RSX-11M

Executive Reference Manual

Order No. DEC-11-OMERA-A-D

RSX-11M Version 1

Order additional copies as directed on the Software Information page at the back of this document.

digital equipment corporation · maynard, massachusetts

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of DIGITAL's copyright notice) only for use in such system, except as may otherwise be provided in writing by DIGITAL.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by DIGITAL.

The HOW TO OBTAIN SOFTWARE INFORMATION page, located at the back of this document explains the various services available to DIGITAL software users.

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.

The following are trademarks of Digital Equipment Corporation.

CDP	DIGITAL	INDAC	PS/8
COMPUTER LAB	DNC	KA10	QUICKPOINT
COMSYST	EDGRIN	LAB-8	RAD-8
COMTEX	EDUSYSTEM	LAB-8/e	RSTS
DDT	FLIP CHIP	LAB-K	RSX
DEC	FOCAL	OMNIBUS	RTM
DECCOMM	GLC-8	OS/8	RT-11
DECTAPE	IDAC	PDP	SABR
DIBOL	IDACS	PHA	TYPESET-10
			UNIBUS

PREFACE

MANUAL OBJECTIVES AND READER CLASS ASSUMPTION

The intent of this manual is to provide experienced MACRO-11 or FORTRAN IV programmers the technical details necessary to use the services provided by the RSX-11M Executive. The manual is not self-contained. The MACRO-11 Reference Manual (for the MACRO-11 programmer), the FORTRAN IV Reference Manual (for the FORTRAN IV programmer), and the Task Builder Reference Manual are prerequisite sources of information. Further the Introduction to RSX-11M, the RSX-11M System Generation Manual, the RSX-11M I/O Operations Reference Manual, the RSX-11M Operators Guide, and the How To Write an I/O Driver Manual are closely allied to the purposes of this manual. All the details in the manual may not be relevant to the strictly FORTRAN IV programmer, but, without question, his exposure to them cannot but add to his ability to obtain the most efficient results from his programming efforts.

The reader is assumed to understand PDP-11 Processors and Processor-related terminology.

The manual is tutorial in bias, but is not meant to train programmers. Experience on DEC or other manufacturers realtime systems is assumed.

NOTE
The How To Write an I/O Driver Manual will be published as a follow-on document.

CONTENTS

FIGURES

TABLES

CHAPTER 1

FUNDAMENTAL CONCEPTS

1.1 INTRODUCTION

Executive services exist to permit users to access structures and facilities inherently available in the hardware, but because of multiprogramming and realtime constraints, must be disbursed by the Executive on a controlled or shared basis.

A typical example is I/O. If many independent tasks seek access to I/O devices, then to prevent chaos, and to provide access based on the importance of the request, an intermediary is required between the independent requests of the tasks and the actual hardware device being accessed: the RSX-11M Executive is this intermediary.

The objective of the Executive is to provide to the user as many of the facilities that are inherent in the hardware as possible, and, where desirable, augment these facilities; the Executive aims to provide these services without impacting the throughput capabilities of the raw hardware. The system provides these facilities through instruction-like constructs called directives.

These Executive directives are analogous to similar hardware facilities (like I/O request directives) or desirable augmentations to the hardware (like SEND and RECEIVE directives for communication between tasks). In any event, the directives are used just as the instruction set is used. The combination of the instruction set and the directives can be viewed as an extended machine.

The applications programmer uses the directives to control the execution and interaction of tasks. These directives are usually implemented via macros in the System Macro Library (SML). The FORTRAN programmer invokes system directives through subroutine calls which are listed with each Executive directive.

Directives are implemented via the EMT 377 instruction. Programs using EMT 0 through EMT 367 can be run via the non-RSX EMT system trap. Any EMT, other than EMT's 370-377, which are reserved for system use, will trap to a task-contained service routine, which may simulate another environment to whatever degree is desired; for example, the emulation of another operating system interface. It should be noted that if the EMT numbers (370-376) are issued by a user task, the resulting trap will be directed to the user task. User tasks should consider these EMT's as internal program errors, since system conventions reserve these EMT's.

Note that by using macros instead of coding the directives, the programmer need only re-assemble to re-adjust programs if changes are made in the directive specifications.

1.2 DIRECTIVE IMPLEMENTATION

A brief discussion of how directives are implemented will help the programmer better understand and use the macros which are associated with the directives.

The EMT 377 is issued with either the address of a Directive Parameter Block (DPB), or a DPB itself, on the top of the issuing task's stack.

The first word of a DPB contains a Directive Identification Code (DIC), and a DPB size. The DIC indicates which directive is to be performed; the size 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.

Figures 1-1 and 1-2 illustrate the alternatives for issuing directives and also show the relationship between the stack pointer and the DPB.

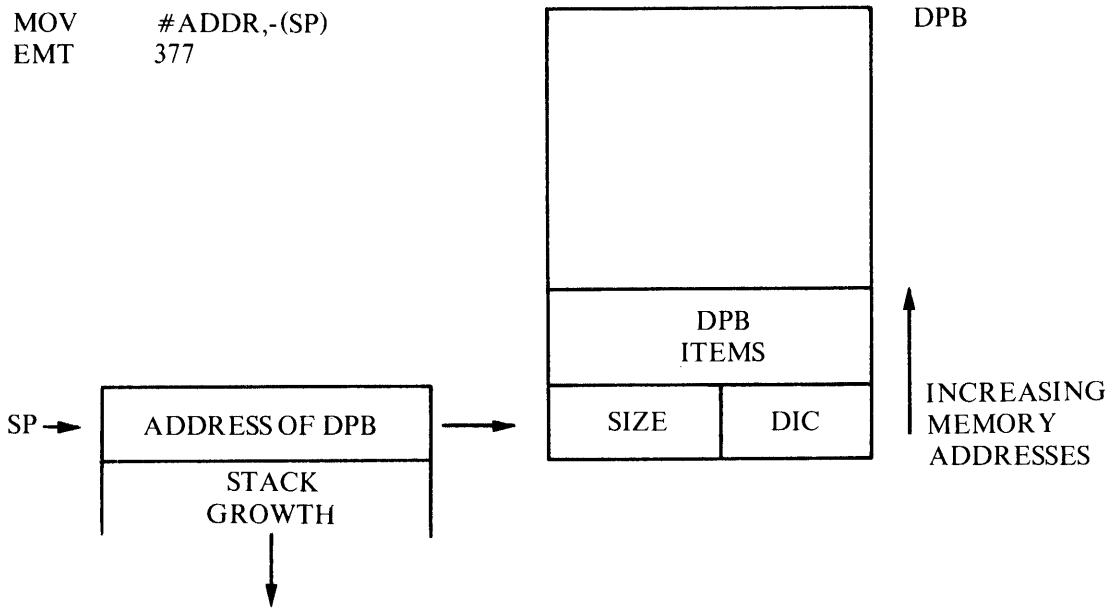

Figure 1-1
Directive Parameter Block (DPB) Pointer On The Stack

```
MOV         XX,-(SP)
    PUSH REQUIRED
    DPB ITEMS ON THE
    STACK IN
    REVERSE ORDER

MOV         (PC)+,-(SP)
BYTE        DIC,SIZE
EMT         377
```

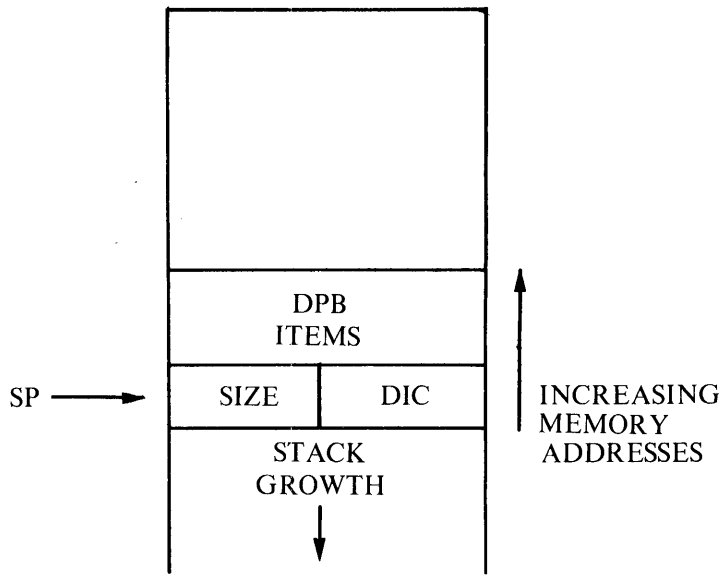

Figure 1-2
Directive Parameter Block (DPB) On The Stack

When the stack contains a DPB address, the address is removed after the directive is processed. When the stack contains a DPB, the entire DPB is removed after the directive is processed. In both cases the removal occurs prior to the Executive returning control to the task. The Executive distinguishes an actual DPB word from a DPB pointer by determining if the first word on the stack is even or odd. An even word specifies a DPB pointer, an odd word indicates the DPB is on the stack.

With the exception of the EXIT and EXITIF and RECEIVE DATA or EXIT directive, control is returned to the instruction following the EMT, with the carry condition code cleared or set indicating the directive has been accepted (cleared) or rejected (set). Further, the Directive Status Word (DSW) which is always referred to symbolically as $DSW*, is set to indicate a more specific cause for acceptance or rejection of the specific directive involved. The DSW is usually +1 for exceptance and has a range of negative values when the directive has been rejected. The detailed return values are listed with each directive.

---

*The Task Builder resolves the address of $DSW. Users addressing the DSW with a physical address are not guaranteed upward compatibility with RSX-11D and may experience incompatibilities with future RSX-11M releases.

1.2.1 Directive Conventions (MACRO-11 and FORTRAN IV)

The following conventions and assumptions are standard for all directives.

1. For MACRO-11 programs decimal radix is used in all cases except hardware addresses and device unit numbers. Octal is assumed in MACRO-11 code examples if the number is not followed by a decimal point.

   For FORTRAN IV, type integer*2 is used in all cases unless specifically noted otherwise.

2. For MACRO-11 programs task and partition names may be up to six characters long and are always represented as two words in Radix-50 form.

   For FORTRAN IV, task and partition names are specified by a variable of type REAL (single precision) which contains the task or partition name in radix-50 representation. Radix-50 representation may be established at compile time by use of the DATA statement, or at runtime by means of the IRAD50 subprogram or RAD50 Function.

3. Device names are two characters long and are represented by one word in ASCII code.

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

5. Optional parameters are enclosed in square brackets.

6. Trailing optional arguments that are null may be omitted.

7. Certain parameters are stated as being ignored, yet required. This convention is needed to maintain RSX-11M, RSX-11D compatibility.

8. Consecutive commas denote omitted arguments.

9. Legal range of Logical Unit Numbers (LUN's) is 1-255(10).

10. Event Flags are numbered 1-64(10).

Directives are listed according to category, in sections 2.2.1 through 2.2.6.

1.2.2 Specialized FORTRAN Subroutines

This section contains Fortran subroutine calls which may be used for simplifying interfacing with the system's Executive directives.

1.2.2.1 GETADR

The primary intent of this call is to facilitate the construction of the parameter array for the QIO Directive subroutine.

Calling Sequence:

CALL GETADR(ipm,[arg1,],[arg2],...,[argn])

lpm	is an integer array of dimension n
arg1 ,... argn	are arguments whose addresses are to be inserted in ipm. Arguments are inserted in the order specified. If a null argument is specified then the corresponding entry in ipm is left unchanged.

1.3 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, which is always negative, is returned in the DSW which is at symbolic location $DSW. 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 of -20 through -99.

All error codes in RSX-11M are defined symbolically. The mnemonics used reflect the cause of the error. In the text of the manual, the symbolics are used exclusively. The macro, DRERR$, which is expanded in Appendix B, provides a correspondence between the symbolic error name and its numeric value.

Table 1-1 summarizes general interpretations of error codes. Others are described in individual directive descriptions.

Table 1-1
General Error Codes

Code	Reason For Rejection
IE.UPN	Insufficient Dynamic Memory
	User tasks cannot request dynamic memory; however, several Executive requests require it for their execution. When it cannot be obtained, this error return results. The user can try again later by suspending himself. (Note: WAITFOR SIGNIFICANT EVENT is recommended, since most other suspend-type directives themselves require dynamic memory.)
IE.INS	Task Name Not In The STD or Undefined Partition Name
	Indicates the task has not been installed in the system, that a partition has not been defined at SYSGEN, or has not been specified in a Set command.
IE.ULN	Unassigned LUN
	The LUN (Logical Unit Number) in the request has not been assigned to a physical device. Recovery is possible by issuing a valid ASSIGN LUN directive, then re-issuing the rejected request.
IE.ACT	Task Is Active/Not Active
	An attempt is made to cause a task state-transition which is a task state inconsistent with the existing task state. For example, an attempt is made to ABORT a task which is not active. Or a task has attempted to request a task which is already active.
IE.ITS	Redundant Request
	Occurs when the request is such that it duplicates an existing task state. For example, the task attempts to enable AST's but AST recognition is already enabled.
IE.CKP	Task is: Checkpointable/Not Checkpointable
	This error occurs if the task is not checkpointable, and the task attempts to enable or disable checkpointability.

Table 1-1 (Cont.)
General Error Codes

IE.ITI Invalid Time Parameter

A time parameter consists of two words:

1. A magnitude word, and
2. A units word.

The legal value of the magnitude is related to the value of the units word, which is encoded as:

1 = Ticks. A tick causes a clock interrupt and the rate at which interrupts occur depends on the type of clock installed on the system.

For a line frequency clock, the tick rate is either 50 or 60 per second, corresponding to the power-line frequency.

For a programmable clock a maximum of 1000 ticks per second is available (frequency is selectable at SYSGEN).

2 = Seconds
3 = Minutes
4 = Hours

The magnitude is the number of units to be clocked, but the magnitude value cannot exceed 24 hours in the specified units.

Units = 1
Any positive value is valid (maximum of 15 bits)

Units = 2
Any positive value is valid (maximum of 15 bits)

Units = 3
1440(10) is maximum magnitude

Units = 4
24(10) is maximum magnitude

Table 1-1 (Cont.)
General Error Codes

IE.ILU Invalid Logical Unit Number

A Logical Unit Number has been specified which is invalid for the issuing task. For example, if the task has established only five LUN's and has attempted to use a LUN greater than five, then this error will occur.

IE.IEF Invalid Event Flag Number

An event flag number has been illegally specified. In the case where the EFN was required, it was less than 1 or greater than 64; in the case when it was not required, it was less than 0 or greater than 64. The only valid non-specification is 0.

IE.ADP Invalid Address

A buffer has been specified in the directive and the buffer lies outside the user's address space or has an improper alignment (not on a word boundary). Also returned if part of the DPB is outside of the task's address space.

