pdp11

RSX-11D

Executive Reference Manual

Order No. DEC-11-OXERA-B-D

digital

RSX-11D

Executive Reference Manual

Order No. DEC-11-OXERA-B-D

RSX-11D Version 6

digital equipment corporation · maynard, massachusetts

First Printing, May 1975

Associated Documents

Refer to RSX-11D Documentation Directory, DEC-11-OXUGA-B-D.

The postage prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CONTENTS

FIGURES

TABLES

PREFACE

The RSX-11D Executive Reference Manual provides information required to prepare user programs written in MACRO-11 Assembler Language for execution under the RSX-11D Operating System. It touches only briefly on preparing FORTRAN programs, because the FORTRAN Special Subroutines Reference Manual covers this material.

The manual is organized as follows.

Chapters 1 and 2 present overview information, first on the whole RSX-11D system, then on the RSX-11D Executive, respectively. Details on the philosophy of RSX-11D can be obtained by reading the introductory chapters of the RSX-11D User's Guide.

Chapters 3 and 4 deal with specific Executive functions and how to use them.

Appendixes A, B and C cover system list formats, a glossary, and a summary of the file QIOMAC.MAC. QIOMAC.MAC provides all symbolic values for QUEUE I/O function codes as well as symbolic definitions of status return codes. Appendix D contains directive parameter block formats for the directives.

CHAPTER 1

INTRODUCTION

RSX-11D is an event-driven multiprogramming operating system with features that make it appropriate for a wide range of applications involving real-time operations. RSX-11D provides the basis for process control systems, online business systems, and communications systems.

The modular construction of RSX-11D allows the user to configure available hardware and software resources to fit a particular processing requirement. The use of memory partitions and priority scheduling facilitates user control over the execution of many parallel real-time functions.

RSX-11D features include:

Fast interrupt response and servicing,

Simultaneous monitoring of multiple activities,

250 priority levels for task execution,

Priority servicing of I/O requests,

Convenient storage and recall of disk-resident programs,

Efficient, convenient task scheduling facilities,

Dynamic memory partitions to contain tasks of varying sizes,

Event flags for task synchronization and notification,

Checkpointing (roll-in/roll-out), a form of memory sharing,

Online program development, concurrent with task execution,

FORTRAN, COBOL*, and MACRO-11 programming languages and utilities,

Asynchronous execution of I/O-dependent code,

Support of multiuser programs and re-entrant code,

Dynamic shared global areas.

*Separate license

1.1 SYSTEM EXECUTIVE SOFTWARE

The following paragraphs provide a brief description of RSX-11D software.

1.1.1 Tasks

The basic program unit under RSX-11D is called a task. A task consists of one or more programs that have been written in FORTRAN or MACRO-11 Assembly Language. Relocatable object modules are generated and installed into the system online, making them available in memory-image format on the disk. A task can initiate another task's execution in various ways. The following are examples:

1. Request immediate execution,
2. Request execution contingent upon available memory,
3. Schedule at a future time, with optional rescheduling at periodic intervals.

All these task initiation functions can be accomplished from an operator's console as well as from a currently executing task.

1.1.2 Memory Management

There are three basic functional uses for which memory is allocated. The amount of memory allocated to each function is specified by the user at SYSGEN time.

The three functional memory areas follow.

1. RSX-11D Executive code.
2. Partitions space for tasks and shared global areas.
    a. User-controlled partitions in which only one task or shared global area can occupy the partition at a time
    b. System-controlled partitions in which the Executive controls allocation of memory in the partition and allows multiple tasks or shared global areas within the partition at one time
3. System lists and system tables.

1.1.3 Significant Events and System Traps

RSX-11D is an event-driven system in which task execution is governed by the occurrence of significant events. A significant event is any change in system status that affects the execution of a task. For example, completion of an I/O operation is a significant event.

One of the ways that significant events are signalled is through event flags. There are 64 event flags. Flags 1 through 32 are local to the task, while event flags 33 through 64 are common to all tasks. A task can set, clear, test, and wait for any event flag or combination of event flags, to achieve efficient synchronization between itself and other tasks in the system.

When a significant event occurs, the Executive scans an active task list, described in Chapter 2, seeking the highest priority task that can be executed. When an eligible task is found, it is run until it exits, suspends execution, waits for a significant event, or a significant event occurs.

System traps are another means of governing task execution. While significant events have a system-wide scope, traps are local to a task. Traps interrupt the sequence of instruction execution in the task and cause control to be transferred to a prespecified point in the program. Traps can be either synchronous or asynchronous.

Synchronous system traps (SSTs) allow servicing of fault conditions, such as memory protection violation, that can occur internally in a task.

Asynchronous system traps (ASTs) are executed as the result of significant events. Because the task has no control over when the significant event is to occur, ASTs execute asynchronously with respect to the task.

Trap service routines may or may not be provided by the user to handle the synchronous and asynchronous traps. If no synchronous trap service routine is provided, the faulting task is aborted. If no asynchronous trap service routine is provided, the task continues to execute without interruption.

1.1.4 System Directives

System directives are instructions to RSX-11D to perform functions for an executing task. System directives allow tasks to perform the following:

Schedule other tasks,

Communicate with other tasks,

Measure time intervals,

Perform I/O functions,

Suspend execution,

Exit.

Directives are generated by MACRO-11 programs via macro calls and are supported for FORTRAN by library routines supplied by DIGITAL. Refer to Chapter 3 of this manual for details of directive usage in MACRO-11 programs.

1.2 DEVICE HANDLERS

Device Handlers are tasks that support I/O devices. These tasks are similar to normal tasks within the system with the following additional features:

They usually contain an interrupt service routine to respond to hardware interrupts,

They are allowed to gain access to any memory areas including privileged ones.

By convention, device handler task names consist of two alphabetic characters followed by four dots. For example, the line printer handler is named as follows.

LP....

Device handler tasks are loaded into memory on command from the operator as needed. Requests from user tasks are queued by the Executive to the device handler according to the priority of the I/O request. If no priority is specified, that of the requesting task is used by default. When necessary, however, the requesting task can reserve a device for its exclusive use for a period of time by attaching it using the ATTACH function.

CHAPTER 2

RSX-11D EXECUTIVE

This chapter acquaints the user with the basic design elements of the RSX-11D Executive. It is not intended to provide the detail required by a systems programmer for modification of RSX-11D.

The primary functions of the Executive include memory and disk management, supervision of task execution and scheduling, intertask communication, I/O queuing, console command monitoring, and maintenance of system integrity.

The basic program unit under RSX-11D is called a task and consists of a program or set of programs that have been written in FORTRAN and/or MACRO-11 assembly language. Relocatable task modules are created either online or offline and are installed into an RSX-11D system. This process results in the recording of the task on the system disk in memory image form; i.e., the task is in executable form.

2.1 MEMORY MANAGEMENT

There are three different functional memory spaces. The size of each is specified during system generation. These areas are:

1. The Executive,
2. Partitions,
    a. System-controlled partitions
    b. User-controlled partitions
3. System communication area (SCOM).

2.1.1 Partitions

Partitions are areas of contiguous real memory that are used for task execution. There are two modes of partition usage: user-controlled where only one task at a time can occupy the partition and system-controlled where the system controls allocation of memory within the partition for execution of one or more tasks. The name, base address, size, and mode of each partition are specified during system generation and cannot be changed online. Tasks are installed to run in a particular partition but, upon specific request, can run in any partition that is large enough.

Normally, an active task remains resident in its memory space until its execution is completed. A checkpointable task, however, can be forced to relinquish its memory for execution of a higher priority task.

2.1.2 Shared Global Areas

Shared global areas (libraries, global common blocks, and pure areas of multiuser tasks) require space in partitions. They can be fixed in memory or can be assigned memory by the Executive when tasks that use them are activated.

Libraries normally are read-only and are used for code.

Global common blocks can be addressed on a per-task basis as read/write or read-only. This is a characteristic of the task rather than the global area and is specified during task building. The global area includes FORTRAN COMMON space and, normally, is used for intertask exchange of large amounts of data. SEND and SEND AND REQUEST also can be used to exchange small amounts of data among tasks. They should not be used for large amounts of data.

Multiuser tasks consist of a pure area and an impure area. The pure area of multiuser task is the area that is not modified during task execution and can be shared among multiple versions of the task. The impure area changes during execution; one copy of the impure area exists for each simultaneous user of the task.

2.1.3 System Communication Area (SCOM)

This memory space contains the tables, lists, system subroutines, and other information required by the Executive to perform its functions and maintain control of the system. It consists of a number of fixed tables or lists and code, with the remaining space being available in variable-length nodes. These nodes are used by the Executive and for intertask communication.

2.2 CONTROL OF TASK EXECUTION

RSX-11D is event-driven, in contrast to systems which use a time slice mechanism for determining a task's eligibility to execute. Under RSX-11D, the highest priority task can run continuously until some event or condition in the system causes it to be suspended. Another event or change in system status can reactivate the task.

Tasks can be activated either by the operator or by another task. Activation can be conditional, based on currently available partition space (EXECUTE), or it can occur as soon as possible (REQUEST). Also it can occur as soon as possible after some future time (SYNC, SCHEDULE, and RUN).

2.2.1 Multiprogramming

Effective multiprogramming is achieved when many tasks reside in memory simultaneously, spending some of their residency waiting for I/O completion, waiting for synchronization with other tasks, or in some way being unable to continue execution. While one or more tasks are waiting, another task can utilize the central processor's resources.

Under RSX-11D, tasks are run at a software priority level ranging from a low of 1 through a high of 250. The Executive grants central processor resources to the highest priority task capable of execution. When a task becomes ready to execute and it has a higher priority than the currently executing task, the Executive interrupts the lower priority task and allows the higher priority task to run. Execution of the interrupted task continues when it once again becomes the highest priority task capable of execution. The environment of an interrupted task is preserved; except for elapsed time, interruption is transparent to an interrupted task.

This multiprogramming scheme normally applies only to memory-resident tasks. Once a task is in memory, the Executive allows it to run to completion in a multiprogramming fashion even if its memory becomes required for the execution of a higher priority, non-resident task. However, if it is desirable to free memory for execution of a higher priority task, a task can be declared checkpointable when it is task built. A checkpointable task is swapped out when its memory is required for a higher priority task and swapped in when it once again becomes the highest priority task requiring its memory.

Normally, a task is brought into memory when requested, executes, and is removed from memory upon completion. This process frees memory for another task to execute. However, a task can be fixed in memory to permit faster response to requests for its execution. It remains in the partition until it is explicitly removed by an UNFIX directive. Tasks fixed in a system-controlled partition have no effect on the rest of the partition which remains available for execution of other tasks.

2.2.2 Significant Events

A task is considered active from the time its execution starts until the time it has exited. While the task is active, it is included in a priority-ordered list of active tasks called the active task list (ATL). The system uses the ATL in the following way.

When a significant event is declared, the Executive interrupts the executing task and scans the active task list examining the status of tasks until a task capable of execution is found. Execution of that task is then initiated, or continued, until one of the following occurs.

1. The task exits.
2. The task must wait for another event (e.g., I/O completion).
3. A significant event occurs and a higher priority task is capable of execution.

Task switching occurs as a result of a significant event, and significant events occur only when declared explicitly or implicitly by tasks.

NOTE

Task switching occurs implicitly when a lower priority task is eligible for execution and the currently executing task performs one of the following actions:

Suspends itself,
Waits for an event
(e.g., I/O completion),
Exits.

Event flags are associated with significant events. Declaration of a significant event indicates that something has happened in the system, and the possible setting of a particular event flag indicates what has happened. For example, upon completion of I/O requests, a device handler task normally sets a requester-indicated event flag and declares a significant event. If a requesting task instructs the system that it cannot run until an event flag is set (signalling task I/O completion), other eligible tasks of lower priority may run. In the scan of the active task list, a task that is awaiting I/O completion is by passed until a significant event is declared through the setting of a event flag upon task I/O completion.

Each task has access to 64 event flags of which 1 through 32 are unique to each task and 33 through 64 are common to all tasks. The use of event flags is detailed in Chapter 4.

2.2.3 System Traps

The ability to service certain conditions without continuously testing for their existence is provided via system traps. As discussed in Chapter 1, two types of traps are defined: synchronous and asynchronous (also see Chapter 4). A trap is a linkage method for optional in-task service routines. Service routines must be included as a part of the task, limited by the same restrictions as the task, and run at the task's priority as a result of a system trap condition, (e.g., fault, I/O completion). This facility also provides a means of responding to the execution of privileged instructions and non-RSX-11D EMT's.

If the system is not explicitly notified of the existence of a system trap service routine, the system trap does not occur.

2.3 SYSTEM TABLES AND SYSTEM LISTS

RSX-11D uses linked lists and fixed-length tables to maintain system information. Fixed-length tables are lists with elements that reside in consecutive memory locations. This format is used when lists are static, when list scan time is critical, or both.

Most linked lists are linked as double-ended queues and are called deques (pronounced "decks"). Deques allow list elements to be added or deleted from either end, since they include backward and forward pointers. An RSX-11D deque consists of a listhead and list elements (nodes), circularly linked by both backward and forward pointers. The first word of a node (or listhead) is a forward pointer, i.e., the address of the next node (or listhead) looking forward. The second word of a node (or listhead) is a backward pointer, i.e., the address of the next node (or listhead) looking backward.

A listhead is a node that consists of only a forward and a backward pointer, and is used as a reference point. Hence, a listhead identifies a deque, and indicates both the beginning and end of the circularly linked list. All nodes are a multiple of eight words in length. Usually the first two words contain pointers and the third defines the node's owner. The following paragraphs describe the major lists used in RSX-11D. Their formats and contents are described in Appendix A.

2.3.1 Clock Queue

The clock queue is a linked list with its listhead in SCOM. It consists of one node for each operation to be performed at some time in the future. A ticks-till-due count in the first node of the clock queue is decremented at each clock tick until the node becomes due (i.e., until the count is zero). Then the indicated operation is performed. The nodes are linked in the order in which they come due. Each node is 16 words.

2.3.2 I/O Request Queue

The I/O request queue is a linked list with a listhead in the physical unit directory entry for the unit to which the request has been queued. Each entry is 16 words.

2.3.3 Global Common Directory

The global common directory (GCD) is a linked list of entries for each global common block and library installed in the system. The GCD listhead is in SCOM. Each entry is 16 words.

The GCD entries are created by INSTALL for the pure area of multiuser tasks and for global common areas (libraries and global common). GCD entries are linked into the GCD at run time and are pointed to by the task's STD.

2.3.4 Physical Unit Directory

The physical unit directory (PUD) is a table of entries for each physical unit specified during system generation. When a logical unit number is assigned to a physical unit, the physical unit is represented by the address of the corresponding PUD entry. Each entry is 25 words.

2.3.5 System Task Directory

The system task directory (STD) is a table that provides information about each task installed in the system. The information recorded in a task's STD entry includes the following:

1. Information required when the task is not active (viz., receive linked list listhead),

2. Information required to load a task into memory (viz., task name, disk address of image).

Under RSX-11D, tasks are referred to by name, and the STD is searched for an indicated task name at each reference. The STD is structured to enable this search to be performed rapidly, without imposing naming conventions, order of installation, or the dedication of a large memory area.

The STD consists of a table of entry pointers (alpha table) for the maximum number of installed tasks and a 16-word entry for each task that is installed. The table is maintained by the programs that install and remove tasks such that the number of entries is known and consecutive table words point to task STD entries ordered alphabetically by task name. Thus, a task name can be found rapidly using a binary search and memory is not dedicated for STD entries until it is needed. The maximum size of the STD is specified during system generation.

The 16-word block of memory for an STD entry is taken from the pool when a task is installed and returned when a task is removed.

2.3.6 Send/Receive Queues

The send/receive queues are linked lists with listheads in the STD entries for each task. Entries are created and queued in priority order by the SEND directives and removed by the RECEIVE directives. Entries are variable in length up to 255 words.

2.3.7 Task Partition Directory

The task partition directory (TPD) is a table of entries for each task partition defined during system generation. Each entry is 10 words.

2.3.8 Node Pool

A node is a block of memory that is a multiple of eight words in length. Empty nodes for use in any deque are initially provided by the system generation routine in the form of a long block called the pool. When a node is needed to expand a list, it is taken from the pool. When a node is no longer needed, it is returned to the pool.

2.3.9 Interrupt Connect Node

Interrupt connect nodes connect the trap vector to the interrupt service routine of a device handler task. Each node is 16 words.

2.3.10 Memory Required List (MRL)

The memory required list is a priority-ordered linked list of active task list nodes for active tasks that require memory in a partition. Its listhead is in the task partition directory (TPD). There is an MRL for each partition. Whenever a nonfixed task exits, the MRL associated with that partition is scanned, and an attempt is made to assign memory to the highest-priority task in the list. If the attempt is successful, the task's node is moved from the MRL to the active task list. Each node is 24 words.

2.3.11 Checkpointable Task List (CTL)

For each partition, there is a priority-ordered list of checkpointable tasks that are active in that partition. Actually, this list is a set of links threaded through the ATL and not a distinct physical set of nodes. The CTL listhead is in the TPD. Each entry is 24 words.

2.3.12 Fixed Task List (FTL)

The fixed task list (FTL) is a deque of active task list nodes for tasks that have been fixed in memory but are not active. The FTL listhead is in SCOM. When a fixed task is made active, its node is relinked from the FTL to the ATL. When the task exits it is relinked into the FTL. Each node is 24 words.

2.3.13 MCR Command Buffer

The MCR command buffer is 96-byte buffer that holds the data for a requested MCR function task. The buffer is set up by the MCR dispatch task. The nodes required for the buffer are returned to the pool after the GET MCR COMMAND LINE directive passes the command line to the MCR function task.

2.3.14 Batch Command Buffer

The batch command buffer is a 96-byte buffer that holds data for the batch processor. It functions in the same manner as the MCR command buffer.

2.3.15 Asynchronous System Trap Queue

The asynchronous system trap queue (ASQ) is a linked list that operates on a first-in/first-out basis. Its listhead is in ATL entries. It consists of one node for each AST (asynchronous system trap) to be executed for the task as defined by the STD entry. Each node is eight words.

2.4 I/O OPERATIONS

The Executive's main function in I/O operations is to handle I/O requests from tasks and pass the requests to the appropriate device handler task. The general method follows.

1. A QIO directive (see Chapter 3) is issued by a task. The task specifies a number of parameters that are required in processing the I/O request. One of these parameters is the logical unit number (LUN) assigned to a device by the task.

   The directive is issued by means of a normal software emulated trap. It follows the normal PDP-11 trap sequence.

2. The Executive examines the LUN parameter of the QIO directive to determine which device handler is to process the request. The particular device handler is chosen by mapping the LUN of a particular task into an entry in the physical unit directory using the logical unit table.

3. The I/O request is put in the request queue of a device handler (one of a set of special tasks).

The requesting task can either suspend operation until the I/O request is completed or continue to operate until interrupted by an asynchronous system trap (see Chapter 4). RSX-11D permits parallel I/O requests to be issued by the same task. That is, the task continues executing after issuing a QIO; subsequently the task can issue further QIO requests without waiting for the previous request to be completed.

Some device handlers operate in conjunction with the file control primitives (FCP) to manipulate files. When an FCP routine is required, the device handler issues a SEND/REQUEST which initiates operation of the specified FCP routine.

I/O requests are queued for each unit by priority at requester task priority unless otherwise specified. The handler tasks pick requests from the top of request queues. Thus, preferential service is given to high priority requests. However, when appropriate, devices can be attached to a task, in which case only requests from the attached task or express request are dequeued. This continues until a detach-unit-from-task request is dequeued, causing requests to be dequeued by priority from the top of the I/O request queue once again.

The right to attach and detach devices is controlled by access privileges defined for each device. Requests to attach a device are rejected if the requester does not have the proper access rights. Because device handler tasks can service many units, they are not themselves attached.

The interface between a device handler task and the RSX-11D system is accomplished by directives and system subroutines which attach and detach devices and dequeue I/O requests.

CHAPTER 3

SYSTEM DIRECTIVES

3.1 INTRODUCTION

System directives are instructions to the Executive to perform an indicated operation. The applications programmer uses them to control the execution and interaction of tasks. The system macro library contains macro calls, which the programmer can use to execute directives. The FORTRAN programmer invokes system directives through a subroutine call, as described in the FORTRAN Special Subroutines Reference Manual.

Directives are implemented solely through the EMT 377 instruction. By using only EMT 377, programs using EMT 0 through EMT 376 can be run via a non-RSX system trap. Any EMT other than EMT 377 traps to a task-contained service routine that can be written to simulate another environment to whatever degree is desired.

By using macro calls, instead of executing the directive, the programmer need only reassemble his program if changes are made in the directive specifications, rather than being required to edit the source code.

3.2 DIRECTIVE IMPLEMENTATION

A brief discussion of how directives are implemented will help the programmer understand and use the macro calls associated with the directives.

Directive processing consists of five parts.

1. The user issues a directive. The directive identifier and the directive parameters are placed in the directive parameter block (DPB).

   The DPB can either be on the stack or be in the user task space.

2. An EMT 377 is issued.

3. The Executive traps the instruction and performs three steps:

    1. Determines whether the instruction is an EMT 377.

    2. If it is a 377, transfers it to the directive processor.

    3. If it is not a 377, determines whether user is capable of handling the directive. If yes, user is given control. If not, an error is returned.

4. The Executive processes the directive.

5. The Executive returns to the issusing task through "common exit," if necessary, and processes any system events.

The EMT 377 is issued with the address of a directive parameter block (DPB), or a DPB itself, on the top of the issuing task's stack. When the stack contains a DPB pointer (address), the pointer is removed (popped) after the directive is processed. When the stack contains a DPB, the entire DPB is removed as the directive is processed. In either case, the DPB is not altered when the directive is processed.

With the exception of EXIT directives, control is returned to the instruction following the EMT, with the C condition code set or cleared and the directive status word (DSW) set to indicate performance or rejection. The DSW is at virtual location zero of each task.

In the case of EXIT directives, control is not returned, but the C condition code and DSW are set.

When a directive is performed properly, the C condition code in the PS (CC-C) is cleared and the directive status word (DSW) contains a positive number, unless otherwise noted in the directive description. When a directive is rejected, the C condition code is set and the DSW contains a negative number. The number's value indicates the reason for rejection.

The first word of all DPB's contains a directive identification code (DIC) byte and a DPB size byte. The DIC indicates which directive is to be performed. The size byte indicates the DPB length in words. The DIC is in the low-order byte of the word and the size is in the high-order byte.

3.3 CONVENTIONS

The following conventions and assumptions are standard for all directives.

1. Task and partition names can be up to six characters long and are always represented as two words in radix-50 form.

2. Device names are two characters long and are represented by one word in ASCII form.

3. Time unit indicators, used for initial and repeated requests, are 1 for clock ticks, 2 for seconds, 3 for minutes, and 4 for hours.

4. The term "background task" indicates a task that is initiated from batch.

3.4 ERROR RETURNS

Directive rejections are divided into two classes: those where a programmed recovery would be common and those where it would be unlikely. The error code, always negative, is returned in the DSW at virtual location 0. Rejections with expected programmed recoveries (i.e., where a branch is taken to an error routine) have values between -1 and -19. Error codes indicating errors for which programmed recoveries are not feasible are in the range from -20 through -99.

The error codes that can be received for a particular directive are listed with the individual directive descriptions.

3.5 USING THE DIRECTIVE MACROS

This discussion applies to MACRO-11 programs. FORTRAN programmers should refer to the FORTRAN Special Subroutines Reference Manual for a description of the library subroutines which support the directives for that language.

Directives are issued by including appropriate macro calls in the program. Macro names consist of up to four letters followed by a dollar sign and, optionally, one letter. The optional letter specifies which of three possible expansions of the macro is desired.

If the optional letter is S, the macro produces code to push a DPB onto the stack, followed by an EMT 377.

If the optional letter is omitted, the macro produces only the directive's DPB. The macro expansion is inserted at the point of macro invocation, but it does not contain executable code.

When the user uses the $ or $C form of the macro call, specifying the generation of a DPB at assemble time, it is assumed that the parameters required for DPB construction are valid expressions to be used in MACRO-11 data storage instructions (e.g., .BYTE, .WORD, .RAD50).

If, however, the $S form is used, specifying the generation of code to store the DPB in the stack, the parameters must be valid source operands to be placed directly in MOV instruction.

Only the $S form produces the DPB dynamically. The other two forms produce the DPB at assembly time.

If the user has a predefined DPB, i.e., has used the $ or $C form of the macro, and wishes to avoid the creation of another one, the DIR$ macro should be used instead of one that identifies the function. This macro pushes the DPB address onto the stack using MOV SSS,-(SP), where the macro parameter (shown here as SSS), is any valid representation of the DPB address. The instruction is followed by EMT 377.

In addition to the macro routines that correspond to the directives, the DIR$ macro is useful to the programmer, particularly in cases where the DPB has been defined independently of the execution of the directive.

DIR$ generates an RSX-11D Executive trap with a predefined DPB.

Macro Call: DIR$ adr,err

Three forms are possible, with the following interpretation:

DIR$	Assumes that the address of the DPB or the DPB itself has already been pushed onto the stack.
DIR$ adr	Generates the code to push the parameter adr onto the stack.
DIR$ adr,err	Generates the code to push the parameter adr onto the stack, executes an EMT 377, generates a branch on carry clear to the address of the branch +4 (or +6 if necessary) and generates a JSR PC to the err address.

The argument adr is optional but, if present, must be a valid assembler source operand used to push the DPB address onto the stack.

The argument err is optional. If defined, it must be a valid assembler destination operand to permit a Jump to Subroutine (JSR) instruction to an error handler if the directive is rejected.

If the optional letter is C, the macro generates a DPB in a separate program section, called $DPB$$. The DPB is followed by a return to the original program section, an instruction to push the DPB address onto the stack, and an EMT 377. To ensure that the correct program section is re-entered, the user must specify its name in the argument list immediately following the required DPB parameters. If the argument is not specified, the blank (unnamed) program section is assumed.

The $C form of the macro call accepts an optional argument, PSECT. This argument allows return to the PSECT specified rather than to the default PSECT.

The $C and $S forms of macro calls and DIR$ accept an optional final argument. If included, the argument must be a valid assembler destination operand to call a user error routine. The argument generates the following code.

```
OPEN$W  #FDBADR,,,,,,OPEERR

BCC     .+6             ;BRANCH ON DIRECTIVE ACCEPTANCE

JSR     PC,OPEERR       ;ELSE, CALL ERROR SERVICE ROUTINE
```

This option is ignored when the user specifies the generation of the DPB only.

3.5.1 Symbolic Offsets

Most system directive macro calls generate local symbolic offsets. The symbols are unique to each directive and are assigned the values of the byte offset from the start of the directive's DPB to the DPB elements.

Because the offsets are defined symbolically, the programmer who must refer to or modify DPB elements can do so without calculating the offsets. Symbolic offsets also preclude the necessity of rewriting programs to accommodate changes in DPB specifications.

All $ and $C forms of macros that generate DPB's longer than one word generate local offsets. All informational directives, including the $S, form generate local symbolic offsets for the parameter block in question.

If any of the $ or $C forms of the macros is invoked and the user defined symbol $$$GLB is included, the DPB is not expanded. Furthermore, if the macro produces symbolic offsets, they are generated as global symbols, unless previously defined. The symbol $$$GLB has no effect on the expansion of $S macros.

