                    449 *             caller of GDUCB and it always dispatches the UCB      * RLA03930
                                    450 *             supplied by that entry point.                         * RLA03940
                                    451 *                                                                   * RLA03950
                                    452 *             If a checkpoint operation is required, the saved state* RLA03960
                                    453 *             information is in page 2, which is really real page 0.* RLA03970
                                    454 *             PGMFLIH moved the running user's state information to * RLA039B0
                                    455 *             a user savearea in real page 0.                       * RLA03990
                                    456 *                                                                   * RLA04000
                                    457 *             Copy the state data into its corresponding UCB fields.* RLA04010
                                    458 *             The last dispatch UCB is the first on the list; real  * RLA04020
                                    459 *             page zero is attached as virtual page 2.              * RLA04030
                                    460 *-------------------------------------------------------------------* RLA04040

000390                              462 CKPTUSER DS    0H            Save the state of the running user:       RLA04060

000390 D403 C3E0 C3E0 003E0 003E0   464          NC    CHECKPTR,CHECKPTR Is a checkpoint operation required?   RLA04080
000396 47B0 C3CB      003C8         465          BZ    CKPTDONE      No  - Then just return.                  RLA04090
00039A 5870 C418      00418         466          L     R7,PAGETWO    Yes - Get address of page 1 (real page 0). RLA04100
                                    467          USING PSA,R7        Make page 0 fields accessible.          RLA04110
                            00000
00039E 5810 C3E0      003E0         469          L     R1,CHECKPTR   Get pointer to dispatched UCB.          RLA04130

0003A2 D23F 100B 74A0 0000B 004A0   471          MVC   UCBGREGS(16*4),SAVGREGS  Save the user's floating-point RLA04150
0003A8 D21F 1048 74E0 00048 004E0   472          MVC   UCBFREGS(4*8),SAVFREGS     and general registers.     RLA04160
0003AE D207 1068 749B 00068 0049B   473          MVC   UCBPSW,SAVPSW            Save his PSW as well.       RLA04170
0003B4 91B0 7500      00500         474          TM    SAVTIMER,X'B0'           If the CPU timer has popped,RLA04180
0003B8 4710 C3C8      003C8         475          B0    CKPTDONE                 don't save it!              RLA04190
0003BC D207 1070 7500 00070 00500   476          MVC   UCBTIMER,SAVTIMER        Otherwise, save it as well. RLA04200

0003C2 D703 C3E0 C3E0 003E0 003E0   47B          XC    CHECKPTR,CHECKPTR Checkpoint not required 'til next dsp RLA04220

0003CB                              480 CKPTDONE DS    0H            We have saved the user's environment.     RLA04240

0003C8 07FE                         4B2          BR    R14           Return to caller.                       RLA04260
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                ASM 0201 20.43 12/04/86
                                    484 *-------------------------------------------------------------------* RLA04280
                                    485 * Constants and variables:                                          * RLA04290
                                    486 *-------------------------------------------------------------------* RLA04300

000300                              488          DS    0D             Align constant area.                   RLA04320
000300 C609C5C5                     489 FREE     DC    CL4'FREE'      Eye-catcher indicating UC8 space is free. RLA04330
000304 FFFFFFFF                     490 NULL     DC    F'-1'          Null pointer.                          RLA04340
0003D8 00000006                     491 COUNT    DC    A((ENDOFUSR-UC8AREA)/SLOTSIZE) # UC8s on the Run List. RLA04350
0003DC 00000000                     492 FIRST    DC    F'0'           Pointer to first UCB on run list.      RLA04360
0003E0 00000000                     493 CHECKPTR DC    F'0'           Pointer to UC8 we have to checkpoint   RLA04370
0003E4 0000000000001000             494 CLEARPP  DC    F'0',F'4096',2F'0' Used by MVCL to clear parameter page. RLA04380
0003F4 050605C5                     495 NOWORK   DC    CL4'NONE'      Indicates no dispatchable UC8 in system. RLA04390
0003F8                              496 SAVEAREA DS    XL(8*4)        Holds part of parameter page while tracing RLA04400
000418 00002000                     497 PAGETWO  DC    XL4'00002000' Virtual address of real page zero.      RLA04410
                           00001    498 YES      EQU   X'01'          Startup flag value                     RLA04420
                           00000    499 NO       EQU   X'00'          Startup flag value                     RLA04430
00041C 01                           500 STARTUP  DC    X'01'          Indicate system is just coming up.     RLA04440
000420                              501          LTORG                                                       RLA04450
000420 5CC505085C404040             502                 =CL8'*ENQ*'
000428 5CC4C5085C404040             503                 =CL8'*DEQ*'
000430 5CC7C4E4C3C25C40             504                 =CL8'*GOUCB*'
000438 00000002                     505                 =A(*-RLAS)
00043C 00000102                     506                 =A(*-RLAS)
000440 E309C1C3                     507                 =CL4'TRAC'
000444 00000202                     508                 =A(*-RLAS)
000448 00000302                     509                 =A(*-RLAS)

                                    511 *-------------------------------------------------------------------* RLA04470
                                    512 * UCB pool / Run List:                                             * RLA04480
                                    513 *-------------------------------------------------------------------* RLA04490

00044C                     00500    515          ORG   RLAS+X'500'                                          RLA04510

                           00090    517 SLOTSIZE EQU   X'90'          Slightly larger than a UCB.            RLA04530

000500                              519 UC8AREA  DS    XL(SLOTSIZE)   Define enough space for 1 UC8.         RLA04550

000590                     00500    521          ORG   UC8AREA                                               RLA04570

000500 FFFFFFFFFFFFFFFF             523          DC    2F'-1'         Pretend this is a UC8:                 RLA04590
000508                              524          DS    16F            GREGS                                  RLA04600
000548                              525          DS    8F             FREGS                                  RLA04610
000568 0769000000010000             526          DC    XL4'07690000',XL4'00010000'  PSW                     RLA04620
000570 0000000100000000             527          DC    F'1',XL4'00000000'  Dispatch quantum                 RLA04630
000578 0607C509C1E30609             528          DC    CL8'OPERATOR'  Ident                                 RLA04640
000580 00000000                     529          DC    A(0PST)        STO                                    RLA04650
000584 80                           530          DC    XL1'80'        Priority                               RLA04660

000585                     00590    532          ORG   UCBAREA+SLOTSIZE                                      RLA04680

000590 FFFFFFFFFFFFFFFF             534          DC    2F'-1'         Pretend this is a UC8:                 RLA04700
000598                              535          DS    16F            GREGS                                  RLA04710
000508                              536          DS    8F             FREGS                                  RLA04720
0005F8 0769000000010000             537          DC    XL4'07690000',XL4'00010000'  PSW                     RLA04730
000600 0000000100000000             538          DC    F'1',XL4'00000000'  Dispatch quantum                 RLA04740
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 20.43 12/04/86
000608 6CE4E2C5D9F16C40            539          DC    CL8'%USER1%'  Ident                                    RLA04750
000610 00000000                    540          DC    A(USER1ST)    STO                                      RLA04760
000614 80                          541          DC    XL1'80'       Priority                                 RLA04770

000615                     00620   543          ORG   UCBAREA+(SLOTSIZE*2)                                   RLA04790

000620 FFFFFFFFFFFFFFFF            545          DC    2F'-1'        Pretend this is a UCB:                   RLA04810
000628                             546          DS    16F           GREGS                                    RLA04820
000668                             547          DS    8F            FREGS                                    RLA04830
000688 0769000000010000            548          DC    XL4'07690000',XL4'00010000'  PSW                       RLA04840
000690 0000000100000000            549          DC    F'1',XL4'00000000'  Dispatch quantum                   RLA04850
000698 4CE4E2C5D9F26E40            550          DC    CL8'<USER2>'  Ident                                    RLA04860
0006A0 D0000000                    551          DC    A(USER2ST)    STO                                      RLA04870
0006A4 80                          552          DC    XL1'80'       Priority                                 RLA04880

0006A5                     006B0   554          ORG   UCBAREA+(SLOTSIZE*3)                                   RLA04900

0006B0 FFFFFFFFFFFFFFFF            556          DC    2F'-1'        Pretend this is a UCB:                   RLA04920
0D06B8                             557          DS    16F           GREGS                                    RLA04930
0006F8                             558          DS    8F            FREGS                                    RLA04940
000718 0769000000010000            559          DC    XL4'07690000',XL4'00010000'  PSW                       RLA04950
000720 0000000100000000            560          DC    F'1',XL4'00000000'  Dispatch quantum                   RLA04960
000728 7EE4E2C5D9F37E4D            561          DC    CL8'=USER3='  Ident                                    RLA04970
000730 00000000                    562          DC    A(USER3ST)    STO                                      RLA04980
000734 80                          563          DC    XL1'80'       Priority                                 RLA04990

000735                     00740   565          ORG   UCBAREA+(SLOTSIZE*4)                                   RLA05010

000740 FFFFFFFFFFFFFFFF            567          DC    2F'-1'        Pretend this is a UCB:                   RLA05030
000748                             568          DS    16F           GREGS                                    RLA05040
000788                             569          DS    8F            FREGS                                    RLA05050
0007A8 0769000000010000            570          DC    XL4'07690000',XL4'00010000'  PSW                       RLA05060
0007B0 0000000100000000            571          DC    F'1',XL4'00000000'  Dispatch quantum                   RLA05070
0007B8 7BE4E2C5D9F47B40            572          DC    CL8'#USER4#'  Ident                                    RLA05080
0007C0 00000000                    573          DC    A(USER4ST)    STO                                      RLA05090
0007C4 80                          574          DC    XL1'80'       Priority                                 RLA05100

0007C5                     007D0   576          ORG   UCBAREA+(SLOTSIZE*5)                                   RLA05120

0007D0 FFFFFFFFFFFFFFFF            578          DC    2F'-1         Pretend this is a UCB:                   RLA05140
0007D8                             579          DS    16F           GREGS                                    RLA05150
000818                             580          DS    8F            FREGS                                    RLA05160
000838 0769000000010000            581          DC    XL4'07690000',XL4'00010000'  PSW                       RLA05170
000840 0000000100000000            582          DC    F'1',XL4'00000000'  Dispatch quantum                   RLA05180
000848 6FE4E2C5D9F56F40            583          DC    CL8'?USER5?'  Ident                                    RLA05190
000850 D0000000                    584          DC    A(USER5ST)    STO                                      RLA05200
000854 80                          585          DC    XL1'80'       Priority                                 RLA05210