IE.SDP Invalid DIC number or DPB size

Either the DIC number, or the DPB size or both were incorrect. DICs range from 1-127 and are always odd.

1.4 USING THE DIRECTIVE MACROS

This discussion applies to MACRO-11 programmers. FORTRAN programmers execute directives via subroutine calls and therefore need not concern themselves with the details of this section.

Directives are issued by including appropriate macro calls in the program. The macros which generate RSX-11M directives, are contained in the System Macro Library (SY:[1,1]RSXMAC.SML). The user makes the macros available to his program by supplying the .MCALL assembler directive, and using as arguments to .MCALL all the system macros used in his program. Figure 1-3 is an example of calling and subsequently using macros in the System Macro Library (SML).

Example:

```
;
; CALLING DIRECTIVES OUT OF THE SYSTEM MACRO LIBRARY
; AND INVOKING THEM.
;

        .MCALL  MRKT$S,WTSE$S
        .
        .
        Additional .MCALL's or code
        .
        .
        MRKT$S  #1,#1,#2,,ERR       ;MARK TIME FOR 1 SECOND
        WTSE$S  #1                  ;WAIT FOR MARK TIME TO COMPLETE
        .
```

Figure 1-3
Calling Directives From Macro Library

Directive 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 omitted ($ form), the macro will produce only the directive's DPB. The DPB is inserted at the point of macro invocation, but does not contain executable code. This form allows for dynamic modification of the DPB, but is not re-entrant and is usually used in conjunction with the DIR$ macro discussed below. It should be noted that DPB's should not appear embedded in instruction sequences, since the Executive always returns to the instruction immediately following the EMT 377, with three exceptions: EXIT, EXITIF, and RECEIVE DATA or EXIT (when the IF condition holds). If the $ form of macro is used, it is assumed that the parameters required for DPB construction are valid expressions to be used in assembler data storage directives (e.g., .BYTE, .WORD, .RAD50).

If the optional letter is "S" ($S form), the macro produces code to push a DPB on the stack, followed by an EMT 377. This form can be used by a program with re-entrancy requirements. If the $S form is used, the parameters must be valid source operands to be placed directly in MOV instructions.

If the optional letter is "C" ($C form), 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 on 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 p-section is assumed. The $C form is used when the program has no re-entrancy requirements and plans to use the DPB on a one-shot basis. This form has low overhead since the DPB is generated at assembly time, thereby eliminating the run time requirement to push the parameters on the stack. The DPB, however cannot be accessed from another part of the program since its address is not

known. If the $C form of macro is used, it is assumed that the parameters required for DPB construction are valid expressions to be used in assembler data storage directives (e.g., .BYTE, .WORD, .RAD50).

Note that only the $S form (also referred to as s-form) produces the DPB dynamically. The other two forms produce the DPB at assembly time.

If the user has a predefined DPB and wishes to avoid the creation of another one, the DIR$ macro can be used. This macro generates the code to push the DPB address on the stack using MOV SSS,-(SP), where the macro parameter (shown here as SSS), represents a valid assembler source operand, followed by an EMT 377.

The $C, $S and DIR$ forms of macro calls will accept an optional final argument. If included, it must be a valid assembler destination operand to call a user error routine. It generates the following code (assume DDD is the macro parameter in the following example):

```
BCC     .+n             ;BRANCH ON DIRECTIVE ACCEPTANCE
JSR     PC,DDD          ;ELSE, CALL ERROR SERVICE ROUTINE
```

This option is not permitted when the user specifiesthe generation of the DPB only.

1.4.1 Symbolic Offsets

Most system directive macros 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 with no need of calculating the offsets. Symbolic offsets also do away with the necessity of rewriting programs to accommodate changes in DPB specifications.

All $ and $C forms of macros that generate DPBs longer than one word generate local offsets.

If any of the $ or $C forms of the macros are invoked, and the symbol $$$GLB has been defined elsewhere in the program (i.e., $$$GLB=0), the DPB is not expanded. Instead the macro produces the symbolic offsets as global symbols. The symbol $$$GLB has no effect on the expansion of $S macros.

1.4.2 Examples of Macro Calls

Example 1 - Generate Only A DPB in the Current Program section

```
MRKT$   1,5,2,MTRAP
```

Generated Code:

```
.BYTE   23.,5.          ;MRKT$ MACRO DIC & DPB SIZE
.WORD   1               ;EVENT FLAG NUMBER
```

```
.WORD   5               ;TIME INTERVAL MAGNITUDE
.WORD   2               ;TIME INTERVAL UNIT (SECONDS)
.WORD   MTRAP           ;AST ENTRY POINT ADDRESS
```

Example 2 - Generate a DPB in a Separate Program section

```
MRKT$C  1,5,2,MTRAP,PROG1,ERR
```

Generated Code:

```
.PSECT      $DPB$$

$$$=.

.BYTE   23.,5.          ;MRKT$ MACRO DIC & DPB SIZE
.WORD   1               ;EVENT FLAG NUMBER
.WORD   5               ;TIME INTERVAL MAGNITUDE
.WORD   2               ;TIME INTERVAL UNIT (SECONDS)
.WORD   MTRAP           ;AST ENTRY POINT ADDRESS

.PSECT  PROG1           [return to the original Program
                        section]

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

Example 3 - Generate a DPB on the Stack

```
MRKT$S  #1,#5,#2,R2,ERR
```

Generated Code:

```
MOV     R2,-(SP)        ;PUSH AST ENTRY POINT
MOV     #2,-(SP)        ;TIME INTERVAL UNIT (SECONDS)
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 EXECUTIVE
BCC     .+6             ;BRANCH ON DIRECTIVE ACCEPTANCE
JSR     PC,ERR          ;ELSE,CALL ERROR SERVICE ROUTINE
```

Example 4 - DPB Already Defined

```
DIR$    R1,(R3)          ;R1 CONTAINS DPB ADDRESS
```

Generated Code:

```
MOV     R1,-(SP)         ;PUSH DPB ADDRESS ON STACK
EMT     377              ;TRAP TO THE EXECUTIVE
BCC     .+4              ;BRANCH ON DIRECTIVE ACCEPTANCE
JSR     PC,(R3)          ;ELSE, CALL ERROR SERVICE ROUTINE
```

1.5 TASK STATES

Throughout this manual references will be made to directives and events that cause the state of a task to change. The following discussion is intended to enhance the reader's understanding of the various internal transitions his task is subject to in the multiprogramming environment maintained by RSX-11M.

An RSX-11M task has four basic states:

Dormant;

Active;

Ready-to-Run, and

Blocked.

The Task Builder creates a task image on disk. To the Executive, however, a task exists only after it has been successfully installed and has an entry in the System Task Directory. (Task installation is a process whereby a task is made known to the system.) Task states are defined as follows:

Dormant:

The task has an entry in the STD, but its activation has not been requested (a RQST$ or RUN$ macro has not been issued for it). Immediately following the Monitor Console Routine's (MCR) processing of an INStall command, a task is known to the system, but is dormant.

Active:

A task is Active from the time it is requested, until it exits. A task passes from the dormant state to the active state as a result of the run-time issuance of the RQST$, or RUN$ macros, or by an operator issuing the MCR RUN command. "Active"implies that the task is eligible for scheduling, while dormant (or equivalently inactive) implies the task is not eligible for scheduling. An Active task may be in one of two states: ready-to-run or blocked.

Ready-To-Run:

The task is capable of competing with other tasks for the CPU on the basis of priority. There is no "Running" state in RSX-11M. The highest priority ready-to-run task will obtain the CPU, thus becoming the current task.

Blocked:

The task, due to unavailability of a needed resource, or requirements of synchronization is unable to compete for access to the CPU.

CHAPTER 2

DIRECTIVE DESCRIPTIONS

Each directive description consists of a narrative explanation of its function and use, the name of the associated macro and its parameters, and possible return values of the Directive Status Word (DSW), which, as previously noted, must be referenced symbolically as $DSW.

In general the $ form of the macro name is given, although all three options are available unless otherwise specified. Certain macros have an s-form only, and are so specified in their descriptions. When Digital Equipment Corporation supplies only the s-form, the programmer is not restricted from either hand-coding the other forms or using the $DIR macro to execute them. The absence of the other macro forms occurs only when the s-form will always require less space and execute at least as fast as the other two forms.

In addition to the macros which correspond to the directives, the DIR$ macro will be of use to the programmer, particularly in cases where the DPB has been defined independently of the execution of the directive.

DIR$ generates an RSX-11M Executive trap with a pre-defined DPB.

Macro Call:

```
DIR$    adr,err
```

Three forms are possible, with the following interpretation:

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

The argument adr is optional but, if present, must be a valid assembler source operand pointing to a DPB that will be pushed on the stack. The argument err is optional. If defined, it must be a valid assembler destination operand to permit a JUMP TO SUBROUTINE instruction to an error handler if the directive is rejected.

The directive descriptions have been organized into categories based on functional similarity. Within these groups they are ordered alphabetically. Six general categories are defined, and are summarized below under the section identifier of each.

2.1 DIRECTIVE CATEGORIES

2.1.1 Task Execution Control

The task execution control directives deal principally with starting and stopping tasks. Each of these requests result in a change of the task's state (unless the task is already in the state being requested). The requests are:

Macro	Directive Name
ABRT$	ABORT TASK
CSRQ$	CANCEL TIME BASED INITIATION REQUESTS
EXIT$S	TASK EXIT (only s-form supplied)
RQST$	REQUEST TASK
RSUM$	RESUME TASK
RUN$	RUN TASK
SPND$S	SUSPEND (only s-form supplied)

2.1.2 Task Status Control

These two directives alter the checkpointable attribute of a task. They are:

DSCP$S	DISABLE CHECKPOINTING (only s-form supplied)
ENCP$S	ENABLE CHECKPOINTING (only s-form supplied)

2.1.3 Informational Directives

The four informational directives provide to the requesting task data retained by the system. These requests provide the time of day, task parameters, the console switch settings, and partition parameters. The directives are:

GPRT$	GET PARTITION PARAMETERS
GSSW$S	GET SENSE SWITCHES (only s-form supplied)
GTIM$	GET TIME PARAMETERS
GTSK$	GET TASK PARAMETERS

2.1.4 Event-Associated Directives

The event and event flag directives are the means provided in the system for inter- and intra-task synchronization and signalling. These directives must be used carefully, since software

faults resulting from erroneous signalling and synchronization are often obscure and difficult to isolate. These are:

CLEF$	CLEAR EVENT FLAG
CMKT$S	CANCEL MARK-TIME REQUESTS (only s-form supplied)
DECL$S	DECLARE SIGNIFICANT EVENT (only s-form supplied)
EXIF$	EXITIF
MRKT$	MARK TIME
RDAF$	READ ALL EVENT FLAGS
SETF$	SET EVENT FLAG
WSIG$S	WAIT FOR SIGNIFICANT EVENT (only s-form supplied)
WTLO$	WAIT FOR LOGICAL 'OR' OF EVENT FLAGS
WTSE$	WAIT FOR SINGLE EVENT FLAG

2.1.5 Trap-Associated Directives

These directives provide the user the same facilities inherent in the PDP-11 hardware trap system. They provide true interrupts to the executing tasks. These are:

ASTX$S	AST (ASYNCHRONOUS SYSTEM TRAP) SERVICE EXIT (only s-form supplied)
DSAR$S	DISABLE AST RECOGNITION (only s-form supplied)
ENAR$S	ENABLE AST RECOGNITION (only s-form supplied)
SFPA$	SPECIFY FLOATING POINT EXCEPTION AST
SPRA$	SPECIFY POWER RECOVERY AST
SVDB$	SPECIFY SST VECTOR TABLE FOR DEBUGGING AID
SVTK$	SPECIFY SST VECTOR TABLE FOR TASK

2.1.6 I/O and Inter-task Communications Related Directives

These directives allow tasks to access I/O devices at the driver interface level, communicate with other tasks in the system, and retrieve command lines sent via MCR to the task. These are:

ALUN$	ASSIGN LUN
GLUN$	GET LUN INFORMATION
GMCR$	GET MCR COMMAND LINE
RCVD$	RECEIVE DATA
RCVX$	RECEIVE DATA OR EXIT
SDAT$	SEND DATA
QIO$	QUEUE I/O

2.2. DIRECTIVE DESCRIPTIONS

Each directive description consists of six elements:

Name:

The directive's intent within the system is described.

Fortran Call:

The Fortran subroutine call is shown, and each parameter defined.

Macro Call:

The macro call is shown, each parameter is defined, and the defaults for optional parameters are in parentheses following the definition of the parameter. Since zero is supplied for most defaulted parameters, only non-zero default values are shown. The ignored parameters are present for RSX-11D compatibility.

Macro Expansion:

The $-form of the macro is expanded. Eleven macros have only the s-form of an expansion and for these the s-form is presented.

Local Symbol Definitions:

Macro expansions usually generate local symbol definitions whose assigned value equals the byte offset from the start of the DPB to the respective DPB element. These symbols are listed. The length in bytes of the datum pointed to by the symbol appears in parentheses following the symbol's description. Thus:

A.BTTN - Task name (4)

defines A.BTTN as pointing to task name in the DPB and the task name datum has a length of 4-bytes.

DSW return code:

All valid return codes are listed.

Notes:

A list of special considerations that may prove helpful in assisting the programmer in the proper use of the directive.

2.2.1 Task Execution Control Directives

ABORT TASK **ABRT$**

This directive instructs the system to terminate the execution of the indicated task. ABRT$ is intended for use as an emergency or fault exit. A termination notification printout occurs at the terminal from which the task was requested or at the operator console (device CO:) if the task was started internally from another task. A task may abort any task, including itself. Aborted tasks are not removed from the system; hence, they may be requested.

Fortran Call:

```
CALL  ABORT (tsk,[ids])

    tsk = Task name
    ids = Directive status
```

Macro Call:

```
ABRT$    tsk

    tsk = Task name
```

Macro Expansion:

```
ABRT$     ALPHA
.BYTE     83.,3          ;ABRT$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50    /ALPHA/        ;TASK 'ALPHA'
```

Local Symbol Definitions:

A.BTTN - Task name (4)*

DSW Return Codes:

IS.SUC -- Successful completion
IE.INS -- Task is not installed
IE.ACT -- Task is not active
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. An aborted task is no longer active; it moves from the active state to the dormant state.

---

* The number in parentheses is the length of the datum to which the symbolic offset definition points.

CANCEL TIME BASED INITIATION REQUESTS CSRQ$

This directive instructs the system to cancel all time-synchronized initiation requests for a specified task regardless of the source of the request; these requests result from a RUN directive, or any of the time-synchronized variations of the RUN MCR function.

Fortran Call:

```
CALL CANALL (tsk,[ids])

tsk  =  Task name
ids  =  Directive status
```

Macro Call:

```
CSRQ$   tsk

tsk = Scheduled (target) task name,
```

Macro Expansion:

```
CSRQ$    ALPHA
.BYTE    25.,3              ;CSRQ$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50   /ALPHA/            ;TASK 'ALPHA'
```

Local Symbol Definitions:

C.SRTN - Target task name (4)

DSW Return Codes:

IS.SUC -- Successful completion
IE.INS -- Task is not installed
IE.ADP-- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. If an error routine address is specified when using the $C or $S macro form, then a null argument must be included for RSX-11D compatibility. For example:

```
CSRQ$S   ALPHA,,ERR        ;CANCEL REQUESTS FOR 'ALPHA'
```

TASK EXIT (only s-form supplied) **EXIT$S**

This directive instructs the system to terminate the execution of the issuing task.

Fortran Call:

```
STOP
```

Macro Call:

```
EXIT$S  [err]
```

err = Error routine address

Macro Expansion:

```
EXIT$S  ERR
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK
.BYTE   51.,1           ;EXIT$S MACRO DIC, DPB SIZE=1 WORD
EMT     377             ;TRAP TO THE EXECUTIVE
JSR     PC,ERR          ;CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. A return to the task occurs if and only if the directive is rejected. Therefore, no branch on carry clear instruction is generated if an err routine address is given, since the return will only occur with carry set.