3.5.2 Examples of Macro Calls

```
1.  MRKT$       1,5,2,MTRAP [generate DPB only in current PSECT]
    .BYTE       23.,5.      ;"MARK-TIME" DIC AND DPB SIZE
    .WORD       1           ;EVENT FLAG NUMBER
    .WORD       5           ;TIME INTERVAL MAGNITUDE
    .WORD       2           ;TIME INTERVAL UNIT
    .WORD       MTRAP       ;AST ENTRY POINT

2.  MRKT$C  1,5,2,MTRAP,PROG1,ERR [generates DPB in separate PSECT]

    .PSECT      $DPB$$

$$$=.                       ;DEFINE TEMPORARY SYMBOL

    .BYTE       23.,5.      ;"MARK TIME" DIC AND DPB SIZE
    .WORD       1           ;EVENT FLAG NUMBER
    .WORD       5           ;TIME INTERVAL MAGNITUDE
    .WORD       2           ;TIME INTERVAL UNIT
    .WORD       MTRAP       ;AST ENTRY POINT

    .PSECT      PROG1       [return to the original PSECT]

    MOV         #$$$,-(SP)  ;PUSH DPB ADDRESS ON STACK
    EMT         377         ;TRAP TO THE MONITOR
    BCC         .+6         ;BRANCH ON DIRECTIVE ACCEPTANCE
    JSR         PC,ERR      ;ELSE, CALL ERROR SERVICE ROUTINE

3.  MRKT$S      #1,#5,#2,R2,ERR [push DPB onto stack]

    MOV         R2,-(SP)    ;PUSH AST ENTRY POINT
    MOV         #2,-(SP)    ;TIME INTERVAL UNIT
    MOV         #5,-(SP)    ;TIME INTERVAL MAGNITUDE
    MOV         #1,-(SP)    ;EVENT FLAG NUMBER
    MOV         (PC)+,-(SP) ;AND MARK TIME" DIC & DPB SIZE
    .BYTE       23.,5.      ;ON THE STACK
    EMT         377         ;TRAP TO THE MONITOR
    BCC         .+6         ;BRANCH ON DIRECTIVE ACCEPTANCE
    JSR         PC,ERR      ;ELSE,CALL ERROR SERVICE ROUTINE

4.  DIR$        @R1,(R3)    [DPB already defined. DPB address in R1.]
    MOV         @R1,-(SP)   ;PUSH DPB ADDRESS ONTO STACK
    EMT         377         ;TRAP TO THE MONITOR
    BCC         .+4         ;BRANCH ON DIRECTIVE ACCEPTANCE
    JSR         PC,(R3)     ;ELSE, CALL ERROR SERVICE ROUTINE
```

3.6 DIRECTIVE SUMMARIES

Each directive description consists of a narrative explanation of its function and use, the name of the macro associated with it and its parameters, and the possible return values of the directive status word (DSW), which is virtual memory location zero.

Only the $ form of the macro name is given, although all three options are available unless otherwise specified.

The directive descriptions are presented in alphabetic order by macro call for ease of reference. However, the directives can be categorized according to their function. Table 3-1 lists the directives by function and describes them briefly.

Table 3-1
RSX-11D Directives

DIRECTIVE	ASSOCIATED MACRO CALL	FUNCTION
DIRECTIVE	DIR$	Generates an RSX-11D trap with a predefined DPB.
Task Execution Control		
EXECUTE	EXEC$	Executes a task only if sufficient memory is available at the present time. If memory is not available, the request is not queued.
REQUEST	RQST$	Runs a task contingent upon priority and memory availability. If the task cannot be run immediately, the request is queued.
SCHEDULE	SCHD$	Requests a task using the RQST$ directive at a specific future time and, optionally, repeats the request periodically; e.g., the request can be scheduled for 11:35 a.m.
RUN	RUN$	Requests a task using the RQST$ directive at a specified interval from the current time and, optionally, repeats the request periodically.
SYNCHRONIZE	SYNC$	Requests a task using RQST$ at a specific interval from a specified future time and, optionally, repeats the request periodically.
CANCEL SCHEDULED REQUESTS	CSRQ$	Cancels scheduled requests for task executions.
SUSPEND	SPND$	Suspends execution of the task issuing the directive.

Table 3-1 (Cont.)
RSX-11D Directives

DIRECTIVE	ASSOCIATED MACRO CALL	FUNCTION
RESUME	RSUM$	Resumes the execution of a task that has suspended itself.
TASK EXIT	EXIT$	Terminates execution of the task issuing the directive and causes a significant event.
ABORT TASK	ABRT$	Terminates execution of another task.
Task Status Control		
FIX IN MEMORY	FIX$	Fixes in memory (makes permanently resident) an inactive, installed task.
UNFIX	UFIX$	Negates a FIX$ directive and frees the memory allocated to the task.
DISABLE	DSBL$	Rejects future attempts to execute or fix an indicated task using any of the following directives: RQST$, EXEC$, SCHD$, RUN$, SYNC$, or FIX$.
ENABLE	ENBL$	Makes a specified disabled task capable of being run.
DISABLE CHECKPOINTING	DSCP$	Disables the checkpointability of the issuing task.
ENABLE CHECKPOINTING	ENCP$	Makes the issuing task checkpointable if checkpointing was previously disabled.
ALTER PRIORITY	ALTP$	Alters the priority of the specified active task to the new priority indicated in the directive.
Informational Directives		
GET TIME PARAMETERS	GTIM$	Fills an indicated 8-word buffer with current time and date information.
GET COMMON BLOCK PARAMETERS	GCOM$	Fills an indicated 8-word buffer with information for a specific common block.
GET PARTITION PARAMETERS	GPRT$	Fills an indicated 3-word buffer with information for a specific partition.
GET TASK PARAMETERS	GTSK$	Fills an indicated 16-word buffer with information about the task issuing the directive.

Table 3-1 (Cont.)
RSX-11D Directives

DIRECTIVE	ASSOCIATED MACRO CALL	FUNCTION
GET SENSE SWITCHES	GSSW$	Obtains the status of the console sense switches and stores it in the issuing task's directive status word.
GET LUN	GLUN$	Fills a 6-word buffer with information about a physical unit to which the task is assigned
Event-Associated Directives		
DECLARE SIGNIFICANT EVENT	DECL$	Declares a significant event and, optionally, sets an event flag and reports the status of the flag before it was set. DECL$ causes an ATL scan.
SET EVENT FLAG	SETF$	Sets an indicated event flag and reports the status of the flag before it was set. SETF$ does not cause a significant event.
CLEAR EVENT FLAG	CLEF$	Clears an indicated event flag and reports the flag's status before clearing. CLEF$ does not cause a significant event.
READ EVENT FLAG	RDEF$	Reads a specified event flag and indicates by the return code in the directive status word whether the flag is set or cleared.
READ ALL FLAGS	RDAF$	Reads all 64 event flags and records their status by setting or clearing corresponding bits in a 64-bit (4-word) buffer.
WAIT FOR SINGLE EVENT FLAG	WTSE$	Suspends execution of the issuing task until the indicated event flag is set.
WAIT FOR LOGICAL OR OF FLAGS	WTLO$	Suspends execution of the issuing task until any indicated event flag in one of five groups is set.
WAIT FOR SIGNIFICANT EVENT	WSIG$	Suspends execution of the issuing task until the next significant event occurs.
EXIT IF	EXIF$	Terminates the execution of the issuing task if an indicated event flag is not set.

Table 3-1 (Cont.)
RSX-11D Directives

DIRECTIVE	ASSOCIATED MACRO CALL	FUNCTION
Trap-Associated Directives		
MARK TIME	MRKT$	Declares a significant event after an indicated time interval starting at issuance of the directive. An event flag also can be set when the significant event is declared.
CANCEL MARK TIME REQUESTS	CMKT$	Cancels any mark time requests made by the issuing task.
INHIBIT AST RECOGNITION	IHAR$	Inhibits recognition of asynchronous system traps for the issuing task.
ENABLE AST RECOGNITION	ENAR$	Recognizes asynchronous system traps for the issuing task.
SPECIFY POWER RECOVERY AST	SPRA$	Tells the system whether or not power recovery AST's are desired for the issuing task. If desired, this directive indicates where control is to be transferred when the AST occurs.
SPECIFY FLOATING POINT EXCEPTION AST	SFPA$	Tells the system whether or not PDP-11/45 floating point exception AST's are desired for the task. If desired, the directive indicates where control is to be transferred when the AST occurs.
SPECIFY RECEIVE AST	SRDA$	Allows a task to determine whether another task has sent data to it without waiting for an event flag or continually checking the buffer. When data is detected, an AST is executed.
SPECIFY SST VECTOR TABLE FOR DEBUGGING AID	SVDB$	Specifies the virtual address of a table of synchronous system trap service routine entry points for use by ODT or other debugging aids. SVDB$ takes precedence over SVTK$.
SPECIFY SST VECTOR TABLE FOR TASK	SVTK$	Specifies the virtual address of a table of synchronous system trap service routine entry points for use by the issuing task.
TERMINATE AST EXECUTION	ASTX$	Terminates execution of an asynchronous system trap service routine.

Table 3-1 (Cont.)
RSX-11D Directives

DIRECTIVE	ASSOCIATED MACRO CALL	FUNCTION
I/O Related Directives		
QUEUE I/O	QIO$	Places an I/O request for an indicated device in a priority-ordered queue of requests for that unit.
QUEUE I/O AND WAIT	QIOW$	Performs the functions of both WTSE$ (see above) and QIO$.
ASSIGN LUN	ALUN$	Assigns a logical unit number (LUN) to a physical device unit.
SEND DATA	VSDA$	Queues a variable-length data block by priority for a task to receive.
SEND DATA AND RESUME OR REQUEST RECEIVER	VSDR$	Queues a variable-length data block by priority for a task to receive and resumes or requests execution of the receiving task.
RECEIVE DATA	VRCD$	Receives a variable-length data block that was queued by another task.
RECEIVE DATA OR EXIT	VRCX$	Receives a variable-length data block that has been queued by another task or exits if none is queued.
RECEIVE DATA OR SUSPEND	VRCS$	Receives a variable-length data block if one is queued for the task or suspends the task.
GET MCR COMMAND LINE	GMCR$	Transfers an 80-byte command line to the issuing task.

NOTE

For Information About:	Refer To:
SDAT$	VSDA$
SDRQ$	VSDR$
RCVD$	VRCD$
RCVX$	VRCX$

3.6.1 SEND and RECEIVE Directives

The following additional information about SEND and RECEIVE directives should be noted.

1. Variable-length data up to 255 words can be sent and received.

2. If the receiver's buffer is too small to hold all the data sent, the excess is lost and the receiver is notified by an error code in the directive status word (DSW).

3. The TI of a task is the same as the TI of the terminal from which the task was initiated. If the task was initiated by another task, the TI of the requesting task becomes the TI of the requested task also.

   Certain directives can be issued only to affect tasks with the same TI: RESUME, ABORT TASK, ALTER PRIORITY, and CANCEL SCHEDULED REQUEST.

4. Multiuser tasks issuing receive directives are passed only that data sent with the same TI as the multiuser task. This approach ensures the proper flow of data among several multiuser tasks and a single-user task when the single-user task is receiving from and sending to the multiuser tasks.

5. Up to three data words can be transmitted in a single 8-word node; the other five words are used for SEND/RECEIVE overhead.

6. The default buffer length (13 words) requires a 24-word node, rather than 16 words.

3.7 ABRT$ (ABORT TASK)

The ABORT TASK directive terminates the execution of the indicated task. This directive is intended for use as an emergency exit or fault exit. It causes a printout at the console each time it is invoked. A task can abort any other task. If the task being aborted is a multiuser task, it is aborted only if its TI matches that of the task issuing the ABORT directive. ABORT TASK cannot be issued by a background task.

Macro Call: ABRT$ task

task = Task name.

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

A.BTTN = (Length is 4 bytes) Task name

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed
-07	--	Task not active
-08	--	Task loading or exiting
-10	--	Task is not abortable
-80	--	Directive issued by background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
ABRTS   YOURS
.BYTE   83.,5
.RAD50  /YOURS/
```

3.8 ALTP$ (ALTER PRIORITY)

The ALTER PRIORITY directive alters the priority of the specified active task to the new priority indicated in the directive. If the task is multiuser, its priority is altered only if its TI matches that of the calling task.

Macro Call: ALTP$ task,pri

task = the name of the task whose priority is to be changed. If task is not specified, the calling task's priority is changed to its default priority.

pri = the new priority (1-250) for the task. If pri is not specified, the new priority is the priority specified at task build or installation. If a priority was not specified during task build or installation, a system default of 50 (decimal) is used.

The following symbols are defined locally with their assigned values equal to the byte offset of the DPB.

A.LTTN = task name (4 bytes)

A.LTPR = priority (2 bytes)

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed
-07	--	Task not active
-08	--	Task in process of being loaded or exiting
-80	--	Directive issued by background task
-95	--	Invalid priority
-98	--	Part of DPB is out of issuing task's address space

Macro Expansion:

```
ALTP$   TASK10,100
.BYTE   9.,4
.RADSO  /TASK10/
.WORD   100
```

3.9 ALUN$ (ASSIGN LUN)

The ASSIGN LUN directive assigns a logical unit number (LUN) to a physical device unit.

Macro Call: ALUN$ lun,devnam,devnum

lun = Logical unit number
devnam = Physical device name (two characters)
devnum = Physical device unit number

If devnam and devnum are omitted, the LUN is assigned as null. If it has previously been assigned, omitting devnam and devnum deassigns the LUN.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

A.LULU = (Length 2 bytes) logical unit number
A.LUNA = (2) Physical device name
A.LUNU = (2) Physical device unit number

DSW return codes:

+01 -- Successful completion
-90 -- LUN usage interlocked (via file open)
-92 -- Invalid device and/or unit
-96 -- Invalid logical unit number
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid

Macro Expansion:

```
ALUN$   5,LP,0
.BYTE   7,4
.WORD   5
.ASCII  /L/
.ASCII  /P/
.WORD   0
```

3.10 ASTX$ (AST SERVICE EXIT)

The AST SERVICE EXIT directive terminates execution of an asynchronous system trap service routine. ASTs are described in Section 4.2.

If another AST is queued, and ASTs are not inhibited, the next AST is executed. Otherwise, the task's pre-AST state is restored.

When an AST service routine is entered, the stack contains certain information. This information is required because the AST routine runs with the same ATL node, task header, and DSW as the main part of the task. Those portions of control areas that can be used to effect requests from AST service routines are saved on the stack. The following information is on the stack.

```
SP+14  --   Event flag mask word for flags 1-16
SP+12  --   Event flag mask word for flags 17-32
SP+10  --   Event flag mask word for flags 33-48
SP+06  --   Event flag mask word for flags 49-64
SP+04  --   The pre-AST task's program status (PS)
SP+02  --   The pre-AST task's program counter (PC)
SP+00  --   The pre-AST directive status (virtual zero)
Additional parameters, if any
```

The stack can contain additional information as well. For power recovery AST's, no information is added, but for I/O done the stack contains the address of the I/O status block; for MARK TIME, the stack contains the event flag number; and for an 11/45 floating point exception, the stack contains the exception code, and the exception address.

AST service routines must save and restore all registers used.

The AST service routine must remove any information on the stack that is additional to the seven words shown above before issuing an AST SERVICE EXIT directive. The following example shows how this is done when an AST routine is used on I/O completion:

```
;MAIN BODY OF PROGRAM
START:  .                ;PROCESS
        .
        .
        QIO$S   #IO.WVB,#2,,,#IOSTBK,#ASTSER,<#BUFFER,#60.,#40>
        .                ;PROCESS & WAIT
        .
        EXIT$S           ;EXIT TO MONITOR

IOSTBK: .BLKW   2
BUFFER: .BLKW   60.

;AST SERVICE
ASTSER: .
        .
        .
        .
        TST(SP)+         ;REMOVE ADDR OF I/O STATUS BLOCK
        ASTX$S           ;AST EXIT MACRO
```

Macro Call: ASTX$

DSW return codes:

+01	--	Successful completion
-80	--	Directive not issued from an AST service routine
-98	--	Part of DPB or table is out of task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
ASTX$
.BYTE   115.,1
```

3.11 CLEF$ (CLEAR EVENT FLAG)

The CLEAR EVENT FLAG directive clears an indicated event flag and reports the flag's polarity before clearing in the DSW. Clearing an event flag does not cause a significant event.

Macro Call: CLEF$ efn

efn = Event flag number

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

C.LEEF = (Length 2 bytes) event flag number

DSW return codes:

+00	--	Flag was already clear
+02	--	Flag was set
-97	--	Event flag number <1 or >64
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
CLEF$   4
.BYTE   31.,2
.WORD   4
```

3.12 CMKT$ (CANCEL MARK TIME REQUESTS)

The CANCEL MARK TIME REQUEST directive cancels MARK TIME requests that have been made by the issuing task. If no parameters are supplied with the macro call, all MARK TIME requests that have been made by the issuing task are cancelled. Parameters are specified to cancel only mark time requests that set an indicated event flag or cause an AST at an indicated location.

Macro Call: CMKT$ efn,ast

efn = event flag number (0 implies no event flag)
ast = AST service routine entry address

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to respective DPB elements.

C.MKEF = (length 2 bytes) Event flag number
C.MKAE = AST service routine entry address

DSW return codes:

+1 -- Successful completion
-90 -- No EFN or AST entry specified
-97 -- Invalid event flag number (event flag number <1 or >64)
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid

Macro Expansion:

```
CMKT$   5,CMTAST
.BYTE   27.,5
.WORD   5
.WORD   CMTAST
```

3.13 CSRQ$ (CANCEL SCHEDULED REQUESTS)

The CANCEL SCHEDULED REQUESTS directive cancels scheduled requests for task executions. All requests to run a specified task can be cancelled, or only those issued for a specified task by another specified task. If the task to be cancelled is a multiuser task, it is cancelled only if its TI matches that of the scheduler. This directive cannot be issued by a background task.

Macro Call: CSRQ$ ttask,rtask

ttask = Scheduled (requested) task name,
rtask = Scheduler (requester) task name.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

C.SRTN = (length 4 bytes) target task name
C.SRRN = (4) Requester task name

The issuing task is taken as the scheduler task when not specified.

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed
-80	--	Directive issued by background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
CSRQ$   TASK1,TASK2
.BYTE   25.,5
.RAD50  /TASK1/
.RAD50  /TASK2/
```

3.14 DIR$ (DIRECTIVE)

DIR$ generates an RSX-11D Executive trap with a predefined DPB.

Macro Call: DIR$ adr,err

Three forms are possible, with the following interpretation:

DIR$	**assumes the address of the DPB or the DPB itself has already been pushed onto the stack.**
DIR$ adr	**generates the code to push the parameter adr onto the stack.**
DIR$ adr,err	**generates the code to push the parameter adr onto the stack, executes an EMT 377, generates a branch on carry clear to the address of the branch +4 (or +6 if necessary) and generates a JSR PC to the err address.**

The argument adr is optional but, if present, must be a valid assembler source operand used to push the DPB address on the stack.

The argument err is optional. If defined, it must be a valid assembler destination operand to permit a jump to subroutine (JSR) instruction to an error handler if the directive is rejected.

Macro Expansion:

```
DIR$    #A
MOV     #A,-(SP)
EMT     377
```

3.15 DECL$ (DECLARE SIGNIFICANT EVENT)

The DECLARE SIGNIFICANT EVENT directive declares a significant event and, optionally, sets an event flag and reports its state before it was set. Declaring a significant event causes a scan of the active task list (ATL). The directive performs four functions:

1. Tests an event flag,
2. Sets the event flag,
3. Declares a significant event,
4. Reports the flag's polarity prior to being set in the DSW.

Macro Call: DECL$ efn

efn = event flag number (an event flag number of 0 implies no event flag number)

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

D.CLEF = (Length 2 bytes) event flag number

DSW return codes:

+1	--	No event flag specified
+0	--	Specified flag was cleared
+2	--	Specified flag was set
-97	--	Event flag number <1 or >64
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
DECL$   40
.BYTE   35.,2
.WORD   40
```

3.16 DSBL$ (DISABLE)

The DISABLE directive instructs the system to reject future attempts to run or fix an indicated task (REQUEST, EXECUTE, SCHEDULE, RUN, SYNCHRONIZE, and FIX-IN-MEMORY directives). It is used effectively to remove a task from a system without actually deleting its STD entry. DISABLE cannot be issued by a background task. If the task to be disabled is active, it is not disabled until it becomes inactive.

When a task is installed into a system, it is not disabled, i.e., it is runnable.

Macro Call: DSBL$ task

task = Task name,

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

D.SBTN = (Length 4 bytes) task name

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed
-08	--	Task is already disabled
-10	--	Task is not to be disabled
-80	--	Directive issued by a background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
DSBL$   DQIT
.BYTE   91.,5
.RAD50  /DQIT/
```

3.17 DSCP$ (DISABLE CHECKPOINTING)

The DISABLE CHECKPOINTING directive disables the checkpointability of an issuing task that has been task built as being checkpointable. This directive cannot be issued by a background task.

When a checkpointable task's execution is started, checkpointing is not disabled, i.e., the task can be checkpointed.

Macro Call: DSCP$

DSW return codes:

+1	--	Successful completion
-08	--	Task checkpointing already disabled
-10	--	Issuing task not installed as checkpointable
-80	--	Directive issued by a background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
DSCP$
.BYTE   95.,1
```

3.18 ENAR$ (ENABLE AST RECOGNITION)

The ENABLE AST RECOGNITION directive allows recognition of asynchronous system traps for the issuing task, i.e., nullifies an INHIBIT AST RECOGNITION directive. AST's that have occurred while recognition was inhibited are effected at issuance. When a task's execution is started, AST recognition is not disabled. ASTs are described in Section 4.2.

AST service routines must save and restore all registers used.

Macro Call: ENAR$

DSW return codes:

+1	--	Successful completion
-08	--	AST recognition not inhibited
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
ENAR$
.BYTE   1,1,,1
```

3.19 ENBL$ (ENABLE)

This directive instructs the system to make an indicated disabled task runnable, i.e., to nullify a DISABLE directive. ENABLE cannot be issued by a background task.

When a task is installed into a system, it is not disabled, i.e., it is runnable.

Macro Call: ENBL$ task

task = Task name,

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

E.NBTN = (Length 4 bytes) Task name

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed (No STD entry found)
-08	--	Task is not disabled
-80	--	Directive issued by a background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
ENBL$   TASK52
.BYTE   93.,6
.RAD50  /TASK52/
```

3.20 ENCP$ (ENABLE CHECKPOINTING)

The ENABLE CHECKPOINTING directive makes the issuing task checkpointable after its checkpointability has been disabled, i.e., to nullify a DISABLE CHECKPOINTING directive. Checkpoint cannot be enabled for a task that was not built as being checkpointable.

Macro Call: ENCP$

DSW return codes:

+1	--	Successful completion
-08	--	Checkpointing not disabled
-80	--	Directive issued by background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
ENCP$
.BYTE   97.,1
```

3.21 EXEC$ (EXECUTE)

The EXECUTE directive activates a task only if the memory required for its execution is presently available. It cannot be issued by a background task.

```
Macro Call:    EXEC$ task,part,pri,ugc,upc

task   = Task name,
part   = Partition name,
pri    = Priority,
ugc    = UIC group code,
upc    = UIC programmer code.
```

A partition cannot be specified for a multiuser task; i.e., the task must be requested to execute in its default partition.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

```
E.XCTN = (length 4 bytes) task name
E.XCPN = (4)  Partition name
E.XCPR = (2)  Priority
E.XCGC = (1)  UIC group code
E.XCPC = (1)  UIC programmer code
```

DSW return codes:

```
 +1 -- Successful completion
-01 -- Insufficient pool nodes available to requester
-02 -- Task not installed
-03 -- No memory for execution
-06 -- Handler task not resident to load task
-07 -- Task is active
-08 -- Task is disabled
-80 -- Directive issued by background task
-91 -- invalid UIC
-94 -- Partition not in system
-95 -- Invalid Priority specified (<0 or >250)
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid
```

Macro Expansion:

```
EXEC$   MYTASK,PART,50,200,200
.BYTE   13.,7
.RAD50  /MYTASK/
.RAD50  /PART/
.WORD   50
.BYTE   200,200
```

3.22 EXIF$ (EXITIF)

The EXITIF directive terminates the execution of the issuing task if an indicated event flag is NOT set. Control is returned if the specified event flag is set.

Macro Call: EXIF$ efn

efn = event flag number

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

E.XFEF = (length 2 bytes) event flag number

DSW return codes:

+1	--	Indicated event flag cleared, task exited
+2	--	Indicated event flag set, task not exited
-97	--	No event flag specified in mask word(s), or invalid event flag number (event flag number <1 or >64)
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
EXIF$   50
.BYTE   53.,2
.WORD   50
```

3.23 EXIT$ (TASK EXIT)

The TASK EXIT directive terminates the execution of the issuing task.

Macro Call: EXIT$

DSW return codes:

The DSW return codes can be tested only if the EXIT$ directive fails.

+1	--	Successful completion
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
EXIT$
.BYTE   51.,1
```

3.24 FIX$ (FIX IN MEMORY)

The FIX IN MEMORY directive fixes an inactive, installed task in memory. Once fixed in memory, the task cannot be checkpointed, and it does not relinquish its memory space until removed by the UNFIX directive.

The use of this directive is desirable when the timing of a task's execution is critical, or when it is requested very frequently. when fixed, a fresh copy of the task is not loaded for each request. FIX-IN-MEMORY may not be issued by a background task.

Macro Call: FIX$ task

task = Task name,

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

F.IXTN = (length 4 bytes) task name

DSW return codes:

+1 -- Successful completion
-01 -- Insufficient pool nodes available to requester
-02 -- Task not installed
-03 -- Partition too small for task
-06 -- Handler task not resident to load task
-07 -- Task is active
-08 -- Task is disabled
-09 -- Task is already fixed
-10 -- Task not fixable
-11 -- Task is checkpointable
-80 -- Directive issued by background task
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid.

Macro Expansion:

```
FIX$    TASK2
.BYTE   85.,3
.BADSW  /TASK2/
```

3.25 GCOM$ (GET COMMON BLOCK PARAMETERS)

The GET COMMON BLOCK PARAMETERS directive fills an indicated 8-word buffer with common block parameters.

The 8-word buffer is filled as follows:

```
Wd.  00  --  1/64th base address of common block
Wd.  01  --  1/64th size of common block
Wd.  02  --  Creation year
Wd.  03  --  Creation month
Wd.  04  --  Creation day
Wd.  05  --  Owner identification code (UIC)
Wd.  06  --  Starting ASR (APR)
Wd.  07  --  Common block flags byte (low order)
             and status byte (high order)
```

```
Macro Call:  GCOM$    blknam,bufadr

blknam = Name of the common block
bufadr = Adress of 8-word buffer
```

The following symbols are locally defined with their assigned value equal to the byte offset from the start of the DPB element.

```
G.COBN - (4) Common block name
G.COBA - (2) Buffer address
```

The following offsets are assigned relative to the start of the common block parameters buffer.

```
G.COBB - (2) 1/64 common block base address
G.COBS - (2) 1/64 common block size
G.COYR - (2) Creation year
G.COMO - (2) Creation month and day
G.COSA - (2) Status and starting APR
G.COUI - (2) Owner identification (UIC)
G.COTP - (2) TPD
G.COFW - (2) Common block flags byte address
```

The flags byte and the status byte are in the same word (G.COFW). The flags byte is in the low-order position and the status byte is in the high-order position.

The following bits are defined for the flags byte.

Bit	Meaning When Set
0	Library or common area loaded
1	1 =library, 0 = common area
2	GCA is position independent
3	Nonowner has write access
4	Nonowner has read access

The following values are defined for the status byte.