                                   587 * Always ORG up where the next UCB would start so #-of-UCBs calc works. RLA05230

000855                     00860   589          ORG   UCBAREA+(SLOTSIZE*6)                                   RLA05250

                                   591 * Always place ENDOFUSR after ORG to next would-be UCB.                RLA05270

                           00860   593 ENDOFUSR EQU   *             Marker used to compute # of UCBs in system RLA05290
```

LOC	OBJECT CODE	ADDR1	AODR2	STMT	SOURCE STATEMENT	ASM 0201 20.43 12/04/86
				595	PRINT OFF	RLA05310

POS.ID	REL.ID	FLAGS	ADDRESS
0007	0001	0C	000610
0007	0002	0C	0006A0
0007	0003	0C	000730
0007	0004	0C	0007C0
0007	0005	0C	000850
0007	0006	0C	000580

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00061	00059
BEGIN	00002	0000018C	00172	00159
CHECKPTR	00004	000003E0	00493	00381 00464 00464 00469 00478 00478
CKPTDONE	00002	000003C8	00480	00465 00475
CKPTUSER	00002	00000390	00462	00229 00361
CLEARPP	00004	000003E4	00494	00235 00367
COUNT	00004	00000308	00491	00194 00196 00217 00217 00291 00293 00339 00377 00377 00409 00409
DOTRACE	00002	0000037C	00413	00410
END	00002	0000019A	00181	00160
ENDOFUSR	00001	00000860	00593	00491
FINDUCB	00002	00000134	00103	00108
FIRST	00004	0000030C	00492	00131 00178 00242 00271 00271 00276 00338 00380
FOUNDUCB	00002	00000146	00114	00106
FREE	00004	00000300	00489	00105 00285
FRETUCB	00002	00000270	00282	00272
INSERT	00002	00000170	00156	00139 00142
INSERTED	00002	000001A4	00192	00166 00179
LINKIT	00002	00000320	00344	00349
MODID	00008	0000000A	00060	00089 00258 00416
MOREUCBS	00002	0000026A	00274	00270
NEWFIRST	00002	00000192	00176	00164
NO	00001	00000000	00499	00336
NODSP	00002	00000354	00395	00378
NOWORK	00004	000003F4	00495	00397 00397
NULL	00004	00000304	00490	00174 00184 00269
OPST	00001	00000000	00044	00529
PAGETWO	00004	00000418	00497	00466
PSA	00001	00000000	00758	00467 00759 00767 00773 00775 00777 00779 00781 00784
RLAS	00001	00000000	00050	00057 00074 00208 00324 00505 00506 00508 00509 00515
R0	00001	00000000	00610	00083 00085 00086 00086 00087 00087 00089 00090 00093 00116 00246 00252 00254 00255 00255 00256 00256 00258 00259 00262 00385 00397 00405 00407 00408 00408 00411 00411 00416 00417 00419
R1	00001	00000001	00611	00099 00101 00107 00107 00129 00162 00165 00170 00178 00183 00186 00194 00195 00195 00196 00242 00244 00291 00292 00293 00305 00337 00338 00346 00347 00347 00348 00380 00381 00383 00469
R12	00001	0000000C	00622	00052 00053 00054 00055 00057 00078 00079 00080 00212 00213 00214 00328 00329 00330
R14	00001	0000000E	00624	00229 00361 00482
R2	00001	00000002	00612	00235 00236 00367 00368
R4	00001	00000004	00614	00236 00368
R5	00001	00000005	00615	00131 00134 00138 00138 00143 00144 00158 00158 00162 00168 00235 00339 00340 00349 00367
R6	00001	00000006	00616	00132 00132 00143 00163 00163 00165 00183 00342 00346 00351
R7	00001	00000007	00617	00466 00467
R9	00001	00000009	00619	00126 00126 00127 00141
SAVEAREA	00032	000003F8	00496	00083 00093 00093 00252 00262 00262 00405 00419 00419
SAVENV	00002	0000032E	00359	00334
SAVEUSER	00008	00000218	00227	00218
SAVFREGS	00008	000004E0	00799	00472
SAVGREGS	00004	000004A0	00798	00471
SAVPSW	00008	00000498	00797	00473
SAVTIMER	00008	00000500	00800	00474 00476
SLOTSIZE	00001	00000090	00517	00107 00284 00347 00491 00519 00532 00543 00554 00565 00576 00589
SORT	00002	00000158	00136	00145
STARTUP	00001	0000041C	00500	00333 00336
TRACEIT	00002	0000035A	00403	00387
UCB	00001	00000000	00841	00086 00101 00105 00116 00134 00162 00165 00170 00183 00244 00246 00255 00284 00284 00342

ASM 0201 20.43 12/04/86

SYMBOL	LEN	VALUE	OEFN	REFERENCES
				00383 00385 00411 00855
UCBAREA	00144	00000500	00519	00099 00337 00491 00521 00532 00543 00554 00565 00576 00589
UCBFREGS	00004	00000048	00845	00472
UCBFWO	00004	00000000	00842	00144 00162 00165 00174 00183 00184 00269 00276 00285 00348
UCBGREGS	00004	00000008	00844	00471
UCBIO	00008	00000078	00848	00086 00255 00411
UCBPRI	00001	00000084	00850	00127 00141
UCBPSW	00008	00000068	00846	00473
UCBSIZE	00001	00000085	00855	00116 00246 00385
UCBTIMER	00008	00000070	00847	00476
USER1ST	00001	00000000	00044	00540
USER2ST	00001	00000000	00044	00551
USER3ST	00001	00000000	00044	00562
USER4ST	00001	00000000	00044	00573
USER5ST	00001	00000000	00044	00584
YES	00001	00000001	00498	00333

```
  LOC  OBJECT COOE    AOOR1 AODR2  STMT   SOURCE STATEMENT                                       ASM 0201 14.46 11/22/86

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* OPE00030
                                     4 *+-------------------------------------------------------------------+* OPE00040
                                     5 *|                                                                   |* OPE00050
                                     6 *|Module:   OPERATOR                                                 |* OPE00060
                                     7 *|                                                                   |* OPE00070
                                     8 *|          System Operator Address Space.                           |* OPE00080
                                     9 *|                                                                   |* OPE00090
                                    10 *|Function: The system Operator is an address space treated exactly  |* OPE00100
                                    11 *|          like a user address space as far as most of the system   |* OPE00110
                                    12 *|          is concerned.  The only difference between the Operator  |* OPE00120
                                    13 *|          and a general user is that Operator is authorized to     |* OPE00130
                                    14 *|          invoke certain system services not available to the      |* OPE00140
                                    15 *|          general user.  At present, these operator services are   |* OPE00150
                                    16 *|          reading from the console and shutting the system down.   |* OPE00160
                                    17 *|                                                                   |* OPE00170
                                    18 *|          The Operator is actually just a big loop that announces  |* OPE00180
                                    19 *|          its availability then blocks itself pending console      |* OPE00190
                                    20 *|          input.  Command text entered by the human operator is    |* OPE00200
                                    21 *|          echoed back to the terminal and analyzed.  If the text   |* OPE00210
                                    22 *|          is not a valid command, an error message is issued and   |* OPE00220
                                    23 *|          the loop is performed again.  Otherwise, the requested   |* OPE00230
                                    24 *|          command function is performed.  At present, the only     |* OPE00240
                                    25 *|          Operator command implemented is SHUTOOWN.  In response   |* OPE00250
                                    26 *|          to the SHUTOOWN command, the system confirms receipt of  |* OPE00260
                                    27 *|          the shutdown request, announces that shutdown is in pro- |* OPE00270
                                    28 *|          gress and issues a shutdown SVC.                         |* OPE00280
                                    29 *|                                                                   |* OPE00290
                                    30 *|Caller:   Oispatched by the system at startup time.                |* OPE00300
                                    31 *|                                                                   |* OPE00310
                                    32 *|Mode:     Key 6.                                                   |* OPE00320
                                    33 *|                                                                   |* OPE00330
                                    34 *+-------------------------------------------------------------------+* OPE00340
                                    35 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* OPE00350

                                    37          CSECTION OPERATOR   The System Operator Address Space:          OPE00370

                                    39+         PUNCH 'SPB'             Make module start on a page boundry.  CSE00040

000000                              41+OPERATOR CSECT                   Establish R12 as module overall base: CSE00060

000000 05C0                         43+         BALR  R12,0             Give R12 a temporary base value;      CSE00080
                            00002   44+         USING *,R12             R12 now points @ SL instruction       CSE00090
000002 5FC0 C2CE      00200         45+         SL    R12,=A(*-OPERATOR) R12 = R12 - offset from CSECT start  CSE00100
                                    46+         OROP  R12               R12 doesn't point at SL anymore;      CSE00110

                            00000   48+         USING OPERATOR,R12      Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012         50+         8     AROUNO            Jump past symbolic module identifier. CSE00150
00000A 0607C509C1E30609             51+M0010    OC    CL8'OPERATOR'     Handy when poking around in storage.  CSE00160
000012                              52+AROUNO   OS    OH                End of CSECTION expansion.            CSE00170

000012 4180 004A      0004A         54          LA    R8,L'BUFFER-6 Maximum command echo buffer length.       OPE00390
000016 41A0 C09E      0009E         55          LA    R10,BUFFER+6  Start of reply area in echo buffer.       OPE00400
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 14.46 11/22/86

00001A                                 57 READY    DS    0H            Here to issue the "System ready" msg:      OPE00420

00001A D232 0000 C198 00000 00198      59          MVC   0(L'READYMSG,R0),READYMSG Move ready msg to msg buffer. OPE00440
000020 4100 0033      00033            60          LA    R0,L'READYMSG Put length of ready message in R0 for SVC. OPE00450
000024 0A04                            61          SVC   4             Have control program write msg to console. OPE00460

000026 0A03                            63          SVC   3             Listen for command reply:                  OPE00480

000028 1200                            65          LTR   R0,R0         Did operator simply press CR?              OPE00500
00002A 4780 C01A      0001A            66          BE    READY         Yes - No command given; re-issue ready msg OPE00510
00002E D24F C0E8 0000 000E8 00000      67          MVC   CMDBUF(80),0(R0) No  - Copy command reply to cmd buffer. OPE00520
000034 064F C0E8 C138 000E8 00138      68          OC    CMD8UF(80),UPPER Xlate cmd text to upper case.           OPE00530