2. EXIT will cause a significant event.

3. On Exit the Executive frees task resources; in particular;

   1 - All attached devices are detached;
   2 - The AST queue is flushed;
   3 - The receiver queue is flushed;
   4 - All open files are closed;
   5 - I/O is run-down, and
   6 - If the task is not fixed, its partition is freed.

4. This directive requires a 1-word DPB, thus the EXIT$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

REQUEST **RQST$**

This directive instructs the system to make a task active. The task is activated and will subsequently run contingent upon priority and memory availability. Request is the basic mechanism used by running tasks for initiating other installed (dormant) tasks. REQUEST is a frequently used subset of the RUN directive.

Fortran Call:

```
CALL REQUES (tsk,[opt],[ids])

  tsk = Task name
  opt = 4-word integer array
        opt(1) = Partition name first half; ignored, but must be present
        opt(2) = partition name second half; ignored, but must be present
        opt(3) = priority; ignored, but must be present
        opt(4) = user identification code
  ids = Directive status
```

Macro Call:

```
RQST$   tsk,[prt],[pri],[ugc],[uoc]

  tsk = Task name
  prt = Partition name; ignored, but must be present
  pri = Priority; ignored, but must be present
  ugc = UIC group code
  uoc = UIC owner code
```

Macro Expansion:

```
RQST$   ALPHA,,,20,10
.BYTE   11.,7           ;RQST$ MACRO DIC, DPB SIZE=7 WORDS
.RAD50  /ALPHA/         ;TASK 'ALPHA'
.WORD   0,0             ;PARTITION IGNORED
.WORD   0               ;PRIORITY IGNORED
.BYTE   10,20           ;UIC UNDER WHICH TO RUN TASK
```

Local Symbol Definitions:

```
R.QSTN - Task name (4)
R.QSPN - Partition name (4)
R.QSPR - Priority (2)
R.QSGC - UIC group (1)
R.QSPC - UIC owner (1)
```

DSW Return Codes:

IS.SUC	--	Successful completion
IE.INS	--	Task is not installed
IE.ACT	--	Task is already active
IE.ADP	--	Part of the DPB is out of the issuing task's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. The requested task must be installed in the system.

2. A requested task whose partition is busy is queued to the list of tasks waiting for the partition and will run, based on priority and resource availability, when the partition becomes free. If the requested task requires a partition that is currently occupied, checkpointing may occur. If the current occupant of the partition is checkpointable, has checkpointing enabled, and is of lower priority than the requested task, then it will be written to disk when its current outstanding I/O completes and the requested task will then be read into the partition.

3. Successful completion means that the task has been made active not that the task is actually running.

4. A task may be requested under any UIC regardless of the UIC of the requesting task. If no UIC is specified in the request, the default UIC from the requested task's header is used. The priority is always that specified in the requested task's Task Control Block.

RESUME **RSUM$**

This directive instructs the system to resume the execution of a task that has issued a SUSPEND Directive.

Fortran Call:

```
CALL RESUME (tsk,[ids])

  tsk = Task name
  ids = Directive status
```

Macro Call:

```
RSUM$   tsk

  tsk  =  Task name
```

Macro Expansion:

```
RSUM$   ALPHA
.BYTE   47.,3           ;RSUM$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50  /ALPHA/         ;TASK 'ALPHA'
```

Local Symbol Definitions:

```
R.SUTN  -  Task name (4)
```

DSW Return Codes:

```
IS.SUC   -- Successful completion
IE.INS   -- Task is not installed
IE.ACT   -- Task is not active
IE.ITS   -- Task is not suspended
IE.ADP   -- Part of the DPB is out of the issuing task's address space
IE.SDP   -- DIC or DPB size is invalid
```

RUN **RUN$**

This 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. If the udc, smg, snt, rmg, and rnt parameters are omitted, RUN is the same as REQUEST except that the task will be initiated 1 clock tick from issuance when using the RUN directive.

Fortran Call:

```
CALL RUN (tsk,[opt],[smg],[snt],[rmg],[rnt],[ids])
```

tsk = Task name
opt = 4-word integer array
opt(1) = Partition name first half; ignored, but must be present.
opt(2) = Partition name second half; ignored, but must be present
opt(3) = Priority; ignored, but must be present
opt(4) = User identification code
smg = Schedule delta magnitude
snt = Schedule delta unit
rmg = Reschedule interval magnitude
rnt = Reschedule interval unit
ids = Directive Status

The ISA standard call for initiating a task is also provided.

```
CALL START(tsk,smg,snt,[ids])
```

tsk = Taskname
smg = Schedule delta magnitude
snt = Schedule delta unit
ids = Directive status

Macro Call:

```
RUN$    tsk,[prt],[pri],[ugc],[uoc],[smg],[snt],[rmg],[rnt]
```

tsk = Task name
prt = Partition name; ignored, but must be present
pri = Priority; ignored, but must be present
ugc = UIC group code
uoc = UIC owner code
smg = Schedule delta magnitude
snt = Schedule delta unit
rmg = Reschedule interval magnitude
rnt = Reschedule interval unit

Macro Expansion:

```
RUN$    ALPHA,,,20,10,20.,3,10.,3
.BYTE   17.,11.                 ;RUN$ MACRO DIC, DPB SIZE=11.'WORDS

.RAD50  /ALPHA/                 ;TASK 'ALPHA'
.WORD   0,0                     ;PARTITION IGNORED
.WORD   0                       ;PRIORITY IGNORED
.BYTE   10,20                   ;UIC TO RUN TASK UNDER
.WORD   20.                     ;SCHEDULE MAGNITUDE=20.
.WORD   3                       ;SCH. DELTA TIME UNIT=MINUTE (=3)
.WORD   10.                     ;RESCH. INTERVAL MAGNITUDE=10.
.WORD   3                       ;RESCH. INTERVAL UNIT=MINUTE (= 3)
```

Local Symbol Definitions:

```
R.UNTN  - Task name (4)
R.UNPN  - Partition name (4)
R.UNPR  - Priority (2)
R.UNGC  - UIC group (1)
R.UNPC  - UIC owner (1)
R.UNSM  - Schedule magnitude (2)
R.UNSU  - Schedule unit (2)
R.UNRM  - Reschedule magnitude (2)
R.UNRU  - Reschedule unit (2)
```

DSW Return Codes:

```
IS.SUC  -- Successful completion
IE.UPN  -- Insufficient dynamic memory
IE.INS  -- Task is not installed
IE.ITI  -- Invalid time parameter
IE.ADP  -- Part of the DPB is out of the issuing task's address space
IE.SDP  -- DIC or DPB size is invalid
```

Notes:

1. The target task must be installed in the system.

2. A task requested to run in a partition that is busy is queued in the list of tasks waiting for the partition and will run, based on priority, and resource and availability, when the partition becomes free. If the requested task requires a partition that is currently occupied, checkpointing may occur. If the current occupant of the partition is checkpointable, has checkpointing enabled, and is of lower priority than the requested task, then it will be written to disk when its current outstanding I/O completes. The requested task will then be read into the partition.

3. Successful completion means the task has been made active, not that the task is actually running.

4. RUN requires dynamic memory for the clock queue entry used to start the task after the specified delta time.

5. If optional rescheduling is not desired, then the macro arguments rmg and rmt must be omitted.

6. A task may be run under any UIC regardless of the UIC of the requesting task. If no UIC is specified in the request, the default UIC from the requested task's header is used. The priority is always that specified in the requested task's Task Control Block.

SUSPEND (only s-form supplied) **SPND$S**

This directive instructs the system to suspend the execution of the issuing task. A task can suspend only itself, not another task. The task can only be restarted by a RESUME directive, or RESume MCR command.

Fortran Call:

```
CALL SUSPND
```

Macro Call:

```
SPND$S [err]

    err = Error routine address
```

Macro Expansion:

```
SPND$S    ERR
MOV       (PC)+,-(SP) ;PUSH DPB ONTO THE STACK
.BYTE     45.,1       ;SPND$S MACRO DIC, DPB SIZE=1 WORD
EMT       377         ;TRAP TO THE EXECUTIVE
BCC       .+6         ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR       PC,ERR      ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC	-- Successful completion
IE.ADP	-- Part of the DPB is out of the issuing task's address space
IE.SDP	-- DIC or DPB size is invalid

Notes:

1. A suspended task retains control of the system resources allocated to it. No attempt is made to free the resources. A task which has exited will result in the executive checking if its resources can be freed.

2. A suspended task is eligible for checkpointing unless fixed or not checkpointable.

3. This directive requires a 1-word DPB, thus the SPND$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

2.2.2 Task Status Control Directives

DISABLE CHECKPOINTING (only s-form supplied) **DSCP$S**

This directive instructs the system to disable the checkpointability of a task that has been installed as a checkpointable task. This directive can only be issued by the task to be affected. A task cannot disable the checkpointability of another task.

Fortran Call:

```
CALL DISCKP
```

Macro Call:

```
DSCP$S [err]

    err = Error routine address
```

Macro Expansion:

```
DSCP$S    ERR
MOV       (PC)+,-(SP) ;PUSH DPB ONTO THE STACK
.BYTE     95.,1       ;DSCP$S MACRO DIC, DPB SIZE=1 WORD
EMT       377         ;TRAP TO THE EXECUTIVE
BCC       .+6         ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR       PC,ERR      ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

```
IS.SUC -- Successful completion
IE.ITS -- Task checkpointing is already disabled
IE.CKP -- Issuing task is not checkpointable
IE.ADP -- Part of the DPB is out of the issuing task's
          address space
IE.SDP -- DIC or DPB size is invalid
```

Notes:

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

2. This directive requires a 1-word DPB, thus the DSCP$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

ENABLE CHECKPOINTING (only s-form supplied) **ENCP$S**

This directive instructs the system to make the issuing task checkpointable after its checkpointability has been disabled, i.e., to nullify a DSCP$S directive.

Fortran Call:

```
CALL ENACKP
```

Macro Call:

```
ENCP$S [err]
```

err = Error routine address

Macro Expansion:

```
ENCP$S  ERR
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK
.BYTE   97.,1           ;ENCP$S MACRO DIC, DPB SIZE=1 WORD
EMT     377             ;TRAP TO THE EXECUTIVE
BCC     .+6             ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR     PC,ERR          ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC -- Successful completion
IE.ITS -- Checkpointing is not disabled
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. This directive requires a 1-word DPB, thus the ENCP$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

2.2.3 Informational Directives

GET PARTITION PARAMETERS **GPRT$**

This directive instructs the system to fill an indicated 3-word buffer with partition parameters. If a partition is not specified, the partition of the issuing task is assumed.

Fortran Call:

```
CALL GETPAR ([prt], buf ,[[ids])
```

prt = Partition name
buf = 3-word integer array to receive partition parameters
ids = Directive status

Macro Call:

```
GPRT$   [prt],buf
```

prt = Partition name
buf = Address of a 3-word buffer

The buffer has the following format:

WD. 0 -- Partition Base Address expressed as a multiple of 32 words (partitions are always aligned on 32-word boundaries. Thus a partition starting at 1000(8) will have 10(8) returned in this word.
WD. 1 -- Partition size expressed as a multiple of 32-words.
WD. 2 -- Partition flags word. This word is always returned equal to 1 to indicate a user-controlled partition. This is done for RSX-11D compatibility.

Macro Expansion:

```
GPRT$ ALPHA,DATBUF
.BYTE   65.,4           ;GPRT$ DIC, DPB SIZE=4 WORDS
.RAD50  /ALPHA/         ;PARTITION 'ALPHA'
.WORD   DATBUF          ;ADDRESS OF 3-WORD BUFFER
```

Local Symbol Definitions:

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

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

G.PRPB - Partition Base Address expressed as a multiple of 32-words (2)
G.PRPS - Partition Size expressed as a multiple of 32-words (2)
G.PRFW - Partition flags word expressed as a multiple of 32-words (2)

DSW Return Codes:

Successful completion is indicated by carry clear and the starting address of the partition is returned in the DSW. In unmapped systems, the returned address is physical, in mapped systems it is virtual. Unsuccessful completion is indicated by carry set, and one of the following codes in the DSW:

IE.INS -- Specified partition not in system
IE.ADP -- Part of the DPB or buffer is out of the issuing tasks' address space
IE.SDP -- DIC or DPB size is invalid

GET TASK PARAMETERS GTSK$

This directive instructs the system to fill an indicated 16-word buffer with parameters relating to the issuing task.

Fortran Call:

```
CALL GETTSK (buf,[ids])

   buf = 16-word integer array to receive the task parameters
   ids = Directive status
```

Macro Call:

```
GTSK$  buf

   buf  =  Address of a 16-word buffer
```

The buffer has the following format:

```
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 -- Undefined in RSX-11M. This word exists
          for RSX-11D compatibility.
WD. 05 -- Undefined in RSX-11M. This word exists
          for RSX-11D compatibility.
WD. 06 -- Run priority
WD. 07 -- User identification code of issuing task
WD. 10 -- Number of logical I/O units (LUN's)
WD. 11 -- Undefined in RSX-11M. This word exists for
          RSX-11D compatibility.
WD. 12 -- Undefined in RSX-11M. This word exists for
          RSX-11D compatibility.
WD. 13 -- (Address of task SST vector tables)*
WD. 14 -- (Size of task SST vector table (in words)*
WD. 15 -- (Reserved)
WD. 16 -- (Reserved)
WD. 17 -- (Reserved)
```

Macro Expansion:

```
GTSK$   DATBUF
.BYTE   63.,2           ;GTSK$ DIC, DPB=2-WORDS
.WORD   DATBUF          ;ADDRESS OF 16-WORD BUFFER
```

---

*These words will contain valid data if word 14 is non zero. If word 14 is zero, the contents of word 15 is meaningless.

Local Symbol Definitions:

G.TSTN = Task name (4)
G.TSPN = Partition name (4)
G.TSPR = Priority (2)
G.TSGC = UIC Group code (2)
G.TSPC = UIC Programmer code (1)
G.TSNL = Number of logical units (2)
G.TSVA = Task's SST vector address (2)
G.TSVL = Task's SST vector length in words (2)

DSW Return Codes:

IS.SUC -- Successful completion
IE.ADP -- Part of the DPB or buffer is out of the issuing task's address space
IE.SDP -- DIC or DPB is invalid.

GET SENSE SWITCHES (only s-form supplied) **GSSW$S**

This directive instructs the system to obtain the contents of the console switch register and store it in the issuing task's Directive Status Word.

Fortran Call:

```
CALL READSW (isw)
```

isw = Integer to receive the console switch settings

Macro Call:

```
GSSW$S [err]
```

err = Error routine address

Macro Expansion:

```
GSSW$S  ERR
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK
.BYTE   125.,1          ;GSSW$S MACRO DIC, DPB SIZE=1 WORD
EMT     377             ;TRAP TO THE EXECUTIVE
BCC     .+6             ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR     PC,ERR          ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

Successful completion is indicated by carry clear and the contents of the console switch register is returned in the DSW. Unsuccessful completion is indicated by carry set, and one of the following codes in the DSW:

IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. This directive requires a 1-word DPB, thus the GSSW$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

GET TIME PARAMETERS GTIM$

This directive instructs the system to fill an indicated 8-word buffer with the current time parameters. All time parameters are delivered as binary numbers. The value ranges (in decimal) are shown in the table below.