Value	Meaning
0	Global area not in use
2	Load request queued
4	Load request succeeded
6	Load request failed
52	Record request queued
54	Record request succeeded
56	Record request failed

DSW Status:

+1	--	Successful completion
-02	--	Indicated common block not in system
-98	--	Part of DPB or buffer is out of task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
GCOM$   SYSRES,COMBUF
.BYTE   67.,4
.RAD50  /SYSRES/
.WORD   COMBUF
```

3.26 GLUN$ (GET LUN INFORMATION)

The GET LUN INFORMATION directive fills a 6-word buffer with information about a physical unit to which the specified logical unit is assigned for the requesting task. If requests to the physical unit have been redirected to another unit, the information returned describes the effective assignment.

The 6-word buffer of LUN information comprises the first six words of the PUD. See Appendix A.

Macro Call: GLUN$ lun,bufadr

lun = Logical unit number
bufadr = Address of 6-word buffer which holds LUN information

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

G.LULU - (length 2 bytes) Logical unit number
G.LUBA - (2) Buffer address

DSW return codes:

+1	--	Successful completion
-05	--	Unassigned LUN
-96	--	Invalid logical unit number
-98	--	Part of DPB or buffer is out of task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
GLUN$   5,LUNBUF
.BYTE   5,3
.WORD   5
.WORD   LUNBUF
```

3.27 GMCR$ (GET MCR COMMAND LINE)

The GET MCR COMMAND LINE directive instructs the system to transfer a command line to the issuing task. The issuing task is an MCR function task requested by the MCR dispatcher task. The command line can be 1 through 80 bytes in length. It was placed in a list in SCOM by the MCR dispatcher. The GMCR$ macro call should be issued as soon as possible by MCR function tasks (...xxx) in order to free pool space.

If the command line is terminated by a carriage return, a flag is set in the task's header to cause the MCR dispatcher task to be requested upon exit.

The command line is read into a buffer that is picked from the pool by the MCR dispatcher task. It is linked into the MCR queue or the batch queue by the batch processor.

The format of the buffer is a standard MCR buffer.

The DSW contains the character count for the command line excluding the carriage control characters if the directive executes successfully.

Macro Call: GMCR$

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the respective DPB element.

G.MCRB = (Length 80 bytes) MCR line buffer

DSW return codes.

+nn -- Character count
-80 -- No buffer exists or illegal task name
-98 -- Part of DPB or buffer is out of issuing task's address space
-99 -- DIC or DPB size is invalid

NOTE

No "$S" macro form exists for this directive.

Macro Expansion:

```
GMCR$
.BYTE   127.,41.
```

3.28 GPRT$ (GET PARTITION PARAMETERS)

The GET PARTITION PARAMETERS directive fills an indicated 3-word buffer with partition parameters. If a partition is not specified, the partition of the issuing task is assumed.

The 3-word buffer is filled as follows:

```
Wd.  00  --   1/64th base address of partition
Wd.  01  --   1/64th size of partition
Wd.  02  --   partition flags byte
```

```
Macro call:     GPRT$    prtnam,bufadr

prtnam = Partition name
bufadr = Address of buffer
```

The following symbols are locally defined with their assigned value equal to the byte offset from the start of the DPB to the DPB element.

```
G.PRPN - (4) Partition name
G.PRBA - (2) Buffer address
```

The following offsets are assigned relative to the start of the partition parameters buffer:

```
G.PRPB - (2) 1/64 partition base address
G.PRPS - (2) 1/64 partition size
G.PRFW - (2) Partition flags byte
```

The following bits are defined for the flags byte.

Bit	Meaning When Set
0	User-controlled partition
1	Occupied user-controlled partition
2	System-controlled partition
3	Active system-controlled partition

DSW:

```
 +0  --   Successful completion
-02  --   Indicated partition not in system
-98  --   Part of DPB or buffer is out of task's address
          space
-99  --   DIC or DPB size is invalid
```

Macro Expansion:

```
GPRT$   GENRL,PARBUF
.BYTE   65.,4
.RAD50  /GENRL/
.WORD   PARBUF
```

3.29 GSSW$ (GET SENSE SWITCHES)

This directive instructs the system to get the status of the console sense switches and store it in the issuing task's directive status word.

Macro call: GSSW$

DSW Codes:

Successful completion is indicated if CC-C is clear. Switch values will be found in the DSW. Unsuccessful completion is indicated by CC-C set and one of the following codes in the DSW.

-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid

Macro Expansion:

```
GSSW$
.BYTE   125.,1
```

3.30 GTIM$ (GET TIME PARAMETERS)

The GET TIME PARAMETERS directive fills an indicated 8-word buffer with current time and date parameters. All values are in binary. The 8-word buffer is filled as follows.

```
WD.  0   --   Year (since 1900),
WD.  1   --   Month of year,
WD.  2   --   Day of month,
WD.  3   --   Hour of day,
WD.  4   --   Minute of hour,
WD.  5   --   Second of minute,
WD.  6   --   Tick of second,
WD.  7   --   Ticks per second (depends on frequency of
               clock).
```

Macro Call: GTIM$ bufadr

bufadr = Address of 8-word buffer

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

G.TIBA = (length 2 bytes) Buffer address

The following offsets are assigned relative to the start of the time parameters buffer.

```
G.TIYR  =  (2)  Year
G.TIMO  =  (2)  Month
G.TIDA  =  (2)  Day
G.TIHR  =  (2)  Hour
G.TIMI  =  (2)  Minute
G.TISC  =  (2)  Second
G.TICT  =  (2)  Clock tick
G.TICP  =  (2)  Clock ticks per second
```

DSW return codes:

```
 +1  --   Successful completion
-98  --   Part of DPB or buffer is out of task's address
           space
-99  --   DIC or DPB size is invalid
```

Macro Expansion:

```
GTIM$   TIMBUF
.BYTE   61.,2
.WORD   TIMBUF -
```

3.31 GTSK$ (GET TASK PARAMETERS)

The GET TASK PARAMETERS directive fills an indicated 16-word buffer with parameters relating to the issuing task.

The 16-word buffer is filled as follows:

```
WD. 00  --  Issuing task's name (first half),
WD. 01  --  Issuing task's name (second half),
WD. 02  --  Partition name (first half),
WD. 03  --  Partition name (second half),
WD. 04  --  Name of the task to which the ATL
            node is accounted; usually task name of
            requester of issuing task (first half),
WD. 05  --  Name of the task to which the ATL node is
            accounted; usually task name of requester
            of issuing task (second half),
WD. 06  --  Run priority
WD. 07  --  User identification code (UIC for file
            control services)
WD. 10  --  Number of logical I/O units (LUN's)
WD. 11  --  Machine type indicator e.g., 45, for PDP-11/45
WD. 12  --  STD flags words,
WD. 13  --  [Address of task SST vector tables],
WD. 14  --  [Size of task SST vector table (in words)],
WD. 15  --  Zero (reserved)
WD. 16  --  Zero (reserved)
WD. 17  --  Zero (reserved)

Macro Call:  GTSK$  bufadr

bufadr = Address of a 16-word buffer.
```

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

```
G.TSBA = (length 2 bytes) Buffer Address,
```

The following offsets are assigned relative to the start of the task parameters buffer:

```
G.TSTN = (4) Task name,
G.TSPN = (4) Partition name,
G.TSRN = (4) Name of task's requester,
G.TSPR  = (2) Priority,
G.TSGC = (1) UIC Group code
G.TSPC = (1) UIC Programmer code
G.TSNL = (2) Number of logical units,
G.TSMT = (2) Machine type,
G.TSFW = (2) STD flags word
G.TSVA = (2) Task's SST vector address
G.TSVL = (2) Task's SST vector (word) length
```

The following bits are defined for the flags word.

Bit	Meaning When Set
0	STD is 24 words long rather than 16 words long
1	Task is fixed in memory
3	Task is disabled
4	Task is being fixed in memory
6	Task is multiuser
7	Task is privileged
8	Network attribute bit
9	Restricted usage level one (background batch jobs)
10	Restricted usage level two
11	Task cannot be aborted
12	Task cannot be disabled
13	Task cannot be fixed in memory
14	Task cannot be checkpointed

DSW:

+01	--	Successful completion
-98	--	Part of DPB or buffer is out of task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
GTSK$   TSKBUF
.BYTE   63.,2
.WORD   TSKBUF
```

3.32 IHAR$ (INHIBIT AST RECOGNITION)

The INHIBIT AST RECOGNITION directive inhibits recognition of asynchronous system traps for the issuing task. The AST's are queued as they occur and are effected when AST recognition is enabled. There is an implied AST inhibit whenever an AST service routine is executing. When a task's execution is started, AST recognition is not disabled. ASTs are described in Section 4.2.

It is the recognition only which is inhibited. The ASTs are still queued by the system. They are queued FIFO and occur in that order.

AST service routines must save and restore any registers used.

Macro Call: IHAR$

DSW return codes:

+1	--	Successful completion
-08	--	AST recognition already inhibited
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
IHAR$
.BYTE   99.,1
```

3.33 MRKT$ (MARK TIME)

The MARK TIME directive declares a significant event after an indicated time interval. The interval begins at issuance of the directive. If an event flag is specified, it is cleared at issuance and set at the time of the significant event. If an AST entry point is specified, an asynchronous system trap occurs at the time of the significant event. At the AST, the task's PS, PC, virtual zero (directive status), and the event flag mask are pushed onto the task's (user) stack. If neither an event flag number nor an AST service entry point is specified, the significant event still occurs after the indicated time interval. ASTs are described in Section 4.2.

Macro Call: MRKT$ efn,timmag,timunit,ast

efn	=	Event flag number (0 implies no event flag)
timmag	=	Time interval magnitude; i.e., how many time interval units
timunit	=	Time interval unit 1 = clock ticks 2 = seconds 3 = minutes 4 = hours
ast	=	AST entry address

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

M.KTEF	=	(Length 2 bytes) event flag
M.KTMG	=	(2) time magnitude
M.KTUN	=	(2) Time unit
M.KTAE	=	(2) AST entry address

DSW return codes:

+1	--	Successful completion
-01	--	Unavailable pool node
-93	--	Invalid time specified
-97	--	Invalid event flag number (event flag number <1 or >64)
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
MRKT$   4,20,2,MTAST
.BYTE   25.,5
.WORD   4
.WORD   20
.WORD   2
.WORD   MTAST
```

3.34 QIO$ (QUEUE I/O)

The QUEUE I/O directive places an I/O request for an indicated device in a queue of priority-ordered requests for that device unit. The device unit is specified as a logical unit number (LUN).

Normally, a significant event is declared by handler tasks (via system subroutines) upon I/O completion. If an event flag is specified, it is cleared when the request is queued, and set at the significant event. The I/O Status block is also cleared when the request is queued.

If an AST service entry point is specified, the AST occurs upon I/O completion with the task's PS, PC, virtual zero (directive status), and the address of the I/O status block is pushed onto the task's (user) stack. Refer to Section 4.2.

Macro Call: QIO$ fnc,lun,efn,pri,iost,ast,prmlst

```
fnc    = I/O function code (see Appendix C for symbols)
lun    = Logical unit number
efn    = Event flag number (0 implies no event flag)
         It can be either global or local.
pri    = Priority
iost   = Address of I/O status block
ast    = Address of I/O done AST entry point
prmlst = Parameter list of the form <P1,....,P6> .
```

Three error indicators are used in conjunction with QIO$:

1. The C bit,
2. The directive status word,
3. The I/O status block.

The C bit is set to indicate that the format of the macro call was incorrect; i.e., it indicates that the Executive rejected the macro call because it resulted in an incorrect DPB. If the C bit is clear, format of the macro call is correct, but the I/O has not necessarily been successful.

The directive status word can be tested to determine whether the Executive successfully queued the I/O request. It is tested following the WTSE$ macro call associated with the QIO$. If the request was not queued, the directive status word contains one of the error codes listed below.

The I/O status block can be tested upon I/O completion to determine the success or failure of the actual transfer. The format of the I/O status block is illustrated below. I/O status error codes are provided in Appendix C.

The first call of this macro defines the following symbols and assigns, as values, their byte offset from the beginning of the DPB.

```
Q.IOFN  =  (length 2 bytes) I/O function
Q.IOLU  =  (2) Logical unit number
Q.IOEF  =  (1) Event flag number
Q.IOPR  =  (1) Priority
Q.IOSB  =  (2) Address of I/O status block
Q.IOAE  =  (2) Address of I/O done AST entry point
Q.IOPL  =  (12) Parameter list (up to 6 words).
```

DSW return codes:

```
 +1  --  Successful completion
-01  --  Unavailable pool node for request queue
-05  --  Unassigned LUN
-06  --  Handler task not resident
-95  --  Invalid Priority (>250)
-96  --  Invalid LUN
-97  --  Invalid event flag number (<1 or >64)
-98  --  Part of DPB is out of issuing task's address space
-99  --  DIC or DPB size is invalid
```

Figure 3-1 illustrates the content of the I/O status block.

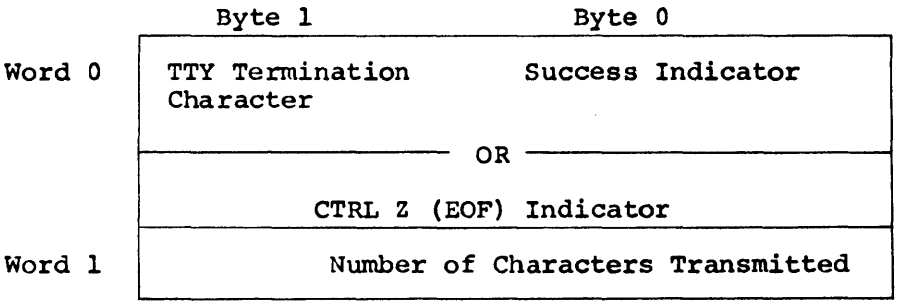

Figure 3-1
I/O Status Block

Word 0 can contain two 1-byte entries (TTY termination character and success indicator) or it can contain a -1O to indicate CTRL Z (EOF). If two entries are placed in word 0, the success indicator contains a 1 to indicate that the operation was successful or an error code, as described in Appendix C.

The TTY termination character is used by teletype handler tasks and similar handler tasks (e.g., DECwriter) and can be one of the following:

```
15(8) = CR,

33(8) = ALT,

  0   = maximum number of characters received.
```

Word 1 contains the number of characters transmitted to or from the I/O device.

Macro Expansion:

```
QIOS    IO,WV8,3,5,50,IOSB,QIOAST,<BUF,SIZ,40>
.BYTE   1,3$$ARG
.WORD   IO,WV8
.WORD   3
.BYTE   5,50
.WORD   IOSB
.WORD   QIOAST
.WORD   BUF
.WORD   SIZ
.WORD   40
```

3.35 QIOW$ (QUEUE I/O AND WAIT)

The QUEUE I/O AND WAIT directive performs the functions of both QUEUE I/O and WAIT FOR SINGLE EVENT FLAG (WTSE$). The format of the call and other related information is identical with that of QIO$, with the following exception: if event flag 0 is specified, the queue I/O is performed, but the wait is not performed.

The user should check the C bit and the I/O status block immediately after the macro call.

Macro Call: QIOW$

DSW return codes:

All those returned for QIO
-97 -- Event flag >64

Macro Expansion:

```
QIOW$   IO.RLB,5,1,200,IOSB1,RDAST,<INBUF,SIZE>
.BYTE   3,$$$ARG
.WORD   IO.RLB
.WORD   5
.BYTE   1,200
.WORD   IOSB1
.WORD   RDAST
.WORD   INBUF
.WORD   SIZE
```

3.36 RDAF$ (READ ALL FLAGS)

The READ ALL FLAGS directive reads all 64 event flags for the issuing task and records their polarities in a 64-bit (4-word) buffer.

Macro Call: RDAF$ bufadr

bufadr = Address of 4-word buffer.

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

R.DABA = (Length 2 bytes) buffer address

DSW return codes:

+1	--	Successful completion
-98	--	Part of DPB or buffer is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
RDAF$   FLAGS
.BYTE   39.,2
.WORD   FLAGS
```

3.37 RDEF$ (READ EVENT FLAG)

The READ EVENT FLAG directive tests an indicated event flag and reports its polarity in the DSW.

Macro Call: RDEF$ efn

efn = Event flag number

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

R.DEEF = (Length 2 bytes) event flag number

DSW return codes:

+0	--	Flag was cleared
+2	--	Flag was set
-97	--	Event flag number <1 or >64
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
RDEF$   6
.BYTE   37.,2
.WORD   6
```

3.38 RQST$ (REQUEST)

The REQUEST directive activates a task, i.e., enables and runs the task contingent upon priority and memory availability. If the task cannot be run immediately, the request is retained so that the task executes when conditions change, i.e., when more memory is available. It cannot be issued by a background task.

Macro call: RQST$ task,part,pri,ugc,upc

task = Task name,
part = Partition name,
pri = Priority
ugc = UIC group code,
upc = UIC programmer code.

A partition cannot be specified for a multiuser task; i.e., the task must be requested to execute in its default partition.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

R.QSTN = (length 4 bytes) task name in radix 50
R.QSPN = (4) Partition name
R.QSPR = (2) Priority
R.QSGC = (1) UIC group code
R.QSPC = (1) UIC programmer code

DSW return codes:

+1 -- Successful completion
-01 -- Insufficient pool nodes available to requester
-02 -- Task not installed
-03 -- Partition too small for task
-06 -- Handler task not resident to load task
-07 -- Task is active
-08 -- Task is disabled
-80 -- Directive issued by background task
-91 -- Invalid UIC
-94 -- Partition not in system (no TPD entry found)
-95 -- Invalid priority specified (<0 or >250)
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid

Macro Expansion:

```
RQST$   MYTASK,GENRL,5,100,50
.BYTE   11.,7
.RAD50  /MYTASK/
.RAD50  /GENRL/
.WORD   5
.BYTE   50,100
```

3.39 RSUM$ (RESUME)

The RESUME directive instructs the system to resume the execution of a task that has issued a suspend directive. If the task being resumed is a multiuser task, it is resumed only if its TI matches that of the task issuing the RESUME directive. The RESUME directive cannot be issued by a background task.

NOTE

It is possible for a task to RESUME itself using the asynchronous trap feature.

Macro Call: RSUM$ task

task = Task name.

The following symbol is defined locally with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

R.SUTN = (length 4 bytes) task name

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed
-07	--	Task not active
-08	--	Task not suspended
-80	--	Directive issued by background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
RSUM$   TASK5
.BYTE   27.,3
.RADSO  /TASK5/
```

3.40 RUN$ (RUN)

The RUN directive causes a task to be requested at a specified future time, and optionally repeated periodically. The schedule time is specified in terms of delta time from issuance. The RUN directive cannot be issued by a background task.

Macro Call: RUN$ tsk,prt,pri,ugc,upc,smg,snt,rmg,rnt

tsk = Task name,
prt = Partition name,
pri = Priority
ugc = UIC group code
upc = UIC programmer code,
smg = Schedule delta magnitude i.e., how many schedule units,
snt = Schedule delta unit,
1= clock ticks
2= seconds
3= minutes
4= hours
rmg = Reschedule interval magnitude,
rnt = Reschedule interval unit

A partition cannot be specified for a multiuser task; i.e., the task must be requested to execute in its default partition.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

R.UNTN = (length 4 bytes) task name
R.UNPN = (4) Partition name
R.UNPR = (2) Priority
R.UNGC = (1) UIC group
R.UNPC = (1) UIC programmer
R.UNSM = (2) Schedule magnitude
R.UNSU = (2) Schedule unit
R.UNRM = (2) Reschedule magnitude
R.UNRU = (2) Reschedule unit

DSW return codes:

+1 -- Successful completion
-01 -- Insufficient pool nodes available to requester
-02 -- Task not installed
-03 -- Partition too small for task
-80 -- Directive issued by background task
-91 -- Invalid UIC
-93 -- Invalid time parameter specified
-94 -- Partition not in system
-95 -- Invalid priority specified (<0 or >250)
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid

Macro Expansion:

```
RUN$    DDIT,MDSPAR,25,31,40,10,3,15,3
.BYTE   17.,11.
.RAD50  /DDIT/
.RAD50  /MDSPAR/
.WORD   25
.BYTE   40,31
.WORD   10
.WORD   3
.WORD   15
.WORD   3
```

3.41 SCHD$ (SCHEDULE)

The SCHEDULE directive causes a task to be requested at a specified future time, and optionally, repeated periodically. The schedule time is specified in terms of absolute time-of-day. The SCHEDULE, RUN, and SYNC directives are the same, differing only in the form in which the schedule data is presented. SCHEDULE cannot be issued by a background task.

Macro Call: SCHD$ tsk,prt,pri,ugc,upc,hrs,min,sec,tck,rmag,rnt

tsk = Task name,
prt = Partition name,
pri = Priority,
ugc = UIC group code,
upc = UIC programmer code,
hrs = Schedule hours,
min = Schedule minutes,
sec = Schedule seconds,
tck = Schedule clock ticks,
rmag = Reschedule interval magnitude (how many of the units defined by rnt),
rnt = Reschedule interval unit.
1 = clock ticks
2 = seconds
3 = minutes
4 = hours

A partition cannot be specified for a multiuser task; i.e., the task must be requested to execute in its default partition.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

S.CHTN = (length 4 bytes) Task name
S.CHPN = (4) Partition name
S.CHPR = (2) Priority
S.CHGC = (1) UIC group
S.CHPC = (1) UIC programmer
S.CHHR = (2) Hours
S.CHMI = (2) Minutes
S.CHSC = (2) Seconds
S.CHCT = (2) Clock ticks
S.CHRM = (2) Reschedule magnitude
S.CHRU = (2) Reschedule unit

DSW return codes:

+1 -- Successful completion
-01 -- Insufficient pool nodes available to requester
-02 -- Task not installed
-03 -- Partition too small for task
-80 -- Directive issued by background task
-91 -- Invalid UIC
-93 -- Invalid time parameter specified
-94 -- Partition not in system
-95 -- Invalid priority specified (<0 or >250)
-98 -- Part of DPB is out of issuing task's address space
-99 -- DIC or DPB size is invalid

Macro Expansion:

```
SCHD$   POOL,CORE,200,40,40,10,30,00,00,5,3
.BYTE   15.,13.
.RAD50  /POOL/
.RAD50  /CORE/
.WORD   200
.BYTE   40,40
.WORD   10
.WORD   30
.WORD   00
.WORD   00
.WORD   5
.WORD   3
```

3.42 SETF$ (SET EVENT FLAG)

The SET EVENT FLAG directive sets an indicated event flag and reports the flag's polarity before setting in the DSW. Setting an event flag does not cause an ATL scan.

Macro Call: SETF$ efn

efn = event flag number

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

S.ETEF = (Length 2 bytes) event flag number

DSW:

+0	--	Flag was cleared
+2	--	Flag was set
-97	--	Event flag number <1 or >64
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

NOTE

It is recommended that the user avoid specifying event flag numbers from 59 to 64, which are reserved for RSX-11D.

Macro Expansion:

```
SETF$   33
.BYTE   33.,2
.WORD   33
```

3.43 SFPA$ (SPECIFY FLOATING POINT EXCEPTION AST)

The SPECIFY FLOATING POINT EXCEPTION AST directive instructs the system to record one of the following two items:

1. That floating point exception ASTs for the issuing task are desired and where control is to be transferred when a floating point exception AST occurs,

2. The floating point exception ASTs for the issuing task are no longer desired.

When an AST service entry point is specified, future floating point exception ASTs occur for the issuing task, and control is transferred to the indicated location whenever a floating point exception AST occurs.

When an AST service entry point is not specified (zero in the second DPB word), future floating point exception ASTs do not occur until an AST entry point is specified again. ASTs are described in Section 4.2.

If SFPA$ is not used, no AST is queued.

AST service routines must save and restore all registers used.

Macro Call: SFPA$ ast

ast = AST service entry address.

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

S.FPAE = (Length 2 bytes) AST entry address

DSW return codes:

+1	--	Sucessful completion
-01	--	No pool node available
-08	--	AST entry already un-specified
-80	--	Directive issued during AST service
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

NOTE

This directive applies only for PDP-11/45. It has no effect on PDP-11/40. For the PDP-11/40, refer to SVTK$.

Macro Expansion:

```
$FPA$   FPAST
.BYTE   111.,2
.WORD   FPAST
```

3.44 SPND$ (SUSPEND)

The SUSPEND directive suspends the execution of the issuing task. A task can suspend only itself and not another task. The SUSPEND directive cannot be issued by a background task.

Macro Call: SPND$

DSW return codes:

+2	--	Successful completion
-80	--	Directive issued by background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
SPND$
.BYTE   45.,1
```

3.45 SPRA$ (SPECIFY POWER RECOVERY AST)

The SPECIFY POWER RECOVERY AST directive instructs the system to record either of the following:

1. That power recovery AST's for the issuing task are desired, and where control is to be transferred when a power recovery AST occurs, or

2. That power recovery AST's for the issuing task are no longer desired.

When an AST service routine entry point is specified, future power recovery ASTs occur for the issuing task, and control is transferred to the indicated location whenever a power recovery occurs. When an AST service entry point is not specified (zero in the second DPB word), future power recovery ASTs do not occur until an AST entry point is specified again. ASTs are described in Section 4.2.

If SPRA$ is not used, no AST is queued.

AST service routines must save and restore all registers used.

Macro Call: SPRA$ ast

ast = AST service routine entry address or zeros if none is desired.

The following symbol is locally defined with its assigned value equal to the offset from the start of the DPB to the DPB element.

S.PRAE = (Length 2 bytes) AST entry address

DSW return codes:

+1	--	Successful completion
-01	--	No pool node available
-08	--	AST entry already unspecified
-80	--	Directive issued during AST service
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
$PRA$   PWRAST
.BYTE   109.,2
.WORD   PWRAST
```

3.46 SRDA$ (SPECIFY RECEIVE AST)

The SPECIFY RECEIVE AST directive allows a task to determine whether another task has sent data to it. When the sending of data is detected, the AST is executed. Use of this directive eliminates the need for the receiving task to check a buffer constantly for the presence of data. ASTs are described in Section 4.2.

Macro Call: SRDA$ ast

ast = AST service entry point. If ast is specified, receive occurs for the issuing task and control is transferred to the entry point. If ast is not specified, ASTs do not occur.

The following symbol is defined locally with its assigned value equal to the byte offset from the start of the DPB.

S.RDAE = AST entry address (two bytes)

DSW return codes:

+1	--	Successful completion
-01	--	No pool node available
-08	--	AST entry already unspecified
-80	--	Directive issued during AST service
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

AST service routines must save and restore all registers used.

Macro Expansion:

```
SRDA$   RCVAST
.BYTE   107.,2
.WORD   RCVAST
```

3.47 SVDB$ (SPECIFY SST VECTOR TABLE FOR DEBUGGING AID)

The SPECIFY SST VECTOR TABLE FOR DEBUGGING AID directive specifies the virtual address of a table of synchronous system trap service routine entry points for use by an intratask debugging aid (e.g., ODT). SSTs are described in Section 4.2.

When both the issuing task table and the ODT exist and both contain an entry for a particular trap, the ODT table takes precedence.

The table can contain up to eight entry points. Each entry point corresponds to a type of error that could occur or an instruction. The table is in the following format.

```
WD.  00 -- Odd address error or other trap through 4
WD.  01 -- Segment fault
WD.  02 -- T-bit trap or execution of a BPT instruction
WD.  03 -- Execution of an 'IOT' instruction
WD.  04 -- Execution of a reserved instruction
WD.  05 -- Execution of non-RSX EMT.
WD.  06 -- Execution of a trap instruction,
WD.  07 -- PDP-11/40 floating point exception
```

If the table does not exist and one of the errors or instructions listed in the table above occurs, the task is aborted. Likewise, the task is aborted if the table exists but does not contain an entry point that corresponds to a particular error or instruction and that error or instruction occurs.

The table does not exist if SVDB$ was not specified or if the macro call contained incorrect parameters.