00003A 1890                            70          LR    R9,R0         Get number of characters typed in.         OPE00550
00003C 1998                            71          CR    R9,R8         Is command longer than echo buffer?        OPE00560
00003E 47D0 C044      00044            72          BNH   LENGTHOK      No  - Then it'll fit okay.                 OPE00570
000042 1898                            73          LR    R9,R8         Yes - Truncate it to length of cmd buffer. OPE00580

000044                                 75 LENGTHOK OS    0H            Here to copy command to echo buffer:       OPE00600

000044 0690                            77          BCTR  R9,0          Subtract 1 from length for execute.        OPE00620
000046 4490 C08A      0008A            78          EX    R9,BUFFCOPY   Copy command to buffer.                   OPE00630
00004A 4199 0001      00001            79          LA    R9,1(R9)      Restore length (add one).                  OPE00640

00004E 9825 C188      00188            81          LM    R2,R5,CLRPP   Get MVCL clear parameters.               OPE00660
000052 0E24                            82          MVCL  R2,R4         Clear out the parameter page.              OPE00670

000054 4199 0005      00005            84          LA    R9,5(R9)      Length of echo message - 1 for execute.    OPE00690
000058 4490 C090      00090            85          EX    R9,COPYPP     Copy echo message to parameter page.       OPE00700
00005C 4109 0001      00001            86          LA    R0,1(R9)      Get full length of echo message.           OPE00710
000060 0A04                            87          SVC   4             Echo the command to the system console.    OPE00720

000062 D54F C1CB C0E8 001CB 000E8      89          CLC   SHUTDOWN,CMDBUF Is this the shutdown command?              OPE00740
000068 4780 C07C      0007C            90          BE    KILLSYS       Yes - Shut her down.                       OPE00750

00006C D230 0000 C21B 00000 0021B      92          MVC   0(L'UNKNOWN,R0),UNKNOWN No  - Move unknown cmd msg to pp OPE00770
000072 4100 0031      00031            93          LA    R0,L'UNKNOWN  Put length of unknown msg in R0 for SVC.   OPE00780
000076 0A04                            94          SVC   4             Have control program write msg to console. OPE00790
000078 47F0 C01A      0001A            95          B     READY         Go listen for next console command.        OPE00800

00007C                                 97 KILLSYS  DS    0H            Here to shut down system per operator cmd: OPE00820

00007C 0234 0000 C24C 00000 0024C      99          MVC   0(L'SHUTMSG,R0),SHUTMSG Move shutdown msg to parm page.  OPE00840
000082 4100 0035      00035           100          LA    R0,L'SHUTMSG  Put length of shutdown msg in R0 for SVC.   OPE00850
000086 0A04                           101          SVC   4             Have control program write msg to console. OPE00860

000088 0A06                           103          SVC   6             Issue shutdown SVC.                        OPE00880

                                      105 *-------------------------------------------------------------------* OPE00900
                                      106 * Executed instructions:                                            * OPE00910
                                      107 *-------------------------------------------------------------------* OPE00920
00008A                                108 BUFFCOPY DS    0H                                                     OPE00930
00008A D200 A000 0000 00000 00000     109          MVC   0(0,R10),0(R0)                                         OPE00940
000090                                110 COPYPP   DS    0H                                                     OPE00950
000090 0200 0000 C098 00000 00098     111          MVC   0(0,R0),BUFFER                                         OPE00960
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                    ASM 0201 14.46 11/22/86

000098                             112           DS    0D                                                    OPE00970
000098 C5C3C8067A404040            113 BUFFER    DC    CL80'ECHO: '                                          OPE00980
0000E8                             114 CMDBUF    DS    CL80                                                  OPE00990
000138 4040404040404040            115 UPPER     DC    80X'40'                                               OPE01000
000188 0000000000001000            116 CLRPP     DC    F'0',F'4096',2F'0'                                    OPE01010
000198 6E6E6E4040E38885            117 READYMSG  DC    C'>>>  The System is ready for Operator commands. <<<' OPE01020
0001CB E2C8E4E3C4D6E6D5            118 SHUTDOWN  DC    CL80'SHUTDOWN'                                        OPE01030
00021B C396949481958440            119 UNKNOWN   DC    C'Command text is unrecognized and will be ignored.'  OPE01040
00024C E2C8E4E3C406E605            120 SHUTMSG   DC    C'SHUTDOWN request accepted; Normal shutdown initiated.' OPE01050
000281 4040404040404040            121 BLANKS    DC    CL79' '                                               OPE01060
000200                             122           LTORG                                                       OPE01070
000200 00000002                    123                 =A(*-OPERATOR)
                                   124           PRINT OFF                                                   OPE01080
```

ASM 0201 14.46 11/22/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00052	00050
BUFFCOPY	00002	0000008A	00108	00078
BUFFER	00080	00000098	00113	00054 00055 00111
CLRPP	00004	00000188	00116	00081
CMDBUF	00080	000000E8	00114	00067 00068 00089
COPYPP	00002	00000090	00110	00085
KILLSYS	00002	0000007C	00097	00090
LENGTHOK	00002	00000044	00075	00072
OPERATOR	00001	00000000	00041	00048 00123
READY	00002	0000001A	00057	00066 00095
READYMSG	00051	00000198	00117	00059 00059 00060
R0	00001	00000000	00139	00059 00060 00065 00065 00067 00070 00086 00092 00093 00099 00100 00109 00111
R10	00001	0000000A	00149	00055 00109
R12	00001	0000000C	00151	00043 00044 00045 00046 00048
R2	00001	00000002	00141	00081 00082
R4	00001	00000004	00143	00082
R5	00001	00000005	00144	00081
R8	00001	00000008	00147	00054 00071 00073
R9	00001	00000009	00148	00070 00071 00073 00077 00078 00079 00079 00084 00084 00085 00086
SHUTDOWN	00080	000001C8	00118	00089
SHUTMSG	00053	0000024C	00120	00099 00099 00100
UNKNOWN	00049	00000218	00119	00092 00092 00093
UPPER	00001	00000138	00115	00068

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                    4 *+----------------------------------------------------------------------+* USE00040
                                    5 *|                                                                      |* USE00050
                                    6 *|Module:   USER1                                                       |* USE00060
                                    7 *|                                                                      |* USE00070
                                    8 *|          User 1 module.                                              |* USE00080
                                    9 *|                                                                      |* USE00090
                                   10 *|Function: Represents a user of the system.  It repetatively writes    |* USE00100
                                   11 *|          a message to the console and goes to sleep for a period      |* USE00110
                                   12 *|          of time.                                                     |* USE00120
                                   13 *|                                                                      |* USE00130
                                   14 *|Caller:   Oispatched when the system starts operation.                |* USE00140
                                   15 *|                                                                      |* USE00150
                                   16 *|Mode:     Key 6.                                                      |* USE00160
                                   17 *|                                                                      |* USE00170
                                   18 *+----------------------------------------------------------------------+* USE00180
                                   19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00190

                                   21          CSECTION USER1                                                    USE00210

                                   23+         PUNCH 'SPB'             Make module start on a page boundry.      CSE00040

000000                             25+USER1    CSECT                    Establish R12 as module overall base:    CSE00060

000000 05C0                        27+         BALR  R12,0              Give R12 a temporary base value;         CSE00080
                         00002     28+         USING *,R12              R12 now points @ SL instruction          CSE00090
000002 5FC0 C04E       00050       29+         SL    R12,=A(*-USER1)    R12 = R12 - offset from CSECT start      CSE00100
                                   30+         OROP  R12                R12 doesn't point at SL anymore;         CSE00110

                         00000     32+         USING USER1,R12          Now R12 has A(This CSECT) in it.         CSE00130

000006 47F0 C012       00012       34+         B     AROUND             Jump past symbolic module identifier.    CSE00150
00000A E4E2C509F1404040            35+MODID    OC    CL8'USER1'         Handy when poking around in storage.     CSE00160
000012                             36+AROUND   DS    OH                 End of CSECTION expansion.                CSE00170

000012 021F 0000 C030 00000 00030  38          MVC   O(L'SLEEPMSG,RO),SLEEPMSG Move message to parameter page    USE00230

000018                             40 NITENITE DS    OH            Main user loop:                             USE00250

000018 4100 0020       00020       42          LA    RO,L'SLEEPMSG Get length of message.                      USE00270
00001C 0A04                        43          SVC   4             Write message to console.                   USE00280

00001E 9801 C028       00028       45          LM    RO,R1,SLEEPY  Get nap time.                               USE00300
000022 0A05                        46          SVC   5             Go to sleep.                                USE00310

000024 47F0 C018       00018       48          B     NITENITE      Do everything all over again.               USE00330

000028                             50          OS    OD            Align constant area.                        USE00350
000028 0000000001312000            51 SLEEPY   DC    F'0',F'20000000' Twenty million microseconds!             USE00360
000030 C79689958740A396            52 SLEEPMSG DC    C'Going to sleep for 20 seconds...'                       USE00370
000050                             53          LTORG                                                           USE00380
000050 00000002                    54                =A(*-USER1)
                                   55          PRINT OFF                                                       USE00390
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00036	00034
NITENITE	00002	00000018	00040	00048
R0	00001	00000000	00070	00038 00042 00045
R1	00001	00000001	00071	00045
R12	00001	0000000C	00082	00027 00028 00029 00030 00032
SLEEPMSG	00032	00000030	00052	00038 00038 00042
SLEEPY	00004	00000028	00051	00045
USER1	00001	00000000	00025	00032 00054

```
  LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                          ASM 0201 20.58 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                      4 *+-------------------------------------------------------------------------+* USE00040
                                      5 *|                                                                         |* USE00050
                                      6 *|Module:   USER2                                                          |* USE00060
                                      7 *|                                                                         |* USE00070
                                      8 *|          User 2 module.                                                 |* USE00080
                                      9 *|                                                                         |* USE00090
                                     10 *|Function: Represents a user of the system.  It repetatively writes       |* USE00100
                                     11 *|          a message to the console and goes to sleep for a period       |* USE00110
                                     12 *|          of time.                                                       |* USE00120
                                     13 *|                                                                         |* USE00130
                                     14 *|Caller:   Oispatched when the system starts operation.                   |* USE00140
                                     15 *|                                                                         |* USE00150
                                     16 *|Mode:     Key 6.                                                         |* USE00160
                                     17 *|                                                                         |* USE00170
                                     18 *+-------------------------------------------------------------------------+* USE00180
                                     19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00190

                                     21          CSECTION USER2                                            USE00210

                                     23+         PUNCH 'SPB'             Make module start on a page boundry.  CSE00040

000000                               25+USER2    CSECT                   Establish R12 as module overall base: CSE00060