Fortran Call:

FORTRAN IV provides several subroutines for obtaining the time in a number of formats. See the RSX-11M FORTRAN IV Reference Manual DEC manual number DEC-11-LFLRA-A-D.

Macro Call:

```
GTIM$   buf
```

buf = Address of 8-word buffer

The buffer has following format:

WD. 0 -- Year (since 1900)
WD. 1 -- Month of year (1-12)
WD. 2 -- Day of month (1-31)
WD. 3 -- Hour of day (0-23)
WD. 4 -- Minute of hour (0-59)
WD. 5 -- Second of minute (0-59)
WD. 6 -- Tick of second (depends on the frequency of the clock)
WD. 7 -- Ticks per second (depends on the frequency of the clock)

Macro Expansion:

```
GTIM$   DATBUF
.BYTE   61.,2           ;GTIM$ DIC, DPB SIZE=2 WORDS
.WORD   DATBUF          ;ADDRESS OF 8.-WORD BUFFER
```

Local Symbol Definitions:

G.TIBA - buffer address (2)

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

G.TIYR - Year (2)
G.TIMO - Month (2)
G.TIDA - Day (2)
G.TIHR - Hour (2)
G.TIMI - Minute (2)
G.TISC - Second (2)
G.TICT - Clock Tick of Second (2)
G.TICP - Clock Ticks per Second (2)

DSW Return Codes:

IS.SUC -- Successful completion
IE.ADP -- Part of the DPB or buffer is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

2.2.4 Event-Associated Directives

Significant events and system traps are the means by which communication is effected between various parts of the system. Significant Events and traps serve distinctly different functions within the system; three points will help to clarify their uses:

1. A significant event is a change in system status; it causes the RSX-11M Executive to re-evaluate the eligibility for execution of all tasks. Significant events are also the major means by which one task communicates and synchronizes its execution with other tasks and the system.

2. System traps are exclusive to a single task; they are useful for intra-task communication and control. System traps are the task level representation of the PDP-11 hardware trapping mechanism.

3. The occurrence of an event may change the eligibility of a task to run, but that is all. A trap, however, is a real interrupt; the sequence of instructions being executed by the task will be interrupted and control will be transferred to another instruction sequence in the program. This may be transparent to the user in some cases, but it occurs, nonetheless, and is a difference between events and traps.

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

Event flags are the means by which RSX-11M 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 reset only as a result of that task's operation. The second 32 flags (33-64) are common to all tasks, and may be set or reset as a result of any task's operation. The two sets of event flags are termed local (1-32) and common (33-64) respectively. Each event flag has a corresponding Event Flag Number (EFN) which uniquely identifies the flag.

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

Some system processes running on behalf of the user need event flags. The last eight local (25-32) and common (57-64) event flags are reserved, by convention, for use by RSX-11M System Software.

All significant events occur as the result of a task having issued a system directive (the one exception is power failure). Some directives will have the event explicitly noted, while in others it is implicit.

Setting and resetting event flags must be carefully planned and carefully executed; this is particularly true of the global event flags. Erroneous or multiple setting/resetting of event flags can result in obscure, difficult to locate software faults. A typical application program can be written without explicitly accessing or modifying event flags, since many of the directives implicitly modify an event flag. The implicit setting of event flags provide a discipline which substantially reduces the opportunity for multiple setting/resetting of event flags.

2.2.4.1 Directives Which Result In The Setting Of An Event Flag

Several directives automatically cause an event flag to be set, and the specification of an EFN is required in their macro call. The programmer must provide an EFN, usually out the 24 local to his task, if he expects the directive to set an EFN. The selection should be unique to a specific directive, or at least never result in the possibility of multiple setting or resetting.

The following directives optionally cause the alteration of an event flag:

1. The SEND DATA directive causes a significant event at directive issuance; if an event flag is specified, it will be set at the time the directive is issued.

2. The MARK TIME directive optionally clears an event flag at issuance; after the specified time interval has elapsed, a significant event is declared, and if an event flag has been specified, it will be set.

3. I/O operations (initiated by the QUEUE I/O directive) optionally clear an event flag at issuance; at I/O completion a significant event is declared and if an event flag was specified, it will be set.

Examples 1 and 2 below show the usage of the common (33-64) event flags for task synchronization. Examples 3 and 4 illustrate the use of local (1-32) flags.

Example 1

Task B specifies a common event flag (for example, 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

Task A specifies task B and a common event flag in a SEND directive. Task B has specified the same common event flag in a WAITFOR directive and issues a RECEIVE directive when activated because its WAITFOR has been satisfied. The effect is to synchronize the transmission of data between TASK A and TASK B.

Note that task A and task B have intimate knowledge of each other's requirements for synchronization and communication. The selection of an event flag is a mutual and unique choice for the two tasks.

Example 3

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

The specified event flag is set when the I/O request is completed, and the task's execution will be resumed at the instruction following the WAITFOR. Note that task execution continues after the I/O request is queued. The EFN is used to ensure that the task does not attempt to manipulate the incoming data until the transfer has actually completed.

Example 4

If a task-local event flag is specified in a MARK TIME and associated WAITFOR directive, the flag will be cleared at MARK TIME issuance and set after the indicated time has elapsed. When the task executes a WAITFOR predicated on the same event flag and the time interval has not yet elapsed, execution of the task will be suspended.

In examples 3 and 4, the choice of one of the first 32 (unique to task) local event flags is the normal choice used to avoid possible interference by other tasks.

In examples (1-4) computation and/or event flag testing is 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. Hence it is not necessary to issue a WAITFOR directive immediately following the issuance of a QUEUE I/O or a MARK TIME directive.

CLEAR EVENT FLAG — CLEF$

This directive instructs the system to clear an indicated event flag and report the flag's polarity before clearing.

Fortran Call:

```
CALL CLFEF (efn,[ids])

  efn = Integer containing an event flag number
  ids = Directive status
```

Macro Call:

```
CLEF$   efn

  efn  =  Event flag number
```

Macro Expansion:

```
CLEF$   52.
.BYTE   31.,2           ;CLEF$ MACRO DIC, DPB SIZE=2 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52.
```

Local Symbol Definitions:

```
C.LEEF - Event flag number (2)
```

DSW Return Codes:

```
IS.CLR  --  Flag was already clear
IS.SET  --  Flag was set
IE.IEF  --  Invalid event flag number (EFN.GT.64 or EFN.LT.1)
IE.ADP  --  Part of the DPB is out of the issuing task's address space
IE.SDP  --  DIC or DPB size is invalid
```

CANCEL MARK TIME REQUESTS (only s-form supplied) **CMKT$S**

This directive instructs the system to cancel all MARK TIME requests that have been made by the issuing task.

Fortran Call:

```
CALL CANMT (,[ids])

  ids  =  Directive status
```

Macro Call:

```
CMKT$S  [,,err]

  err = Error routine address
```

Macro Expansion:

```
CMKT$S   ,,ERR        ;NOTE: THERE ARE TWO IGNORED ARGUMENTS
MOV      (PC)+,-(SP)  ;PUSH DPB ONTO THE STACK
.BYTE    27.,1        ;CMKT$S MACRO DIC, DPB SIZE=1 WORD
EMT      377          ;TRAP TO THE EXECUTIVE
BCC      .+6          ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR      PC,ERR       ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

```
IS.SUC   -- Successful completion
IE.ADP   -- Part of the DPB is out of the issuing task's address space
IE.SDP   -- DIC or DPB size is invalid
```

Notes:

1. This directive requires a 1-word DPB, thus the CMKT$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

DECLARE SIGNIFICANT EVENT (only s-form supplied) **DECL$S**

This directive instructs the system to declare a significant event.

Fortran Call:

```
CALL DECLAR (,[ids])

  ids  =  Directive status
```

Macro Call:

```
DECL$S  [,err]

  err  =  Error routine address
```

Macro Expansion:

```
DECL$S  ,ERR          ;NOTE: THERE IS ONE IGNORED ARGUMENT
MOV     (PC)+,-(SP)   ;PUSH DPB ONTO THE STACK
.BYTE   35.,1         ;DECL$S MACRO DIC, DPB SIZE=1 WORD
EMT     377           ;TRAP TO THE EXECUTIVE
BCC     .+6           ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR     PC,ERR        ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC -- Successful completion
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. Declaration of a significant event causes the Executive to scan the System Task Directory from the beginning, searching for the highest priority task that is ready to run. This directive should be used with caution since excessive scanning overhead may result if used indiscriminately.

2. This directive requires a 1-word DPB, thus the DECL$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

EXITIF **EXIF$**

This directive instructs the system to terminate the execution of the issuing task if, and only if, an indicated event flag is NOT set. Control is returned to the issuing task if the specified event flag is set.

Fortran Call:

```
CALL EXITIF (efn,[ids])

  efn = Event flag number
  ids = Directive status
```

Macro Call:

```
EXIF$   efn

  efn  =  Event flag number
```

Macro Expansion:

```
EXIF$   52.
.BYTE   53.,2          ;EXIF$ MACRO DIC, DPB SIZE=2 WORDS
.WORD   52.            ;EVENT FLAG NUMBER 52.
```

Local Symbol Definitions:

E.XFEF - Event flag number (2)

DSW Return Codes:

IS.SET	--	Indicated EFN set, task not exited
IE.IEF	--	Invalid event flag number (EFN.GT.64 or EFN.LT.1)
IE.ADP	--	Part of the DPB is out of the issuing task's address space
IE.SDP	--	DIC or DPB size is invalid

Notes:

1. The EXITIF directive is useful in avoiding a possible race condition that may occur between two tasks communicating via the SEND and RECEIVE directives. The race condition occurs when one task executes a RECEIVE directive and finds its receive queue empty. But before the task can EXIT, the other task sends it a message. Since the first task has already decided to exit, the message is lost since the receiving queue is flushed during task exit. This condition can be avoided if the sending task specifies a common event flag in the SEND directive and the receiving task executes an EXITIF specifying the same common event flag. The EXITIF directive will return control to the issuing task signalling that something has been sent.

2. If the exit is taken, the Executive frees task resources. In particular

   1 - All attached devices are detached;
   2 - The AST queue is flushed;
   3 - The receive queue is flushed;
   4 - All open files are closed;
   5 - I/O is run-down; and
   6 - If the task is not fixed, its partition is freed.

3. If the exit is taken, a significant event is declared.

MARK TIME **MRKT$**

This directive instructs the system to declare 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 address is specified, an Asynchronous System Trap (see section 2.2.5 below) will occur at the time of the significant event. At the AST, the task's PS, PC, directive status, WAITFOR mask words, and the event flag number specified in the directive will be pushed onto the issuing task's stack. If neither an event flag number, nor an AST service entry point is specified, the significant event will still occur after the indicated time interval.

Fortran Calls:

```
CALL MARK (efn,tmg,tnt,[ids])

  efn = Event flag number
  tmg = Integer time interval magnitude
  tnt = Integer time interval unit
  ids = Directive status
```

The ISA standard call for delaying a task for a specified time interval is also provided:

```
CALL WAIT (tmg,tnt,ids)

  tmg = Integer time interval magnitude
  tnt = Integer time interval unit
  ids = Directive status
```

Macro Call:

```
MRKT$   [efn],tmg,tnt,[ast]

  efn = Event flag number
  tmg = Time interval magnitude
  tnt = Time interval unit
  ast = AST entry point address
```

Macro Expansion:

```
MRKT$   52.,30.,2,MRKAST
.BYTE   23.,5           ;MRKT$ MACRO DIC, DPB SIZE=5 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52.
.WORD   30.             ;TIME MAGNITUDE=30.
.WORD   2               ;TIME UNIT=SECONDS
.WORD   MRKAST          ;ADDRESS OF MARK TIME AST ROUTINE
```

Local Symbol Definitions:

M.KTEF - Event flag (2)
M.KTMG - Time magnitude (2)
M.KTUN - Time unit (2)
M.KTAE - AST entry point address (2)

DSW Return Codes:

IS.SUC -- Successful completion
IE.UPN -- Insufficient dynamic memory
IE.ITI -- Invalid time parameter
IE.IEF -- Invalid event flag number (EFN.GT.64 or EFN.LT.0)
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. MARK TIME requires dynamic memory for the clock queue entry.

2. If an AST entry point address is specified, the AST service routine is entered with the task's stack in the following state:

   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 prior to AST
   SP+04 PC of task prior to AST
   SP+02 DSW of task prior to AST
   SP+00 event flag number, or zero if none was specified
   in the MARK TIME directive

   The event flag number must be removed from the task's stack before an exit AST directive (see section 2.2.5 below) is executed.

3. If the directive is rejected, the specified event flag is not guaranteed to be cleared or set. Thus, if the task indiscriminately executes a WAITFOR directive and the MARK TIME directive is rejected, then the task may wait forever. Care should always be taken to insure that the directive was successfully completed.