SST service routines must save and restore all registers used.

```
Macro Call: SVDB$  adr,len

adr = Address of SST vector table
len = Number of entries in table
```

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

```
S.VDTA = (Length 2 bytes) table address
S.VDTL = (2) Table length
```

DSW return codes:

```
+01  --   Successful completion
-98  --   Part of DPB or table is out of task's address
           space, or table address not specified (zero
           address)
-99  --   DIC or DPB size is invalid
```

Macro Expansion:

```
SVDB$   SSTTBL,4
.BYTE   133.,3
.WORD   SSTTBL
.WORD   4
```

3.48 SVTK$ (SPECIFY SST VECTOR TABLE FOR TASK)

The SPECIFY SST VECTOR TABLE FOR TASK directive specifies the virtual address of a table of synchronous system trap service routine entry points for use by the issuing task. When both the issuing task table and the ODT table exist and both contain an entry for a particular trap, the ODT table takes precedence. SSTs are described in Section 4.2.

The table can contain up to eight entry points. Each entry point corresponds to a type of error that could occur or an instruction. The table is in the following format.

```
WD.00  --  Odd address error,
WD.01  --  Segment fault
WD.02  --  T-Bit trap or execution of a BPT instruction
WD.03  --  Execution of an IOT instruction
WD.04  --  Execution of a reserved instruction
WD.05  --  Execution of non-RSX EMT,
WD.06  --  Execution of a trap instruction
WD.07  --  PDP-11/40 floating point exception.
```

If the table does not exist and one of the errors or instructions listed in the table above occurs, the task is aborted. Likewise, the task is aborted if the table exists but does not contain an entry point that corresponds to a particular error or instruction and that error or instruction occurs.

The table does not exist if SVTK$ was not specified or if the macro call contained incorrect parameters.

SST service routines must save and restore all registers used.

Macro Call: SVTK$ adr,len

adr = Address of SST vector table
len = Length of (number of entries in) table

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

S.VTTA = (Length 2 bytes) table address
S.VTTL = (2) Table length

DSW return codes:

```
 +1  --  Successful completion
-98  --  Part of DPB or table is out of task's address
         space, or table address is not specified (zero
         address)
-99  --  DIC or DPB size is invalid
```

Macro Expansion:

```
SVTK$   SSTTBL,2
.BYTE   105.,3
.WORD   SSTTBL
.WORD   2
```

3.49 SYNC$ (SYNCHRONIZE)

The SYNCHRONIZE directive requests a task at a specified future time, and optionally, repeats it periodically. The schedule time is specified in terms of delta time from clock unit synchronization. Clock unit synchronization is specified for a future time. SYNCHRONIZE differs from RUN in that RUN requests task execution for delta time from the present time, not delta time from a future time. The SYNCHRONIZE directive cannot be issued by a background task.

Macro Call: SYNC$ tsk,prt,pri,ugc,upc,smg,snt,sync,rmg,rnt

```
tsk  = Task name,
prt  = Partition name,
pri  = Priority,
ugc  = UIC group code,
upc  = UIC programmer code,
smg  = Schedule delta magnitude; i.e., how many delta units
snt  = Schedule delta unit,
         1 = clock ticks
         2 = seconds
         3 = minutes
         4 = hours
sync = Synchronization unit,
rmg  = Schedule interval magnitude,
rnt  = Schedule interval unit.
```

A partition cannot be specified for a multiuser task; i.e., the task must be requested to execute in its default partition.

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

```
S.YNTN = (length 4 bytes) task name
S.YNPN = (4) Partition name
S.YNPR = (2) Priority
S.YNGC = (1) UIC group code
S.YNPC = (1) UIC programmer code
S.YNSM = (2) Schedule magnitude
S.YNSU = (2) Schedule unit
S.YNSY = (2) Synchronization
S.YNRM = (2) Reschedule magnitude
S.YNRU = (2) Reschedule unit
```

DSW return codes:

```
 +1  --  Successful completion
-01  --  Insufficient pool nodes available to requester
-02  --  Task not installed
-03  --  Partition too small for task
-80  --  Directive issued by background task
-91  --  Invalid UIC
-93  --  Invalid time parameter specified
-94  --  Partition not in system
-95  --  Invalid priority specified (<0 or >250)
-98  --  Part of DPB is out of issuing task's address space
-99  --  DIC OR DPB size is invalid
```

Macro Expansion:

```
SYNCS   THIS,GENRL,55,100,100,6,2,3,10,2
.BYTE   19.,12.
.RAD50  /THIS/
.RAD50  /GENRL/
.WORD   55
.BYTE   100,100
.WORD   6
.WORD   2
.WORD   3
.WORD   10
.WORD   2
```

3.50 UFIX$ (UNFIX)

The UNFIX directive negates a FIX directive that has been issued to make a task permanently memory-resident. The task specified in the UNFIX directive must have the same TI as the issuing task. The task then can be removed from memory. UNFIX cannot be issued by a background task.

Macro Call: UFIX$ task

task = Task name,

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

U.FXTN = (length 4 bytes) task name

DSW return codes:

+1	--	Successful completion
-02	--	Task not installed
-09	--	Task is already unfixed
-80	--	Directive issued by background task
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
UFIX$   POOL
.BYTE   87.,5
.RAD50  /POOL/
```

3.51 VRCD$ (RECEIVE DATA)

The RECEIVE DATA directive receives a variable-length data block that has been queued for it according to priority. The SEND DATA or SEND DATA AND RESUME OR REQUEST RECEIVER directives queue data for a receiver.

When a sender task is specified, only data sent by the indicated task is received.

If the buffer size is not specified, a default size of 13 words is used. A 2-word sending task name and the data block are placed in the indicated buffer. The task name is in the first two words. The buffer length should not include these two words.

If the location to store the TI is specified, the TI indicator is transferred from the SEND/RECEIVE node to this location.

If the receiver task is multiuser, only data with the same TI assignment is received.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked is charged to the sender. When the data is received, the nodes are returned and subtracted from sender's usage count.

Macro call: VRCD$ task,bufadr,buflen,ti

task = sender task name
bufadr = buffer address
buflen = buffer length
ti = address to store TI

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

R.VDTN = (length 4 bytes) task name
R.VDBA = (2) buffer address
R.VDBL = (2) buffer length
R.VDTI = (2) address in which to store TI

Condition codes:

CC-C - cleared to indicate successful completion.
CC-C - set (with CC-V unaltered) if rejection occurs.
CC-V - set if sender task is privileged

DSW return codes:

+01	-- successful completion
-02	-- sender task not installed
-08	-- no data queued (sent)
-15	-- receiver's buffer too small, data truncated
-89	-- invalid buffer size (>255)
-98	-- part of DPB or buffer is out of task's address space
-99	-- DIC or DPB size is invalid.

An alternate macro call is RCVD$ which receives a 13-word data block.

Macro call: RCVD$ task,bufadr

Macro Expansion of VRCD$:

```
VRCD$   OTHER,DATAIN,25.,TIADDR
.BYTE   75.,$$ST9
.RAD50  /OTHER/
.WORD   DATAIN
.IF GE $$ST9-5,.WORD    25.
.IF EQ $$ST9-6,.WORD    TIADDR
```

3.52 VRCS$ (RECEIVE DATA OR SUSPEND)

The RECEIVE DATA OR SUSPEND directive receives a variable-length data block that has been queued according to priority for it or suspends if no data blocks can be received. The SEND DATA and SEND DATA AND RESUME OR REQUEST RECEIVER are used to queue data for a receiver. This directive cannot be issued by background task.

When a sender task is specified, only data sent by the indicated task is received.

If the buffer size is not specified, a default size of 13 words is used. A 2-word sending task name and the data block are returned in the indicated buffer. The task name is in the first two words. The buffer length should not include these two words.

If the location at which to store TI is specified, the TI indicator is transferred from the SEND/RECEIVE node to this specified location.

If the receiving task is multiuser, only data with the same TI assignment is received.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked is charged to the sender. When the data is received, the nodes are returned and subtracted from sender's usage count.

```
Macro call:  VRCS$  task,bufadr,buflen,ti

task   = sender task name
bufadr = buffer address
buflen = buffer length
ti     = address in which to store TI
```

The following symbols are defined locally with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

```
R.VSTN = (length 4 bytes) task name
R.VSBA = (2) buffer address
R.VSBL = (2) buffer length
R.VSTI = (2) address in which to store TI
```

Condition codes:

```
CC-C  - cleared to indicate successful completion.
CC-C  - set (with CC-V unaltered) if rejection occurs.
CC-V  - set if sender task is privileged
```

DSW return codes:

```
+01   -- successful completion
-02   -- sender task not installed
-15   -- receiver's buffer too small, data truncated
-80   -- directive issued by background task
-89   -- invalid buffer size (>255).
-98   -- part of DPB or buffer is out of task's address
         space
-99   -- DIC or DPB size is invalid.
```

An alternate macro call is RCVS$ which receives a 13-word data block.

Macro call: RCVS$ task,bufadr

Macro Expansion of VRCS$:

```
.VRCS$   TASK2,DATAIN,10.,,TIADDR
.BYTE    79.,$$ST9
.RAD50   /TASK2/
.WORD    DATAIN
.IIF GE  $$ST9-5,.WORD    10.
.IIF EQ  $$ST9-6,.WORD    TIADDR
```

3.53 VRCX$ (RECEIVE DATA OR EXIT)

The RECEIVE DATA OR EXIT directive receives a variable-length data block that has been queued for it according to priority or exits if no data block can be received. The SEND DATA and SEND DATA AND RESUME OR REQUEST RECEIVER directives queue data for a receiver.

When a sender is specified, only data sent by the indicated task is received.

If the buffer size is not specified, a default size of 13 words is used. A 2-word sending task name and the data block are returned in the indicated buffer. The task name is in the first two words. The buffer length should not include these two words.

If the location to store TI is specified, the TI indicator is transferred from the SEND/RECEIVE node to this specified location.

If the receiving task is multiuser, only data with the same TI assignment is received.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked is charged to the sender. When the data is received, the nodes are returned and subtracted from sender's usage count.

```
Macro call:  VRCX$  task,bufadr,buflen,ti

task   = sender task name
bufadr = buffer address
buflen = buffer length
ti     = address at which to store TI
```

The following symbols are locally defined with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

```
R.VXTN = (length 4 bytes) task name
R.VXBA = (2) buffer address
R.VXBL = (2) buffer length
R.VXTI = (2) address in which to store TI
```

Condition codes:

```
CC-C  - cleared to indicate successful completion.
CC-V  - set (with CC-V unaltered) if rejection occurs.
CC-C  - set if sender task is privileged
```

DSW return codes:

```
+01   -- successful completion
-02   -- sender task not installed
-15   -- receiver's buffer too small, data truncated
-89   -- invalid buffer size (>255)
-98   -- part of DPB or buffer is out of task's address
         space
-99   -- DIC or DPB size is invalid.
```

An alternate macro call is RCVX$ which receives a 13-word data block.

```
Macro call:  RCVX$  task,bufadr
```

Macro Expansion of VRCX$:

```
VRCX$   TASK9,BUFIN,8.,,TIADDR
.BYTE   77.,$$ST9
.RAD50  /TASK9/
.WORD   BUFIN
.IIF GE $$ST9-5,.WORD   8.
.IIF EQ $$ST9-6,.WORD   TIADDR
```

3.54 VSDA$ (SEND DATA)

The SEND DATA directive queues a variable-length data block according to priority for a task to receive.

If the buffer size is not specified, a default of 13 words is used.

When an event flag is specified, a significant event is declared if the directive is performed; the indicated event flag is set. Normally, the event flag is used to trigger the receiver into some action. The user must distinguish between the task's event flags (1 through 32) and the systems's event flags (33 through 64). To be effective, the task must set a system event flag.

If no priority is specified, the priority of the sending task is used.

If a TI is specified for the receiving task, the specified TI is inserted in the SEND/RECEIVE node; an AST is declared if an active task with the same TI is found. If no TI is specified, the sender's TI is inserted in the SEND/RECEIVE node; an AST is declared if the task is active regardless of its TI.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked is charged to the sender. When the data is received, the nodes are returned and subtracted from sender's usage count.

```
Macro Call:  VSDA$ task,bufadr,buflen,efn,sndpri,ti

task   = receiver task name
bufadr = address of data block
buflen = length of buffer (1 through 255 words)
efn    = event flag number (0 implies no event flag)
sndpri = priority of send (1 through 250)
ti     = TI indicator
```

The following symbols are defined locally with their assigned values equal to the byte offset from the start of the DPB to the respective DPB elements.

```
S.DATN = (length 4 bytes) task name
S.DABA = (2) buffer address
S.DAEF = (2) event flag number
S.DABL = (2) buffer length
S.DASP = (2) send priority
S.DATI = (2) TI indicator
```

DSW return codes:

+01 -- successful completion
-02 -- receiver task not installed
-04 -- insufficient pool nodes for SEND
-88 -- invalid TI
-89 -- invalid buffer size (size > 255)
-97 -- invalid event flag number (event flag number < 0 or > 64)
-98 -- part of DPB or data block is out of task's address space
-99 -- DIC or DPB size is invalid

An alternate macro call is SDAT$ which sends a 13-word data block.

Macro call: SDAT$ task,bufadr,efn

Macro Expansion of VSDA$:

```
VSDA$   TASK4,DATA,24,36,200,TIADDR
.BYTE   71.,$$ST9
.RAD50  /TASK4/
.WORD   DATA
.WORD   36
.IIF GE $$ST9-6,        .WORD 24
.IIF GE $$ST9-7,        .WORD 200
.IIF EQ $$ST9-8.,       .WORD   TIADDR
```

3.55 VSDR$ (SEND DATA AND RESUME OR REQUEST RECEIVER)

The SEND DATA AND RESUME OR REQUEST RECEIVER queues a variable-length data block according to priority for a task to receive and to request or resume the execution of the receiver. This directive cannot be issued by a background task.

This directive has the effect of issuing a SEND DATA directive followed by a REQUEST directive with the following exceptions:

1. No task switching occurs between the SEND and the RECEIVE or RESUME,

2. If the SEND is not performed, neither the RESUME nor the REQUEST is attempted.

If the buffer size is not specified, a default buffer size of 13 words is used.

When an event flag is specified, a significant event is declared if the directive is performed; the indicated event flag is set. The event flag is used commonly to trigger the receiver into some action. The user must distinguish between task event flags (1 through 32) and system event flags (33 through 64).

To be effective, the task must set a system event flag. If the send priority is not specified, the priority of the sender is used to insert the SEND/RECEIVE node into the receiver's queue.

If a TI is specified for the receiving task, the specified TI is inserted in the SEND/RECEIVE node. An AST is declared if an active task is found with the same TI and the task has specified RECEIVE AST. If no TI is specified, the TI of the sending task is inserted in the SEND/RECEIVE node; an AST is declared if the task is active regardless of its TI.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked is charged to the sender. When the data is received, the nodes are returned and subtracted from sender's usage count.

Macro call: VSDR$ task,part,pri,ugc,upc,bufadr,buflen,efn,sndpri,ti

```
task   = receiver task name
part   = partition
pri    = priority
ugc    = UIC group code
upc    = UIC programmer code
bufadr = address of data block
buflen = length of data block in words
efn    = event flag number (0 implies no event flag)
sndpri = priority of send
ti     = TI indicator
```

The following symbols are locally defined with their assigned values equal to the byte offset for the start of the DPB to the respective DPB elements.

```
S.DRTN = (length 4 bytes) task name)
S.DRPN = (4) partition name
S.DRPR = (2) request priority
S.DRGC = (1) UIC group
S.DRPC = (1) UIC programmer
S.DRBA = (2) buffer address
S.DREF = (2) event flag
S.DRBL = (2) buffer length
S.DRSP = (2) send priority
S.DRTI = (2) TI indicator
```

DSW return codes:

In the following code descriptions, R indicates that the REQUEST or RESUME was rejected, and B indicates that both the SEND and REQUEST or RESUME were rejected.

```
+01 - data sent and task requested
+02 - data sent and task resumed
+03 - data sent to a non-suspended task
-01 - [R] insufficient pool nodes available for REQUEST
-02 - [B] receiver task not installed
-03 - [R] partition too small for receiver task
-04 - [B] insufficient pool nodes for SEND
-06 - [R] handler task not resident to load task
-08 - [R] receiver task is disabled
-80 - [B] directive issued by background task
-88 - [B] invalid TI indicator
-89 - [B] invalid buffer size (>255)
-91 - [R] invalid uic
-94 - [R] partition not in system
-95 - [R] invalid priority specified (<0 or >250)
-97 - [B] invalid event flag number (efn<0 or >64)
-98 - [B] part of DPB or data block is out of the task's
       address space
-99 - [B] DIC or DPB size is invalid.
```

NOTE

The SEND portion of this directive can complete and the REQUEST portion fail.

An alternate macro call is SDRQ$ which sends a 13-word data block.

```
Macro call:  SDRQ$  task,part,pri,ugc,upc,bufadr,efn
```

Macro Expansion of VSDR$:

```
VSDR$   YOU,PART,40,200,200,MYDATA,30,60,30,TIADDR
.BYTE   73.,$$ST9
.RAD50  /YOU/
.IIF LT $$ST1-4,        .WORD   0
.RAD50  /PART/
.WORD   40
.BYTE   200,200
.WORD   MYDATA
.WORD   60
.IIF    GE      $$ST9-10.,.WORD 30
.IIF    GE      $$ST9-11.,.WORD 30
.IIF    EQ      $$ST9-12.,.WORD TIADDR
```

3.56 WSIG$ (WAIT FOR SIGNIFICANT EVENT)

The WAIT FOR SIGNIFICANT EVENT directive suspends execution of the issuing task until the next significant event occurs. It is an especially effective way to suspend a task which cannot continue because of a lack of pool nodes.

Macro Call: WSIG$

DSW return codes:

+1 -- Successful completion

Macro Expansion:

```
WSIG$
.BYTE   49.,1
```

3.57 WTLO$ (WAIT FOR LOGICAL OR OF FLAGS)

The WAIT FOR LOGICAL OR OF FLAGS directive suspends the execution of the issuing task until any indicated event flag of one of the following groups of event flags is set.

```
GR 0  --  FLAGS 1-16
GR 1  --  FLAGS 17-32
GR 2  --  FLAGS 33-48
GR 3  --  FLAGS 49-64
GR 4  --  FLAGS 1-64
```

If the indicated condition is met at issuance, task execution is effectively not suspended.

Mask word bits from right-to-left represent increasing event flag numbers. A set mask word bit indicates that the task is to wait for the corresponding event flag.

Macro Call: WTLO$ set,mask

set = Desired set of event flags

mask = If set is 0,1,2,3, mask is a 16 bit (16-flag) mask word;

If set is 4, mask provides a list of four mask words in the form: < M1, M2, M3, M4>.

If zero is specified in the $S form of the macro, do not use a number sign (#) preceding it.

Example: The following macro is used to wait for flag 19, or flag 20, or flag 21, or flag 32.

```
WTLO$   1,100034
```

Example: This macro waits for 1, 19, 20, 21, 32 or 64.

```
WTLO$   4,<000001,100034,0,100000>
```

DSW return codes:

```
 +1  --   Successful completion
-97  --   No event flag specified in mask word(s)
          or; Flag set indicator other than 0,1,2,3, or 4
-98  --   Part of DPB is out of issuing task's address space
-99  --   DIC or DPB size is invalid
```

Macro Expansion:

```
WTLO$   4,<1,500,2,343>
.BYTE   43.,5
.WORD   1
.WORD   500
.WORD   2
.WORD   343
```

3.58 WTSE$ (WAIT FOR SINGLE EVENT FLAG)

The WAIT FOR SINGLE EVENT FLAG directive suspends the execution of the issuing task until an indicated event flag is set. If the flag is set at issuance, task execution is effectively not suspended.

Macro Call: WTSE$ efn

efn = event flag number

The following symbol is locally defined with its assigned value equal to the byte offset from the start of the DPB to the DPB element.

W.TSEF = (Length 2 bytes) event flag number

DSW return codes

+1	--	Successful completion
-97	--	Invalid event flag number (<1 or >64)
-98	--	Part of DPB is out of issuing task's address space
-99	--	DIC or DPB size is invalid

Macro Expansion:

```
WTSE$   19.
.BYTE   41.,2
.WORD   19.
```

CHAPTER 4

SIGNIFICANT EVENTS AND SYSTEM TRAPS

Understanding of trap logic and EMT handling as described in the PDP-11 Processor Handbook is a prerequisite to understanding this chapter.

Significant events and system traps are the means by which communication is effected among various parts of the system. Events and traps appear to overlap in some areas, but three points may help to clarify their function and use.

1. A significant event causes a change in system status; it causes the Executive to re-evaluate the eligibility for execution of all tasks.

2. System Traps are exclusive to a single task; they are useful for intra-task communication and control.

3. The occurrence of an event can only change the eligibility of a task to run. A trap, however, is a real interrupt; the sequence of instructions being executed by the task is interrupted and control is transferred to another place in the program. This process may be invisible to the user in some cases. However, it is a difference between events and traps.

4.1 SIGNIFICANT EVENTS

Significant events provide a mechanism for achieving dynamic control of task execution. Tasks are able to declare and recognize significant events through event-associated directives. The declaration and occurrence of significant events provide dynamic control over the execution of tasks. Waiting for an event, such as the waiting for an I/O request, can suspend a high priority program until that event occurs. Meanwhile, lower priority programs are allowed to run.

Event flags are the means by which RSX-11D and tasks distinguish one event from another. Associated with each task are 64 event flags. The first 32 flags (1-32) are unique to each task, and are set or cleared only as a result of that task's operation. The integrity of these flags is maintained by the task. They are stored in the task's ATL and are not available to other tasks.

The second 32 flags (33-64) are common to all tasks, and can be set or cleared as a result of any task's operation. Therefore, they require system management to preserve their significance and are stored in SCOM.

Event flags usually are set when significant events occur, and tasks can read and/or clear them using system directives. Also, task execution can be suspended until a particular event flag or a logical combination of event flags is set.

Some system processes running for the user need event flags. The last eight local (25-32) and global (57-64) event flags are reserved by convention for use by RSX-11D. Accounting uses flags 61 and 62.

In addition, device handler tasks use event flag 1 for express requests and event flag 2 for normal requests.

All significant events occur as the result of a task's having issued a system directive; some directives have the event explicitly noted, while in others it is implicit. Refer to Chapter 3 for a complete discussion of the directives.

1. The DECLARE SIGNIFICANT EVENT directive allows any task to stimulate the event-driven system whenever necessary;

2. The SEND, SEND AND RESUME OR REQUEST, or RESUME directives also cause a significant event at issuance;

3. The directives that cause task execution (REQUEST, EXECUTE, SCHEDULE, RUN, and SYNCHRONIZE) also cause a significant event whenever task execution is initiated;

4. The MARK TIME directive causes a significant event at a specified time interval after issuance;

5. Most I/O operations (initiated by the QUEUE I/O directive) cause a significant event upon completion. Successful completion of the QUEUE I/O directive itself causes a significant event.

The following examples show the usage of the common event flags for task synchronization.

Example 1 -- Global Flags

Task B specifies a global event flag (e.g., event flag number 35) in a WAITFOR directive, and task A specifies the same event flag in a SET EVENT FLAG directive at the time it is appropriate for task B to proceed.

Example 2 -- Global Flags

Task A specifies task B and a global event flag in a SEND directive. Task B has specified the same common event flag in a WAITFOR directive and issues a RECEIVE directive (to dequeue a block of data sent by task A) when activated because the WAITFOR is satisfied.

Following are examples of event flag usage to detect I/O completion, and to indicate that a specified period of time has elapsed.

Example 3 -- Local Flags

If an event flag is specified in QUEUE I/O and associated WAITFOR directives, the flag is cleared when the request is queued. When the task executes a WAITFOR predicated on the same event flag and the requested action is not yet completed, execution of the task is suspended.

The task performing the requested service sets the specified event flag when the request is completed, and the task's execution is resumed.

Example 4 -- Local Flags

If an event flag is specified in MARK TIME and WAITFOR directives, the flag is cleared at MARK TIME issuance and set after the indicated time has elapsed, and the task's execution is suspended for the indicated interval of time.

In the above examples, computation and/or event flag testing are not precluded prior to, or instead of, the WAITFOR directive, i.e., specifying an event flag does not imply that a WAITFOR directive must be used. Event flag testing can be performed at any time. The purpose of a WAITFOR directive is to stop execution until an indicated significant event occurs. It is not necessary to issue a WAITFOR directive immediately following the issuance of a QUEUE I/O or a MARK TIME directive.

4.2 SYSTEM TRAPS

System traps are task interrupts initiated by the Executive to allow servicing of certain conditions or situations that can occur.

When a task plans to use the system trap facility, it must contain a trap service routine. This routine is automatically entered when the trap occurs with the task's normal priority and privilege. The action taken by the Executive if a service routine is not supplied is dependent upon the type of trap and is described below.

There are two types of system traps: synchronous system traps (SST's) and asynchronous system traps (AST's).

SST's provide a means of servicing fault conditions within a task, such as memory protection violation and PDP-11/40 floating point exceptions. These conditions, which are internal to a task and are not significant events, occur synchronously with respect to task execution. In these cases, if an SST service routine is not included in the task, the task's execution is aborted.

AST's commonly occur as the result of a significant event and thus occur asynchronously with respect to a task's execution, i.e., a task does not have direct or complete control over when AST's occur. A characteristic of AST's is that they are for information purposes, such as signifying an I/O completion that a task wants to know about immediately and PDP-11/45 floating point exceptions. If an AST service routine is not provided, a trap does not occur and task execution is not interrupted.

It should be emphasized that SST's only are initiated by the Executive; no further action is taken. That is, they appear to the Executive just like normal task execution. The Executive having initiated an SST, cannot determine that the task is in the SST service routine. Thus, an SST service routine can be interrupted by another SST or an AST. SST's can be nested.

SST's are caused by activities internal to the task, while AST's occur as a result of an external event. The Executive keeps track of all AST's, queues them (FIFO) and is aware of when a task is executing an AST.

4.2.1 Synchronous System Traps

Before reading this section, it is advisable to review the vector interrupt logic as described in the PDP-11 Processor Handbook.

When an SST occurs, the task's PSW (program status word) and PC (program location counter) are pushed onto its stack, and return from the SST routine is accomplished by issuing an RTI or RTT instruction.

Execution of an SST service routine is indistinguishable from task execution, and an SST service routine can perform any operation that the task can. However, if a service routine for a system trap is to cause that same system trap to occur, it must be re-entrant.

SST service routine entry points are provided in a trap vector table which is contained in the task. The trap vector table is described in Section 3.41.

A zero address in the vector table is interpreted as if no entry point were specified. If an SST occurs and no entry point is specified, the task's execution is aborted. If an invalid address is specified as an SST service address, an attempt is made to transfer control to the address specified, which probably results in an odd-address or segment fault. The SST vector table is given to the monitor by the task by use of the DECLARE SST VECTOR TABLE Directive.

On entry to an SST service routine, the stack contains the following standard PDP-11 trap vector information:

```
SP+02  PS
SP+00  PC
```

At the start of SST service, the stack can contain additional information depending on the cause of the trap, as follows.

1. Segment Fault -

   Segmentation Status Register 1

   Segmentation Status Register 2

   Segmentation Status Register 0

2. EMT Other Than 377 -

   Instruction Operand (low-order byte) times two

Items 1 and 2, above, must be removed from the stack before the SST exit is taken. Exit from SST's is by means of an RTI or RTT instruction.

The RTI or RTT instruction removes PC and PS from the stack.

The additional data is pushed on the stack by the RSX-11D interrupt service routines. Seven trap service routines are provided as part of the RSX-11D Executive. Table 3-2 lists the ISRs and describes their functions.

Table 4-1
Executive Trap Service Routines

ISR	FUNCTION
Memory Parity Interrupt	Traps at location 114. If a parity error occurs at other than processor priority zero (Executive code), the system is crashed. If a parity error occurs at processor priority zero, the task's status is changed to task parity error and the task is effectively locked in memory and can execute no move.
Odd Address and Miscellaneous Error	Traps at location 4. If the cause of the trap is a red or yellow trap violation, the system crashes.If the trap was in task code and the SST vector table is defined, the SST service routine executes. If the SST vector table is not defined, the task is aborted. For other traps refer to Section 4.2.3.
Segment Faults	Traps at location 250. If the segment fault occurs at other than task level, the system is crashed. If the segment fault was caused by task-level code, the user task is notified by means of an SST or aborted.
EMTs and Traps	Traps at locations 30 and 34. If the instruction is an EMT 377, control is transferred to the RSX-11D EMT handler. If the instruction was a trap and the previous mode was kernel, control is transferred to the directive status return routine in the Executive. For other EMTs or traps, the user task is notified by an SST or is aborted.
T Bit and BPT	Traps at location 14.
IOT and Reserved Instruction	Traps at locations 20 and 10.
Floating Point Exception	Traps at location 244. SST for 11/40 and AST for 11/45.

4.2.2 Asynchronous System Traps

When an AST occurs, the task's PSW and PC are pushed onto its stack. There also can be other parameters pushed onto the stack depending upon the cause of the AST.

After processing an AST, the trap dependent parameters must be removed from the task's stack, and an EXIT AST SERVICE directive is issued with the task's stack set as indicated in the description of the AST SERVICE EXIT directive.

Upon AST service exit, control is returned to one of three places:

1. Another (queued) AST;
2. The task;
3. Another task (e.g., the corresponding task was not in execution).

Five variations on the stack format, depending upon the AST origin, are as follows:

1. If a task is to be notified of power failure recoveries, a SPECIFY POWER RECOVERY AST directive is issued.

   SP+04 -- PS of task at AST (interrupt),
   SP+02 -- PC of task at AST (interrupt),
   SP+00 -- Task's directive status (virtual zero).

   It is not necessary to pop any of this from the stack.

2. When an I/O request is queued, an AST service entry point may be specified in the macro. If specified, an AST occurs upon completion of the request with the task's stack containing the following information:

   SP+16 -- Event flag mask word for flags 1-16
   SP+14 -- Event flag mask word for flags 17-32
   SP+12 -- Event flag mask word for flags 33-48
   SP+10 -- Event flag mask word for flags 49-64
   SP+06 -- PS of task at AST (interrupt),
   SP+04 -- PC of task at AST (interrupt),
   SP+02 -- Task's directive status (virtual zero),
   SP+00 -- Address of I/O status block for request (or zero if none specified). Must be popped off stack before AST exit.

3. When a MARK TIME directive is issued, an AST service entry point can be specified in the DPB. If specified, when the indicated time interval has elapsed, an AST occurs with the task's stack as follows:

   SP+16 -- Event flag mask word for flags 1-16
   SP+14 -- Event flag mask word for flags 17-32
   SP+12 -- Event flag mask word for flags 33-48
   SP+10 -- Event flag mask word for flags 49-64
   SP+06 -- PS of task at AST (interrupt),
   SP+04 -- PC of task at AST (interrupt),
   SP+02 -- Task's directive status (virtual zero),
   SP+00 -- Event Flag number (or zero if none specified). Must be popped off stack before AST exit.

4. If a task is to be notified of PDP-11/45 floating point hardware exceptions, a SPECIFY FLOATING POINT EXCEPTION AST directive is issued, and following floating point exceptions will cause AST's to occur with the task's stack as follows:

```
SP+20 --  Event flag mask word for flags 1-16
SP+16 --  Event flag mask word for flags 17-32
SP+14 --  Event flag mask word for flags 33-48
SP+12 --  Event flag mask word for flags 49-64
SP+10 --  PS of task at AST (interrupt),
SP+06 --  PC of task at AST (interrupt),
SP+04 --  Task's directive status (virtual zero),
SP+02 --  Floating point exception code, must be popped
          off stack before AST exit,
SP+00 --  Floating point exception address, must be
          popped off stack before AST exit.
```

5. For SPECIFY RECEIVE AST, nothing is placed on the stack. The actual RECEIVE can be performed by the AST service routine.

The following are some general notes on the characteristics and use of AST's.

1. Two directives, INHIBIT AST RECOGNITION and ENABLE AST RECOGNITION, allow AST's to be queued for subsequent execution during critical periods. If AST's occur while AST recognition is inhibited, they are queued (FIFO), and are processed when AST recognition is enabled.

2. If an AST occurs while another AST is being processed, it is queued (FIFO), and is processed when the preceding AST service is completed unless AST recognition was disabled by an AST service routine.

3. If an AST occurs while an SST is being processed, the SST service routine execution is not distinguished from task execution, and is interrupted for execution of the AST service routine.

4. If an AST occurs while the related task is suspended, the task remains suspended after execution of the AST service routine unless explicitly resumed by the AST.

5. If an AST occurs while the related task is waiting for an event flag setting (WAITFOR directive), the task remains in a wait state after execution of the AST service routine unless an appropriate event flag is set by the AST or other routine.

6. If an AST occurs while the related task is in execution, the task is interrupted for the execution of the AST service routine. This interrupt is transparent to the task unless the trap service routine changes the context of the task.

7. If an AST occurs while a task is checkpointed, the task remains checkpointed and the AST is queued for execution after the task is reloaded.

4.2.3 Processor Priorities

Seven processor priorities are used by the PDP-11, as described in the PDP-11 Processor Handbook. Peripheral device interrupt service routines run at processor priority levels four through seven. In addition, software modules that cannot be interrupted run at priority level seven for short periods of time.

Priority levels zero through three are used by the system as operation indicators; i.e., no software interrupts occur, and therefore, no precedence is implied or invoked by levels zero through three.

Level zero is used exclusively for task execution. If a segment fault occurs at level zero, a system trap is caused if the task has a service routine to handle it. Otherwise, the task is aborted.

Level one is used for the servicing of trap type instructions, e.g., EMT IOT, TRAP, and is used for the following:

1. The processing of system directives (EMT 377s),

2. The causing of a synchronous system trap if a trap-type other than BPT trap executes at priority one or the priority of the interrupt service routine, whichever is higher.

Level two is used for the recognition of system events. These events are indicated in the system event recognition flag called .SERFG and are recognized only when returning to task execution, e.g., from an interrupt or from a directive. System events are significant event declarations, clock ticks, and power failure recoveries.

Level three is used for execution of routines that cannot be interrupted by significant event recognition or clock tick recognition, but that can be interrupted by peripheral device interrupts, e.g., the system subroutine to dequeue a request for a device handler task. In these cases, the level can be set to three by any means, but it must be lowered to zero by transferring control to subroutine ..ENB0 to allow recognition of system events that might have occurred while running at priority level three. The system trace debugging routine also runs at level three.

APPENDIX A
SYSTEM LISTS AND TABLES

```
                                ; TPD -- TASK PARTITION DIRECTORY
                                ;
                                ; THE "TPD" IS A FIXED LIST OF ENTRIES DESCRIBING EACH PARTITION IN A
                                ; SYSTEM.  THIS DIRECTORY IS CREATED BY THE SYSTEM CONFIGURATION
                                ; ROUTINE (SYSGEN) CONSISTING OF ENTRIES OF THE FOLLOWING FORMAT:
                                ;
000000                          T.PN==00 ; WD. 00 (B 00) -- PARTITION NAME (FIRST HALF)
                                         ; WD. 01 (B 02) -- PARTITION NAME (SECOND HALF)
000004                          T.BA==04 ; WD. 02 (B 04) -- 1/64TH BASE ADDRESS OF PARTITION (IN BYTES)
000006                          T.PZ==06 ; WD. 03 (B 06) -- 1/64TH SIZE OF PARTITION (IN BYTES)
000010                          T.FW==10 ; WD. 04 (B 10) -- PARTITION FLAGS WORD
000012                          T.HP==12 ; WD. 05 (B 12) -- 1/64TH BASE ADR OF FIRST HOLE, OR ZERO IF NO HOLES.
000014                          T.RF==14 ; WD. 06 (B 12) -- MRL LISTHEAD (FORWARD LINKAGE)
000016                          T.RB==16 ; WD. 07 (B 14) -- MRL LISTHEAD (BACKWARD LINKAGE)
000020                          T.CF==20 ; WD. 10 (B 20) -- CTL LISTHEAD (FORWARD LINKAGE)
000022                          T.CB==22 ; WD. 11 (B 22) -- CTL LISTHEAD (BACKWARD LINKAGE)
                                ;
000024                          T.SZ==24                ;SIZE (IN BYTES) OF TPD ENTRIES
                                ;
                                ;       FLAGS WORD BIT DEFINITIONS:
                                ;
000001                          TF.UC==000001   ;[00] SET IF USER CONTROLLED PARTITION.
000002                          TF.OU==000002   ;[01] SET IF OCCUPIED USER CONTROLLED PARTITION.
000004                          TF.TS==000004   ;[02] SET IF A TIME SHARED PARTITION
000010                          TF.AC==000010   ;[03] SET IF A TIME SHARED PARTITION IS ACTIVE
```

```
                                ; GCD -- GLOBAL COMMON DIRECTORY
                                ;
                                ; GLOBAL COMMON AREAS ARE SHARABLE AREAS OF MEMORY FOR USE AS
                                ; LIBRARIES, OR FOR COMMON DATA STORAGE.
                                ; THE "GCD" IS A LINKED LIST OF ENTRIES DESCRIBING EACH GLOBAL COMMON
                                ; BLOCK IN A SYSTEM.  THIS LIST IS CREATED BY INSTALL
                                ;  AND CONSISTS OF ENTRIES OF THE FOLLOWING FORMAT. NOTE THAT GCD TYPE NODES
                                ; ARE CREATED BY INSTALL FOR THE PURE AREAS OF MULTI-USER TASKS, BUT
                                ; THESE NODES ARE NOT LINKED INTO THE GCD, BUT ARE POINTED TO BY THE
                                ; TASKS' STD NODES.
                                ;
                                        ; WD. 00 (B 00) -- FORWARD LINK
                                        ; WD. 01 (B 02) -- BACKWARD LINK
000004                          G.BN==04 ; WD. 02 (B 04) -- COMMON BLOCK NAME (6 CHAR IN RADIX-50, 2-WORDS)
000010                          G.BA==10 ; WD. 04 (B 10) -- 1/64TH BASE ADDRESS OF COMMON BLOCK
000012                          G.CZ==12 ; WD. 05 (B 12) -- 1/64TH SIZE OF COMMON BLOCK
000014                          G.CT==14 ; WD. 06 (B 14) -- CREATION TIME (TWO WORDS: YEAR, MONTH/DAY)
000020                          G.GS==N.SB;WD. 10 (B 20) -- GLOBAL AREA STATUS
000021                          G.SA==21 ;        (B 21) -- STARTING APR
000022                          G.OI==22 ; WD. 11 (B 22) -- OWNER IDENTIFICATION (UIC)
000024                          G.PD==24 ; WD. 12 (B 24) -- GLOBAL AREA TPD ADDRESS
000026                          G.FB==26 ; WD. 13 (B 26) -- FLAGS BYTE
000027                          G.DI==27 ;        (B 27) -- DISK INDICATOR
000030                          G.AC==30 ; WD. 14 (B 30) -- ACTIVE REFERENCE COUNT (BYTE)
000031                          G.IC==31 ;        (B 31) -- INSTALLED REFERENCE COUNT (BYTE)
000032                          G.NA==32 ; WD. 15 (B 32) -- I/O NODE ADDRESS
000034                          G.DA==34 ; WD. 16 (B 34) -- GLOBAL AREA DISK ADDRESS
                                ;
                                ;THE FOLLOWING ARE GLOBAL AREA STATUSES:
                                ;
000000                          GS.NUL==00              ;GLOBAL AREA NOT IN USE
000002                          GS.LRQ==02              ;LOAD REQUEST QUEUED
000004                          GS.LRS==04              ;LOAD REQUEST SUCCEEDED
000006                          GS.LRF==006             ;LOAD REQUEST FAILED
000052                          GS.RRQ==052             ;RECORD REQUEST QUEUED
000054                          GS.RRS==054             ;RECORD REQUEST SUCCEEDED
000056                          GS.RRF==056             ;RECORD REQUEST FAILED
                                ;
                                ;
000040                          G.SZ==40                ;SIZE (IN BYTES) OF GCD ENTRIES
                                ;
                                ;       FLAGS WORD BIT DEFINITIONS
                                ;
000001                          GF.EI==000001   ;[00] EXISTENCE INDICATOR (SET WHEN LIB OR COMMON LOADED)
000002                          GF.LI==000002   ;[01] LIBRARY COMMON INDICATOR -- 1:LIB  0:COM
000004                          GF.RI==000004   ;[02] LIBRARY RELOCATABILITY INDICATOR -- SET FOR PIC CODE
000010                          GF.NW==000010   ;[03] SET WHEN NON-OWNER HAS WRITE ACCESS
000020                          GF.NR==000020   ;[04] SET WHEN NON-OWNER HAS READ ACCESS
```

```
                                ; PUD -- PHYSICAL UNIT DIRECTORY
                                ;
                                ; THE "PUD" IS A FIXED LIST OF ENTRIES DESCRIBING EACH PHYSICAL DEVICE-
                                ; UNIT IN A SYSTEM.  THIS LIST IS CREATED BY THE SYSTEM CONFIGURATION
                                ; ROUTINE (SYSGEN) AND CONSISTS OF ENTRIES OF THE FOLLOWING FORMAT:
                                ;
000000                          U.DN==00    ; WD. 00 (B 00) -- DEVICE NAME (2 ASCII CHARS)
000002                          U.UN==02    ; WD. 01 (B 02) -- UNIT NUMBER (BYTE)
000003                          U.FB==03    ;         (B 03) -- FLAGS (BYTE)
000004                          U.C1==04    ; WD. 02 (B 04) -- CHARACTERISTICS WORD ONE (DEVICE INDEPENDENT INDICATORS)
000006                          U.C2==06    ; WD. 03 (B 06) -- CHARACTERISTICS WORD TWO (DEVICE DEPENDENT INDICATORS)
000010                          U.C3==10    ; WD. 04 (B 10) -- CHARACTERISTICS WORD THREE (DEVICE DEPENDENT INDICATORS)
000012                          U.C4==12    ; WD. 05 (B 12) -- CHARACTERISTICS WORD FOUR (SIZE OF BLOCK, BUFFER, LINE)
000014                          U.AF==14    ; WD. 06 (B 14) -- ATTACH FLAG (ATL NODE ADDRESS OF ATTACHING TASK)
000016                          U.RP==16    ; WD. 07 (B 16) -- REDIRECT POINTER
000020                          U.HA==20    ; WD. 10 (B 20) -- HANDLER TASK ATL NODE ADDRESS
000022                          U.XC==22    ; WD. 11 (B 22) -- COUNT OF EXPRESS REQUESTS IN QUEUE
000024                          U.RF==24    ; WD. 12 (B 24) -- UNIT REQUEST DEQUE LISTHEAD (FWD PNTR)
000026                          U.RB==26    ; WD. 13 (B 26) -- UNIT REQUEST DEQUE LISTHEAD (BKD PNTR)
000030                          U.IV==30    ; WD. 14 (B 30) -- INTERRUPT TRAP VECTOR ADDRESS
000032                          U.IP==32    ; WD. 15 (B 32) -- INTERRUPT PRIORITY (IN BITS 5-7)
000034                          U.DA==34    ; WD. 16 (B 34) -- [DEVICE PAGE ADDRESS]
                                ;
                                ; PHYSICAL UNITS ARE CONSIDERED "VOLUMES" BY THE FILES SYSTEM, AND THE
                                ; REMAINDER OF THE PUD ENTRY IS A "VOLUME CONTROL BLOCK".
                                ;
000036                          U.VA==36    ; WD. 17 (B 36) -- ADDRESS OF VOLUME CONTROL BLOCK EXTENSION
000040                          U.UI==40    ; WD. 20 (B 40) -- USER IDENTIFICATION CODE (UIC)
000040                          U.PC==40    ;         (B 40) -- UIC PROGRAMMER CODE
000041                          U.GC==41    ;         (B 41) -- UIC GROUP CODE
000042                          U.VP==42    ; WD. 21 (B 42) -- VOLUME PROTECTION WORD
000042                          U.CH==42    ;         (B 42) -- CHARACTERISTICS FLAGS
                                            ;         (B 43) -- RESERVED BYTE
000044                          U.AR==44    ; WD. 22 (B 44) -- ACCESS RIGHTS FLAGS WORD
000046                          U.DACP==46  ; WD. 23 (B 46) -- DEFAULT ACP NAME, RAD50 (FIRST WORD)
000050                          U.ACP==50   ; WD. 24 (B 50) -- STD ENTRY ADDRESS OF CURRENT ACP
000052                          U.TF==52    ; WD. 25 (B 52) -- TERMINAL FLAGS WORD
000052                          U.PR==52    ; WD. 25 (B 52) -- TERMINAL PRIVILEGE BYTE
000053                          U.FO==53    ;         (B 53) -- TERMINAL FORMS BYTE
000054                          U.LBH==54   ; WD. 26 (B 54) -- HIGH ORDER = TOTAL # OF BLKS FOR DEVICE
000056                          U.LBN==56   ; WD. 27 (B 56) -- LOW ORDER= TOTAL # OF BLOCKS FOR DEVICE
                                            ; WD. 30 (B 60) -- RESERVED WORD
                                ;
000062                          U.SZ==62                ;SIZE (IN BYTES) OF PUD ENTRIES
```

```
                                ;
                                ;       FLAGS BYTE BIT DEFINITIONS
                                ;
000040          UF.RD==040      ; ************TEMPORARY*******************
000200          UF.RH==200      ; [7] SET WHEN HANDLER TASK IS DECLARED RESIDENT.
000100          UF.TL==100      ; [6] SET WHEN HANDLER TASK RECOGNIZES LOAD AND RECORD
000040          UF.OFL==040     ; [5] SET WHEN DEVICE IS OFFLINE
                                ;
                                ;       BIT DEFINITIONS FOR CHARACTERISTICS WORD ONE
                                ;
000001          UC.REC==000001  ;[00] SET IF RECORD ORIENTED DEVICE (VIZ., TT, LP, CR)
000002          UC.CCL==000002  ;[01] SET IF CARRIAGE CONTROL DEVICE (VIZ., TT LP)
000004          UC.TTY==000004  ;[02] SET IF TTY DEVICE (VIZ., KSR, LA30)


RSX11D -- RESIDENT EXECUTIVE    MACRO D0710  24-APR-75 12:56  PAGE  12-1
EXEC MODULE ONE -- SYMBOLIC DEFINITIONS

000010          UC.DIR==000010  ;[03] SET IF DEVICE IS A DIRECTORY DEVICE
000020          UC.SDI==000020  ;[04] SET IF DEVICE IS A SINGLE DIRECTORY DEVICE
000040          UC.SQD==000040  ;[05] SET IF DEVICE IS A SEQUENTIAL DEVICE
000400          UC.IMB==000400  ;+003 [08] SET IF THE DEVICE IS INTERMEDIATE BUFFERED
001000          UC.SWL==001000  ;[09] SET IF THE DEVICE IS SOFTWARE WRITE LOCKED
002000          UC.ISP==002000  ;[10] SET IF DEVICE IS INPUT SPOOLED
004000          UC.OSP==004000  ;[11] SET IF DEVICE IS OUTPUT SPOOLED
010000          UC.PSE==010000  ;[12] SET IF DEVICE IS PSEUDO DEVICE
020000          UC.COM==020000  ;[13] SET IF DEVICE IS COMMUNICATIONS CHANNEL
040000          UC.F11==040000  ;[14] SET IF DEVICE IS FILES-11
100000          UC.MNT==100000  ;[15] SET IF DEVICE IS MOUNTABLE
                                ;
                                ;       BIT DEFINITIONS FOR CHARACTERISTICS WORD TWO
                                ;
000004          UC.WCK==000004  ;[02] SET IF A READ AFTER WRITE CHECK IS REQUIRED
                                ;
                                ;       BIT DEFINITIONS FOR VOLUME CHARACTERISTICS BYTE U.CH
                                ;
000200          CH.OFF==200     ;VOLUME IS OFF-LINE
000100          CH.FOR==100     ;VOLUME IS FOREIGN
000040          CH.UNL==40      ;DISMOUNT PENDING
000020          CH.NAT==20      ;ATTACH/DETACH NOT PERMITTED
000010          CH.NDC==10      ;DEVICE CONTROL FUNCTIONS NOT PERMITTED
000001          CH.LAB==1       ;VOLUME IS LABELED TAPE
                                ;
                                ;
                                ;       BIT DEFINITIONS FOR TERMINAL PRIVILEGE BYTE
                                ;
000001          UT.PR==1        ;SET IF TTY IS PRIVLEGED
000002          UT.SL==2        ;SET IF TTY IS SLAVED
000004          UT.LG==4        ;SET IF TERMINAL IS LOGGED ON
```

```
                                ; STD-- SYSTEM TASK DIRECTORY
                                ;
                                ; THE SYSTEM TASK DIRECTORY IS A MEMORY RESIDENT DIRECTORY OF ALL TASKS
                                ; WHICH HAVE BEEN INSTALLED INTO A SYSTEM.  THIS DIRECTORY CONSISTS OF TWO
                                ; PARTS: (1) A FIXED SIZE AREA OF ONE WORD FOR EACH TASK THAT MAY
                                ; BE INSTALLED AT ANY TIME, AND (2) AN STD ENTRY FOR EACH TASK THAT IS
                                ; INSTALLED.  THE FIXED SIZED AREA IS  CALLED THE "ALPHA TABLE" AND
                                ; PROVIDES SPACE FOR AN ALPHABETICALLY ORDERED CONTIGUOUS LIST OF POINTERS
                                ; TO STD ENTRIES TO FACILITATE SEACH FOR STD ENTRY BY TASK NAME.
                                ; EACH STD ENTRY IS OF THE FOLLOWING FORMAT:
                                ;
000000                          S.TN==00 ; WD. 00 (B 00) -- TASK NAME (6 CHAR IN RADIX-50, 2 WORDS)
                                         ; WD. 01 (B 02) -- (SECOND HALF OF TASK NAME)
000004                          S.TD==04 ; WD. 02 (B 04) -- DEFAULT TASK PARTITION (TPD ADDRESS)
000006                          S.FW==06 ; WD. 03 (B 06) -- FLAGS WORD
000010                          S.DP==10 ; WD. 04 (B 10) -- DEFAULT PRIORITY (BYTE)
000011                          S.DI==11 ;        (B 11) -- SYSTEM DISK INDICATOR (BYTE)
000012                          S.LZ==12 ; WD. 05 (B 12) -- 1/64TH SIZE OF LOAD IMAGE
000014                          S.TZ==14 ; WD. 06 (B 16) -- 1/64TH MAX TASK SIZE
000016                          S.AV==16 ; WD. 07 (B 16) -- NUMBER OF ACTIVE VERSIONS OF TASK (BYTE)
000017                          S.PV==17 ;        (B 17) -- TASK POOL LIMIT PER VERSION (BYTE)
000020                          S.PU==20 ; WD. 10 (B 20) -- TASK POOL UTILIZATION
000022                          S.RF==22 ; WD. 11 (B 22) -- RECEIVE DEQUE LISTHEAD (FWD PNTR)
000024                          S.RB==24 ; WD. 12 (B 24) -- RECEIVE DEQUE LISTHEAD (BKG PNTR)
000026                          S.DL==26 ; WD. 13 (B 26) -- LOAD IMAGE FIRST BLOCK NUMBER (32-BITS)
                                         ; WD. 14 (B 30)    (SECOND HALF OF DISK ADDRESS)
000032                          S.GC==32 ; WD. 15 (B 32) -- BEGINNING OF GCD POINTER AREA
                                ;
                                ;
                                ; THE SYSTEM DISK INDICATOR SPECIFIES WHICH I/O REQUEST QUEUE IS
                                ; TO RECEIVE A "LOAD TASK IMAGE" REQUEST, BY PROVIDING A "PUD ENTRY INDEX".
                                ; E.G., A ZERO WOULD INDICATE THE REQUEST QUEUE FOR THE DEVICE-UNIT
                                ; REPRESENTED BY THE FIRST (ENTRY ZERO) PUD ENTRY.
                                ;
                                ;       FLAGS WORD BIT DEFINITIONS:
                                ;
000001                          SF.24==000001   ;[00] STD IS 24 WORDS LONG (DEFAULT IS 16)
000002                          SF.FX==000002   ;[01] SET WHEN TASK IS FIXED IN MEMORY
000004                          SF.RM==000004   ;[02] SET WHEN STD IS TO BE REMOVED
000010                          SF.TD==000010   ;[03] SET WHEN TASK IS DISABLED
000020                          SF.BF==000020   ;[04] SET  WHEN A TASK IS BEING FIXED IN MEMORY
000040                          SF.XT==000040   ;[05] SET WHEN A TASK IS TO BE REMOVED ON EXIT
000100                          SF.MU==000100   ;[06] SET WHEN TASK IS MULTI-USER
000200                          SF.PT==000200   ;[07] SET WHEN TASK IS A PRIVILEGED TASK
```

```
686          000400                SF.NT==000400   ;[08]  NETWORK ATTRIBUTE BIT
687          001000                SF.R1==001000   ;[09] RESTRICTED USAGE LEVEL ONE (BACKGROUND BATCH JOBS)
688          002000                SF.R2==002000   ;[10] RESTRICTED USAGE LEVEL TWO (UNIMPLEMENTED)
689          004000                SF.XA==004000   ;[11] SET WHEN TASK IS NEVER TO BE ABORTED
690          010000                SF.XD==010000   ;[12] SET WHEN TASK IS NEVER TO BE DISABLED
691          020000                SF.XF==020000   ;[13] SET WHEN TASK IS NEVER TO BE FIXED IN MEMORY
692          040000                SF.XC==040000   ;[14] SET WHEN TASK IS NEVER TO BE CHECKPOINTED
693                                                ;[15] UNUSED BIT
694                                ;
695          000040                S.SIZ==32.      ;SIZE OF STD IN BYTES


RSX11D -- RESIDENT EXECUTIVE    MACRO D0710  24-APR-75 12:56  PAGE  14
EXEC MODULE ONE -- SYMBOLIC DEFINITIONS

697                                ; ATL -- ACTIVE TASK LIST
698                                ;
699                                ; THE "ATL" IS A PRIORITY ORDERED DEQUE OF "ATL" NODES FOR ACTIVE TASKS
700                                ; THAT HAVE MEMORY ALLOCATED FOR THEIR EXECUTION.  THE TASKS REPRESENTED
701                                ; BY ENTRIES IN THE ATL ARE EITHER MEMORY RESIDENT, OR A REQUEST FOR THEIR
702                                ; LOADING HAS BEEN QUEUED,  THE LISTHEAD FOR THIS DEQUE IS IN THE SYSTEM
703                                ; COMMUNICATIONS AREA (SCOM), AND THE NODES ARE OF THE FOLLOWING FORMAT:
704                                ;
705                                        ; WD. 00 (B 00) -- FORWARD LINKAGE
706                                        ; WD. 01 (B 02) -- BACKWARD LINKAGE
707                                        ; WD. 02 (B 04) -- NODE ACCOUNTING WORD (STD ENTRY ADR OF REQUESTOR)
708          000004                A.RQ==.,AW
709          000006                A.TI==.,TI;WD. 03 (B 06) -- TI IDENTIFICATION = PUD ADDRESS
710          000010                A.RP==10 ; WD. 04 (B 10) -- TASK'S RUN PRIORITY (BYTE)
711          000011                A.IR==11 ;       (B 11) -- TASK I/O IN PROCESS COUNT (BYTE)
712          000012                A.IN==12 ; WD. 05 (B 12) -- TASK I/O PENDING COUNT (BYTE)
713          000013                A.CS==13 ;       (B 13) -- SAVED STATUS OF CHECKPOINTED TASK
714          000014                A.MT==14 ; WD. 06 (B 14) -- TASK MARK TIME PENDING COUNT (BYTE)
715          000015                A.CP==15 ;       (B 15) -- SAVED PRIORITY OF CHECKPOINTED TASK (BYTE)
716          000016                A.HA==16 ; WD. 07 (B 16) -- 1/64TH REAL ADR OF LOAD IMAGE
717          000016                A.NA==A.HA ;             -- I/O NODE ADDRESS WHEN TASK IS IN MRL
718          000020                A.TS==.,SB;WD. 10 (B 20) -- TASK STATUS (BYTE)
719          000021                A.AS==21 ;       (B 21) -- AST INDICATOR (PREVIOUS STATUS) BYTE
720          000022                A.TD==22 ; WD. 11 (B 22) -- SYSTEM TASK DIRECTORY (STD) ENTRY ADDRESS
721          000024                A.EF==24 ; WD. 12 (B 24) -- TASK'S EVENT FLAGS (1-32)
722                                        ; WD. 13 (B 26) -- (SECOND HALF OF TASK'S EVENT FLAGS)
723          000030                A.FM==30 ; WD. 14 (B 30) -- TASK'S EVENT FLAGS MASKS (64-BITS)
724                                        ; WD. 15 (B 32) -- (SECOND WORD OF FLAGS MASK)
725                                        ; WD. 16 (B 34) -- (THIRD WORD OF FLAGS MASK)
726                                        ; WD. 17 (B 36) -- (FOURTH WORD OF FLAGS MASK)
```