000000 05C0                          27+         BALR  R12,0             Give R12 a temporary base value;      CSE00080
                            00002    28+         USING *,R12             R12 now points @ SL instruction       CSE00090
000002 5FC0 C046      00048          29+         SL    R12,=A(*-USER2)   R12 = R12 - offset from CSECT start   CSE00100
                                     30+         OROP  R12               R12 doesn't point at SL anymore;      CSE00110

                            00000    32+         USING USER2,R12         Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012          34+         B     AROUNO            Jump past symbolic module identifier. CSE00150
00000A E4E2C509F2404040              35+M0010    OC    CL8'USER2'        Handy when poking around in storage.  CSE00160
000012                               36+AROUNO   OS    0H                End of CSECTION expansion.            CSE00170

000012 0217 0000 C030 00000 00030    38          MVC   0(L'MSG,R0),MSG Move message to parameter page.         USE00230

000018                               40 NITENITE OS    0H             Main user loop:                          USE00250

000018 4100 0018      00018          42          LA    R0,L'MSG       Get length of message.                   USE00270
00001C 0A04                          43          SVC   4              Write message to console.                USE00280

00001E 9801 C028      00028          45          LM    R0,R1,NAPTIME Get nap time.                             USE00300
000022 0A05                          46          SVC   5              Go to sleep.                             USE00310

000024 47F0 C018      00018          48          B     NITENITE       Oo everything all over again.            USE00330

000028                               50          OS    0O             Align constant area.                     USE00350
000028 0000000003938700              51 NAPTIME  OC    F'0',F'60000000' Sixty million microseconds!            USE00360
000030 E689939340A29385              52 MSG      OC    C'Will sleep for 1 minute.'                             USE00370
000048                               53          LTORG                                                         USE00380
000048 00000002                      54                =A(*-USER2)
                                     55          PRINT OFF                                                     USE00390
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00036	00034
MSG	00024	00000030	00052	00038 00038 00042
NAPTIME	00004	00000028	00051	00045
NITENITE	00002	00000018	00040	00048
R0	00001	00000000	00070	00038 00042 00045
R1	00001	00000001	00071	00045
R12	00001	0000000C	00082	00027 00028 00029 00030 00032
USER2	00001	00000000	00025	00032 00054

```
  LOC  O8JECT CODE    AODR1 ADDR2  STMT   SOURCE STATEMENT                                      ASM 0201 20.58 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                      4 *+----------------------------------------------------------------------+* USE00040
                                      5 *|                                                                      |* USE00050
                                      6 *|Module:   USER3                                                       |* USE00060
                                      7 *|                                                                      |* USE00070
                                      8 *|          User 3 module.                                              |* USE00080
                                      9 *|                                                                      |* USE00090
                                     10 *|Function: Represents a user of the system.  It does absolutely        |* USE00100
                                     11 *|          nothing but loop to burn time.                              |* USE00110
                                     12 *|                                                                      |* USE00120
                                     13 *|Caller:   Dispatched when the system starts operation.                |* USE00130
                                     14 *|                                                                      |* USE00140
                                     15 *|Mode:     Key 6.                                                      |* USE00150
                                     16 *|                                                                      |* USE00160
                                     17 *+----------------------------------------------------------------------+* USE00170
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00180

                                     20          CSECTION USER3                                                  USE00200

                                     22+         PUNCH 'SP8'              Make module start on a page boundry.  CSE00040

000000                               24+USER3    CSECT                      Establish R12 as module overall base: CSE00060

000000 05C0                          26+         8ALR  R12,0                Give R12 a temporary base value;      CSE00080
                            00002    27+         USING *,R12                R12 now points @ SL instruction       CSE00090
000002 5FC0 C016      00018          28+         SL    R12,=A(*-USER3)      R12 = R12 - offset from CSECT start   CSE00100
                                     29+         DROP  R12                  R12 doesn't point at SL anymore;      CSE00110

                            00000    31+         USING USER3,R12            Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012          33+         8     AROUND               Jump past symbolic module identifier. CSE00150
00000A E4E2C5D9F3404040              34+MODID    DC    CL8'USER3'           Handy when poking around in storage.  CSE00160
000012                               35+AROUND   DS    OH                   End of CSECTION expansion.            CSE00170

000012 47F0 C012      00012          37          8     *              A completely mindless load on the system.  USE00220

000018                               39          LTORG                                                           USE00240
000018 00000002                      40                =A(*-USER3)
                                     41          PRINT OFF                                                       USE00250
```

ASM 0201 20.58 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00035	00033
R12	00001	0000000C	00068	00026 00027 00028 00029 00031
USER3	00001	00000000	00024	00031 00040

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*> USE00030
                                    4 *+------------------------------------------------------------------------------------------+* USE00040
                                    5 *|                                                                                          |* USE00050
                                    6 *|Module:   USER4                                                                           |* USE00060
                                    7 *|                                                                                          |* USE00070
                                    8 *|          User 4 module.                                                                  |* USE00080
                                    9 *|                                                                                          |* USE00090
                                   10 *|Function: Represents a user of the system.  It does absolutely                           |* USE00100
                                   11 *|          nothing but expend time.                                                        |* USE00110
                                   12 *|                                                                                          |* USE00120
                                   13 *|Caller:   Dispatched when the system starts operation.                                    |* USE00130
                                   14 *|                                                                                          |* USE00140
                                   15 *|Mode:     Key 6.                                                                          |* USE00150
                                   16 *|                                                                                          |* USE00160
                                   17 *+------------------------------------------------------------------------------------------+* USE00170
                                   18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*> USE00180

                                   20          CSECTION USER4                                                                   USE00200

                                   22+         PUNCH 'SPB'             Make module start on a page boundry.  CSE00040

000000                             24+USER4    CSECT                    Establish R12 as module overall base: CSE00060

000000 05C0                        26+         BALR  R12,0              Give R12 a temporary base value;      CSE00080
                         00002     27+         USING *,R12              R12 now points @ SL instruction       CSE00090
000002 5FC0 C02E         00030     28+         SL    R12,=A(*-USER4)    R12 = R12 - offset from CSECT start   CSE00100
                                   29+         DROP  R12                R12 doesn't point at SL anymore;      CSE00110

                         00000     31+         USING USER4,R12          Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012         00012     33+         B     AROUND             Jump past symbolic module identifier. CSE00150
00000A E4E2C5D9F4404040            34+MODID    DC    CL8'USER4'         Handy when poking around in storage.  CSE00160
000012                             35+AROUND   DS    0H                 End of CSECTION expansion.             CSE00170

000012                             37 NONSENSE DS    0H           A completely mindless load on the system.  USE00220

000012 1755                        39          XR    R5,R5                                                   USE00240
000014 9107 C012         00012     40          TM    NONSENSE,X'07'                                          USE00250
000018 47F0 C01C         0001C     41          B     NEXT                                                    USE00260

00001C                             43 NEXT     DS    0H                                                      USE00280

00001C D506 C012 C01C 00012 0001C  45          CLC   NONSENSE(7),NEXT                                        USE00300
000022 1950                        46          CR    R5,R0                                                   USE00310
000024 1299                        47          LTR   R9,R9                                                   USE00320
000026 47F0 C012         00012     48          B     NONSENSE                                                USE00330

000030                             50          LTORG                                                         USE00350
000030 00000002                    51                =A(*-USER4)
                                   52          PRINT OFF                                                     USE00360
```

ASM 0201 20.58 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00035	00033
NEXT	00002	0000001C	00043	00041 00045
NONSENSE	00002	00000012	00037	00040 00045 00048
R0	00001	00000000	00067	00046
R12	00001	0000000C	00079	00026 00027 00028 00029 00031
R5	00001	00000005	00072	00039 00039 00046
R9	00001	00000009	00076	00047 00047
USER4	00001	00000000	00024	00031 00051

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          ASM 0201 20.58 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                      4 *+----------------------------------------------------------------------+* USE00040
                                      5 *|                                                                      |* USE00050
                                      6 *|Module:   USER5                                                       |* USE00060
                                      7 *|                                                                      |* USE00070
                                      8 *|          User 5 module.                                              |* USE00080
                                      9 *|                                                                      |* USE00090
                                     10 *|Function: Represents a malicious user of the system.  It tries to     |* USE00100
                                     11 *|          read from the console which is a privileged service.        |* USE00110
                                     12 *|          The penalty for this action will be termination.            |* USE00120
                                     13 *|                                                                      |* USE00130
                                     14 *|Caller:   Dispatched when the system starts operation.                |* USE00140
                                     15 *|                                                                      |* USE00150
                                     16 *|Mode:     Key 6.                                                      |* USE00160
                                     17 *|                                                                      |* USE00170
                                     18 *+----------------------------------------------------------------------+* USE00180
                                     19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00190

                                     21          CSECTION USER5      Malicious user 5.                           USE00210

                                     23+         PUNCH 'SPB'             Make module start on a page boundry.  CSE00040

000000                               25+USER5    CSECT                   Establish R12 as module overall base: CSE00060

000000 05C0                          27+         BALR  R12,0             Give R12 a temporary base value;      CSE00080
                            00002    28+         USING *,R12             R12 now points @ SL instruction       CSE00090
000002 5FC0 C05E      00060          29+         SL    R12,=A(*-USER5)   R12 = R12 - offset from CSECT start   CSE00100
                                     30+         DROP  R12               R12 doesn't point at SL anymore;      CSE00110

                            00000    32+         USING USER5,R12         Now R12 has A(This CSECT) in it.      CSE00130

000006 47F0 C012      00012          34+         B     AROUND            Jump past symbolic module identifier. CSE00150
00000A E4E2C509F5404040              35+MODID    DC    CL8'USER5'        Handy when poking around in storage.  CSE00160
000012                               36+AROUNO   DS    0H                End of CSECTION expansion.            CSE00170

000012                               38 READ     DS    0H            Get ourselves killed:                     USE00230

000012 0A03                          40          SVC   3             Post console read request.                USE00250

000014 D231 0000 C028 00000 00028    42          MVC   0(L'HAHAMSG,R0),HAHAMSG Just in case we got through!    USE00270
00001A 4100 0032      00032          43          LA    R0,L'HAHAMSG  Get length of message.                    USE00280
00001E 0A04                          44          SVC   4             Write message to console.                 USE00290

000020 47F0 C012      00012          46          B     REAO                                                    USE00310

000028                               48          DS    0D            Align constant area.                      USE00330
000028 C88140C8816B40A8              49 HAHAMSG  DC    C'Ha Ha, your operating system project doesn''t work!'  USE00340
000060                               50          LTORG                                                         USE00350
000060 00000002                      51                =A(*-USER5)
                                     52          PRINT OFF                                                     USE00360
```