---

* These event flag mask words preserve the waitfor conditions of a task prior to AST entry. A task can, after an AST, return to a waitfor state. Since these flags and the other stack data are in the user task, they can be modified. Such modification is strongly discouraged since if done erroneously or without sufficient comprehension of the task-wide impact of the change, a given task may fault on extremely obscure conditions.

READ ALL EVENT FLAGS **RDAF$**

This directive instructs the system to read all 64 event flags for the issuing task and record their polarity in a 64-bit (4-word) buffer.

Fortran Call:

Only a single event flag may be read by a FORTRAN IV task. The call is:

```
CALL READEF (efn,[ids])
```

efn = Event flag number
ids = Directive status

Macro Call:

```
RDAF$   buf
```

The buffer has the following format:

WD. 00 Task Local Flags 1-16
WD. 01 Task Local Flags 1-32
WD. 02 Task Common Flags 33-48
WD. 03 Task Common Flags 49-64

Macro Expansion:

```
RDAF$   FLGBUF
.BYTE   39.,2           ;RDAF$ MACRO DIC, DPB SIZE=2 WORDS
.WORD   FLGBUF          ;ADDRESS OF 4-WORD BUFFER
```

Local Symbol Definitions:

R.DABA - Buffer address (4)

DSW Return Codes:

IS.SUC -- Successful Completion
IE.ADP -- Part of the DPB or buffer is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

SET EVENT FLAG **SETF$**

This directive instructs the system to set an indicated event flag and report the flag's polarity before setting.

Fortran Call:

```
CALL SETEF (efn,[ids])

  efn = Event flag number
  ids = Directive status
```

Macro Call:

```
SETF$   efn

  efn  =  Event flag number
```

Macro Expansion:

```
SETF$   52.
.BYTE   33.,2           ;SETF$ MACRO DIC, DPB SIZE=2 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52.
```

Local Symbol Definitions:

S.ETEF - Event flag number (2)

DSW Return Codes:

IS.CLR -- Flag was cleared
IS.SET -- Flag was already set
IE.IEF -- Invalid event flag number (EFN.GT.64 or EFN.LT.1)
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. SET EVENT FLAG does not declare a significant event, it merely sets the specified flag.

WAIT FOR SIGNIFICANT EVENT (only s-form supplied) **WSIG$S**

This directive is used to suspend the 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 dynamic memory, since significant events occurring throughout the system often result in the release of dynamic memory.

Fortran Call:

```
CALL WFSNE
```

Macro Call:

```
WSIG$S  [err]
```

err = Error routine address

Macro Expansion:

```
WSIG$S  ERR
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK
.BYTE   49.,1           ;WSIG$S MACRO DIC, DPB SIZE=1 WORD
EMT     377             ;TRAP TO THE EXECUTIVE
BCC     .+6             ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR     PC,ERR          ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC -- Successful completion
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. If a directive is rejected for lack of dynamic memory, this directive is the only technique available for suspending task execution until dynamic memory may again be available.

2. The wait state induced by this directive is satisified by the first significant event which occurs following directive issuance. The significant event which occurs may or may not be related to the issuing task.

3. This directive requires a 1-word DPB, thus the WSIG$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

WAIT FOR LOGICAL 'OR' OF EVENT FLAGS — WTLO$

This directive instructs the system to suspend the execution of the issuing task until any indicated event flag within 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

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

Fortran Call:

```
CALL WFLOR (efn1,efn2,...efnn)
```

efn = List of event flag numbers is taken as the set of flags to be specified in the directive.

Macro Call:

```
WTLO$   grp,msk
```

grp = Desired group of event flags
msk = A 16 bit octal mask word

Macro Expansion:

```
WTLO$   2,160003
.BYTE   43.,3           ;WTLO$ MACRO DIC, DPB SIZE=3 WORDS
.WORD   2               ;FLAGS SET NUMBER 2 (FLAGS 33:48.)
.WORD   160003          ;EVENT FLAGS 33,34,46,47 AND 48.
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC -- Successful completion
IE.IEF -- No event flag specified in the mask word or flag set indicator other than 0,1,2, or 3
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. There is a one to one correspondence between bits in the mask word and the event flags in the specified group. That is, if group 2 were specified, then bit 0 in the mask word would correspond to event flag 17, bit 1 to event flag 18, and so forth.

2. Event flags are not arbitrarily cleared by the Executive when Waitfor conditions are met. Some directives (QIO, for example) implicitly clear a flag; otherwise they must be explicitly cleared by a Clear Event Flag directive.

3. The grp operand must always be absolute regardless of the macro form used. In all other macro calls absolute values for s-form macros have the format:

   #n

   For WTLO$S this would be

   n

4. The argument list specified in the FORTRAN call must contain only efn's that lie with one event flag group. If efn's are specified that lie in more than one group or an invalid efn is specified then a fatal FORTRAN error is generated.

WAIT FOR SINGLE EVENT FLAG — WTSE$

This directive instructs the system to suspend the execution of the issuing task until the indicated event flag is set. If the flag is set at issuance, task execution is not suspended.

Fortran Call:

```
CALL WAITFR (efn,[ids])

  efn  =  Event flag number
  ids  =  Directive status
```

Macro Call:

```
WTSE$   efn

  efn  =  Event flag number
```

Macro Expansion:

```
WTSE$   52.
.BYTE   41.,2           ;WTSE$ MACRO DIC, DPB SIZE=2 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52.
```

Local Symbol Definitions:

```
W.TSEF - Event flag number (2)
```

DSW Return Codes:

```
IS.SUC -- Successful completion
IE.IEF -- Invalid event flag number (EFN.GT.64 or EFN.LT.1)
IE.ADP -- Part of the DPB is out of the issuing task's
          address space
IE.SDP -- DIC or DPB size is invalid
```

2.2.5 Trap-Associated Directives

System traps are task interrupts initiated by the RSX-11M EXECUTIVE to allow servicing of contingencies which are either exceptional events, such as an odd address error, or a signalling event such as the completion of a previous I/O request. They are exclusive to an individual task, i.e., there is nothing one task can do to cause a trap to occur in another task.

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; the task's normal priority and privilege* are in effect. The action taken by the RSX-11M 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. A synchronous condition is one that will re-occur at precisely the same instruction if the sequence of instructions preceding the fault were repeated. An odd address fault is a typical example. If a service routine is not included in the task, and a synchronous fault occurs, the task's execution is aborted.

AST's are closely linked to significant events. They 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 the exact moment of AST occurrence. A characteristic of AST's is that they are for information purposes, such as signalling an I/O completion for which a task desires immediate knowledge. If a service routine is not provided, a trap does not occur and task execution is not interrupted.

It should be emphasized that SST's are initiated by the RSX-11M Executive, but are then forgotten, i.e., they appear just like normal task execution. The RSX-11M 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.

Note that SST's are caused by occurrences within a task, while AST's occur as a result of an external event. The RSX-11M Executive keeps track of all AST's, queues them (FIFO), and is aware when a task is servicing an AST.

---

*Privileged task definition and construction is discussed in the Task Builder Reference Manual (DEC-11-OMTBA-A-D)

SST's are effected by pushing the tasks' PS (Processor Status) word and PC (Program Counter) onto its stack, and return control by issuing an RTI or RTT instruction. Note that the tasks general purpose registers R0-R6 are not saved, and if the user-trap routine intends to make use of them, the user routine itself must save and restore them.

Execution of an SST service routine is indistinguishable from task execution, and an SST service routine may perform any operation that may be performed by the task. However, if a service routine for an SST may cause that SST to occur, it must be coded re-entrantly.

SST service routine entry points are provided in a trap vector table which is contained in the task. The trap vector table has the following format:

WD. 00 -- Odd Address error
WD. 01 -- Memory Protect Violation
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 a Non-RSX EMT instruction
WD. 06 -- Execution of a Trap instruction
WD. 07 -- PDP-11/40 floating point exception

A zero or odd address appearing in the table is interpreted as no entry point specified. If an SST occurs and an entry point is not specified, the task's execution is aborted. The SST vector table is specified to the Executive by use of the SPECIFY SST VECTOR FOR TASK or the SPECIFY SST VECTOR FOR DEBUGGING AID directives.

On entrance to an SST service routine, the stack always contains the following information:

SP+02 -- PS
SP+00 -- PC

The task's stack may also contain additional information depending on the cause:

Memory Protect Violation - Complete stack

SP+10 -- PS
SP+06 -- PC
SP+04 -- Memory protect status register (SR0)*
SP+02 -- Virtual PC of the faulting instruction (SR2)*
SP+00 -- Instruction backup register (SR1)*

---

* For details of SR0, SR1 and SR2 see the memory management unit section of the 11/40 or 11/45 Processor Handbook.

TRAP Instruction and EMT Other Than 377 - Complete stack

SP+04 -- PS
SP+02 -- PC
SP+00 -- Instruction Operand (low-order byte) multiplied by two, non-sign extended.

The additional information must be removed from the stack before an exit from the SST service routine is executed. Exit from an SST is usually via an RTI or RTT instruction.

AST's occur with the task's four WAITFOR mask words, the DSW, the PS, and the PC pushed onto its stack. In effect this saves the state of the task so that the AST service routine has available to it all the services provided by the Executive. The requirement to save the DSW, PS, and PC is obvious. Saving the WAITFOR mask words is necessary to permit the AST routines to execute WAITFOR type directives, since it is these words which establish the waiting conditions that must be met for unblocking the waiting task. There may also be other parameters pushed onto the stack, depending upon the cause of the AST. Note that the tasks general purpose registers R0-R6 are not saved, and if the user-trap routine intends to make use of them, the user routine itself must save and restore them.

After processing an AST, the trap dependent parameters must be removed from the task's stack, and an EXIT AST SERVICE directive issued with the task's stack set as indicated in the description of the AST SERVICE EXIT directive. (Refer to ASTX$S below.)

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

1. Another (queued) AST;

2. The task, or

3. Another task (e.g., the corresponding task was in a wait or suspend state prior to the execution of the AST).

The five variations in the stack format, depending upon the AST cause, are as follows:

1. If a task is to be notified when an 11/45 Floating Point Unit exception trap occurs, a SPECIFY FLOATING POINT EXCEPTION AST directive is issued. If specified, an AST will occur when an 11/45 Floating Point Unit exception trap occurs with the stack containing the following:

   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 prior to AST
   SP+06 -- PC of task prior to AST
   SP+04 -- Task's directive status word
   SP+02 -- Floating exception code
   SP+00 -- Floating exception address

2. If a task is to be notified of power failure recoveries, a SPECIFY POWER RECOVERY AST directive is issued. If specified, an AST will occur when the power is restored with the stack containing the following:

   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 -- PS of task prior to AST
   SP+02 -- PC of task prior to AST
   SP+00 -- Task's Directive Status Word

3. If a task is to be notified when a message is sent to it, a SPECIFY RECEIVE AST directive is issued. If specified, an AST will occur when a message is sent to the task with the stack containing the following:

   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 -- PS of task prior to AST
   SP+02 -- PC of task prior to AST
   SP+00 -- Task's directive status word

4. When an I/O request is queued, an AST service entry point may be specified in the macro. If specified, an AST will occur upon completion of the I/O 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 prior to AST
   SP+04 -- PC of task prior to AST
   SP+02 -- Task's Directive Status Word
   SP+00 -- Address of I/O status block for I/O request (or zero if none specified).

5. When a MARK TIME directive is issued, an AST service entry point may be specified in the macro. If specified, when the indicated time interval has elapsed, an AST will occur 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 prior to AST
SP+04 -- PC of task prior to AST
SP+02 -- Task's Directive status Word
SP+00 -- Event Flag number (or zero if none specified)

The following notes describe general characteristics and use of AST's.

1. Two directives, DISABLE AST RECOGNITION and ENABLE AST RECOGNITION, allow AST's to be queued for subsequent execution during critical sections of code that access data bases that are also accessed by AST service routines. If AST's occur while AST recognition is disabled, they are queued (FIFO), and will be processed when AST recognition is enabled.

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

3. If an AST occurs while an SST is being processed, the SST service routine execution will not be distinguished from task execution, and will be 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 service routine or another task.

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 service routine or another task.

6. If an AST occurs while the related task is in execution, the task is interrupted for the execution of the AST service routine.

7. If an AST occurs for a checkpointed task, the AST is queued (FIFO) and effected when the task is returned to direct competition for processor resources.

8. AST memory is allocated when the AST is specified. Thus, no AST lacks memory for data storage at the time the AST occurs.

AST SERVICE EXIT (only s-form supplied) **ASTX$S**

This directive instructs the system to terminate execution of an Asynchronous System Trap service routine.

If another AST is queued, and AST's are not disabled, then the next AST is immediately effected. Otherwise, the task's pre-AST state is restored.

Fortran Call:

Neither the FORTRAN IV language nor the ISA standard permits direct linking to system trapping mechanisms, therefore, this directive is not available to Fortran tasks.

Macro Call:

ASTX$S [err]

err = Error routine address

Macro Expansion:

```
ASTX$S    ERR
MOV       (PC)+,-(SP)  ;PUSH DPB ONTO THE STACK
.BYTE     115.,1       ;ASTX$S MACRO DIC, DPB SIZE=1 WORD
EMT       377          ;TRAP TO THE EXECUTIVE
BCC       .+6          ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR       PC,ERR       ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC	--	Successful completion
IE.AST	--	Directive was not issued from an AST service routine
IE.ADP	--	Part of the DPB or stack is out of the issuing task's address space
IE.SDP	--	DIC or DPB size invalid

Notes:

1. When an AST occurs, the Executive, pushes, at minimum, the following information onto the task's 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 -- PS of task prior to AST
   SP+02 -- PC of task prior to AST
   SP+00 -- DSW of task prior to AST

   The task stack must be in this same state when the AST SERVICE EXIT directive is executed.

   In addition to the above parameters, supplemental information is also pushed onto the task stack for certain AST's. For I/O completion the stack contains the address of the I/O Status Block; for MARK TIME, the stack contains the Event Flag Number; for 11/45 FLOATING POINT EXCEPTION, the stack contains the exception code and address.

   These AST parameters must be removed from the task's stack prior to issuing an AST exit directive. The following example shows how this is done when an AST routine is used on I/O completion:

Example:

```
;
; EXAMPLE PROGRAM
;
; LOCAL DATA
;

IOSB:     .BLKW    2          ;I/O STATUS DOUBLEWORD
BUFFER:   .BLKW    30.        ;I/O BUFFER

;
; START OF MAIN PROGRAM
;
```

```
START:  .                                       ;PROCESS DATA
        .
        .
        QIO$S   #IO.WVB,#2,,,#IOSB,#ASTSER,<#BUFFER,#60.,#40>
        .                                       ;PROCESS & WAIT
        .
        EXIT$S                                  ;EXIT TO EXECUTIVE

;
; AST SERVICE ROUTINE
;

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

Notes: (cont.)