```
727          000040                A.PD==40 ; WD. 20 (B 40) -- TASK'S RUN PARTITION (TPD ADDRESS)
728          000042                A.AF==42 ; WD. 21 (B 42) -- AST DEQUE LISTHEAD (FWD POINTER)
729          000044                A.AB==44 ; WD. 22 (B 44) -- AST DEQUE LISTHEAD (BKWD POINTER)
730          000046                A.IA==46 ; WD. 23 (B 46) -- TASK IMAGE DISK ADDRESS
731                                        ; WD. 24        -- (SECOND WORD OF IMAGE ADDRESS)
732                                        ; INITIALLY = S.DL
733                                        ; CONTAINS CHECKPOINT ADRS IF TASK IS CHECKPOINTED
734          000052                A.TF==52 ; WD. 25 (B 52 -- TASK FLAGS
735          000054                A.CF==54 ; WD. 26 (B 54) -- CHECKPOINT TASK LIST FORWARD POINTER
736          000056                A.CB==56 ; WD. 27 (B 56) -- CHECKPOINT TASK LIST BACKWARD POINTER
737                                        ;
738                                        ; TASK STATUS VALUES ARE DESCRIBED AT 'ASXDT'
739                                        ;
740                                        ; BEFORE EXECUTION , THE
741                                        ; FLAGS MASK WORDS ARE USED AS FOLLOWS;
742                                        ;
743                                        ;       A.FM+0 -- ADDRESS OF TASK LOAD I/O REQUEST DEQUE LISTHEAD.
744                                        ;       A.FM+4 -- TASK UIC.
745                                        ;
746                                        ; AFTER TASK EXECUTION, 'A.FM+0' IS SET AS FOLLOWS;
747                                        ;
748                                        ;       BIT-8 IS SET WHEN THE LOW ORDER BYTE (BITS 0-7) CONTAIN A
749                                        ;       TERMINATION NOTIFICATION CODE (CODES DESCRIBED AT 'SC.CAC')
750                                        ;
751                                        ;       BIT-9 IS SET WHEN I/O RUNDOWN MESSAGE IS REQUIRED.
752                                        ;
753                                        ;       FLAGS WORD BIT DEFINITIONS

RSX11D -- RESIDENT EXECUTIVE    MACRO D0710  24-APR-75 12:56  PAGE  14-1
EXEC MODULE ONE -- SYMBOLIC DEFINITIONS

754                                        ;
755          000001                AF.CP==001      ; SET WHEN TASK IS CHECKPOINTED
756          000002                AF.RD==002      ; SET WHEN TASK'S I/O IS BEING RUN DOWN
757          000004                AF.AD==004      ; SET WHEN TASK AST RECOGNITION IS INHIBITED
758          000010                AF.CD==010      ; SET WHEN CHECKPOINTING IS DISABLED
759          000020                AF.MC==020      ; SET WHEN TASK IS MARKED FOR CHECKPOINTING
760          000040                AF.GC==040      ; SET WHEN TASK IS HOLDING A COMMAND BUFFER
761          000100                AF.IO==100      ; SET WHEN TASK HAS AN I/O COMPLETION EVENT IN ITS AST QUEUE
762          000200                AF.IT==200      ; SET WHEN TASK HAS AN INTERMEDIATE TRANSFER IN PROGRESS
763          000400                AF.GR==400      ; SET WHEN TASK'S SHARED GLOBAL AREAS HAVE BEEN RELEASED
764          001000                AF.BF==1000     ; SET WHEN A TASK IS TO BE FIXED
765          002000                AF.FX==2000     ; SET WHEN A TASK IS FIXED
766          004000                AF.AS==4000     ; SET WHEN AN AST HAS BEEN DECLARED
767          010000                AF.RA==10000    ; SET WHEN THERE IS A POTENTIAL RECEIVE AST
768                                        ;
769          000060                A.SIZ==48.      ;SIZE OF ATL IN BYTES
```

```
                                    ;
                                    ; MRL -- MEMORY REQUIRED LIST
                                    ;
                                    ; THE "MRL" IS A PRIORITY ORDERED DEQUE OF "ATL" NODES FOR ACTIVE TASKS
                                    ; THAT REQUIRE MEMORY IN A PARTITION.  EACH PARTITION HAS ITS OWN MRL.
                                    ; WHENEVER A NON-FIXED TASK RUNNING IN A PARTITION EXITS, AN
                                    ; ATTEMPT IS MADE TO ASSIGN MEMORY TO THE FIRST (HIGHEST PRIORITY)
                                    ; TASK IN THE LIST.  IF MEMORY IS FOUND, THE TASK'S NODE IS
                                    ; MOVED FROM THE "MRL" TO THE "ATL" DEQUE.  THE MRL LISTHEAD IS IN
                                    ; THE TPD ENTRY FOR THE CORRESPONDING PARTITION.
                                    ;
```



```
                                    ;
                                    ;
                                    ; CTL -- CHECKPOINTABLE TASK LIST
                                    ;
                                    ; THE "CTL" IS A PRIORITY ORDERED DEQUE OF ENTRIES FOR CHECKPOINTABLE
                                    ; TASKS THAT ARE ACTIVE IN A PARTITION.  EACH PARTITION HAS ITS OWN
                                    ; CTL.  THE CTL LISTHEAD IS IN THE TPD ENTRY FOR THE CORRESPONDING
                                    ; PARTITION.
                                    ; THE CTL IS REALLY JUST A RELINKING OF THE ATL, HOWEVER, AND HAS
                                    ; NO UNIQUE NODES OR FORMAT OF ITS OWN.
                                    ; THE CTL FORWARD AND BACKWARD POINTERS ARE THE LAST TWO WORDS IN
                                    ; THE ATL NODE, AND , CONSEQUENTLY, NEGATIVE OFFSETS MUST BE DEFINED
                                    ; SO THAT PARAMETERS MAY BE REFERENCED WITH RESPECT TO THE CTL FORWARD
                                    ; POINTER.
                                    ;
        177730                      K.RQ==A.RQ-A.CF
        177764                      K.PD==A.PD-A.CF
        177734                      K.RP==A.RP-A.CF
        177735                      K.IR==A.IR-A.CF
        177736                      K.IN==A.IN-A.CF
        177740                      K.MT==A.MT-A.CF
        177741                      K.CP==A.CP-A.CF
        177742                      K.NA==A.NA-A.CF
        177742                      K.HA==A.HA-A.CF
        177744                      K.TS==A.TS-A.CF
        177745                      K.AS==A.AS-A.CF
        177746                      K.TD==A.TD-A.CF
```

```
 810          177750                  K.EF==A.EF=A.CF
 811          177754                  K.FM==A.FM=A.CF
 812          177732                  K.TI==A.TI=A.CF
 813          177766                  K.AF==A.AF=A.CF
 814          177770                  K.AB==A.AB=A.CF
 815          177772                  K.IA==A.IA=A.CF
 816          177776                  K.TF==A.TF=A.CF
 817          177737                  K.CS==A.CS=A.CF
 818                                  ;
```



```
 821                                  ; IRQ -- I/O REQUEST QUEUE
 822                                  ;
 823                                  ; THE "IRQ" IS A PRIORITY ORDERED DEQUE OF I/O REQUEST NODES WITH ITS
 824                                  ; LISTHEAD IN THE PUD ENTRY OF THE PHYSICAL UNIT FOR WHICH THE I/O
 825                                  ; REQUEST WAS QUEUED.  EACH PHYSICAL UNIT HAS ITS OWN I/O REQUEST QUEUE.
 826                                  ; I/O REQUEST NODES ARE CREATED AND QUEUED PRIMARILY BY THE "QUEUE I/O"
 827                                  ; DIRECTIVE.  HOWEVER, THE EXEC ALSO CREATES I/O REQUESTS TO:
 828                                  ; (1) LOAD A TASK IMAGE, (2) RECORD A TASK IMAGE (CHECKPOINTING), AND
 829                                  ; (3) TO RUNDOWN I/O ON AN EXIT'ED TASK.  I/O REQUEST NODES ARE OF
 830                                  ; THE FOLLOWING FORMAT.
 831                                  ;
 832                                          ; WD. 00 (B 00) -- FORWARD LINKAGE
 833                                          ; WD. 01 (B 02) -- BACKWARD LINKAGE
 834                                          ; WD. 02 (B 04) -- NODE ACCOUNTING WORD (STD ENTRY ADR OF REQUESTOR)
 835          000004                  R.TD==N.AW
 836          000006                  R.AT==06 ; WD. 03 (B 06) -- ATL NODE OF REQUESTOR ***
 837          000010                  R.PR==10 ; WD. 04 (B 10) -- PRIORITY (BYTE)
 838          000011                  R.DP==11 ;        (B 11) -- DPB SIZE (BYTE) ***
 839          000012                  R.LU==12 ; WD. 05 (B 12) -- LOGICAL UNIT NUMBER (BYTE)
 840          000013                  R.FN==13 ;        (B 13) -- EVENT FLAG NUMBER (BYTE)
 841          000014                  R.FC==14 ; WD. 06 (B 14) -- I/O FUNCTION CODE
 842          000016                  R.SB==16 ; WD. 07 (B 16) -- VIRTUAL ADDRESS OF STATUS BLOCK
 843          000020                  R.AE==20 ; WD. 10 (B 20) -- VIRTUAL ADDRESS OF AST SERVICE ENTRY
 844          000022                  R.UI==22 ; WD. 11 (B 22) -- USER IDENTIFICATION CODE
 845          000022                  R.PC==22 ;        (B 22) -- PROGRAMMER CODE
 846          000023                  R.GC==23 ;        (B 23) -- GROUP CODE
 847          000024                  R.PB==24 ; WD. 12 (B 24) -- PARAMETER #1
 848                                           ; WD. 13 (B 26) -- PARAMETER #2
 849                                           ; WD. 14 (B 30) -- PARAMETER #3
 850                                           ; WD. 15 (B 32) -- PARAMETER #4
 851                                           ; WD. 16 (B 34) -- PARAMETER #5
 852                                           ; WD. 17 (B 36) -- PARAMETER #6
 853          000040                  R.PD==40 ; WD. 20 (B 40) -- PUD POINTER FOR THIS REQUEST
```

```
854          000042                  R.EL==42 ; WD. 21 (B 42) -- ERROR LOG BUFFER POINTER/FLAG
855          000044                  R.WA==44 ; WD. 22 (B 44) -- FLAG BYTE FOR EXEC
856                                           ; WD. 22 (B 45) -- WORK AREA FOR DEVICE HANDLERS
857                                           ; WD. 23 (B 46) -- WORK AREA FOR DEVICE HANDLERS
858                                           ; WD. 24 (B 50) -- WORK AREA FOR DEVICE HANDLERS
859                                           ; WD. 25 (B 52) -- WORK AREA FOR DEVICE HANDLERS
860          000054                  R.IB==54 ; WD. 26 (B 54) -- INTERMEDIATE BUFFER ADDRESS
861          000056                  R.UB==56 ; WD. 27 (B 56) -- USER BUFFER ADDRESS(INTERMEDIATE TRANSFER)
862                                  ;
863                                  ; THE LOW ORDER THREE-BITS OF THE I/O FUNCTION CODE ARE USED BY THE SYSTEM
864                                  ; AS FOLLOWS:
865                                  ;
866          000001                  RF.IT==000001    ;[0] -- RESERVED FOR FUTURE USE
867          000002                  RF.XR==000002    ;[1] -- "EXPRESS REQUEST"
868          000004                  RF.IR==000004    ;[2] -- RESERVED FOR FUTURE USE
869                                  ;
870                                  ; *** WHENEVER AN I/O REQUEST IS QUEUED BY THE "QUEUE I/O" DIRECTIVE, THE
871                                  ; DPB SIZE AND THE REQUESTOR'S ATL NODE ADDRESS ARE RECORDED IN THE I/O
872                                  ; REQUEST NODE.  WHENEVER AN I/O REQUEST IS QUEUED AS A RESULT OF ANOTHER
873                                  ; DIRECTIVE (VIZ., "REQUEST" CAUSING A TASK IMAGE TO BE LOADED), THE DPB
874                                  ; SIZE AND THE REQUESTOR'S ATL NODE ADDRESS ARE SET TO ZERO.  THUS, BOTH
875                                  ; BOTH THE DPB SIZE AND THE ATL NODE ADDRESS ARE ALSO "EXEC REQUEST"
876                                  ; INDICATORS.
```



```
878                                  ; CKQ -- CLOCK QUEUE
879                                  ;
880                                  ; THE CLOCK QUEUE IS A DEQUE CONSISTING OF ONE NODE FOR EACH OPERATION
881                                  ; SCHEDULED TO BE PERFORMED AT SOME FUTURE TIME.  A "SCHEDULE DELTA-
882                                  ; TIME" IN THE FIRST NODE (IF ANY) OF THE CLOCK QUEUE IS DECREMENTED
883                                  ; AT EACH CLOCK TICK UNTIL THE NODE "COMES DUE", AT WHICH TIME THE
884                                  ; INDICATED OPERATION IS PERFORMED.  CLOCK QUEUE NODES ARE LINKED
885                                  ; IN THE ORDER IN WHICH THEY WILL COME DUE, AND THE SCHEDULE DELTA-TIME
886                                  ; IN EACH NODE (EXCEPT THE FIRST) IS RELATIVE TO THE SCHEDULE TIME
887                                  ; OF THE PREVIOUS CLOCK QUEUE NODE.  CLOCK QUEUE NODES ARE OF THE
888                                  ; FOLLOWING FORMAT.
889                                  ;
890                                           ; WD. 00 -- FORWARD LINKAGE
891                                           ; WD. 01 -- BACKWARD LINKAGE
892                                           ; WD. 02 -- NODE ACCOUNTING WORD (STD ENTRY ADR OF REQUESTOR)
893          000004                  C.TD==4.AW
894          000006                  C.AT==06 ; WD. 03 -- ATL NODE ADDRESS OF REQUESTOR
895          000010                  C.SD==10 ; WD. 04 -- SCHEDULE DELTA IN TICKS (64-BITS)
```

```
                                ; WD. 05 -- (LOWER ORDER HALF OF SCHEDULE DELTA)
000014          C.RT==14        ; WD. 06 -- REQUEST TYPE INDICATOR & UNUSED BYTE
                !
000016          C.FM==16        ; WD. 07 -- [MT] FLAG MASK (BIS SRC)
000020          C.FA==20        ; WD. 10 -- [MT] FLAGS WORD ADR (BIS DST ADR)
000022          C.FN==22        ; WD. 11 -- [MT] EVENT FLAG NUMBER
000024          C.AE==24        ; WD. 12 -- [MT] VIRTUAL ADDRESS OF AST SERVICE ENTRY
                                ; (5 UNUSED WORDS)
                !
000016          C.RI==16        ; WD. 07 -- [TS] RESCHEDULE INTERVAL IN TICKS (64-BITS)
                                ; WD. 10 -- [TS] (LOW ORDER HALF OF RESCHEDULE INTERVAL)
000022          C.R2==22        ; WD. 11 -- [TS] STD ENTRY ADR OF REQUESTED TASK (R2 FOR '.REQS')
000024          C.R3==24        ; WD. 12 -- [TS] TPD ENTRY ADR, OR ZERO (R3 FOR '.REQS')
000026          C.R4==26        ; WD. 13 -- [TS] RUN PRIORITY, OR ZERO (R4 FOR '.REQS')
000030          C.UI==30        ; WD. 14 -- [TS] UIC INDICATOR FOR '.REQS'
000032          C.TI==32        ; WD. 15 -- [TS] TI IDENTIFICATION FOR '.REQS'
                                ; (2 UNUSED WORDS)
                !
                ; [MT] -- MARK TIME NODE ENTRIES
                ; [TS] -- TASK SCHEDULING NODE ENTRIES
                !
                ; REQUEST TYPE INDICATORS:
                !
                ;       0 -- MARK TIME
000400          TM.SL==400              ;SUB CODE FOR AN INTERNAL TIME SLICE
                ;       1 -- TASK SCHEDULING (SINGLE SHOT)
                ;       2 -- TASK SCHEDULING WITH PERIODIC RESCHEDULING
                !
                ;       NOTE -- THE CLOCK QUEUE SCAN ROUTINE IN "CANCEL SCHEDULED REQUESTS"
                ;       ASSUMES TASK SCHEDULING IF NON-ZERO REQUEST TYPE INDICATOR.
```

```
                                ; ASQ -- ASYNCHRONOUS SYSTEM TRAP QUEUE
                                ;
                                ; THE "ASQ" IS A DEQUE (FIFO), WITH LISTHEAD IN ATL ENTRIES, CONSISTING
                                ; OF ONE NODE FOR EACH AST (ASYNCHRONOUS SYSTEM TRAP) TO BE EXECUTED FOR
                                ; THE TASK DEFINED BY THE STD ENTRY.  ASQ NODES ARE OF THE FOLLOWING
                                ; FORMAT.
                                ;
                                        ; WD. 00 -- FORWARD LINKAGE
                                        ; WD. 01 -- BACKWARD LINKAGE
                                        ; WD. 02 -- ACCOUNTING WORD (STD ENTRY ADDRESS OF CHARGED TASK)
000006                          Y.TT==06 ; WD. 03 -- AST TYPE & NUMBER OF PARAMETERS **
000010                          Y.AE==10 ; WD. 04 -- AST ENTRY POINT
000012                          Y.P1==12 ; WD. 05 -- AST PARAMETER 1
                                        ; WD. 05 -- AST PARAMETER 2
                                        ; WD. 06 -- AST PARAMETER 3
                                        ; .... ETC.
                                ;
                                ; ** THE AST TYPE & NUMBER OF PARAMETER DEFINITIONS ARE AS FOLLOWS:
                                ;
000400                          YF.MT==0+<400*1>        ;MARK-TIME (PARAMETER: EVENT FLAG NUMBER)
000401                          YF.IC==1+<400*1>        ;I/O COMPLETION (PARAMETER: STATUS BLOCK ADDRESS)
001002                          YF.FE==2+<400*2>        ;F.P. EXCEPTION (PARAMETERS: EXCEPTION CODE & ADDRESS)
000003                          YF.PR==3+<400*0>        ;POWER RECOVERY (NO PARAMETERS)
000004                          YF.RE==4+<400*0>        ;RECEIVE QUEUE'D (NO PARAMETERS)
001407                          YF.PC==7+<400*3>        ;COMMUNICATIONS AST
                                ;
```

```
                                ;
                                ; SRQ -- SEND/RECEIVE QUEUE
                                ;
                                ; THE "SRQ" IS A DEQUE (FIFO), WITH LISTHEAD IN STD ENTRIES. CONSISTING
                                ; ONE NODE FOR EACH BLOCK OF DATA "SENT" (VIA "SEND" OR "SEND & REQUEST"
                                ; DIRECTIVES) TO THE TASK DEFINED BY THE STD ENTRY.  RQS NODES ARE OF
                                ; THE FOLLOWING FORMAT.
                                ;
                                        ; WD. 00 -- FORWARD LINKAGE
                                        ; WD. 01 -- BACKWARD LINKAGE
000004                          D.SI==N.AW ; WD. 02 (B 04) -- SENDER ID (NAW)
000006                          D.TI==N.TI ; WD. 03 (B 06) -- TI INDICATOR
000010                          D.PR==10   ; WD. 04 (B 10) -- PRIORITY OF SEND
000011                          D.BS==11   ;       (B 11) -- BUFFER SIZE (WORDS)
000012                          D.D1==12   ; WD. 05 (B 12) -- FIRST WORD OF DATA BLOCK
```



```
                                ; MCR -- MCR COMMAND BUFFER
                                ;
                                ; THE MCR COMMAND BUFFER IS A 96 BYTE BUFFER THAT HOLDS THE DATA
                                ; FOR A REQUESTED MCR FUNCTION.THE BUFFER IS SET UP BY THE MCR
                                ; DISPATCH FUNCTION AND IS RETURNED TO THE POOL BY THE 'GET MCR
                                ; COMMAND LINE  DIRECTIVEAFTER THE INFORMATION HAS BEEN PASSED
                                ; TO THE MCR FUNCTION.THE BUFFERS ARE LINKED TO THE MCR BUFFER
                                ; LIST BY THE MCR DISPATCH FUNCTION.
                                ;
                                ;       ; WD. 00 -- FORWARD LINKAGE
                                ;       ; WD. 01 -- BACKWARD LINKAGE
                                        ; WD. 02 -- NODE ACCOUNTING WORD
000006                          M.TN==6         ; WD. 03 -- SECOND HALF OF MCR TASK NAME
000010                          M.TI==10        ; WD. 04 -- TI ADDRESS OF MCR FUNCTION
000012                          M.BC==12        ; WD. 05 -- NO. OF BYTES IN COMMAND LINE
000014                          M.BF==14        ; WD. 06 -- START OF DATA AREA IN BUFFER
```

APPENDIX B

GLOSSARY

ATL (Active Task List)	A priority-ordered list of Active Tasks used to drive the system. The ATL is a deque consisting of one node for each Active Task in the system.
CLOCK QUEUE	The Clock Queue is a deque consisting of one node for each item to be done at some time in the future, such as scheduling of Tasks (Via the SCHEDULE and MARK TIME Directives), and rescheduling of Tasks (Clock interrupt service routine). The nodes are linked in the order in which they come due.
COMMON BLOCK, INTERNAL	An area of contiguous memory within a partition, available only to the Task in the partition during its residency.
COMMON BLOCK, SYSTEM	An area of contiguous memory, defined at System Generation time, where data can be stored and referenced by all Tasks. A SYSTEM COMMON BLOCK is referenced by using a COMMON name matching a SYSTEM COMMON BLOCK name and declaring that COMMON as SYSTEM COMMON to the Task Builder.
DEFAULT PRIORITY	A priority given to a Task during Task Building or Task Installation that is used when a priority is not specified and the Task's execution is requested or scheduled.
DEQUE	A double-ended queue consisting of a listhead and list elements (nodes), circularly linked by both forward and backward pointers. Deques or linked lists are used to store system information.
DEVICE HANDLER	A Task in the RSX-11D system which drives or services an I/O device. Handler tasks are activated using the Queue I/O directive.
DIRECTIVE	See SYSTEM DIRECTIVE
DISK-RESIDENT TASK	A Task which normally resides on the disk and is brought into a memory partition to execute.

DPB (DIRECTIVE PARAMETER BLOCK)	A block of up to 12 (decimal) contiguous words containing information needed in processing a System Directive.
EVENT FLAG	One of 64 bits associated with a Task, which is set or cleared to indicate that a particular Significant Event has occurred.
EXECUTIVE	The Executive coordinates all activities in the system including Task scheduling, I/O supervision, resource allocation, and interactive operator communication.
I/O RUNDOWN	A process which delays the availability of a partition until all transfers to and from that partition have been stopped or have been allowed to complete. I/O RUNDOWN is invoked when a Task is terminated by the Executive or by the ABORT MCR Function Task and has outstanding transfers pending to/from its partition.
LISTHEAD	A 2-word memory block with forward and backward pointers pointing to the next and previous list node or to itself if empty. The listhead is a reference point in a circularly-linked list.
LINKED LIST	A deque consisting of nodes and a listhead used to store system information. An empty list consists of only a listhead.
LUN (LOGICAL UNIT NUMBER)	Logical Unit Numbers are used to represent logical I/O device units rather than physical units. Each Logical Unit Number is represented by an entry in the Logical Unit Table.
LUT (LOGICAL UNIT TABLE)	A block of contiguous memory with a 1-word entry, or slot, for each Logical Unit Number. When a LUN is assigned to a physical device unit, the corresponding LUT slot contains the address of the appropriate Physical Unit Directory node.
MCR (MONITOR CONSOLE ROUTINE)	The MCR allows the user to communicate on-line with the system from the console Teletype. The MCR consists of the Resident MCR Task, which accepts user's commands, and the MCR Functions, which actually carry out the indicated requests.

MEMORY-RESIDENT TASK	A Task which has been fixed-in-memory or which is assembled as part of the Executive.
MONITOR CONSOLE	The control Teletype of the RSX-11D system where MCR Function requests may be issued by the operator.
NODES	The list elements of a deque. All nodes (of dynamic lists) consist of the listhead, followed by data (list elements).
PARTITION	An area of contiguous memory within which Tasks are executed; defined at System Generation time.
PUD (PHYSICAL UNIT DIRECTORY)	A table constructed during System Generation to describe the I/O devices and units in the system. When a logical I/O number is assigned to a physical unit, the device and unit are set in a LUT entry corresponding to the LUN.
POOL (POOL OF EMPTY NODES)	Empty 17-word nodes for use in any deque. The pool is generated by System Generation from a core area not specified for other use.
SIGNIFICANT EVENT	An event which results in the scanning of the active task list. The following events are considered significant events: 1) I/O queuing, 2) normal I/O request completion, 3) a task request, 4) a scheduled RUN, SCHEDULE, or SYNC coming due, 5) a Mark Time expiration, 6) a task resumption (Resume directive), and 7) a task exit(Exit directive).
STD (SYSTEM TASK DIRECTORY)	A directory of all tasks installed in the system.
SYSTEM GENERATION	The process through which the user tailors the RSX-11D system to best fit his requirements.
SYSTEM DIRECTIVES	Instructions to the RSX-11D Executive to perform special functions, such as I/O, etc.
TASK	A program written by the user or supplied by Digital which is built via the Task Builder, installed in the system via the Monitor Console Routine, and scheduled and executed on a priority basis.
TKB (TASK BUILDER)	The Task Builder program is used to build Task files from relocatable binary files.

APPENDIX C

QIOMAC.MAC

```
                                .TITLE  QIOMAC - QIOSYM MACRO DEFINITION
                        ; ALTERED SUNDAY 24-NOV-74 13:00
                        ; ALTERED TUESDAY 28-JAN-75 13:50:00
                        ; ALTERED THURSDAY 06-FEB-75 15:50
                        ; ALTERED MONDAY 24-FEB-75 15:40:00 BY ED MARISON
                        ; ALTERED TUE 25-MAR-75 15:30 EDIT # +001
                        ;
                        ; ***** ALWAYS UPDATE THE FOLLOWING TWO LINES TOGETHER
                                .IDENT  /0304/
000304                          QI.VER=0304
                        ;
                        ; COPYRIGHT 1974,1975, DIGITAL EQUIPMENT CORP., MAYNARD MASS.

                        ; THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE FOR USE
                        ; ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION
                        ; OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT
                        ; AS MAY OTHERWISE BE PROVIDED IN WRITING BY DEC.

                        ; THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
                        ; EQUIPMENT CORPORATION.