ASM 0201 20.58 12/04/86

SYMBOL	LEN	VALUE	DEFN	REFERENCES
AROUND	00002	00000012	00036	00034
HAHAMSG	00050	00000028	00049	00042 00042 00043
READ	00002	00000012	00038	00046
R0	00001	00000000	00067	00042 00043
R12	00001	0000000C	00079	00027 00028 00029 00030 00032
USER5	00001	00000000	00025	00032 00051

```
  LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                       ASM 0201 20.46 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SCH00040
                                      4 *+-----------------------------------------------------------------------+* SCH00050
                                      5 *|                                                                       |* SCH00060
                                      6 *|Module:   SCHPP                                                        |* SCH00070
                                      7 *|                                                                       |* SCH00080
                                      8 *|          Scheduler Parameter Page.                                    |* SCH00090
                                      9 *|                                                                       |* SCH00100
                                     10 *|Function: Declares an area of storage for the transmission of          |* SCH00110
                                     11 *|          data into and out of SCH.   .                                |* SCH00120
                                     12 *|                                                                       |* SCH00130
                                     13 *|Caller:   Not called.  Contains no executable code.                    |* SCH00140
                                     14 *|                                                                       |* SCH00150
                                     15 *|Mode:     Key 5.                                                       |* SCH00160
                                     16 *|                                                                       |* SCH00170
                                     17 *+-----------------------------------------------------------------------+* SCH00180
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SCH00190

                                     20          PUNCH 'SPB'       Make module start on a page boundry.        SCHOC210

000000                               22 SCHPP    CSECT              Scheduler Parameter Page.                  SCH00230
000000                               23          DS    XL4096                                                  SCH00240
                                     24          ENO                                                           SCH00250
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          ASM 0201 20.44 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* RLA00040
                                      4 *+----------------------------------------------------------------------+* RLA00050
                                      5 *|                                                                      |* RLA00060
                                      6 *|Module:   RLASPP                                                      |* RLA00070
                                      7 *|                                                                      |* RLA00080
                                      8 *|          Run List Address Space Parameter Page.                      |* RLA00090
                                      9 *|                                                                      |* RLA00100
                                     10 *|Function: Oeclares an area of storage for the transmission of        |* RLA00110
                                     11 *|          data into and out of Run List Address Space.                |* RLA00120
                                     12 *|                                                                      |* RLA00130
                                     13 *|Caller:   Not called.  Contains no executable code.                   |* RLA00140
                                     14 *|                                                                      |* RLA00150
                                     15 *|Mode:     Key 5.                                                      |* RLA00160
                                     16 *|                                                                      |* RLA00170
                                     17 *+----------------------------------------------------------------------+* RLA00180
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* RLA00190

                                     20          PUNCH 'SPB'        Make module start on a page boundry.        RLA00210

000000                               22 RLASPP   CSECT               Run List A. S. Parameter Page.              RLA00230
000000                               23          DS    XL4096                                                    RLA00240
                                     24          END                                                             RLA00250
```

```
  LOC  OBJECT CODE    ADDR1 ADOR2  STMT   SOURCE STATEMENT                                          ASM 0201 20.33 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* OSP00040
                                      4 *+-------------------------------------------------------------------+* DSP00050
                                      5 *|                                                                   |* DSP00060
                                      6 *|Module:   OSPPP                                                    |* DSP00070
                                      7 *|                                                                   |* DSP00080
                                      8 *|          Oispatcher Parameter Page.                               |* DSP00090
                                      9 *|                                                                   |* DSP00100
                                     10 *|Function: Oeclares an area of storage for the transmission of      |* DSP00110
                                     11 *|          data into OSP address space.                             |* DSP00120
                                     12 *|                                                                   |* DSP00130
                                     13 *|Caller:   Not called.  Contains no executable code.                |* DSP00140
                                     14 *|                                                                   |* DSP00150
                                     15 *|Mode:     Key 5.                                                   |* DSP00160
                                     16 *|                                                                   |* DSP00170
                                     17 *+-------------------------------------------------------------------+* DSP00180
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* DSP00190

                                     20          PUNCH 'SPB'        Make module start on a page boundry.        DSP00210

000000                               22 DSPPP    CSECT               Dispatcher Parameter Page.                 DSP00230
000000                               23          DS    XL4096                                                   DSP00240
                                     24          END                                                            DSP00250
```

```
  LOC  DBJECT CODE    ADDR1 ADDR2  STMT   SDURCE STATEMENT                                  ASM D2D1 2D.29 12/D4/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* AUTDDD3D
                                      4 *+-------------------------------------------------------------------+* AUTDDD4D
                                      5 *|                                                                   |* AUTDDD5D
                                      6 *|Module:   AUTHPP                                                   |* AUTDDD6D
                                      7 *|                                                                   |* AUTDDD7D
                                      8 *|          Authentication Routine's Parameter Page.                 |* AUTDDD8D
                                      9 *|                                                                   |* AUTDDD9D
                                     1D *|Function: Declares an area of storage for the transmission of      |* AUTDD1DD
                                     11 *|          data from SVCFLIH to AUTH.                               |* AUTDD11D
                                     12 *|                                                                   |* AUTDD12D
                                     13 *|Caller:   Not called.  Contains no executable code.                |* AUTDD13D
                                     14 *|                                                                   |* AUTDD14D
                                     15 *|Mode:     Key 1 = SVCFLIH key.                                     |* AUTDD15D
                                     16 *|                                                                   |* AUTDD16D
                                     17 *+-------------------------------------------------------------------+* AUTDD17D
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* AUTDD18D

                                     2D          PUNCH 'SP8'       Make module start on a page boundry.       AUTDD2DD

DDDDDD                               22 AUTHPP   CSECT              Authentication Routine's Parameter Page.   AUTDD22D
DDDDDD                               23          DS    XL4D96                                                  AUTDD23D
                                     24          END                                                           AUTDD24D
```

```
  LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                              ASM 0201 20.51 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLE00040
                                      4 *+---------------------------------------------------------------------+* SLE00050
                                      5 *|                                                                     |* SLE00060
                                      6 *|Module:   SLEEPPP                                                    |* SLE00070
                                      7 *|                                                                     |* SLE00080
                                      8 *|          Sleep Address Space Parameter Page.                        |* SLE00090
                                      9 *|                                                                     |* SLE00100
                                     10 *|Function: Oeclares an area of storage for the transmission of        |* SLE00110
                                     11 *|          data into and out of Sleep Address Space.                  |* SLE00120
                                     12 *|                                                                     |* SLE00130
                                     13 *|Caller:   Not called.  Contains no executable code.                  |* SLE00140
                                     14 *|                                                                     |* SLE00150
                                     15 *|Mode:     Key 5.                                                     |* SLE00160
                                     16 *|                                                                     |* SLE00170
                                     17 *+---------------------------------------------------------------------+* SLE00180
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLE00190

                                     20          PUNCH 'SP8'        Make module start on a page boundry.        SLE00210

000000                               22 SLEEPPP  CSECT               Sleep Parameter Page.                      SLE00230
000000                               23          OS    XL4096                                                   SLE00240
                                     24          ENO                                                            SLE00250
```

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRU00040
                                    4 *+-----------------------------------------------------------------------+* TRU00050
                                    5 *|                                                                       |* TRU00060
                                    6 *|Module:   TRUPP                                                        |* TRU00070
                                    7 *|                                                                       |* TRU00080
                                    8 *|          Terminate Running User Parameter Page.                       |* TRU00090
                                    9 *|                                                                       |* TRU00100
                                   10 *|Function: Declares an area of storage for the transmission of          |* TRU00110
                                   11 *|          data from PGMFLIH to TRU.                                     |* TRU00120
                                   12 *|                                                                       |* TRU00130
                                   13 *|Caller:   Not called.  Contains no executable code.                    |* TRU00140
                                   14 *|                                                                       |* TRU00150
                                   15 *|Mode:     Key 5.                                                       |* TRU00160
                                   16 *|                                                                       |* TRU00170
                                   17 *+-----------------------------------------------------------------------+* TRU00180
                                   18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRU00190