2. The task may alter its return state by manipulating the information on its stack prior to executing an AST exit directive. For example, to return to task state at an address other than the PC prior to the AST, the task may simply replace the PC word on the stack. This may be useful in cases where error conditions are discovered in the AST routine, but, this alteration should be exercised with extreme caution since AST service routine bugs are difficult to isolate.

3. This directive requires a 1-word DPB, thus the ASTX$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

DISABLE AST RECOGNITION (only s-form supplied) DSAR$S

This directive instructs the system to disable recognition of Asynchronous System Traps for the issuing task. The AST's are queued as they occur, and will be effected when AST recognition is enabled. There is an implied AST disable whenever an AST service routine is executing. When a task's execution is started, AST recognition is not disabled.

Fortran Call:

```
CALL DSASTR
```

Macro Call:

```
DSAR$S [err]

    err = Error routine address
```

Macro Expansion:

```
DSAR$S    ERR
MOV       (PC)+,-(SP) ;PUSH DPB ONTO THE STACK
.BYTE     99.,1       ;DSAR$S MACRO DIC, DPB SIZE=1
                      ;WORD
EMT       377         ;TRAP TO THE EXECUTIVE
BCC       .+6         ;BRANCH IF DIRECTIVE SUCCESS-
                      ;FUL
JSR       PC,ERR      ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC	-- Successful completion
IE.ITS	-- AST recognition is already disabled
IE.ADP	-- Part of the DPB is out of the issuing task's address space
IE.SDP	-- DIC or DPB size is invalid

Notes:

1. It is only the recognition which is disabled. The AST's are still queued by the system. They are queued FIFO and will occur in that order when AST recognition is re-enabled.

2. This directive requires a 1-word DPB, thus the DSAR$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

3. This Fortran call, as well as ENASTR below, exist solely to control the possible jump to the PWRUP routine (power-up). Fortran is not designed to link to a system's trapping mechanism. The PWRUP routine is strictly controlled by the system. It is the system which both accepts the trap and subsequently dismisses it. The Fortran program is notified by a jump to PWRUP but must use DSASTR and ENASTR to ensure the integrity of Fortran data structures, most importantly the stack, during PWRUP processing.

ENABLE AST RECOGNITION (only s-form supplied) **ENAR$S**

This directive instructs the system to recognize Asynchronous System Traps for the issuing task, i.e., to nullify a DISABLE AST RECOGNITION directive. AST's that have been queued while recognition was disabled are effected at issuance. When a task's execution is started, AST recognition is enabled.

Fortran Call:

```
CALL ENASTR
```

Macro Call:

```
ENAR$S [err]
```

err = Error routine address

Macro Expansion:

```
ENAR$S  ERR
MOV     (PC)+,-(SP) ;PUSH DPB ONTO THE STACK
.BYTE   101.,1      ;ENAR$S MACRO DIC, DPB SIZE=1
                    ;WORD
EMT     377         ;TRAP TO THE EXECUTIVE
BCC     .+6         ;BRANCH IF DIRECTIVE SUCCESS-
                    ;FUL
JSR     PC,ERR      ;OTHERWISE, CALL ROUTINE 'ERR'
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC -- Successful completion
IE.ITS -- AST recognition is not disabled
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. This directive requires a 1-word DPB, thus the ENAR$S form of the macro is recommended since it will always require less space and executes with the same speed as the DIR$ macro form.

SPECIFY FLOATING POINT EXCEPTION AST — SFPA$

This directive instructs the system to record either:

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

2. That floating point exception AST's for the issuing task are no longer desired.

When an AST service routine entry point address is specified, future floating point exception AST's will occur for the issuing task, and control will be transferred to the indicated location whenever a floating point exception AST occurs. When an AST service entry point address is not specified, future floating point exception AST's will not occur until an AST entry point is specified again.

Fortran Call:

Neither the FORTRAN IV language nor the ISA standard permits direct linking to system trapping mechanisms; therefore, this directive is not available to Fortran tasks.

Macro Call:

```
SFPA$     [ast]

ast  =  Ast service routine entry point address (0)
```

Macro Expansion:

```
SFPA$     FLTAST
.BYTE     111.,2          ;SFPA$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     FLTAST          ;ADDRESS OF FLOATING POINT AST
```

Local Symbol Definitions:

```
S.FPAE  -    AST Entry address  (2)
```

DSW Return Codes:

```
IS.SUC   --   Successful completion
IE.UPN   --   Insufficient dynamic memory
IE.ITS   --   AST entry point address is already unspecified
IE.AST   --   Directive was issued from an AST service routine
              or AST's are disabled
IE.ADP   --   Part of the DPB is out of the issuing task's
              address space
IE.SDP   --   DIC or DPB size is invalid
```

Notes:

1. SPECIFY FLOATING POINT EXCEPTION AST requires dynamic memory.

2. Floating point exception AST's are queued when a floating point exception trap occurs. No future floating point exception AST's will be queued for the task until the first one queued has actually been effected.

3. The floating point exception AST service routine is entered with the task stack in the following state:

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 prior to AST
SP+06	PC of task prior to AST
SP+04	DSW of task prior to AST
SP+02	Floating exception code
SP+00	Floating exception address

   The floating exception code and address must be removed from the task's stack before an AST SERVICE EXIT directive is executed.

4. This directive cannot be issued when AST's are disabled or from an AST service routine.

5. This directive applies only to the 11/45 Floating Point Unit.

SPECIFY POWER RECOVERY AST — SPRA$

This directive instructs the system to record either:

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 address is specified, future power recovery AST's will occur for the issuing task, and control will be transferred to the indicated location whenever a power recovery AST occurs. When an AST service entry point address is not specified, future power recovery AST's will not occur until an AST entry point is specified again.

Fortran Call:

To establish an AST:

```
EXTERNAL  sub
CALL PWRUP  (sub)
```

sub = name of a subroutine to be executed upon power recovery. The PWRUP subroutine will effect a

CALL sub (no arguments).

sub is called as a result of a power recovery AST (Asynchronous System Trap), and therefore may be controlled at critical points by using DISABLE and ENABLE AST recognition directives.

To remove an AST:

```
CALL PWRUP
```

Macro Call:

```
SPRA$  [ast]
```

ast = Ast service routine entry point address (0)

Macro Expansion:

```
SPRA$       PWRAST
.BYTE       109.,2      ;SPRA$ MACRO DIC, DPB SIZE=2 WORDS
.WORD       PWRAST      ;ADDRESS OF POWER RECOVERY AST
```

Local Symbol Definitions:

S.PRAE - AST Entry address (2)

DSW Return Codes:

IS.SUC -- Successful completion
IE.UPN -- Insufficient dynamic memory
IE.ITS -- AST entry point address is already unspecified or AST's are disabled
IE.AST -- Directive was issued from an AST service routine
IE.ADP -- Part of the DPB is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. SPECIFY POWER RECOVERY AST requires dynamic memory.

2. Power recovery AST's are queued when the power-up interrupt occurs following a power failure. No future powerfail AST's will be queued for the task until the first one queued has actually been effected.

3. The power fail AST service routine is entered with the task stack in the following state:

   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 PS of task prior to AST
   SP+02 PC of task prior to AST
   SP+00 DSW of task prior to AST

   No trap-dependent parameters accompany a powerfail AST, and thus the AST SERVICE EXIT directive must be executed with the stack in the same state as when the AST was effected.

4. If a power recovery AST entry point is specified by a checkpointable task and the power fails while the task is checkpointed, the AST is not effected or queued. A checkpointable task should disable checkpointing over critical regions where power recovery AST's are essential.

5. This directive cannot be issued when AST's are disabled or from an AST service routine.

SPECIFY RECEIVE AST **SRDA$**

This directive instructs the system to record either:

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

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

When an AST service routine entry point address is specified, future receive AST's will occur for the issuing task, and control will be transferred to the indicated location whenever a receive AST occurs. When an AST service entry point address is not specified, future receive AST's will not occur until an AST entry point is specified again.

Fortran Call:

Neither the FORTRAN IV language nor the ISA standard permits direct linking to system trapping mechanisms; therefore, this directive is not available to Fortran tasks.

Macro Call:

```
SRDA$     [ast]

ast  =  Ast service routine entry point address (0)
```

Macro Expansion:

```
SRDA$     RECAST
.BYTE     107.,2          ;SRDA$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     RECAST          ;ADDRESS OF RECEIVE AST
```

Local Symbol Definitions:

```
S.RDAE  --     Ast Entry address (2)
```

DSW Return codes:

```
IS.SUC  --     Successful completion
IE.UPN  --     Insufficient dynamic memory
IE.ITS  --     AST entry point address is already unspecified
IE.AST  --     Directive was issued from an AST service routine
               or AST's are disabled.
IE.ADP  --     Part of the DPB is out of the issuing task's
               address space
IE.SDP  --     DIC or DPB size is invalid
```

Notes:

1. SPECIFY RECEIVE AST requires dynamic memory.

2. Receive AST's are queued when a message is sent to the task. No future receive AST's will be queued for the task until the first one queued has actually been effected.

3. The receive AST service routine is entered with the task stack in the following state:

   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 PS of task prior to AST
   SP+02 PC of task prior to AST
   SP+00 DSW of task prior to AST

   No trap-dependent parameters accompany a receive AST, and thus the AST SERVICE EXIT directive must be executed with the stack in the same state as when the AST was effected.

4. If a receive AST entry point is specified by a checkpointable task and a message is sent to the task while it is checkpointed, the AST is not effected or queued. A checkpointable task should disable checkpointing over critical regions where receive AST's are essential.

5. This directive cannot be issued when AST's are disabled or from an AST service routine.

SPECIFY SST VECTOR TABLE FOR DEBUGGING AID — SVDB$

This directive instructs the system to record the address of a table of Synchronous System Trap service routine entry points for use by an intra-task debugging aid (e.g., ODT). If the vector table is to be de-assigned, then the adr and len parameters are omitted from the macro invocation.

Whenever an SST service routine entry is specified in both the table used by the task, and the table used by a debugging aid, the trap occurs for the debugging aid, and not for the task.

Fortran Call:

Neither the FORTRAN IV language not the ISA standard permits direct linking to system trapping mechanisms; therefore, this directive is not available to Fortran tasks.

Macro Call:

```
SVDB$   [adr],[len]

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

The vector table is of the following format:

```
WD.  00 -- Odd address error
WD.  01 -- Memory protect violation
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 a non-RSX EMT instruction
WD.  06 -- Execution of a TRAP instruction
WD.  07 -- PDP-11/40 Floating Point exception
```

Macro Expansion:

```
SVDB$   SSTTBL,4
.BYTE   103.,3          ;SVDB$ MACRO DIC, DPB SIZE=3 WORDS
.WORD   SSTTBL          ;ADDRESS OF SST TABLE
.WORD   4               ;SST TABLE LENGTH=4 WORDS
```

Local Symbol Definitions:

```
S.VDTA -  Table address (2)
S.VDTL -  Table length (2)
```

DSW Return Codes:

```
IE.SUC  --  Successful completion
IE.ADP  --  Part of the DPB or table is out of the issuing TASK'S address space.
IE.SDP  --  DIC or DPB size is invalid
```

SPECIFY SST VECTOR TABLE FOR TASK — SVTK$

This directive instructs the system to record the address of a table of Synchronous System Trap service routine entry points for use by the issuing task.

If the vector table is to be de-assigned, then the adr and len parameters are omitted from the macro invocation.

Whenever an SST dervice routine entry is specified in both the table used by the task, and the table used by a debugging aid, the trap occurs for the debugging aid, and not for the task.

Fortran Call:

> Neither the FORTRAN IV language nor the ISA standard permits direct linking to system trapping mechanisms; therefore; this directive is not available to Fortran tasks.

Macro Call:

```
SVTK$   [adr],[len]

adr  =  Address of SST Vector table
len  =  Length of (number of entries in) the table in words
```

The vector table is of the following format:

```
WD.00 -- Odd address error
WD.01 -- Memory protect violation
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 a non-RSX EMT instruction
WD.06 -- Execution of a TRAP instruction
WD.07 -- PDP-11/40 floating point exception,
```

Macro Expansion:

```
SVTK$       SSTTBL,4
.BYTE       105.,3          ;SVTK$ MACRO DIC, DPB SIZE=3 WORDS
.WORD       SSTTBL          ;ADDRESS OF SST TABLE
.WORD       4               ;SET TABLE LENGTH=4 WORDS
```

Local Symbol Definitions:

```
S.VTTA - Table address (2)
S.VTTL - Table length (2)
```

DSW Return Codes:

IS.SUC -- Successful completion
IE.ADP-- Part of the DPB or table is out of the issuing task's address space.
IE.SDP -- DIC or DPB size is invalid

2.2.6 I/O Related Directives

ASSIGN LUN **ALUN$**

This directive instructs the system to assign a physical device unit to a Logical Unit Number (LUN). ASSIGN LUN connects a LUN identifier with a physical device. It does not necessarily indicate that the task has possession of the device.

Fortran Call:

```
CALL ASNLUN (lun,dev,unt,[ids])

lun = Integer containing a Logical Unit Number.
dev = Integer containing a device name (format: 1A2).
unt = Integer containing a device unit number.
ids = Integer variable to receive the Direct Status Word.
```

Macro Call:

```
ALUN$ lun,dev,unt

lun = Logical Unit Number
dev = Physical device name (two ASCII characters)
unt = Physical device unit number
```

Macro Expansion:

```
ALUN$   7,TT,0          ;ASSIGN LOGICAL UNIT NUMBER
.BYTE   7,4             ;ALUN$ MACRO DIC, DPB SIZE=4 WORDS
.WORD   7               ;LOGICAL UNIT NUMBER 7
.ASCII  /TT/            ;DEVICE NAME IS TT (TELETYPE)
.WORD   0               ;DEVICE UNIT NUMBER=0
```

Local Symbol Definitions:

```
A.LULU - Logical Unit Number (2)
A.LUNA - Physical device name (2)
A.LUNU - Physical device unit number (2)
```

DSW Return Codes:

```
IS.SUC -- Successful completion
IE.LNL -- LUN usage is interlocked (see note 1)
IE.IDU -- Invalid device and/or unit
IE.ILU -- Invalid Logical Unit Number
IE.ADP -- Part of the DPB is out of the issuing task's
          address space
IE.SDP -- DIC or DPB size is invalid
```

Notes:

1. A return code of IE.LNL means that the LUN may not be reassigned to another device because it is already assigned to a device and a file is currently open on that device for the specified LUN, or that the device is attached to the issuing task.

2. On successful reassignment all I/O requests for the issuing task in the previous device queue are cancelled.

GET LUN INFORMATION — GLUN$

This directive instructs the system to fill a 6-word buffer with information about a physical device unit to which a LUN is assigned. If requests to the physical device unit have been redirected to another unit, the information returned will describe the effective assignment.

Fortran Call:

```
CALL GETLUN (lun,dat,[ids])

  lun  =  Integer containing a logical unit number
  dat  =  6-word integer array to receive LUN information
  ids  =  Directive status
```

Macro Call:

```
GLUN$   lun,buf

  lun  =  Logical unit number
  buf  =  Address of 6-word buffer which will receive the LUN
          information
```

Buffer Format:

```
WD. 00 -- Name of Assigned Device
WD. 01 -- Unit Number of Assigned Device and flags byte
WD. 02 -- First Device Characteristics Word
    Bit 0 -- Record Oriented Device (1=yes) [FD.REC]*
    Bit 1 -- Carriage Control Device (1=yes)[FD.CCL]
    Bit 2 -- Terminal device (1=Yes)[FD.TTY]
    Bit 3 -- Directory Device (1=yes)[FD.DIR]
    Bit 4 -- Single Directory Device (1=yes)[FD.SDI]
    Bit 5 -- Sequential Device (1=yes)[FD.SDG]
    Bits 6-11   Reserved
    Bit 12 -- Pseudo Device (1=yes)
    Bit 13 -- Device Mountable as a
              Communications Channel (1=yes)
    Bit 14 -- Device mountable as a Files-11
              device (1=Yes)
    Bit 15 -- Device mountable (1=yes)
WD. 03 -- Second Device Characteristics Word
WD. 04 -- Third Device Characteristics Word
          (Words 2 and 3 are device driver specific)
WD. 05 -- Standard device buffer size
```

---

* Bits having symbolics associated with them have the symbols shown in square brackets. These symbols may be defined for use by a task via the FCSBT$ macro. See the I/O Operations Reference Manual (DEC-11-OMFSA-A-D).

Macro Expansion:

```
GLUN$   7,LUNBUF
.BYTE   5,3             ;GLUN$ MACRO DIC, DPB SIZE=3 WORDS
.WORD   7               ;LOGICAL UNIT NUMBER 7
.WORD   LUNBUF          ;ADDRESS OF 6-WORD BUFFER
```

Local Symbol Definitions:

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

The following offsets are assigned relative to the start of the LUN information buffer.

G.LUNA - Device name (2)
G.LUNU - Device unit number (1)
G.LUFB - Flags byte* (1)
G.LUCW - Four device characteristics words (8)

DSW Return Codes:

IS.SUC -- Successful completion
IE.ITS -- No data currently queued
IE.ADP -- Part of the DPB or buffer is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

---

* Always returned as 200(8) for RSX-11D compatibility.

GET MCR COMMAND LINE — GMCR$

This directive instructs the system to transfer an 80-byte command line to the issuing Task.

Fortran Call:

```
CALL GETMCR (buf,[ids])

buf = 80-byte array to receive command line
ids = Directive status
```

Macro Call:

```
GMCR$
```

Macro Expansion:

```
GMCR$
.BYTE   127.,41.    ;GMCR$ MACRO DIC, DPB SIZE=41. WORDS
.BLKW   40.         ;80. CHARACTER MCR COMMAND LINE BUFFER
```

Local Symbol Definitions:

G.MCRB = MCR line buffer (80)

DSW Return Codes:

+n	-- Successful completion; n is the number of data bytes transferred (excluding termination character). The termination character is, however, in the buffer.
IE.AST	-- Directive not issued by the last task requested by MCR dispatch
IE.ADP	-- Part of the DPB is out of the issuing task's address space
IE.SDP	-- DIC or DPB size is invalid

Notes:

1. The GMCR$S forms of the macro is not supplied since the DPB receives the actual command line.

RECEIVE DATA **RCVD$**

This directive instructs the system to dequeue a 13-word data block for the issuing task that has been queued (FIFO) for it via a SEND DATA Directive.

A 2-word sender task name (in RAD50) and the 13-word data block are returned in an indicated 15-word buffer, with the task name in the first two words.

Fortran Call:

```
CALL RECEIV (,buf,[ids])
```

buf = 15-word integer array for received data
ids = Directive status

Macro Call:

```
RCVD$ ,buf
```

buf = Address of 15-word buffer

Macro Expansion:

```
RCVD$   ,DATBUF     ;NOTE: ONE ARGUMENT IS IGNORED
.BYTE   75.,4       ;RCVD$ MACRO DIC, DPB SIZE=4 WORDS
.WORD   0,0         ;SENDER TASK NAME (IGNORED)*
.WORD   DATBUF      ;ADDRESS OF 15.-WORD BUFFER
```

Local Symbol Definitions:

R.VDTN - Task name (4)
R.VDBA - Buffer address (2)

DSW Return Codes:

IS.SUC -- Successful completion
IE.ADP -- Part of the DPB or buffer is out of the issuing task's address space
IE.SDP -- DIC or DPB size is invalid

---

* This field exists for RSX-11D compatibility and is not related to the task name delivered in R.VDTN. The task name in R.VDTN is supplied by the Executive as part of its servicing of the SEND DATA directive.

RECEIVE DATA OR EXIT **RCVX$**

This directive instructs the system to dequeue a 13-word data block for the issuing task that has been queued (FIFO) for it via a SEND DATA Directive.

A 2-word sender task name (in RAD50) and the 13-word data block are returned in an indicated 15-word buffer, with the task name in the first two words.

If no data has been sent, a task exit is effected.

Fortran Call:

```
CALL RECOEX   (,buf,,[ids])

buf  =  15-word integer array for received data
ids  =  Directive status
```

Macro Call:

```
RCVX$     .buf

buf  =  Address of 15-word buffer
```

Macro Expansion:

```
RCVX$     .DATBUF     ;NOTE:  ONE ARGUMENT IS IGNORED
.BYTE     77.,4       ;RCVX$ MACRO DIC, DPB SIZE=4 WORDS
.WORD     0,0         ;SENDER TASK NAME (IGNORED)*
.WORD     DATBUF      ;ADDRESS OF 15.-WORD BUFFER
```

Local Symbol Definitions:

```
R.VXTN  -   Task name (4)
R.VXBA  -   Buffer address (2)
```

DSW Return Codes:

```
IS.SUC   --   Successful completion
IE.ADP   --   Part of the DPB or buffer is out of the
              issuing task's address space
IE.SDP   --   DIC or DPB size is invalid
```

* This field exists for RSX-11D compatibility and is not related to the task name delivered in R.VXTN. The task name in R.VXTN is supplied by the Executive as part of its servicing of the SEND DATA directive.

Notes:

1. If no data has been sent, a task exit is effected.

2. The RECEIVE DATA OR EXIT directive is useful in avoiding a possible race condition that may occur between two tasks communicating via the SEND and RECEIVE directives. The race condition occurs when one task executes a RECEIVE directive and finds its receive queue empty. But before the task can exit the other task sends it a message. Since the first task has already decided to exit, the message is lost since the receiving queue is flushed during task exit. This condition can be avoided by the receiving task executing a RECEIVE DATA OR EXIT directive. If the receive queue is found to be empty a task exit is effected before the other task can send any data and thus no loss of data can occur.

3. If the exit is taken, the Executive frees task resources. In particular:

   1-All attached devices are detached;
   2-The AST queue is flushed;
   3-All open files are closed;
   4-I/O is rundown; and
   5-If the task is not fixed, its partition is freed.

4. If the exit is taken, a significant event is declared.

QUEUE I/O — QIO$

This directive instructs the system to place an I/O request for an indicated physical device unit in a queue of priority-ordered requests for that device unit. The physical device unit is specified as a logical unit number (LUN). A significant event is declared by device drivers 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 and set to the final I/O status when the I/O request is completed. If an AST service routine entry point address is specified, the AST will occur upon I/O completion with the task's WAITFOR mask words, PS, PC, DSW (directive status), and the address of the I/O status block pushed onto the task's stack. The description below deals solely with the Executive directive; the device dependent information can be found in the I/O Drivers Reference Manual (DEC-11-OMDRA-A-D).

Fortran Call:

```
CALL QIO (fnc,lun,[efn],[pri],[isb],[prl],[ids])
```

fun	=	Integer I/O function code
lun	=	Integer logical unit number
efn	=	Integer event flag number
pri	=	Integer priority; ignored, but must be present
prl	=	A 6-word integer array containing device dependent parameters to be placed in parameter words 1 to 6 of the Directive Parameter Block (DPB).
ids	=	Directive status

Macro Call:

```
QIO$    fnc,lun,[efn],[pri],[isb],[ast],[prl]
```

fnc	=	I/O function code (DEC-11-OMDRA-A-D)
lun	=	Logical unit number
efn	=	Event flag number
pri	=	Priority; ignored, but must be present
isb	=	Address of I/O status block
ast	=	Address of AST service routine entry point
prl	=	Parameter list of the form <P1,...,P6>

Macro Expansion:

```
QIO$    IO.RVB,7,52.,,IOSTAT,IOAST,<IOBUFR,512.>
.BYTE   1,12.           ;QIO$ MACRO DIC, DPB SIZE=12.
.WORD   IO.RVB          ;FUNCTION=READ VIRTUAL BLOCK
.WORD   7               ;LOGICAL UNIT NUMBER 7
.BYTE   52.,0           ;EFN 52., PRIORITY IGNORED
.WORD   IOSTAT          ;ADDRESS OF 2-WORD I/O STATUS BLOCK
.WORD   IOAST           ;ADDRESS OF I/O AST ROUTINE
.WORD   IOBUFR          ;ADDRESS OF DATA BUFFER
.WORD   512.            ;BYTE COUNT=512.
.WORD   0               ;ADDITIONAL PARAMETERS...
.WORD   0               ;...NOT USED IN...
.WORD   0               ;...THIS PARTICULAR...
.WORD   0               ;...INVOCATION OF QUEUE I/O
```

Local Symbol Definitions:

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

DSW Return Codes:

IS.SUC -- Successful completion
IE.UPN -- Insufficient dynamic memory
IE.ULN -- Unassigned LUN
IE.ILU -- Invalid LUN
IE.IEF -- Invalid event flag number (EFN.GT 64 or EFN.LT.0)
IE.ADP -- Part of the DPB or I/O status block is out of the issuing Task's address space
IE.SDP -- DIC or DPB size is invalid

Notes:

1. If an AST entry point address is specified, the AST service routine is entered with the task stack in the following state:

   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 prior to AST
   SP+04 PC of task prior to AST
   SP+02 DSW of task prior to AST
   SP+00 Address of I/O status block or zero if none was specified in the QIO directive.

The address of the I/O status block, which is a trap-dependent parameter, must be removed from the task's stack before an exit AST directive is executed.

2. If the directive is rejected, the specified event flag is not guaranteed to be cleared or set. Thus, if the task indiscriminately executes a WAITFOR directive and the QIO directive is rejected, then the task may wait forever. Care should always be taken to insure that the directive was successfully completed.

SEND DATA **SDAT$**

This directive instructs the system to declare a significant event and to queue (FIFO) a 13-word block of data for a task to receive. When an event flag is specified, the indicated event flag is set; a significant event is always declared.

Fortran Call:

```
CALL SEND (tsk,buf,[efn],[ids])

  tsk = Task name
  buf = 13-word integer array of data to be sent
  efn = Event flag number
  ids = Directive status
```

Macro Call:

```
SDAT$ tsk,buf,[efn]

  tsk = Receiver task name
  buf = Address of 13-word data buffer
  efn = Event flag number
```

Macro Expansion:

```
SDAT$   ALPHA,DATBUF,52.
.BYTE   71.,5           ;SDAT$ MACRO DIC, DPB SIZE=5 WORDS
.RAD50  /ALPHA/         ;RECEIVER TASK NAME
.WORD   DATBUF          ;ADDRESS OF 13.-WORD BUFFER
.WORD   52.             ;EVENT FLAG NUMBER 52.
```

Local Symbol Definitons:

```
S.DATN - Task name (4)
S.DABA - Buffer address (2)
S.DAEF - Event flag number (2)
```

DSW Return Codes:

```
IS.SUC  -- Successful completion
IE.INS  -- Receiver Task is not installed
IE.UPN  -- Insufficient dynamic memory
IE.IEF  -- Invalid event flag number (EFN.GT.64 or EFN.LT.0)
IE.ADP  -- Part of the DPB or data block is out of the issuing
           task's address space
IE.SDP  -- DIC or DPB size is invalid
```

Notes:

1. SEND DATA requires dynamic memory.

APPENDIX A

DIRECTIVE SUMMARY - ALPHABETIC ORDER

ABORT TASK **ABRT$**

Fortran Call:

CALL ABORT (tsk,[ids])

ABRT$ tsk

Macro Call:

tsk = Task name

ASSIGN LUN **ALUN$**

Fortran Call:

CALL ASNLUN (lun,dev,unt,[ids])

lun = Integer containing a Logical Unit Number.
dev = Integer containing a device name (format 1A2).
unt = Integer containing a device unit number.
ids = Integer variable to receive the Directive Status Word.

Macro Call:

ALUN$ lun,dev,unt

lun = Logical Unit Number
dev = Physical device name (two characters)
unt = Physical device unit number

AST SERVICE EXIT (only s-form supplied) **ASTX$S**

Fortran Call:

Neither the FORTRAN IV language nor the ISA standard permits direct linking to system trapping mechanics, therefore, this directive is not available to Fortran tasks.

Macro Call:

ASTX$S [err]

err = Error routine address

CLEAR EVENT FLAG **CLEF$**

Fortran Call:

CALL CLFEF (efn,[ids])

efn = Integer containing an event flag number
ids = Directive status

Macro Call:

CLEF$ efn

efn = Event flag number

CANCEL MARK TIME REQUESTS (only s-form supplied) **CMKT$S**

Fortran Call:

CALL CANMT (,[ids])

ids = Directive status

Macro Call:

CMKT$S [,,err]

err = Error routine address

CANCEL TIME BASED INITIATION REQUESTS **CSRQ$**

Fortran Call:

```
CALL CANALL (tsk,[ids])

  tsk  =  Task name
  ids  =  Directive status
```

Macro Call:

```
CSRQ$  tsk

  tsk = Task name
```

DECLARE SIGNIFICANT EVENT (only s-form supplied) **DECL$S**

Fortran Call:

```
CALL DECLAR (,[ids])

  ids  =  Directive status
```

Macro Call:

```
DECL$S  [,err]

  err = Error routine address
```

DISABLE AST RECOGNITION (only s-form supplied) **DSAR$S**

Fortran Call:

```
CALL DSASTR
```

Macro Call:

```
DSAR$S  [err]

  err = Error routine address
```

DISABLE CHECKPOINTING (only s-form supplied) **DSCP$S**

Fortran Call:

```
CALL DISCKP
```

Macro Call:

```
DSCP$S  [err]
```

err = Error routine address

ENABLE AST RECOGNITION (only s-form supplied) **ENAR$S**

Fortran Call:

```
CALL ENASTR
```

Macro Call:

```
ENAR$S  [err]
```

err = Error routine address

ENABLE CHECKPOINTING (only s-form supplied) **ENCP$S**

Fortran Call:

```
CALL ENACKP
```

Macro Call:

```
ENCP$S  [err]
```

err = Error routine address

TASK EXIT (only s-form supplied) **EXIT$S**

Fortran Call:

STOP

Macro Call:

EXIT$S [err]

err = Error routine address

EXITIF **EXIF$**

Fortran Call:

CALL EXITIF (efn,[ids])

efn = Event flag number
ids = Directive status

Macro Call:

EXIF$ efn

efn = Event flag number

GET LUN INFORMATION **GLUN$**

Fortran Call:

CALL GETLUN (lun,dat,[ids])

lun = Integer containing a logical unit number
dat = 6-word integer array to receive LUN information
ids = Directive status

Macro Call:

GLUN$ lun,buf

lun = Logical unit number
buf = Address of 6-word buffer which will receive the LUN information

GET MCR COMMAND LINE **GMCR$**

Fortran Call:

CALL GETMCR (buf,[ids])

buf = 80-byte array to receive command line
ids = Directive status

Macro Call:

GMCR$

GET PARTITION PARAMETERS **GPRT$**

Fortran Call:

CALL GETPAR ([prt],buf,[ids])

prt = a two word RADIX-50 partition name
buf = a 3-word integer array to receive partition parameters
ids = Directive status

Macro Call:

GPRT$ [prt],buf

prt = Partition name
buf = Address of a 3-word buffer

GET SENSE SWITCHES (only s-form supplied) **GSSW$S**

Fortran Call:

CALL READSW (isw)

isw = Integer to receive the console switch settings

Macro Call:

GSSW$S [err]

err = Error routine address

GET TIME PARAMETERS **GTIM$**

Fortran Call:

FORTRAN IV provides several subroutines for obtaining the time in a number of formats. See the RSX-11M FORTRAN IV Reference Manual DEC manual number DEC-11-LFLRA-A-D.

Macro Call:

GTIM$ buf

buf = Address of 8-word buffer

GET TASK PARAMETERS **GTSK$**

Fortran Call:

CALL GETTSK (buf,[ids])

buf = 16-word integer array to receive the task parameters
ids = Directive status

Macro Call:

GTSK$ buf

buf = Address of a 16-word buffer

MARK TIME **MRKT$**

Fortran Call:

```
CALL MARK (efn,tmg,tnt,[ids])

  efn  =  Event flag number
  tmg  =  Integer time interval magnitude
  tnt  =  Integer time interval unit
  ids  =  Directive status
```

The ISA standard call for delaying a task for a specified time interval is also provided:

```
CALL WAIT (tmg,tnt,ids)

  tmg  =  Integer time interval magnitude
  tnt  =  Integer time interval unit
  ids  =  Directive status
```

Macro Call:

```
MRKT$  efn,tmg,tnt,[ast]
  efn = Event flag number
  tmg = Time interval magnitude
  tnt = Time interval unit
  ast = AST entry point address
```

QUEUE I/O **QIO$**

Fortran Call:

```
CALL QIO (fnc,lun,[efn],[pri],[isb],[prl],[ids])

  fun  =  Integer I/O function code
  lun  =  Integer logical unit number
  efn  =  Integer flag number
  pri  =  Integer priority; ignored, but must be present
  isb  =  2-word integer array to receive final I/O status
  prl  =  6-word integer array containing device dependent
          parameters to be placed in parameter words 1 to 6
          of the Directive Parameter Block (DPB).
  ids  =  Directive status
```

Macro Call:

QIO$ fnc,lun,[efn],[pri],[isb],[ast],[pri]

fnc = I/O function code (see DEC-11-OMFSA-A-D)
lun = Logical unit number
efn = Event flag number
pri = Priority; ignored, but must be present
isb = Address of I/O status block
ast = Address of AST service routine entry point
pri = Parameter list of the form <P1,...,P6>

RECEIVE DATA **RCVD$**

Fortran Call:

CALL RECEIV (,buf,[ids])

buf = 15-word integer array for received data
ids = Directive status

Macro Call:

RCVD$,buf

buf = Address of 15-word buffer

RECEIVE DATA OR EXIT **RCVX$**

Fortran Call:

CALL RECDEX (,buf,,[ids])

buf = 15-word integer array for received data
ids = Directive status

Macro Call:

RCVX$,buf

buf = Address of 15-word buffer

READ ALL EVENT FLAGS **RDAF$**

Fortran Call:

Only a single event flag may be read by a FORTRAN IV task. The call is:

```
CALL READEF (efn,[ids])
```

efn = Event flag number
ids = Directive status

Macro Call:

```
RDAF$ buf
```

buf = Address of 4-word buffer

REQUEST **RQST$**

Fortran Call:

```
CALL REQUES (tsk,[opt],[ids])
```

tsk = Task name
opt = 4-word integer array
- opt(1) = Partition name first half; ignored, but must be present
- opt(2) = partition name second half; ignored, but must be present
- opt(3) = priority; ignored, but must be present
- opt(4) = user identification code

ids = Directive status

Macro Call:

```
RQST$ tsk,[prt],[pri],[ugc],[uoc]
```

tsk = Task name
prt = Partition name; ignored, but must be present
ugc = UIC group code
uoc = UIC owner code

RESUME **RSUM$**

Fortran Call:

```
CALL RESUME (tsk,[ids])

  tsk  =  Task name
  ids  =  Directive status
```

Macro Call:

```
RSUM$  tsk

  tsk = Task name
```

RUN **RUN$**

Fortran Calls:

```
CALL RUN (tsk,[opt],[smg],[snt],[rmg],[rnt],[ids])

  tsk  =  Task name
  opt  =  4-word integer array
            opt(1)  =  Partition name first half; ignored, but must be present
            opt(2)  =  Partition name second half; ignored, but must be present
            opt(3)  =  Priority; ignored, but must be present
            opt(4)  =  User identification code
  smg  =  Schedule delta magnitude
  snt  =  Schedule delta unit
  rmg  =  Reschedule interval magnitude
  rnt  =  Reschedule interval unit
  ids  =  Directive status
```

The ISA standard call for initiating a task is also provided:

```
CALL START (tsk,smg,snt,ids)

  tsk = Taskname
  smg = Schedule delta magnitude
  snt = Schedule delta unit
  ids = Directive status
```

Macro Call:

RUN$ tsk,[prt],[pri],[ugc],[uoc],[smg],[snt],[rmg],[rnt]

tsk = Task name
prt = Partition name; ignored, but must be present
pri = Priority; ignored, but must be present
ugc = UIC group code
uoc = UIC owner code
smg = Schedule delta magnitude
snt = Schedule delta unit
rmg = Reschedule interval magnitude
rnt = Reschedule interval unit

SEND DATA **SDAT$**

Fortran Call:

CALL SEND (tsk,buf,[efn],[ids])

tsk = Task name
buf = 13-word integer array of data to be sent
efn = Event flag number
ids = Directive status

Macro Call:

SDAT$ tsk,buf,[efn]

tsk = Receiver task name
buf = Address of 13-word data buffer
efn = Event flag number

SET EVENT FLAG **SETF$**

Fortran Call:

CALL SETEF (efn,[ids])

efn = Event flag number
ids = Directive status

Macro Call:

SETF$ efn

efn = Event flag number

SUSPEND (only s-form supplied) **SPND$S**

Fortran Call:

CALL SUSPND

Macro Call:

SPND$S [err]

err = Error routine address

SPECIFY FLOATING POINT EXCEPTION AST **SFPA$**

Fortran Call:

Not supported.

Macro Call:

SFPA$ [ast]
ast = Ast service routine entry point address

SPECIFY POWER RECOVERY AST **SPRA$**

Fortran Call:

CALL PWRUP (sub)

sub = name of a subroutine to be executed upon power recovery. The PWRUP subroutine will effect a

CALL sub (no arguments)

sub is called as a result of a power recovery AST (Asynchronous System Trap), and therefore may be controlled at critical points by using the DSABLE and ENABLE AST recognition directives.

Macro Call:

SPRA$ [ast]

ast = Ast service routine entry point address

SPECIFY RECEIVE AST **SRDA$**

Fortran Call:

Not supported.

Macro Call:

SRDA$ [ast]
ast = Ast service routine entry point address

SPECIFY SST VECTOR TABLE FOR DEBUGGING AID **SVDB$**

Fortran Call:

Not supported.

Macro Call:

SVDB$ [adr],[len]

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

SPECIFY SST VECTOR TABLE FOR TASK **SVTK$**

Fortran Call:

Not supported.

Macro Call:

SVTK$ [adr],[len]

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

WAIT FOR SIGNIFICANT EVENT (only s-form supplied) **WSIG$S**

Fortran Call:

CALL WFSNE

Macro Call:

WSIG$S [err]

err = Error routine address

WAIT FOR LOGICAL "OR" OF EVENT FLAGS **WTLO$**

Fortran Call:

CALL WFLOR (efn1,efn2,...efnn)

efn = LIST of event flag numbers is taken as the set of flags to be specified in the directive.

Macro Call:

WTLO$ grp,msk

grp = Desired group of event flags
msk = A 16 bit octal mask word

WAIT FOR SINGLE EVENT FLAG **WTSE$**

Fortran Call:

```
CALL WAITFR (efn,[ids])

  efn  =  Event flag number
  ids  =  Directive status
```

Macro Call:

```
WTSE$  efn

  efn = Event flag number
```

APPENDIX B

STANDARD ERROR CODES

The symbol definitions below are the directive status codes that are returned by the RSX-11M executive. To include these definitions in a MACRO-11 program the following coding sequence is used:

```
        .MCALL  DRERR$
        DRERR$


;
; STANDARD ERROR CODES RETURNED BY DIRECTIVES IN THE DIRECTIVE
; STATUS WORD
        IS.CLR  +00     EVENT FLAG WAS CLEAR
        IS.SUC  +01     OPERATION COMPLETE, SUCCESS
        IS.SET  +02     EVENT FLAG WAS SET
;
;
;
        IE.UPN  -01.    INSUFFICIENT DYNAMIC STORAGE
        IE.INS  -02.    SPECIFIED TASK NOT INSTALLED
        IE.ULN  -05.    UN-ASSIGNED LUN
        IE.ACT  -07.    TASK NOT ACTIVE
        IE.ITS  -08.    DIRECTIVE INCONSISTENT WITH TASK STATE
        IE.CKP  -10.    ISSUING TASK NOT CHECKPOINTABLE
;
;
;
        IE.AST  -80.    DIRECTIVE ISSUED/NOT ISSUED FROM AST
        IE.LNL  -90.    LUN LOCKED IN USE
        IE.IDU  -92.    INVALID DEVICE OR UNIT
        IE.ITI  -93.    INVALID TIME PARAMETERS
        IE.ILU  -96.    INVALID LUN
        IE.IEF  -97.    INVALID EVENT (.GT.64.)
        IE.ADP  -98.    PART OF DPB OUT OF USER'S SPACE
        IE.SDP  -99.    DIC OR DPB SIZE INVALID
```

INDEX

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes software newsletters for the various DIGITAL products. Newsletters are published monthly, and keep the user informed about customer software problems and solutions, new software products, documentation corrections, as well as programming notes and techniques.

There are two similar levels of service:

- The Software Dispatch
- The Digital Software News

The Software Dispatch is part of the Software Maintenance Service. This service applies to the following software products:

PDP-9/15
RSX-11D
DOS/BATCH
RSTS-E
DECsystem-10

A Digital Software News for the PDP-11 and a Digital Software News for the PDP-8/12 are available to any customer who has purchased PDP-11 or PDP-8/12 software.

A collection of existing problems and solutions for a given software system is published periodically. A customer receives this publication with his initial software kit with the delivery of his system. This collection would be either a Software Dispatch Review or Software Performance Summary depending on the system ordered.

A mailing list of users who receive software newsletters is also maintained by Software Communications. Users must sign-up for the newsletter they desire. This can be done by either completing the form supplied with the Review or Summary or by writing to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to DIGITAL's software should be reported as follows:

North and South American Submitters:

Upon completion of Software Performance Report (SPR) form remove last copy and send remainder to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

The acknowledgement copy will be returned along with a blank SPR form upon receipt. The acknowledgement will contain a DIGITAL assigned SPR number. The SPR number or the preprinted number should be referenced in any future correspondence. Additional SPR forms may be obtained from the above address.

All International Submitters:

Upon completion of the SPR form, reserve the last copy and send the remainder to the SPR Center in the nearest DIGITAL office. SPR forms are also available from our SPR Centers.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computers Users Society, maintains a user exchange center for user-written programs and technical application information. The Library contains approximately 1,900 programs for all DIGITAL computer lines. Executive routines, editors, debuggers, special functions, games, maintenance and various other classes of programs are available.

DECUS Program Library Catalogs are routinely updated and contain lists and abstracts of all programs according to computer line:

- PDP-8, FOCAL-8, BASIC-8, PDP-12
- PDP-7/9, 9, 15
- PDP-11, RSTS-11
- PDP-6/10, 10

Forms and information on acquiring and submitting programs to the DECUS Library may be obtained from the DECUS office.

In addition to the catalogs, DECUS also publishes the following:

DECUSCOPE	-The Society's technical newsletter, published bi-monthly, aimed at facilitating the interchange of technical information among users of DIGITAL computers and at disseminating news items concerning the Society. Circulation reached 19,000 in May, 1974.
PROCEEDINGS OF THE DIGITAL EQUIPMENT USERS SOCIETY	-Contains technical papers presented at DECUS Symposia held twice a year in the United States, once a year in Europe, Australia, and Canada.
MINUTES OF THE DECsystem-10 SESSIONS	-A report of the DECsystem-10 sessions held at the two United States DECUS Symposia.
COPY-N-Mail	-A monthly mailed communique among DECsystem-10 users.
LUG/SIG	-Mailing of Local User Group (LUG) and Special Interest Group (SIG) communique, aimed at providing closer communication among users of a specific product or application.

Further information on the DECUS Library, publications, and other DECUS activities is available from the DECUS offices listed below:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS EUROPE
Digital Equipment Corp. International
(Europe)
P.O. Box 340
1211 Geneva 26
Switzerland

RSX-11M Executive Service Reference Manual
DEC-11-OMERA-A-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 (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

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 do not 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:

Software Communications
P. O. Box F
Maynard, Massachusetts 01754