                        ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
                        ; OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
                        ;
                        ;
                        ;
                        ;+
                        ; MACRO TO DEFINE STANDARD QUEUE I/O DIRECTIVE FUNCTION VALUES
                        ; AND IOSB RETURN VALUES.  TO INVOKE AT ASSEMBLY TIME (WITH LOCAL
                        ; DEFINITION) USE:
                        ;
                        ;       QIOSYS          ;DEFINE SYMBOLS
                        ;
                        ; TO OBTAIN GLOBAL DEFINITION OF THESE SYMBOLS USE:
                        ;
                        ;       QIOSYS DEF$G    ;SYMBOLS DEFINED GLOBALLY
                        ;
                        ; THE MACRO CAN BE CALLED ONCE ONLY AND THEN
                        ; REDEFINES ITSELF AS NULL.
                        ;-

                                .MACRO  QIOSYS  $$$GBL,$$$MSG
                                .IIF    IDN,<$$$GBL>,<DEF$G>,   .GLOBL  QI.VER
                                .IF     IDN,<$$$MSG>,<DEF$S>
                                $$$MAX=0
                                $$MSG=1
                                .IFF
                                $$MSG=0
                                .ENDC
                                .MCALL  IOERR$
                                IOERR$  $$$GBL          ;I/O ERROR CODES FROM HANDLERS, FCP, FCS
                                .MCALL  DRERR$
                                DRERR$  $$$GBL          ;DIRECTIVE STATUS WORD ERROR CODES
                                .IF     DIF,<$$$MSG>,<DEF$S>
                                .MCALL  FILIO$
                                FILIO$  $$$GBL          ;DEFINE GENERAL QI/O FUNCTION CODES
```

```
                        .MCALL  SPCIO$
                        SPCIO$  $$$GBL          ;DEVICE DEPENDENT I/O FUNCTION CODES
                        .MACRO  QIOSY$  ARG,ARG1,ARG2   ;RECLAIM MACRO STORAGE
                        .ENDM   QIOSY$
                        .ENDC
                        .ENDM   QIOSY$
```

```
                        ;
                        ; DEFINE THE ERROR CODES RETURNED BY DEVICE HANDLER AND FILE PRIMITIVES
                        ; IN THE FIRST WORD OF THE I/O STATUS BLOCK
                        ; THESE CODES ARE ALSO RETURNED BY FILE CONTROL SERVICES (FCS) IN THE
                        ; BYTE F.ERR IN THE FILE DESCRIPTOR BLOCK (FDB)
                        ;       THE BYTE F.ERR+1 IS 0 IF F.ERR CONTAINS A HANDLER OR FCP ERROR CODE.
                        ;
                                .MACRO  IOERR$  $$$GBL
                                .MCALL  .IOER,,DEFIN$
                                .IF     IDN,<$$$GBL>,<DEF$G>
                                ...GBL=1
                                .IFF
                                ...GBL=0
                                .ENDC
                                .IIF    NDF,$$MSG,$$MSG=0
                        ;
                        ; SYSTEM STANDARD CODES, USED BY ALL FUNCTIONS
                        ;
                                .IOER,  IE.BAD,-01.,,<BAD PARAMETERS>
                                .IOER,  IE.IFC,-02.,,<INVALID FUNCTION CODE>
                                .IOER,  IE.DNR,-03.,,<DEVICE NOT READY>
                                .IOER,  IE.VER,-04.,,<PARITY ERROR ON DEVICE>
                                .IOER,  IE.ONP,-05.,,<HARDWARE OPTION NOT PRESENT>
                                .IOER,  IE.SPC,-06.,,<ILLEGAL USER BUFFER>
                                .IOER,  IE.DNA,-07.,,<DEVICE NOT ATTACHED>
                                .IOER,  IE.DAA,-08.,,<DEVICE ALREADY ATTACHED>
                                .IOER,  IE.DUN,-09.,,<DEVICE NOT ATTACHABLE>
                                .IOER,  IE.EOF,-10.,,<END OF FILE DETECTED>
                                .IOER,  IE.EOV,-11.,,<END OF VOLUME DETECTED>
                                .IOER,  IE.WLK,-12.,,<WRITE ATTEMPTED TO LOCKED UNIT>
                                .IOER,  IE.DAO,-13.,,<DATA OVERRUN>
                                .IOER,  IE.SRE,-14.,,<SEND/RECEIVE FAILURE>
                                .IOER,  IE.ABO,-15.,,<REQUEST TERMINATED>
                                .IOER,  IE.PRI,-16.,,<PRIVILEGE VIOLATION>
                                .IOER,  IE.RSU,-17.,,<SHARABLE RESOURCE IN USE>
                                .IOER,  IE.OVR,-18.,,<ILLEGAL OVERLAY REQUEST>
                                .IOER,  IE.BYT,-19.,,<ODD BYTE COUNT (OR VIRTUAL ADDRESS)>
                                .IOER,  IE.BLK,-20.,,<LOGICAL BLOCK NUMBER TOO LARGE>
                                .IOER,  IE.MOD,-21.,,<INVALID UDC MODULE #>
                                .IOER,  IE.CON,-22.,,<UDC CONNECT ERROR>
                                .IOER,  IE.BBE,-56.,,<BAD BLOCK ON DEVICE>
                                .IOER,  IE.STK,-58.,,<NOT ENOUGH STACK SPACE (FCS OR FCP)>
                                .IOER,  IE.FHE,-59.,,<FATAL HARDWARE ERROR ON DEVICE>
                                .IOER,  IE.EOT,-62.,,<END OF TAPE DETECTED>
                                .IOER,  IE.OFL,-65.,,<DEVICE OFF LINE>
                                .IOER,  IE.BCC,-66.,,<BLOCK CHECK OR CRC ERROR>


                        ;
                        ; FILE PRIMITIVE CODES
                        ;

                                .IOER,  IE.NOD,-23.,,<CALLER'S NODES EXHAUSTED>
                                .IOER,  IE.DFU,-24.,,<DEVICE FULL>
                                .IOER,  IE.IFU,-25.,,<INDEX FILE FULL>
                                .IOER,  IE.NSF,-26.,,<NO SUCH FILE>
```

```
                        .IOER,  IE.LCK,-27.,,<LOCKED FROM WRITE ACCESS>
                        .IOER,  IE.HFU,-28.,,<FILE HEADER FULL>
                        .IOER,  IE.WAC,-29.,,<ACCESSED FOR WRITE>
                        .IOER,  IE.CKS,-30.,,<FILE HEADER CHECKSUM FAILURE>
                        .IOER,  IE.WAT,-31.,,<ATTRIBUTE CONTROL LIST FORMAT ERROR>
                        .IOER,  IE.RER,-32.,,<FILE PROCESSOR DEVICE READ ERROR>
                        .IOER,  IE.WER,-33.,,<FILE PROCESSOR DEVICE WRITE ERROR>
                        .IOER,  IE.ALN,-34.,,<FILE ALREADY ACCESSED ON LUN>
                        .IOER,  IE.SNC,-35.,,<FILE ID, FILE NUMBER CHECK>
                        .IOER,  IE.SQC,-36.,,<FILE ID, SEQUENCE NUMBER CHECK>
                        .IOER,  IE.NLN,-37.,,<NO FILE ACCESSED ON LUN>
                        .IOER,  IE.CLO,-38.,,<FILE WAS NOT PROPERLY CLOSED>
                        .IOER,  IE.DUP,-57.,,<ENTER - DUPLICATE ENTRY IN DIRECTORY>
                        .IOER,  IE.BVR,-63.,,<BAD VERSION NUMBER>
                        .IOER,  IE.BHD,-64.,,<BAD FILE HEADER>
                        .IOER,  IE.EXP,-75.,,<FILE EXPIRATION DATE NOT REACHED>
                        .IOER,  IE.BTF,-76.,,<BAD TAPE FORMAT>

                ;
                ; FILE CONTROL SERVICES CODES
                ;

                        .IOER,  IE.NBF,-39.,,<OPEN - NO BUFFER SPACE AVAILABLE FOR FILE>
                        .IOER,  IE.RBG,-40.,,<ILLEGAL RECORD SIZE>
                        .IOER,  IE.NBK,-41.,,<FILE EXCEEDS SPACE ALLOCATED, NO BLOCKS>
                        .IOER,  IE.ILL,-42.,,<ILLEGAL OPERATION ON FILE DESCRIPTOR BLOCK>
                        .IOER,  IE.BTP,-43.,,<BAD RECORD TYPE>
                        .IOER,  IE.RAC,-44.,,<ILLEGAL RECORD ACCESS BITS SET>
                        .IOER,  IE.RAT,-45.,,<ILLEGAL RECORD ATTRIBUTES BITS SET>
                        .IOER,  IE.RCN,-46.,,<ILLEGAL RECORD NUMBER - TOO LARGE>
                        .IOER,  IE.MBK,-47.,,<MULTIPLE BLOCK READ/WRITE - NOT IMPLEMENTED YET>
                        .IOER,  IE.2DV,-48.,,<RENAME - 2 DIFFERENT DEVICES>
                        .IOER,  IE.FEX,-49.,,<RENAME - NEW FILE NAME ALREADY IN USE>
                        .IOER,  IE.BDR,-50.,,<BAD DIRECTORY FILE>
                        .IOER,  IE.RNM,-51.,,<CAN'T RENAME OLD FILE SYSTEM>
                        .IOER,  IE.BDI,-52.,,<BAD DIRECTORY SYNTAX>
                        .IOER,  IE.FOP,-53.,,<FILE ALREADY OPEN>
                        .IOER,  IE.BNM,-54.,,<BAD FILE NAME>
                        .IOER,  IE.BDV,-55.,,<BAD DEVICE NAME>
                        .IOER,  IE.NFI,-60.,,<FILE ID WAS NOT SPECIFIED>
                        .IOER,  IE.ISQ,-61.,,<ILLEGAL SEQUENTIAL OPERATION>
                        .IOER,  IE.NNC,-77.,,<NOT ANSI 'D' FORMAT BYTE COUNT>
                ;
                ; NETWORK ACP CODES
                ;
                        .IOER,  IE.AST,-67.,,<NO AST SPECIFIED IN CONNECT>
                        .IOER,  IE.NNN,-68.,,<NO SUCH NODE>
                        .IOER,  IE.NFW,-69.,,<PATH LOST TO PARTNER> ;+001 THIS CODE MUST BE ODD
                        .IOER,  IE.BLB,-70.,,<BAD LOGICAL BUFFER> ;+001
                        .IOER,  IE.TMM,-71.,,<TOO MANY OUTSTANDING MESSAGES>
                        .IOER,  IE.NDR,-72.,,<NO DYNAMIC SPACE AVAILABLE>
                        .IOER,  IE.CNR,-73.,,<CONNECTION REJECTED>
                        .IOER,  IE.TMO,-74.,,<TIMEOUT ON REQUEST>
                        .IOER,  IE.NNL,-78.,,<NOT A NETWORK LUN> ;+001

                ;
                ; SUCCESSFUL RETURN CODES---
```

```
                        ;
                        ;
                                DEFIN$  IS.PND,+00.     ;OPERATION PENDING
                                DEFIN$  IS.SUC,+01.     ;OPERATION COMPLETE, SUCCESS
                                DEFIN$  IS.BV,+05.      ;ON A/D READ, AT LEAST ONE BAD VALUE
                                                        ;WAS READ (REMAINDER MAY BE GOOD),
                                                        ;BAD CHANNEL IS INDICATED BY A
                                                        ;NEGATIVE VALUE IN THE BUFFER.
                        ;
                        ; TTY SUCCESS CODES:
                        ;
                                DEFIN$  IS.CR,<15+400+1> ;CARRIAGE RETURN WAS TERMINATOR
                                DEFIN$  IS.ESC,<33+400+1> ;ESCAPE (ALTMODE) WAS TERMINATOR
                        ;
                        ; ******
                        ;
                        ; THE NEXT AVAILABLE ERROR NUMBER IS: -79.
                        ; ALL LOWER NUMBERS ARE IN USE!!
                        ;
                        ; *****
                                .IF     EQ,$$MSG
                                .MACRO  IOERR$  A
                                .ENDM   IOERR$
                                .ENDC
                                .ENDM   IOERR$
```

```
                        ;
                        ; DEFINE THE DIRECTIVE ERROR CODES RETURNED IN THE DIRECTIVE STATUS WORD
                        ;
                        ;       FILE CONTROL SERVICES (FCS) RETURNS THESE CODES IN THE BYTE F.ERR
                        ; OF THE FILE DESCRIPTOR BLOCK (FDB).  TO DISTINGUISH THEM FROM THE
                        ; OVERLAPPING CODES FROM HANDLER AND FILE PRIMITIVES, THE BYTE
                        ; F.ERR+1 IN THE FDB WILL BE NEGATIVE FOR A DIRECTIVE ERROR CODE.
                        ;
                                .MACRO  DRERR$  $$$GBL
                                .MCALL  .QIOE.,DEFIN$
                                .IF     IDN,<$$$GBL>,<DEF$G>
                                ...GBL=1
                                .IFF
                                ...GBL=0
                                .ENDC
                                .IIF    NDF,$$MSG,$$MSG=0
                        ;
                        ; STANDARD ERROR CODES RETURNED BY DIRECTIVES IN THE DIRECTIVE STATUS WORD
                        ;
                                .QIOE.  IE.UPN,-01.,<INSUFFICIENT DYNAMIC STORAGE>
                                .QIOE.  IE.INS,-02.,<SPECIFIED TASK NOT INSTALLED>
                                .QIOE.  IE.ULN,-05.,<UN-ASSIGNED LUN>
                                .QIOE.  IE.HWR,-06.,<HANDLER TASK NOT RESIDENT>
                                .QIOE.  IE.ACT,-07.,<TASK NOT ACTIVE>
                                .QIOE.  IE.ITS,-08.,<DIRECTIVE INCONSISTENT WITH TASK STATE>
                                .QIOE.  IE.CKP,-10.,<ISSUING TASK NOT CHECKPOINTABLE>
                        ;
                        ;
                        ;
                                .QIOE.  IE.AST,-80.,<DIRECTIVE ISSUED/NOT ISSUED FROM AST>
                                .QIOE.  IE.LNL,-90.,<LUN LOCKED IN USE>
                                .QIOE.  IE.IDU,-92.,<INVALID DEVICE OR UNIT>
                                .QIOE.  IE.ITI,-93.,<INVALID TIME PARAMETERS>
                                .QIOE.  IE.IPR,-95.,<INVALID PRIORITY ( .GT. 250.)>
                                .QIOE.  IE.ILU,-96.,<INVALID LUN>
                                .QIOE.  IE.IEF,-97.,<INVALID EVENT ( .GT. 64.)>
                                .QIOE.  IE.ADP,-98.,<PART OF DPB OUT OF USER'S SPACE>
                                .QIOE.  IE.SDP,-99.,<DIC OR DPB SIZE INVALID>
                        ;
                        ; SUCCESS CODES FROM DIRECTIVES - PLACED IN THE DIRECTIVE STATUS WORD
                        ;
                                DEFIN$  IS.CLR,0        ;EVENT FLAG WAS CLEAR
                                                        ;FROM CLEAR EVENT FLAG DIRECTIVE
                                DEFIN$  IS.SET,2        ;EVENT FLAG WAS SET
                                                        ;FROM SET EVENT FLAG DIRECTIVE
                                DEFIN$  IS.SPD,2        ;TASK WAS SUSPENDED
                        ;
                        ;
                                .IF     EQ,$$MSG
                                .MACRO  DRERR$  A
                                .ENDM   DRERR$
                                .ENDC
                                .ENDM   DRERR$
```

```
                        ;
                        ; DEFINE THE GENERAL QI/O FUNCTION CODES - DEVICE INDEPENDENT
                        ;
                                .MACRO  FILIOS  $$$GBL
                                .MCALL  .WORD,,DEFIN$
                                .IF     IDN,<$$$GBL>,<DEF$G>
                                ...GBL=1
                                .IFF
                                ...GBL=0
                                .ENDC
                        ;
                        ; GENERAL QI/O QUALIFIER BYTE DEFINITIONS
                        ;
                                .WORD.  IQ.X,001,000    ;NO ERROR RECOVERY
                                .WORD.  IQ.Q,002,000    ;QUEUE REQUEST IN EXPRESS QUEUE
                        ;;;     .WORD.  IQ.,004,000     ;RESERVED
                        ;
                        ; EXPRESS QUEUE COMMANDS
                        ;

                                .WORD.  IO.KIL,012,000  ;KILL CURRENT REQUEST
                                .WORD.  IO.RDN,022,000  ;I/O RUNDOWN
                                .WORD.  IO.UNL,042,000  ;UNLOAD I/O HANDLER TASK
                                .WORD.  IO.LTK,050,000  ;LOAD A TASK IMAGE FILE
                                .WORD.  IO.RTK,060,000  ;RECORD A TASK IMAGE FILE
                        ;
                        ; GENERAL DEVICE HANDLER CODES
                        ;
                                .WORD.  IO.WLB,000,001  ;WRITE LOGICAL BLOCK
                                .WORD.  IO.RLB,000,002  ;READ LOGICAL BLOCK
                                .WORD.  IO.LOV,010,002  ;LOAD OVERLAY (DISK DRIVER)
                                .WORD.  IO.ATT,000,003  ;ATTACH A DEVICE TO A TASK
                                .WORD.  IO.DET,000,004  ;DETACH A DEVICE FROM A TASK
                        ;
                        ; DIRECTORY PRIMITIVE CODES
                        ;
                                .WORD.  IO.FNA,000,011  ;FIND FILE NAME IN DIRECTORY
                                .WORD.  IO.RNA,000,013  ;REMOVE FILE NAME FROM DIRECTORY
                                .WORD.  IO.ENA,000,014  ;ENTER FILE NAME IN DIRECTORY
                        ;
                        ; FILE PRIMITIVE CODES
                        ;
                                .WORD.  IO.CLN,000,007  ;CLOSE OUT LUN
                                .WORD.  IO.ACR,000,015  ;ACCESS FOR READ
                                .WORD.  IO.ACW,000,016  ;ACCESS FOR WRITE
                                .WORD.  IO.ACE,000,017  ;ACCESS FOR EXTEND
                                .WORD.  IO.DAC,000,020  ;DE-ACCESS FILE
                                .WORD.  IO.RVB,000,021  ;READ VIRITUAL BLOCK
                                .WORD.  IO.WVB,000,022  ;WRITE VIRITUAL BLOCK
                                .WORD.  IO.EXT,000,023  ;EXTEND FILE
                                .WORD.  IO.CRE,000,024  ;CREATE FILE
                                .WORD.  IO.DEL,000,025  ;DELETE FILE
                                .WORD.  IO.RAT,000,026  ;READ FILE ATTRIBUTES
                                .WORD.  IO.WAT,000,027  ;WRITE FILE ATTRIBUTES
                                .WORD.  IO.APV,010,030  ;PRIVILEGED ACP CONTROL
                                .WORD.  IO.APC,000,030  ;ACP CONTROL
                        ;
```

```
320                                     ;
321                                             .MACRO  FILIO$  A
322                                             .ENDM   FILIO$
323                                             .ENDM   FILIO$
```

```
                        ;
                        ; DEFINE THE QI/O FUNCTION CODES THAT ARE SPECIFIC TO INDIVIDUAL DEVICES
                        ;
                                .MACRO  $PCIO$  $$$GBL
                                .MCALL  .WORD.,.DEFIN$
                                .IF     IDN,<$$$GBL>,<DEFSG>
                                ...GBL=1
                                .IFF
                                ...GBL=0
                                .ENDC
                        ;
                        ; QI/O FUNCTION CODES FOR SPECIFIC DEVICE DEPENDENT FUNCTIONS
                        ;
                                .WORD.  IO.WLV,100,001  ;WRITE LOGICAL REVERSE (DECTAPE)
                                .WORD.  IO.WLS,010,001  ;(COMMUNICATIONS) WRITE PRECEDED BY SYNC TRAIN
                                .WORD.  IO.WNS,020,001  ;(COMMUNICATIONS) WRITE, NO SYNC TRAIN
                                .WORD.  IO.RLV,100,002  ;READ REVERSE (DECTAPE)
                                .WORD.  IO.RNC,040,002  ;READ - NO LOWER CASE CONVERT (TTY)
                                .WORD.  IO.RAL,010,002  ;READ PASSING ALL CHARACTERS (TTY)
                                .WORD.  IO.RNE,020,002  ;READ WITHOUT ECHO (TTY)
                                .WORD.  IO.RDB,200,002  ;READ BINARY MODE (CARD READER)
                                .WORD.  IO.RHD,010,002  ;(COMMUNICATIONS) READ, STRIP SYNC
                                .WORD.  IO.RNS,020,002  ;(COMMUNICATIONS) READ, DON'T STRIP SYNC
                                .WORD.  IO.CRC,040,002  ;(COMMUNICATIONS) READ, DON'T CLEAR CRC
                                .WORD.  IO.RIC,000,005  ;READ SINGLE CHANNEL (AFC, AD01, UDC)
                                .WORD.  IO.INL,000,005  ;(COMMUNICATIONS) INITIALIZATION FUNCTION
                                .WORD.  IO.TRM,010,005  ;(COMMUNICATIONS) TERMINATION FUNCTION
                                .WORD.  IO.RBC,000,006  ;READ MULTICHANNELS (BUFFER DEFINES CHANNELS)
                                .WORD.  IO.MOD,000,006  ;(COMMUNICATIONS) SETMODE FUNCTION FAMILY
                                .WORD.  IO.HDX,010,006  ;(COMMUNICATIONS) SET UNIT HALF DUPLEX
                                .WORD.  IO.FDX,020,006  ;(COMMUNICATIONS) SET UNIT FULL DUPLEX
                                .WORD.  IO.SYN,040,006  ;(COMMUNICATIONS) SPECIFY SYNC CHARACTER
                                .WORD.  IO.RTC,000,007  ;READ CHANNEL - TIME BASED
                                .WORD.  IO.RWD,000,005  ;REWIND (MAGTAPE, DECTAPE)
                                .WORD.  IO.SPB,020,005  ;MAGTAPE, SPACE "N" BLOCKS
                                .WORD.  IO.SPF,040,005  ;MAGTAPE, SPACE "N" EOF MARKS
                                .WORD.  IO.EOF,000,006  ;MAGTAPE, WRITE EOF
                                .WORD.  IO.STC,100,005  ;MAGTAPE, SET CHARACTERISTIC
                                .WORD.  IO.SEC,120,005  ;MAGTAPE, SENSE CHARACTERISTIC
                                .WORD.  IO.RWU,140,005  ;REWIND AND UNLOAD (MAGTAPE, DECTAPE)
                                .WORD.  IO.SMO,160,005  ;MAGTAPE, MOUNT & SET CHARACTERISTICS
                                .WORD.  IO.SAO,000,010  ;UDC SINGLE CHANNEL ANALOG OUTPUT
                                .WORD.  IO.SSO,000,011  ;UDC SINGLE SHOT, SINGLE POINT
                                .WORD.  IO.MSO,000,012  ;UDC SINGLE SHOT, MULTI-POINT
                                .WORD.  IO.SLO,000,013  ;UDC LATCHING, SINGLE POINT
                                .WORD.  IO.MLO,000,014  ;UDC LATCHING, MULTI-POINT
                                .WORD.  IO.LED,000,024  ;LPS11 WRITE LED DISPLAY LIGHTS
                                .WORD.  IO.SDO,000,025  ;LPS11 WRITE DIGITAL OUTPUT REGISTER
                                .WORD.  IO.SDI,000,026  ;LPS11 READ DIGITAL INPUT REGISTER
                                .WORD.  IO.SCS,000,026  ;UDC CONTACT SENSE, SINGLE POINT
                                .WORD.  IO.REL,000,027  ;LPS11 WRITE RELAY
                                .WORD.  IO.MCS,000,027  ;UDC CONTACT SENSE, MULTI-POINT
                                .WORD.  IO.ADS,000,030  ;LPS11 SYNCHRONOUS A/D SAMPLING
                                .WORD.  IO.CCI,000,030  ;UDC CONTACT INT - CONNECT
                                .WORD.  IO.MDI,000,031  ;LPS11 SYNCHRONOUS DIGITAL INPUT
                                .WORD.  IO.DCI,000,031  ;UDC CONTACT INT - DISCONNECT
                                .WORD.  IO.XMT,000,031  ;(COMMUNICATIONS) TRANSMIT SPECIFIED BLOCK WITH ACK
```

```
382                             .WORD.  IO.XNA,010,031  ;(COMMUNICATIONS) TRANSMIT WITHOUT ACK
383                             .WORD.  IO.HIS,000,032  ;LPS11 SYNCHRONOUS HISTOGRAM SAMPLING
384                             .WORD.  IO.RCI,000,032  ;UDC CONTACT INT - READ
385                             .WORD.  IO.RCV,000,032  ;(COMMUNICATIONS) RECEIVE DATA IN BUFFER SPECIFIED
386                             .WORD.  IO.MDO,000,033  ;LPS11 SYNCHRONOUS DIGITAL OUTPUT
387                             .WORD.  IO.CTI,000,033  ;UDC TIMER - CONNECT
388                             .WORD.  IO.CON,000,033  ;(COMMUNICATIONS) COMMUNICATIONS CONNECT FUNCTION
389                             .WORD.  IO.CPR,010,033  ;(COMMUNICATIONS) COMMUNICATIONS CONNECT NO TIMEOUTS
390                             .WORD.  IO.CAS,020,033  ;(COMMUNICATIONS) COMMUNICATIONS CONNECT WITH AST
391                             .WORD.  IO.CRJ,040,033  ;(COMMUNICATIONS) COMMUNICATIONS CONNECT REJECT
392                             .WORD.  IO.CBO,110,033  ;+001 (COMMUNICATIONS) COMMUNICATIONS BOOT CONNECT
393                             .WORD.  IO.CTR,210,033  ;+001 (COMMUNICATIONS) COMMUNICATIONS TRANSPARENT CONNECT
394                             .WORD.  IO.GNI,010,035  ;(COMMUNICATIONS) COMMUNICATIONS GET NODE INFO
395                             .WORD.  IO.GLI,020,035  ;(COMMUNICATIONS) COMMUNICATIONS GET LINK INFO
396                             .WORD.  IO.GLC,030,035  ;(COMMUNICATIONS) GET LINK INFO CLEAR COUNTERS
397                             .WORD.  IO.GRI,040,035  ;(COMMUNICATIONS) GET REMOTE NODE INFO
398                             .WORD.  IO.GRC,050,035  ;+001 (COMMUNICATIONS) GET REMOTE NODE ERROR COUNTS
399                             .WORD.  IO.GRN,060,035  ;+001 (COMMUN.) GET REMOTE NODE NAME
400                             .WORD.  IO.CSM,070,035  ;+001 (COMMUNICATIONS) CHANGE SOLO MODE
401                             .WORD.  IO.CIN,100,035  ;+001 (COMMUN.) CHANGE CONNECTION INHIBIT
402                             .WORD.  IO.CBN,110,035  ;+001 (COMMUNICATIONS) CIRCULAR BUFFER NCS
403                             .WORD.  IO.CBD,120,035  ;+001 (COMMUNICATIONS) CIRCULAR BUFFER DDCMP
404                             .WORD.  IO.DTI,000,034  ;UDC TIMER - DISCONNECT
405                             .WORD.  IO.DIS,000,034  ;(COMMUNICATIONS) COMMUNICATIONS DISCONNECT FUNCTION
406                             .WORD.  IO.MDA,000,034  ;LPS11 SYNCHRONOUS D/A OUTPUT
407                             .WORD.  IO.RTI,000,035  ;UDC TIMER - READ
408                             .WORD.  IO.CTL,000,035  ;(COMMUNICATIONS) NETWORK CONTROL FUNCTION
409                             .WORD.  IO.STP,000,035  ;LPS11 STOP IN PROGRESS FUNCTION
410                             .WORD.  IO.ITI,000,036  ;UDC TIMER - INITIALIZE
411                             .WORD.  IO.WPB,040,001  ; RX01 - FLOPPY DISK WRITE PHYSICAL BLOCK
412                             .WORD.  IO.RPB,040,002  ; RX01 - FLOPPY DISK READ PHYSICAL BLOCK
413                             .WORD.  IO.SHT,010,005  ;SET HORIZONTAL TAB POSITIONS
414                             .WORD.  IO.SST,030,005  ;SET SPECIAL TERMINATOR CHARACTERS
415                             .WORD.  IO.SEM,040,005  ;SET TERMINAL MODE (CHARACTERISTICS)
416                             .WORD.  IO.SNM,050,005  ;SENSE TERMINAL MODE
417                             .WORD.  IO.CCT,060,005  ;CONNECT TO REMOTE TERMINAL (AUTO DIALOUT)
418                             .WORD.  IO.DCT,070,005  ;DISCONNECT FROM REMOTE TERMINAL (HANGUP)
419                             .WORD.  IO.ESA,100,005  ;ENABLE STATUS AST
420                     ;
421                     ;
422                             .MACRO  SPCIO$  A
423                             .ENDM   SPCIO$
424                             .ENDM   SPCIO$
```