                                   20          PUNCH 'SP8'        Make module start on a page boundry.       TRU00210

000000                             22 TRUPP    CSECT               Terminate Running User Parameter Page.     TRU00230
000000                             23          DS    XL4096                                                   TRU00240
                                   24          END                                                            TRU00250
```

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                    4 *+----------------------------------------------------------------------+* USE00040
                                    5 *|                                                                      |* USE00050
                                    6 *|Module:   USER1PP                                                     |* USE00060
                                    7 *|                                                                      |* USE00070
                                    8 *|          User 1's Parameter Page.                                    |* USE00080
                                    9 *|                                                                      |* USE00090
                                   10 *|Function: Declares an area of storage for the transmission of         |* USE00100
                                   11 *|          data from user to system and vice-versa.                     |* USE00110
                                   12 *|                                                                      |* USE00120
                                   13 *|Caller:   Not called.  Contains no executable code.                   |* USE00130
                                   14 *|                                                                      |* USE00140
                                   15 *|Mode:     Key 6.                                                      |* USE00150
                                   16 *|                                                                      |* USE00160
                                   17 *+----------------------------------------------------------------------+* USE00170
                                   18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00180

                                   20          PUNCH 'SP8'        Make module start on a page boundry.       USE00200

000000                             22 USER1PP  CSECT               User 1's Parameter Page.                   USE00220
000000                             23          DS    XL4096                                                   USE00230
                                   24          END                                                            USE00240
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                  ASM 0201 18.30 11/22/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                      4 *+----------------------------------------------------------------------+* USE00040
                                      5 *|                                                                      |* USE00050
                                      6 *|Module:    USER2PP                                                    |* USE00060
                                      7 *|                                                                      |* USE00070
                                      8 *|           User 2's Parameter Page.                                   |* USE00080
                                      9 *|                                                                      |* USE00090
                                     10 *|Function:  Declares an area of storage for the transmission of        |* USE00100
                                     11 *|           data from user to system and vice-versa.                   |* USE00110
                                     12 *|                                                                      |* USE00120
                                     13 *|Caller:    Not called.  Contains no executable code.                  |* USE00130
                                     14 *|                                                                      |* USE00140
                                     15 *|Mode:      Key 6.                                                     |* USE00150
                                     16 *|                                                                      |* USE00160
                                     17 *+----------------------------------------------------------------------+* USE00170
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00180

                                     20          PUNCH 'SPB'        Make module start on a page boundry.        USE00200

 00000D                              22 USER2PP  CSECT                User 2's Parameter Page.                  USE00220
 000000                              23          DS    XL4096                                                   USE00230
                                     24          END                                                            USE00240
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                              ASM 0201 18.30 11/22/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                      4 *+-----------------------------------------------------------------------+* USE00040
                                      5 *|                                                                       |* USE00050
                                      6 *|Module:   USER3PP                                                      |* USE00060
                                      7 *|                                                                       |* USE00070
                                      8 *|          User 3's Parameter Page.                                     |* USE00080
                                      9 *|                                                                       |* USE00090
                                     10 *|Function: Declares an area of storage for the transmission of          |* USE00100
                                     11 *|          data from user to system and vice-versa.                     |* USE00110
                                     12 *|                                                                       |* USE00120
                                     13 *|Caller:   Not called.  Contains no executable code.                    |* USE00130
                                     14 *|                                                                       |* USE00140
                                     15 *|Mode:     Key 6.                                                       |* USE00150
                                     16 *|                                                                       |* USE00160
                                     17 *+-----------------------------------------------------------------------+* USE00170
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00180

                                     20          PUNCH 'SPB'        Make module start on a page boundry.          USE00200

000000                               22 USER3PP  CSECT               User 3's Parameter Page.                     USE00220
000000                               23          DS    XL4096                                                     USE00230
                                     24          END                                                              USE00240
```

```
  LOC  OBJECT CODE    A00R1 A00R2  STMT   SOURCE STATEMENT                                ASM 0201 18.30 11/22/86

                                       3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                       4 *+----------------------------------------------------------------------+* USE00040
                                       5 *|                                                                      |* USE00050
                                       6 *|Module:   USER4PP                                                     |* USE00060
                                       7 *|                                                                      |* USE00070
                                       8 *|          User 4's Parameter Page.                                    |* USE00080
                                       9 *|                                                                      |* USE00090
                                      10 *|Function: Oeclares an area of storage for the transmission of         |* USE00100
                                      11 *|          data from user to system and vice-versa.                     |* USE00110
                                      12 *|                                                                      |* USE00120
                                      13 *|Caller:   Not called.  Contains no executable code.                   |* USE00130
                                      14 *|                                                                      |* USE00140
                                      15 *|Mode:     Key 6.                                                      |* USE00150
                                      16 *|                                                                      |* USE00160
                                      17 *+----------------------------------------------------------------------+* USE00170
                                      18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00180

                                      20          PUNCH 'SPB'        Make module start on a page boundry.       USE00200

000000                                22 USER4PP  CSECT               User 4's Parameter Page.                   USE00220
000000                                23          OS    XL4096                                                   USE00230
                                      24          ENO                                                            USE00240
```

```
  LOC  OBJECT COOE    AOOR1 AOOR2  STMT   SOURCE STATEMENT                                        ASM 0201 15.33 11/23/86

                                     3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00030
                                     4 *+-----------------------------------------------------------------------+* USE00040
                                     5 *|                                                                       |* USE00050
                                     6 *|Module:   USER5PP                                                      |* USE00060
                                     7 *|                                                                       |* USE00070
                                     8 *|          User 5's Parameter Page.                                     |* USE00080
                                     9 *|                                                                       |* USE00090
                                    10 *|Function: Oeclares an area of storage for the transmission of          |* USE00100
                                    11 *|          data from user to system and vice-versa.                     |* USE00110
                                    12 *|                                                                       |* USE00120
                                    13 *|Caller:   Not called.  Contains no executable code.                    |* USE00130
                                    14 *|                                                                       |* USE00140
                                    15 *|Mode:     Key 6.                                                       |* USE00150
                                    16 *|                                                                       |* USE00160
                                    17 *+-----------------------------------------------------------------------+* USE00170
                                    18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* USE00180

                                    20          PUNCH 'SPB'        Make module start on a page boundry.        USE00200

000000                              22 USER5PP  CSECT               User 5's Parameter Page.                   USE00220
000000                              23          OS    XL4096                                                   USE00230
                                    24          ENO                                                            USE00240
```

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT ASM 0201 20.35 12/04/86

```
                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* EXT00040
                                    4 *+-------------------------------------------------------------------------+* EXT00050
                                    5 *|                                                                         |* EXT00060
                                    6 *|Module:   EXTSPP                                                         |* EXT00070
                                    7 *|                                                                         |* EXT00080
                                    8 *|          External Interrupt Second-Level Interrupt Handler              |* EXT00090
                                    9 *|            Parameter Page.                                              |* EXT00100
                                   10 *|                                                                         |* EXT00110
                                   11 *|Function: Declares an area of storage for the transmission of            |* EXT00120
                                   12 *|          data from EXTFLIH to EXTSLIH.                                  |* EXT00130
                                   13 *|                                                                         |* EXT00140
                                   14 *|Caller:   Not called.  Contains no executable code.                      |* EXT00150
                                   15 *|                                                                         |* EXT00160
                                   16 *|Mode:     Key 2 = EXTFLIH key.                                           |* EXT00170
                                   17 *|                                                                         |* EXT00180
                                   18 *+-------------------------------------------------------------------------+* EXT00190
                                   19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* EXT00200

                                   21          PUNCH 'SP8'        Make module start on a page boundry.       EXT00220

000000                             23 EXTSPP   CSECT                 External Interrupt Parameter Page.         EXT00240
000000                             24          DS    XL4096                                                    EXT00250
                                   25          END                                                             EXT00260
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          ASM 0201 20.41 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* OPP00040
                                      4 *+----------------------------------------------------------------------+* OPP00050
                                      5 *|                                                                      |* OPP00060
                                      6 *|Module:   OPPP                                                        |* OPP00070
                                      7 *|                                                                      |* OPP00080
                                      8 *|          Operator Address Space Parameter Page.                      |* OPP00090
                                      9 *|                                                                      |* OPP00100
                                     10 *|Function: Declares an area of storage for the transmission of         |* OPP00110
                                     11 *|          data from Operator Address Space to system and vice-        |* OPP00120
                                     12 *|          versa.                                                      |* OPP00130
                                     13 *|                                                                      |* OPP00140
                                     14 *|Caller:   Not called.  Contains no executable code.                   |* OPP00150
                                     15 *|                                                                      |* OPP00160
                                     16 *|Mode:     Key 6.                                                      |* OPP00170
                                     17 *|                                                                      |* OPP00180
                                     18 *+----------------------------------------------------------------------+* OPP00190
                                     19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* OPP00200

                                     21          PUNCH 'SP8'          Make module start on a page boundry.        OPP00220

000000                               23 OPPP     CSECT                 Operator Address Space Parameter Page.     OPP00240
000000                               24          DS    XL4096                                                     OPP00250
                                     25          END                                                              OPP00260
```

```
                                                                              ASM 0201 20.50 12/04/86
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLA00040
                                      4 *+--------------------------------------------------------------------+* SLA00050
                                      5 *|                                                                    |* SLA00060
                                      6 *|Module:   SLASPP                                                    |* SLA00070
                                      7 *|                                                                    |* SLA00080
                                      8 *|          Sleep List Address Space Parameter Page.                  |* SLA00090
                                      9 *|                                                                    |* SLA00100
                                     10 *|Function: Declares an area of storage for the transmission of      |* SLA00110
                                     11 *|          data into and out of Sleep List Address Space.            |* SLA00120
                                     12 *|                                                                    |* SLA00130
                                     13 *|Caller:   Not called.  Contains no executable code.                 |* SLA00140
                                     14 *|                                                                    |* SLA00150
                                     15 *|Mode:     Key 5.                                                    |* SLA00160
                                     16 *|                                                                    |* SLA00170
                                     17 *+--------------------------------------------------------------------+* SLA00180
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* SLA00190

                                     20          PUNCH 'SPB'        Make module start on a page boundry.      SLA00210

000000                               22 SLASPP   CSECT               Sleep List A. S. Parameter Page.          SLA00230
000000                               23          DS    XL4096                                                   SLA00240
                                     24          ENO                                                            SLA00250
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                     ASM 0201 20.54 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TLA00040
                                      4 *+---------------------------------------------------------------------+* TLA00050
                                      5 *|                                                                     |* TLA00060
                                      6 *|Module:   TLASPP                                                     |* TLA00070
                                      7 *|                                                                     |* TLA00080
                                      8 *|          Terminate List Address Space Parameter Page.               |* TLA00090
                                      9 *|                                                                     |* TLA00100
                                     10 *|Function: Oeclares an area of storage for the transmission of        |* TLA00110
                                     11 *|          data into and out of Terminate List Address Space.         |* TLA00120
                                     12 *|                                                                     |* TLA00130
                                     13 *|Caller:   Not called.  Contains no executable code.                  |* TLA00140
                                     14 *|                                                                     |* TLA00150
                                     15 *|Mode:     Key 5.                                                     |* TLA00160
                                     16 *|                                                                     |* TLA00170
                                     17 *+---------------------------------------------------------------------+* TLA00180
                                     18 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TLA00190

                                     20          PUNCH 'SPB'       Make module start on a page boundry.       TLA00210

000000                               22 TLASPP   CSECT              Terminate List A. S. Parameter Page.       TLA00230
000000                               23          DS    XL4096                                                  TLA00240
                                     24          ENO                                                           TLA00250
```

```
  LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                          ASM 0201 20.55 12/04/86

                                      3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRA00040
                                      4 *+-----------------------------------------------------------------------+* TRA00050
                                      5 *|                                                                       |* TRA00060
                                      6 *|Module:   TRACEPP                                                      |* TRA00070
                                      7 *|                                                                       |* TRA00080
                                      8 *|          System Trace Table Routine's Parameter Page.                 |* TRA00090
                                      9 *|                                                                       |* TRA00100
                                     10 *|Function: Declares an area of storage for the transmission of a        |* TRA00110
                                     11 *|          trace table entry from (most) any address space to           |* TRA00120
                                     12 *|          the Trace Table address space.                               |* TRA00130
                                     13 *|                                                                       |* TRA00140
                                     14 *|Caller:   Not called.  Contains no executable code.                    |* TRA00150
                                     15 *|                                                                       |* TRA00160
                                     16 *|Mode:     Key 5.                                                       |* TRA00170
                                     17 *|                                                                       |* TRA00180
                                     18 *+-----------------------------------------------------------------------+* TRA00190
                                     19 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* TRA00200

                                     21          PUNCH 'SP8'        Make module start on a page boundry.          TRA00220

000000                               23 TRACEPP  CSECT               Trace Table Routine's Parameter Page.        TRA00240
000000                               24          DS    XL4096                                                     TRA00250
                                     25          END                                                              TRA00260
```

```
 LOC  OBJECT CODE    ADDR1 ADDR2  STMT   SOURCE STATEMENT                                   ASM 0201 20.31 12/04/86

                                    3 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* CONC0040
                                    4 *+-----------------------------------------------------------------------+* CON00050
                                    5 *|                                                                       |* CON00060
                                    6 *|Module:   CONSPP                                                       |* CON00070
                                    7 *|                                                                       |* CON00080
                                    8 *|          Console Address Space Parameter Page.                        |* CON00090
                                    9 *|                                                                       |* CON00100
                                   10 *|Function: Declares an area of storage for the transmission of          |* CON00110
                                   11 *|          data into and out of Console Address Space.  Also serves     |* CON00120
                                   12 *|          as the console I/O buffer, and all I/O operations take       |* CON00130
                                   13 *|          place through this page because the data address of the      |* CON00140
                                   14 *|          REAO/WRITE CCW in PGMFLIH points at it.                      |* CON00150
                                   15 *|                                                                       |* CON00160
                                   16 *|Caller:   Not called.  Contains no executable code.                    |* CON00170
                                   17 *|                                                                       |* CON00180
                                   18 *|Mode:     Key 3.                                                       |* CON00190
                                   19 *|                                                                       |* CON00200
                                   20 *+-----------------------------------------------------------------------+* CON00210
                                   21 *<*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*><*>* CON00220

                                   23          PUNCH 'SPB'        Make module start on a page boundry.       CON00240

000000                             25 CONSPP   CSECT              Console Address Space Parameter Page.      CON00260
000000                             26          DS    XL4096                                                 CON00270
                                   27          END                                                          CON00280
```

```
*COPY APP                                                              APP00010
*----------------------------------------------------------------------* APP00020
*                                                                      * APP00030
* Copy file: APP - Authentication Routine's Parameter Page Format.     * APP00040
*                                                                      * APP00050
* Purpose:   Provide a layout of the data in the Authentication        * APP00060
*            routine's parameter page.                                 * APP00070
*                                                                      * APP00080
* Used by:   SVC first level interrupt handler and Authentication      * APP00090
*            routine.                                                  * APP00100
*                                                                      * APP00110
*----------------------------------------------------------------------* APP00120
         SPACE                                                         APP00130
*                                                                      APP00140
*                  +0        +1        +2        +3                   APP00150
*          Offset: +---------+---------+---------+---------+          APP00160
*            X'0'  | APPSTATE| APPNIL1 |       APPCODE     |          APP00170
*                  +---------+---------+---------+---------+          APP00180
*           X'04'  |                 APPSTO                |          APP00190
*                  +---------+---------+---------+---------+          APP00200
*           X'08'  |                APPVADDR               |          APP00210
*                  +---------+---------+---------+---------+          APP00220
*           X'0C'  |                                       |          APP00230
*                  +                                       +          APP00240
*                  .                                       .          APP00250
*                  .               APPREGS                 .          APP00260
*                  .                                       .          APP00270
*                  +                                       +          APP00280
*           X'48'  |                                       |          APP00290
*                  +---------+---------+---------+---------+          APP00300
*                                                                      APP00310
APP      DSECT                                                         APP00320
APPSTATE DS    XL1          System processing state: See SYSTATE      APP00330
APPNIL1  DS    XL1          Unused                                    APP00340
APPCODE  DS    H            SVC interrupt code                        APP00350
APPSTO   DS    F            STO @ time of interrupt                   APP00360
APPVADDR DS    A            Resume virtual address                    APP00370
APPREGS  DS    16F          Registers @ time of interrupt             APP00380
```

```
*COPY EQUATES                                                          EQU00010
*----------------------------------------------------------------------* EQU00020
*                                                                      * EQU00030
* Copy file: EQUATES                                                   * EQU00040
*                                                                      * EQU00050
* Purpose:   Define commonly used (and in many cases public) symbols.  * EQU00060
*                                                                      * EQU00070
* Used by:   Any module.                                               * EQU00080
*                                                                      * EQU00090
*----------------------------------------------------------------------* EQU00100
         SPACE                                                         EQU00110
*                                                                      EQU00120
*   General register equates:                                          EQU00130
*                                                                      EQU00140
R0       EQU   0                                                       EQU00150
R1       EQU   1                                                       EQU00160
R2       EQU   2                                                       EQU00170
R3       EQU   3                                                       EQU00180
R4       EQU   4                                                       EQU00190
R5       EQU   5                                                       EQU00200
R6       EQU   6                                                       EQU00210
R7       EQU   7                                                       EQU00220
R8       EQU   8                                                       EQU00230
R9       EQU   9                                                       EQU00240
R10      EQU   10                                                      EQU00250
R11      EQU   11                                                      EQU00260
R12      EQU   12                                                      EQU00270
R13      EQU   13                                                      EQU00280
R14      EQU   14                                                      EQU00290
R15      EQU   15                                                      EQU00300
         SPACE                                                         EQU00310
*                                                                      EQU00320
*   Floating-point register equates:                                   EQU00330