```
426                             ;
427                             ; HANDLER ERROR CODES RETURNED IN I/O STATUS BLOCK ARE DEFINED THROUGH THIS
428                             ; MACRO WHICH THEN CONDITIONALLY INVOKES THE MESSAGE GENERATING MACRO
429                             ; FOR THE QIOSYM.MSG FILE
430                             ;
431                                     .MACRO  .IOER,  SYM,LO,MSG
432                                     DEFIN$  SYM,LO
433                                     .IF     GT,$$MSG
434                                     .MCALL  .IOMG,
435                                     .IOMG,  SYM,LO,<MSG>
436                                     .ENDC
437                                     .ENDM   .IOER,
438                             ;
439                             ; QI/O ERROR CODES ARE DEFINED THOUGH THIS MACRO WHICH THEN INVOKES THE
440                             ; ERROR MESSAGE GENERATING MACRO, ERROR CODES -129 THROUGH -256
441                             ; ARE USED IN THE QIOSYM.MSG FILE
442                             ;
443                                     .MACRO  .QIOE,  SYM,LO,MSG
444                                     DEFIN$  SYM,LO
445                                     .IF     GT,$$MSG
446                                     .MCALL  .IOMG,
447                                     .IOMG,  SYM,<LO-128.>,<MSG>
448                                     .ENDC
449                                     .ENDM   .QIOE,
450                             ;
451                             ; CONDITIONALLY GENERATE DATA FOR WRITING A MESSAGE FILE FOR MO
452                             ;
453                                     .MACRO  .IOMG,  SYM,LO,MSG
454                                     .WORD   -^O<LO>
455                                     .ASCIZ  ^MSG^
456                                     .EVEN
457                                     .IIF    LT,^O<$$$MAX+<LO>>,$$$MAX=-^O<LO>
458                                     .ENDM   .IOMG,
459                             ;
460                             ; DEFINE THE SYMBOL SYM WHERE LO IS IS THE LOW ORDER BYTE, HI IS THE HIGH BYTE
461                             ;
462                                     .MACRO  .WORD,  SYM,LO,HI
463                                     DEFIN$  SYM,<^O<HI*400+LO>>
464                                     .ENDM   .WORD,
```

```
1 000000                                QIOSYS  DEFSG
2          000001                       .END
```

SYMBOL TABLE

```
IE.ABO= 177761 G        IE.IFU= 177747 G        IE.UPN= 177777 G        IO.FNA= 004400 G        IO.RWU= 002540 G
IE.ACT= 177771 G        IE.ILL= 177726 G        IE.VER= 177774 G        IO.GLC= 016430 G        IO.RIC= 002400 G
IE.ADP= 177636 G        IE.ILU= 177640 G        IE.WAC= 177743 G        IO.GLI= 016420 G        IO.SAO= 004000 G
IE.ALN= 177736 G        IE.INS= 177776 G        IE.WAT= 177741 G        IO.GNI= 016410 G        IO.SCS= 013000 G
IE.AST= 177660 G        IE.IPR= 177641 G        IE.WER= 177737 G        IO.GRC= 016450 G        IO.SDI= 013000 G
IE.BAD= 177777 G        IE.ISQ= 177703 G        IE.WLK= 177764 G        IO.GRI= 016440 G        IO.SDO= 012400 G
IE.BBE= 177710 G        IE.ITI= 177643 G        IE.2DV= 177720 G        IO.GRN= 016460 G        IO.SEC= 002520 G
IE.BCC= 177676 G        IE.ITS= 177770 G        IO.ACE= 007400 G        IO.HDX= 003010 G        IO.SEM= 002440 G
IE.BD1= 177714 G        IE.LCK= 177745 G        IO.ACR= 006400 G        IO.HIS= 015000 G        IO.SHT= 002410 G
IE.BDR= 177716 G        IE.LNL= 177646 G        IO.ACW= 007000 G        IO.INL= 002400 G        IO.SLO= 005400 G
IE.BDV= 177711 G        IE.MBK= 177721 G        IO.ADS= 014000 G        IO.ITI= 017000 G        IO.SMO= 002560 G
IE.BHD= 177700 G        IE.MOD= 177753 G        IO.APC= 014000 G        IO.KIL= 000012 G        IO.SNM= 002450 G
IE.BLB= 177672 G        IE.NBF= 177731 G        IO.APV= 014010 G        IO.LED= 012000 G        IO.SPB= 002420 G
IE.BLK= 177754 G        IE.NBK= 177727 G        IO.ATT= 001400 G        IO.LOV= 001010 G        IO.SPF= 002440 G
IE.BNM= 177712 G        IE.NDR= 177670 G        IO.CAS= 015420 G        IO.LTK= 000050 G        IO.SSO= 004400 G
IE.BTF= 177664 G        IE.NFI= 177704 G        IO.CBD= 016520 G        IO.MCS= 013400 G        IO.SST= 002430 G
IE.BTP= 177725 G        IE.NFW= 177673 G        IO.CBN= 016510 G        IO.MDA= 016000 G        IO.STC= 002500 G
IE.BVR= 177701 G        IE.NLN= 177733 G        IO.CBO= 015510 G        IO.MDI= 014400 G        IO.STP= 016400 G
IE.BYT= 177755 G        IE.NNC= 177663 G        IO.CCI= 014000 G        IO.MDO= 015400 G        IO.SYN= 003040 G
IE.CKP= 177766 G        IE.NNL= 177662 G        IO.CCT= 002460 G        IO.MLO= 006000 G        IO.TRM= 002410 G
IE.CKS= 177742 G        IE.NNN= 177674 G        IO.CIN= 016500 G        IO.MOD= 003000 G        IO.UNL= 000042 G
IE.CLO= 177732 G        IE.NOD= 177751 G        IO.CLN= 003400 G        IO.MSO= 005000 G        IO.WAT= 013400 G
IE.CNR= 177667 G        IE.NSF= 177746 G        IO.CON= 015400 G        IO.RAL= 001010 G        IO.WLB= 000400 G
IE.CON= 177752 G        IE.OFL= 177677 G        IO.CPR= 015410 G        IO.RAT= 013000 G        IO.WLS= 000410 G
IE.DAA= 177770 G        IE.ONP= 177773 G        IO.CRC= 001040 G        IO.RBC= 003000 G        IO.WLV= 000500 G
IE.DAO= 177763 G        IE.OVR= 177756 G        IO.CRE= 012000 G        IO.RCI= 015000 G        IO.WNS= 000420 G
IE.DFU= 177750 G        IE.PRI= 177760 G        IO.CRJ= 015440 G        IO.RCV= 015000 G        IO.WPB= 000440 G
IE.DNA= 177771 G        IE.RAC= 177724 G        IO.CSM= 016470 G        IO.RDB= 001200 G        IO.WVB= 011000 G
IE.DNR= 177775 G        IE.RAT= 177723 G        IO.CTI= 015400 G        IO.RDN= 000022 G        IO.XMT= 014400 G
IE.DUN= 177767 G        IE.RBG= 177730 G        IO.CTL= 016400 G        IO.REL= 013400 G        IO.XNA= 014410 G
IE.DUP= 177707 G        IE.RCN= 177722 G        IO.CTR= 015610 G        IO.RHD= 001010 G        IQ.Q  = 000002 G
IE.EOF= 177766 G        IE.RER= 177740 G        IO.DAC= 010000 G        IO.RLB= 001000 G        IQ.X  = 000001 G
IE.EOT= 177702 G        IE.RNM= 177715 G        IO.DCI= 014400 G        IO.RLV= 001100 G        IS.BV = 000005 G
IE.EOV= 177765 G        IE.RSU= 177757 G        IO.DCT= 002470 G        IO.RNA= 005400 G        IS.CLR= 000000 G
IE.EXP= 177665 G        IE.SDP= 177635 G        IO.DEL= 012400 G        IO.RNC= 001040 G        IS.CR = 006401 G
IE.FEX= 177717 G        IE.SNC= 177735 G        IO.DET= 002000 G        IO.RNE= 001020 G        IS.ESC= 015401 G
IE.FHE= 177705 G        IE.SPC= 177772 G        IO.DIS= 016000 G        IO.RNS= 001020 G        IS.PND= 000000 G
IE.FOP= 177713 G        IE.SOC= 177734 G        IO.DTI= 016000 G        IO.RPB= 001040 G        IS.SET= 000002 G
IE.HFU= 177744 G        IE.SRE= 177762 G        IO.ENA= 006000 G        IO.RTC= 003400 G        IS.SPD= 000002 G
IE.HWR= 177772 G        IE.STK= 177706 G        IO.EOF= 003000 G        IO.RTI= 016400 G        IS.SUC= 000001 G
IE.IDU= 177644 G        IE.TMM= 177671 G        IO.ESA= 002500 G        IO.RTK= 000060 G        QI.VER= 000304 G
IE.IEF= 177637 G        IE.TMO= 177666 G        IO.EXT= 011400 G        IO.RVB= 010400 G        $$MSG = 000000
IE.IFC= 177776 G        IE.ULN= 177773 G        IO.FDX= 003020 G        IO.RWD= 002400 G        ...GBL= 000001

. ABS.  000000      000
        000000      001
ERRORS DETECTED:  0

FREE CORE:  5669. WORDS
,LP:=[156,133]QIOMAC,TI:
```

APPENDIX D

DIRECTIVE PARAMETER BLOCKS

D.1 QUEUE I/O

A 6- to 12-word DPB of the following format is used for the Queue I/O.

```
WD. 00 -- DIC (01.) & DPB SIZE (6-12),
WD. 01 -- I/O FUNCTION CODE,
WD. 02 -- LUN,
WD. 03 -- [EFN] & [PRIORITY],
WD. 04 -- [ADDRESS OF I/O STATUS BLOCK],
WD. 05 -- [I/O DONE AST SERVICE ENTRY POINT],
WD. 06 -- PARAMETER #1,
WD. 07 -- PARAMETER #2,
WD. 10 -- PARAMETER #3,
WD. 11 -- PARAMETER #4,
WD. 12 -- PARAMETER #5,
WD. 13 -- PARAMETER #6.
```

D.2 QUEUE I/O AND WAIT

A 6- to 12-word DPB of the following format is used.

```
WD. 00 -- DIC (03.) & DPB SIZE (6-12),
WD. 01 -- I/O FUNCTION CODE,
WD. 02 -- LUN,
WD. 03 -- [EFN] & [PRIORITY],
WD. 04 -- [ADDRESS OF I/O STATUS BLOCK],
WD. 05 -- [I/O DONE AST SERVICE ENTRY POINT],
WD. 06 -- PARAMETER #1,
WD. 07 -- PARAMETER #2,
WD. 10 -- PARAMETER #3,
WD. 11 -- PARAMETER #4,
WD. 12 -- PARAMETER #5,
WD. 13 -- PARAMETER #6.
```

D.3 GET LUN INFORMATION

```
WD. 00 -- DIC (05.) & DPB SIZE (3.),
WD. 01 -- LUN,
WD. 02 -- ADDRESS OF SIX-WORD BUFFER.
```

D.4 ASSIGN LUN

```
WD. 00 -- DIC (07.) & DPB SIZE (4.),
WD. 01 -- LOGICAL UNIT NUMBER,
WD. 02 -- PHYSICAL DEVICE NAME,
WD. 03 -- PHYSICAL DEVICE UNIT NUMBER.
```

D.5 ALTER PRIORITY

A 4-word DPB in the following format is used.

```
WD. 00 -- DIC (009.) & DPB SIZE (4.).
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).
WD. 03 -- TASK PRIORITY.
```

D.6 REQUEST

A 7-word DPB in the following format is used.

```
WD. 00 -- DIC (11.) & DPB SIZE (7.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF),
WD. 03 -- [PARTITION NAME (FIRST HALF)],
WD. 04 -- [PARTITION NAME (SECOND HALF)],
WD. 05 -- [PRIORITY],
WD. 06 -- [UIC].
```

D.7 EXECUTE

A 7-word DPB in the following format is used.

```
WD. 00 -- DIC (13.) & DPB SIZE (7.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF),
WD. 03 -- [PARTITION NAME (FIRST HALF)],
WD. 04 -- [PARTITION NAME (SECOND HALF)],
WD. 05 -- [PRIORITY],
WD. 06 -- [UIC].
```

D.8 SCHEDULE

A 13-word DPB in the following format is used.

```
WD. 00 -- DIC (15.) & DPB SIZE (13.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF),
WD. 03 -- [PARTITION NAME (FIRST HALF)],
WD. 04 -- [PARTITION NAME (SECOND HALF)],
WD. 05 -- [PRIORITY],
WD. 06 -- [UIC],
WD. 07 -- SCHEDULE HOURS (0-23),
WD. 10 -- SCHEDULE MINUTES (0-59),
WD. 11 -- SCHEDULE SECONDS (0-59),
WD. 12 -- SCHEDULE TICKS (0-59),
WD. 13 -- [RE-SCHEDULE INTERVAL MAGNITUDE],
WD. 14 -- [RE-SCHEDULE INTERVAL UNITS (1-4)].
```

D.9 RUN

An 11-word DPB in the following format is used.

```
WD. 00 -- DIC (17.) & DPB SIZE (11.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF),
WD. 03 -- [PARTITION NAME (FIRST HALF)],
WD. 04 -- [PARTITION NAME (SECOND HALF)],
WD. 05 -- [PRIORITY],
WD. 06 -- [UIC],
WD. 07 -- SCHEDULE DELTA MAGNITUDE,
WD. 10 -- SCHEDULE DELTA UNITS (1-4),
WD. 11 -- [RE-SCHEDULE INTERVAL MAGNITUDE],
WD. 12 -- [RE-SCHEDULE INTERVAL UNITS (1-4)].
```

D.10 SYNC

A 12-word DPB in the following format is used.

```
WD. 00 -- DIC (19.) & DPB SIZE (12.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF),
WD. 03 -- [PARTITION NAME (FIRST HALF)],
WD. 04 -- [PARTITION NAME (SECOND HALF)],
WD. 05 -- [PRIORITY],
WD. 06 -- [UIC],
WD. 07 -- SCHEDULE DELTA MAGNITUDE,
WD. 10 -- SCHEDULE DELTA UNITS (1-4),
WD. 11 -- SYNCHRONIZATION UNITS (1-4),
WD. 12 -- [RE-SCHEDULE INTERVAL MAGNITUDE],
WD. 13 -- [RE-SCHEDULE INTERVAL UNITS (1-4)].
```

D.11 MARK TIME

A 5-word DPB in the following format is used.

```
WD. 00 -- DIC (23.) & DPB SIZE (5.),
WD. 01 -- [EVENT FLAG NUMBER (EFN)],
WD. 02 -- TIME INTERVAL MAGNITUDE,
WD. 03 -- TIME INTERVAL UNITS,
WD. 04 -- [SYSTEM TRAP ENTRY POINT].
```

D.12 CANCEL SCHEDULED REQUESTS

A 3-word DPB in the following format is used to cancel all scheduled requests for an indicated task.

```
WD. 00 -- DIC (25.) & DPB SIZE (3.),
WD. 01 -- SCHEDULED TASK NAME (FIRST HALF),
WD. 02 -- SCHEDULED TASK NAME (SECOND HALF).
```

A 5-word DPB in the following format is used to cancel only those requests issued for an indicated task by an indicated task.

```
WD. 00 -- DIC (25.) & DPB SIZE (5.),
WD. 01 -- SCHEDULED TASK NAME (FIRST HALF),
WD. 02 -- SCHEDULED TASK NAME (SECOND HALF),
WD. 03 -- [SCHEDULER TASK NAME (FIRST HALF)],
WD. 04 -- [SCHEDULER TASK NAME (SECOND HALF)].
```

D.13 CANCEL MARK TIME REQUESTS

A 1-word DPB of the following format is used to cancel all Mark Time requests made by the issuing task.

```
WD. 00 -- DIC (27.) & DPB SIZE (1.).
```

A 3-word directive in the following format is used to cancel only Mark Time request made by the issuing task and that set an indicated event flag or cause an AST at an indicated location.

```
WD. 00 -- DIC (27.) & DPB SIZE (3.),
WD. 01 -- [EVENT FLAG NUMBER (EFN)],
WD. 02 -- [AST SERVICE ROUTINE ENTRY].
```

D.14 CLEAR EVENT FLAG

A 2-word DPB of the following format is used.

```
WD. 00 -- DIC (31.) & DPB SIZE (2.),
WD. 01 -- EVENT FLAG NUMBER (EFN).
```

D.15 SET EVENT FLAG

A 2-word DPB in the following format is used.

```
WD. 00 -- DIC (33.) & DPB SIZE (2.),
WD. 01 -- EVENT FLAG NUMBER (EFN).
```

D.16 DECLARE SIGNIFICANT EVENT

A 2-word DPB of the following format is used to read an event flag, set an event flag, declare a significant event, and to report the pre-event flag polarity.

```
WD. 00 -- DIC (35.) & DPB SIZE (2.),
WD. 01 -- EVENT FLAG NUMBER (EFN).
```

A 1-word DPB in the following format is used to declare a significant event.

```
WD. 00 -- DIC (35.) & DPB SIZE (1.).
```

D.17 READ EVENT FLAG

A 2-word DPB of the following format is used.

WD. 00 -- DIC (37.) & DPB SIZE (2.),
WD. 01 -- EVENT FLAG NUMBER (EFN).

D.18 READ ALL FLAGS

A 2-word DPB of the following format is used.

WD. 00 -- DIC (39.) & DPB SIZE (2.),
WD. 01 -- ADDRESS (VIRTUAL) OF 64-BIT BUFFER.

D.19 WAIT FOR SINGLE EVENT FLAG

A 2-word DPB of the following format is used.

WD. 00 -- DIC (41.) & DPB SIZE (2.),
WD. 01 -- EVENT FLAG NUMBER (EFN).

D.20 WAIT FOR LOGICAL OR OF FLAGS

A 3-word DPB of the following format is used to wait for event flags of sets 0, 1, 2, or 3.

WD. 00 -- DIC (43.) & DPB SIZE (3.),
WD. 01 -- SET INDICATOR (0, 1, 2, 3),
WD. 02 -- SIXTEEN FLAG MASK WORD.

A 5-word DPB of the following format is used to wait for event flags of set 4.

WD. 00 -- DIC (43.) & DPB SIZE (5.),
WD. 01 -- MASK WORD FOR FLAGS 1-16,
WD. 02 -- MASK WORD FOR FLAGS 17-32,
WD. 03 -- MASK WORD FOR FLAGS 33-48,
WD. 04 -- MASK WORD FOR FLAGS 49-64.

D.21 WAIT FOR NEXT SIGNIFICANT EVENT

A 1-word DPB of the following format is used.

WD. 00 -- DIC (49.) & DPB SIZE (1.).

D.22 SUSPEND

A 1-word DPB of the following format is used.

WD. 00 -- DIC (45.) & DPB SIZE (1.).

D.23 RESUME

A 3-word DPB of the following format is used.

WD. 00 -- DIC (47.) & DPB SIZE (3.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).

D.24 EXIT

A 1-word DPB of the following format is used.

WD. 00 -- DIC (51.) & DPB SIZE (1.).

D.25 EXITIF

A 2-word DPB of the following format is used.

WD. 00 -- DIC (53.) & DPB SIZE (2.),
WD. 01 -- EVENT FLAG NUMBER.

D.26 GET TIME PARAMETERS

A 2-word DPB of the following format is used.

WD. 00 -- DIC (61.) & DPB SIZE (2.),
WD. 01 -- ADDRESS OF 8-WORD BUFFER.

D.27 GET TASK PARAMETERS

A 2-word DPB of the following format is used.

WD. 00 -- DIC (63.) & DPB SIZE (2.),
WD. 01 -- ADDRESS OF SIXTEEN WORD BUFFER.

D.28 GET PARTITION PARAMETERS

A 4-word DPB of the following format is used.

WD. 00 -- DIC (65.) & DPB SIZE (4.),
WD. 01 -- [PARTITION NAME (FIRST HALF)],
WD. 02 -- [PARTITION NAME (SECOND HALF)],
WD. 03 -- ADDRESS OF THREE WORD BUFFER.

D.29 GET COMMON BLOCK PARAMETERS

A 4-word DPB of the following format is used.

WD. 00 -- DIC (67.) & DPB SIZE (4.),
WD. 01 -- COMMON BLOCK NAME (FIRST HALF),
WD. 02 -- COMMON BLOCK NAME (SECOND HALF),
WD. 03 -- ADDRESS OF EIGHT WORD BUFFER.

D.30 SEND DATA

A 5- to 8-word DPB of the following format is used.

WD. 00 -- DIC (71.) & DPB SIZE (5.-8.),
WD. 01 -- RECEIVER TASK NAME (FIRST HALF),
WD. 02 -- RECEIVER TASK NAME (SECOND HALF),
WD. 03 -- ADDRESS OF DATA BLOCK,
WD. 04 -- [EVENT FLAG NUMBER],
WD. 05 -- [BUFFER SIZE - 1-255.],
WD. 06 -- [PRIORITY OF SEND],
WD. 07 -- [RECEIVER TI].

D.31 SEND AND REQUEST OR RESUME

A 9- to 12-word DPB of the following format is used.

WD. 00 -- DIC (73.) & DPB SIZE (9.-12.),
WD. 01 -- RECEIVER TASK NAME (FIRST HALF),
WD. 02 -- RECEIVER TASK NAME (SECOND HALF),
WD. 03 -- [PARTITION NAME (FIRST HALF)],
WD. 04 -- [PARTITION NAME (SECOND HALF)],
WD. 05 -- [PRIORITY],
WD. 06 -- [UIC],
WD. 07 -- ADDRESS OF DATA BLOCK,
WD. 10 -- [EVENT FLAG NUMBER],
WD. 11 -- [BUFFER SIZE - 1-255.],
WD. 12 -- [PRIORITY OF SEND],
WD. 13 -- [RECEIVER TI].

D.32 RECEIVE DATA

A 4- to 6-word DPB of the following format is used.

WD. 00 -- DIC (75.) & DPB SIZE (4-6),
WD. 01 -- [SENDER TASK NAME (FIRST HALF)],
WD. 02 -- [SENDER TASK NAME (SECOND HALF)],
WD. 03 -- ADDRESS OF BUFFER.
WD. 04 -- [BUFFER SIZE - 1-255.],
WD. 05 -- [LOC. TO STORE TI].

D.33 RECEIVE DATA OR EXIT

A 4- to 6-word DPB of the following format is used.

WD. 00 -- DIC (77.) & DPB SIZE (4-6),
WD. 01 -- [SENDER TASK NAME (FIRST HALF)],
WD. 02 -- [SENDER TASK NAME (SECOND HALF)],
WD. 03 -- ADDRESS OF BUFFER.
WD. 04 -- [BUFFER SIZE - 1-255.],
WD. 05 -- [LOC. TO STORE TI].

D.34 RECEIVE DATA OR SUSPEND

A 4- to 6-word DPB of the following format is used.

```
WD. 00 -- DIC (79.) & DPB SIZE (4-6),
WD. 01 -- [SENDER TASK NAME (FIRST HALF)],
WD. 02 -- [SENDER TASK NAME (SECOND HALF)],
WD. 03 -- ADDRESS OF  BUFFER.
WD. 04 -- [BUFFER SIZE - 1-255.],
WD. 05 -- [LOC. TO STORE II].
```

D.35 ABORT

A 3-word DPB of the following format is used.

```
WD. 00 -- DIC (83.) & DPB SIZE (3.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).
```

D.36 FIX-IN-MEMORY

A 3-word DPB of the following format is used.

```
WD. 00 -- DIC (85.) & DPB SIZE (3.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).
```

D.37 UNFIX

A 3-word DPB of the following format is used.

```
WD. 00 -- DIC (87.) & DPB SIZE (3.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).
```

D.38 DISABLE

A 3-word DPB of the following format is used.

```
WD. 00 -- DIC (91.) & DPB SIZE (3.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).
```

D.39 ENABLE

A 3-word DPB of the following format is used.

```
WD. 00 -- DIC (93.) & DPB SIZE (3.),
WD. 01 -- TASK NAME (FIRST HALF),
WD. 02 -- TASK NAME (SECOND HALF).
```

D.40 DISABLE CHECKPOINTING

A 1-word DPB of the following format is used.

WD. 00 -- DIC (95.) & DPB SIZE (1.).

D.41 ENABLE CHECKPOINTING

A 1-word DPB of the following format is used.

WD. 00 -- DIC (97.) & DPB SIZE (1.).

D.42 INHIBIT AST RECOGNITION

A 1-word DPB of the following format is used.

WD. 00 -- DIC (99.) & DPB SIZE (1.).

D.43 ENABLE AST RECOGNITION

A 1-word DPB of the following format is used.

WD. 00 -- DIC (101.) & DPB SIZE (1.).

D.44 SPECIFY SST VECTOR TABLE FOR DEBUGGING AID

A 3-word DPB of the following format is used.

WD. 00 -- DIC (103.) & DPB SIZE (3.),
WD. 01 -- ADDRESS OF SST VECTOR TABLE,
WD. 02 -- NUMBER OF TABLE ENTRIES (8).

D.45 SPECIFY SST VECTOR TABLE FOR TASK

A 3-word DPB of the following format is used.

WD. 00 -- DIC (105.) & DPB SIZE (3.),
WD. 01 -- ADDRESS OF SST VECTOR TABLE,
WD. 02 -- NUMBER OF TABLE ENTRIES (8).

D.46 SPECIFY RECEIVE AST

A 2-word DPB of the following format is used.

WD. 00 -- DIC (107.) & DPB SIZE (2.),
WD. 01 -- [AST SERVICE ENTRY POINT].

D.47 SPECIFY POWER FAIL AST

A 2-word DPB of the following format is used.

```
WD. 00 -- DIC (109.) & DPB SIZE (2.),
WD. 01 -- [AST SERVICE ENTRY POINT].
```

D.48 SPECIFY FLOATING POINT EXCEPTION AST

A 2-word DPB of the following format is used.

```
WD. 00 -- DIC (111.) & DPB SIZE (2.),
WD. 01 -- [AST SERVICE ENTRY POINT].
```

D.49 AST SERVICE EXIT

A 1-word DPB of the following format is used.

```
WD. 00 -- DIC (115.) & DPB SIZE (1.).
```

D.50 GET SENSE SWITCHES

A 1-word DPB of the following format is used.

```
WD. 00 -- DIC (125.) & DPB SIZE (1.).
```

D.51 GET MCR COMMAND LINE

A 41-word DPB of the following format is used.

```
WD. 00 -- DIC (127.) & DPB SIZE (41.),
WD. 01 -- FIRST WORD OF 80-BYTE BUFFER.
```

INDEX

RSX-11D Executive
Reference Manual
DEC-11-OXERA-B-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you require a written reply, please check here. ☐

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

digital

digital equipment corporation

Printed in U.S.A.