*                                                                      EQU00340
F0       EQU   0                                                       EQU00350
F2       EQU   2                                                       EQU00360
F4       EQU   4                                                       EQU00370
F6       EQU   6                                                       EQU00380
         SPACE                                                         EQU00390
*                                                                      EQU00400
*   Values associated with the use of the Monitor Call instruction:    EQU00410
*                                                                      EQU00420
MCEVENT  EQU   X'40'      Program interrupt code indicating monitor     EQU00430
*                         event.                                       EQU00440
         SPACE                                                         EQU00450
*   The following are also indices into a branch table:                EQU00460
ACTAUTH  EQU   0          Monitor Call code: "Activate Auth"            EQU00470
SDP      EQU   1          Monitor Call code: "Set Dispatch Parameters"  EQU00480
DISPATCH EQU   2          Monitor Call code: "Dispatch"                 EQU00490
ABEND    EQU   3          Monitor Call code: "ABEND"                    EQU00500
BIND     EQU   4          Monitor Call code: "XMIT Bind Request"        EQU00510
DETACH   EQU   5          Monitor Call code: "XMIT Detach Request"      EQU00520
SCK      EQU   6          Monitor Call code: "Set Clock Comparator"     EQU00530
IDLE     EQU   7          Monitor Call code: "Idle No Work Wait"        EQU00540
SAVEREGS EQU   8          Monitor Call Code: "Save registers"           EQU00550
CONSIO   EQU   9          Monitor Call Code: "Console Start I/O"        EQU00560
PURGECON EQU   10         Monitor Call Code: "Purge Console Input Data" EQU00570
SHUTSYS  EQU   11         Monitor Call Code: "Shut Down the System"     EQU00580
         SPACE                                                         EQU00590
```

```
*                                                                      EQU00600
*   Values associated with the state of the System (SYSTATE):          EQU00610
*                                                                      EQU00620
RUNUSER  EQU   X'80'      System is running a user                     EQU00630
RUNSYS   EQU   X'40'      System is running a module of itself         EQU00640
RUNIDLE  EQU   X'20'      System is in a no-work idle wait state       EQU00650
*                                                                      EQU00660
*   Values associated with Dynamic Address Translation:               EQU00670
*                                                                      EQU00680
DATON    EQU   B'00000100' PSW mask to check for DAT off or on        EQU00690
```

```
*COPY PSA                                                              PSA00010
*----------------------------------------------------------------------* PSA00020
*                                                                      * PSA00030
* Copy file: PSA - Prefixed Storage Area                               * PSA00040
*                                                                      * PSA00050
* Purpose:   Associate symbolic names with low-core assigned storage   * PSA00060
*            locations and page 0 variables.                           * PSA00070
*                                                                      * PSA00080
* Used by:   Modules that need to access low storage and/or page 0:    * PSA00090
*              - System initialization code                            * PSA00100
*              - First-level interrupt handlers                        * PSA00110
*                                                                      * PSA00120
*----------------------------------------------------------------------* PSA00130
        SPACE                                                          PSA00140
*  System/370 Assigned Storage Locations used by this Control Program: PSA00150
*   Note: ---> Consult S/370 Principles of Operation for complete      PSA00160
*              description of assigned storage locations.              PSA00170
*                                                                      PSA00180
*    -             +0   +2   +4   +6   +8   +A   +C   +E               PSA00190
*   Real address:  +----+----+----+----+----+----+----+----+           PSA00200
*          X'0'    | / / / / / / / / / / / / / / / / / / / / |         PSA00210
*                  +---------+---------+---------+---------+         PSA00220
*         X'10'    | / / / / / / / / / / | External Old PSW  |         PSA00230
*                  +---------+---------+---------+---------+         PSA00240
*         X'20'    |    SVC Old PSW    | Program Old PSW   |         PSA00250
*                  +---------+---------+---------+---------+         PSA00260
*         X'30'    | / / / / / / / / / / |    I/O Old PSW    |       PSA00270
*                  +---------+---------+---------+---------+         PSA00280
*         X'40'    |        CSW        |   CAW   | / / / / |         PSA00290
*                  +---------+---------+---------+---------+         PSA00300
*         X'50'    | / / / / / / / / / / | External New PSW  |         PSA00310
*                  +---------+---------+---------+---------+         PSA00320
*         X'60'    |    SVC New PSW    | Program New PSW   |         PSA00330
*                  +---------+---------+---------+---------+         PSA00340
*         X'70'    | / / / / / / / / / / |    I/O New PSW    |       PSA00350
*                  +---------+----+----+----+----+----+----+         PSA00360
*         X'80'    | / / / / / / / /| EC |/ / | SC |/ / | PC |      PSA00370
*                  +---------+----+----+----+----+----+----+         PSA00380
*         X'90'    | / / / / / / / / / / / / / / / / | MC |/ / |      PSA00390
*                  +---------+---------+---------+----+----+         PSA00400
*         X'A0'    | / / / / / / / / / / / / / / / / / / / / |       PSA00410
*                  +---------+---------+----+----+----+----+         PSA00420
*         X'B0'    | / / / / / / / / / / / / / /| IC | / / / / |      PSA00430
*                  +---------+---------+----+----+----+----+         PSA00440
*                                                                      PSA00450
*         Notes:   EC = External interrupt code.                       PSA00460
*                  SC = SVC interrupt code.                            PSA00470
*                  PC = Program interrupt code.                        PSA00480
*                  MC = Monitor Call code.                             PSA00490
*                  IC = Interrupting I/O Device Address                PSA00500
*                                                                      PSA00510
*  All storage following MC up to the Page 0 Variables section is      PSA00520
*  unused.                                                             PSA00530
*                                                                      PSA00540
*                                                                      PSA00550
*  Page 0 Variables for use by the Control Program:                    PSA00560
*                                                                      PSA00570
*   Real address:  +----+----+----+----+----+----+----+----+           PSA00580
*         X'400'   |                                       |           PSA00590
```

```
*                +              Savearea for               +            PSA00600
*         X'410' |                                         |            PSA00610
*                +            General  Purpose             +            PSA00620
*         X'420' |                                         |            PSA00630
*                +              Registers                  +            PSA00640
*         X'430' |                                         |            PSA00650
*                +---------+---------+---------+---------+            PSA00660
*         X'440' |              Savearea for                |            PSA00670
*                +            Floating-Point               +            PSA00680
*         X'450' |               Registers                  |            PSA00690
*                +---------+---------+---------+---------+            PSA00700
*         X'460' | CPU Timer Savearea| SYSTATE | SYSSTO  |            PSA00710
*                +---------+---------+---------+---------+            PSA00720
*         X'470' |      SYSPSW       | DUCBRA  | USRSTO  |            PSA00730
*                +---------+---------+---------+---------+            PSA00740
*         X'480' |      USRPSW       |     USRTIMER      |            PSA00750
*                +---------+---------+---------+---------+            PSA00760
*         X'490' |INTSTATE |INTCREG1 |       SAVPSW      |            PSA00770
*                +---------+---------+---------+---------+            PSA00780
*         X'4A0' |                                         |            PSA00790
*                +              Savearea for               +            PSA00800
*         X'4B0' |                                         |            PSA00810
*                +         User General Purpose            +            PSA00820
*         X'4C0' |                                         |            PSA00830
*                +              Registers                  +            PSA00840
*         X'4D0' |                                         |            PSA00850
*                +---------+---------+---------+---------+            PSA00860
*         X'4E0' |              Savearea for                |            PSA00870
*                +           User Floating Point           +            PSA00880
*         X'4F0' |               Registers                  |            PSA00890
*                +---------+---------+---------+---------+            PSA00900
*         X'500' |      SAVTIMER     |                                 PSA00910
*                +---------+---------+                                 PSA00920
*                                                                      PSA00930
PSA      DSECT                                                         PSA00940
         ORG   PSA+X'18'                                               PSA00950
EXTOLD   DS    D             External interrupt old PSW                PSA00960
SVCOLD   DS    D             Supervisor Call interrupt old PSW         PSA00970
PGMOLD   DS    D             Program interrupt old PSW                 PSA00980
MCHOLD   DS    D             Machine Check interrupt old PSW (Not Used) PSA00990
IOOLD    DS    D             I/O interrupt old PSW                     PSA01000
CSW      DS    D             Channel Status Word                       PSA01010
CAW      DS    F             Channel Address Word                      PSA01020
         ORG   PSA+X'58'                                               PSA01030
EXTNEW   DS    D             External interrupt new PSW                PSA01040
SVCNEW   DS    D             Supervisor Call interrupt new PSW         PSA01050
PGMNEW   DS    D             Program interrupt new PSW                 PSA01060
MCHNEW   DS    D             Machine Check interrupt new PSW (Not Used) PSA01070
IONEW    DS    D             I/O interrupt new PSW                     PSA01080
         ORG   PSA+X'86'                                               PSA01090
EXTCODE  DS    H             External interrupt code                   PSA01100
         ORG   PSA+X'8A'                                               PSA01110
SVCCODE  DS    H             SVC interrupt code                        PSA01120
         ORG   PSA+X'8E'                                               PSA01130
PGMCODE  DS    H             Program interrupt code                    PSA01140
         ORG   PSA+X'9C'                                               PSA01150
MONCODE  DS    H             Monitor Call code                         PSA01160
         ORG   PSA+X'BA'                                               PSA01170
IOCODE   DS    H             Interrupting I/O Device Address           PSA01180
```

```
         SPACE                                                          PSA01190
         ORG   PSA+X'400'                                               PSA01200
PSAGREGS DS    16F          General register savearea                  PSA01210
PSAFREGS DS    4D           Floating-point register savearea           PSA01220
CPUTIMER DS    D            CPU Timer savearea                         PSA01230
SYSTATE  DS    F            System processing state (only 1st byte...) PSA01240
SYSSTO   DS    F            System module STO                          PSA01250
SYSPSW   DS    D            System module dispatch PSW                 PSA01260
DUCBRA   DS    F            Real address of dispatched user's UCB      PSA01270
USRSTO   DS    F            Dispatched user's STO                      PSA01280
USRPSW   DS    D            Dispatch PSW for running user              PSA01290
USRTIMER DS    D            User's CPU Timer value at dispatch         PSA01300
INTSTATE DS    F            System processing state @ interrupt point  PSA01310
INTCREG1 DS    F            Control register 1 @ interrupt point       PSA01320
SAVPSW   DS    D            Saves user PSW @ interrupt point           PSA01330
SAVGREGS DS    16F          Saves user general registers @ interrupt   PSA01340
SAVFREGS DS    4D           Saves user floating point registers "      PSA01350
SAVTIMER DS    D            Saves user CPU timer @ interrupt point.    PSA01360
```

```
*COPY SERV                                                              SER00010
*-----------------------------------------------------------------------* SER00020
*                                                                       * SER00030
* Copy file: SERV   User/System SERVice Table Entry Format.             * SER00040
*                                                                       * SER00050
* Purpose:   Provide a template describing the format of an entry in    * SER00060
*            one of two tables defined in and referenced by the         * SER00070
*            Authentication routine.  Each table entry denotes the      * SER00080
*            existence of either a system module or a user service      * SER00090
*            provided by a module or modules of the system.             * SER00100
*                                                                       * SER00110
* Used by:   The Authentication routine only.                           * SER00120
*                                                                       * SER00130
*-----------------------------------------------------------------------* SER00140
         SPACE                                                          SER00150
*                                                                       SER00160
*                  +0        +2        +4        +6                     SER00170
*          Offset: +---------+---------+---------+---------+           SER00180
*            X'0'  |      SERVID       |     SERVASTO      |           SER00190
*                  +----+----+---------+---------+---------+           SER00200
*           X'08'  | ** |   SERVADDR   |     SERVLCNT      |           SER00210
*                  +----+----+---------+---------+---------+           SER00220
*           X'10'  |               SERVLIST                |           SER00230
*                  +                                       +           SER00240
*                  .                                       .           SER00250
*                  .                                       .           SER00260
*                  .                                       .           SER00270
*                                                                       SER00280
*                                                                       SER00290
*                     **Note: @+8 - SERVKEY                             SER00300
*                                                                       SER00310
SERV     DSECT                                                          SER00320
SERVID   DS    F            Requested service identifier               SER00330
SERVASTO DS    A            Pointer to service's STO                   SER00340
SERVKEY  DS    XL1          PSW storage key for running address space  SER00350
SERVADDR DS    AL3          Virtual address of service entry point     SER00360
SERVLCNT DS    F            # of authorized users                      SER00370
SERVLIST DS    A            1st authorization list STO                 SER00380
```

```
*COPY UCB                                                              UCB00010
*----------------------------------------------------------------------* UCB00020
*                                                                      * UCB00030
* Copy file: UCB - User Control Block.                                 * UCB00040
*                                                                      * UCB00050
* Purpose:   Retain information describing the state and features of   * UCB00060
*            a user address space.                                     * UCB00070
*                                                                      * UCB00080
* Used by:   Modules that need to access the very information that     * UCB00090
*            defines a user's address space:                           * UCB00100
*              - First-level interrupt handlers                        * UCB00110
*              - Various user service routines                         * UCB00120
*              - List manipulation routines                            * UCB00130
*              - Scheduler                                             * UCB00140
*              - Dispatcher                                            * UCB00150
*                                                                      * UCB00160
*----------------------------------------------------------------------* UCB00170
         SPACE                                                         UCB00180
*                                                                      UCB00190
*                  +0   +2   +4   +6   +8   +A   +C   +E               UCB00200
*          Offset: +----+----+----+----+----+----+----+----+           UCB00210
*            X'0'  | UCBFWD  | UCBBKWD |                   |           UCB00220
*                  +---------+---------+                   +           UCB00230
*           X'10'  |                                       |           UCB00240
*                  +                                       +           UCB00250
*           X'20'  |               UCBGREGS                |           UCB00260
*                  +                                       +           UCB00270
*           X'30'  |                                       |           UCB00280
*                  +                   +---------+---------+           UCB00290
*           X'40'  |                   |                   |           UCB00300
*                  +---------+---------+                   +           UCB00310
*           X'50'  |               UCBFREGS                |           UCB00320
*                  +                   +---------+---------+           UCB00330
*           X'60'  |                   |      UCBPSW       |           UCB00340
*                  +---------+---------+---------+---------+           UCB00350
*           X'70'  |     UCBTIMER      |       UCBID       |           UCB00360
*                  +---------+-+-------+---------+---------+           UCB00370
*           X'80'  | UCBSTO  |*|                                       UCB00380
*                  +---------+-+                                       UCB00390
*                            * - UCBPRI                                UCB00400
UCB       DSECT                                                       UCB00410
UCBFWD    DS    A             Pointer to next UCB                     UCB00420
UCBBKWD   DS    A             Pointer to previous UCB                 UCB00430
UCBGREGS  DS    16F           General purpose registers               UCB00440
UCBFREGS  DS    8F            Floating-point registers                UCB00450
UCBPSW    DS    D             User's dispatch PSW                     UCB00460
UCBTIMER  DS    D             User's CPU Timer quantum                UCB00470
UCBID     DS    CL8           Symbolic name of this user              UCB00480
UCBSTO    DS    A             Real STO for this UCB                   UCB00490
UCBPRI    DS    XL1           User's runlist enqueue priority         UCB00500
* Assign priorities by urgency of associated event:                   UCB00510
UCBPRIWK  EQU   X'80'          Priority: Waking up (Highest priority)  UCB00520
UCBPRIIO  EQU   X'40'          Priority: Done with I/O (2nd highest prty) UCB00530
UCBPRITE  EQU   X'20'          Priority: Time Slice End (lowest priority) UCB00540
UCBSIZE   EQU   *-UCB                                                 UCB00550
```