IAS System Directives Reference Manual

Order Number: AA–H002C–TC

This document describes the system directives that allow experienced MACRO-11 and FORTRAN programmers to use IAS Executive services to control the execution and interaction of tasks.

Operating System Version: IAS Version 3.4

May 1990

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

This document was prepared using VAX DOCUMENT, Version 1.2

Contents

FIGURES

TABLES

Preface

Purpose of the Manual

The *IAS System Directives Reference Manual* describes the system directives that allow experienced MACRO-11 and FORTRAN programmers to use IAS Executive services to control the execution and interaction of tasks.

Document Structure

Chapter 1 defines the system directives and describes their use in both MACRO-11 and FORTRAN programs.

Chapter 2 introduces the concept of extended logical address space and describes the associated memory management directives.

Chapter 3 contains a brief summary of all directives, grouped according to category.

Chapter 4 contains detailed descriptions of each directive, arranged alphabetically according to macro call.

Appendix A lists the standard Error Returns of the Directive Status Word (DSW).

Associated Documents

The following manuals are prerequisite sources of information for readers of this manual:

- *IAS Task Builder Reference Manual*
- *IAS Executive Facilities Reference Manual*
- *PDP-11 MACRO-11 Reference Manual*
- *PDP-11 FORTRAN Language Reference Manual*

Other documents related to the contents of this manual are described briefly in the *IAS Master Index and Documentation Directory*. The directory defines the intended readership of each manual in the IAS manual set and provides a brief summary of the contents of each document.

Manual Conventions

Macro calls with corresponding expansions and examples of macro source code are printed in red throughout the manual.

1 Using System Directives

This chapter describes the use of system directives and how they are processed.

1.1 Introduction

A system directive is a request from a task to the Executive to perform an indicated operation. The programmer uses the directives to control the execution and interaction of tasks. The MACRO-11 programmer usually invokes directives through macros defined in the system macro library. The FORTRAN programmer invokes system directives through calls to subroutines contained in the system object module library.

System directives enable tasks to perform the following functions:

1 Obtain task and system information

2 Measure time intervals

3 Perform I/O functions

4 Communicate with other tasks

5 Manipulate a task's logical and virtual address space

6 Suspend and resume execution, and

7 Exit

Directives are implemented by means of the EMT (Emulator Trap) 377 instruction. EMT 0 through EMT 376 are considered to be non-IAS EMT synchronous system traps. The Executive aborts the task unless the task specifies that it wants to receive control when such traps occur. Note that IAS reserves EMT 370 and above for possible use as special system traps in the future.

To issue system directives, a MACRO-11 programmer uses the calls supplied in the system macro library for directive calls, rather than hand-coding calls to directives. Then the programmer only needs to re-assemble the program to incorporate any changes in the directive specifications.

Section 1.2, 1.3, and 1.4 pertain to all users. Section 1.5 describes the use of macros. Section 1.6 describes the use of FORTRAN subroutine calls. Programmers using another supported language must refer to the appropriate language reference manual.

1.2 Directive Processing

Directive processing consists of the following parts:

1 The user task issues a directive (by issuing an EMT 377). The directive identifier and the directive parameters must be in a Directive Parameter Block (DPB). The DPB itself can be either on the user task's stack or in a user task's data section.

2 The Executive traps the instruction and checks to see if it is an EMT 377 instruction. If so, control transfers to the directive processor and the Executive processes the directive and returns to the user task with the directive status information. If the EMT is not an EMT 377, the Executive determines whether the user task is capable of handling the trap. If so, the task's SST service routine is entered. If not, the task is aborted.

The Executive preserves all task registers (R0-R5) when a task issues a directive. The user task issues an EMT 377 with the address of a Directive Parameter Block, or a DPB itself, on the top of its stack. When the stack contains a DPB pointer (address), the pointer is removed (popped) when the directive is processed. In this case, the DPB does not alter when the directive is processed. When the stack contains a DPB, the entire DPB is removed as the directive is processed.

The first word of each DPB contains a Directive Identification Code (DIC) byte, and a DPB size byte. The DIC indicates which directive is to be performed; the size byte indicates the DPB length in words. The DIC is in the low-order byte of the word, and the size is in the high-order byte.

The DIC is always odd; thus the Executive can determine whether the word on the top of the stack (before EMT 377 was issued) was the address of the DPB (even word) or the first word of the DPB (odd word).

With the exception of the AST SERVICE EXIT, EXIT, EXITIF, RECEIVE DATA OR EXIT, and certain RECEIVE BY REFERENCE directives, the Executive returns control to the instruction following the EMT. The exceptional RECEIVE BY REFERENCE directives are those that specify the receive-or-exit option. The Executive also clears or sets the carry condition code in the Processor Status word (PS) to indicate acceptance or rejection, respectively, of the directive. Further, the Directive Status Word (DSW) is set to indicate a more specific cause for acceptance or rejection of the directive. The DSW usually has a value of IS.SUC (+1) for acceptance and a range of negative values for rejection (exceptions are success return codes for the directives CLEF$, SETF$, GPRT$, among others). IAS defines the DSW values symbolically, using mnemonics that reflect either successful completion or the cause of an error (see Section 1.4). The ISA FORTRAN calls CALL START and CALL WAIT are exceptions; ISA requires positive numeric error codes. The detailed return values are listed with each directive.

In the case of successful EXIT directives, the Executive does not return control to the task. If an EXIT directive fails, control returns with an error status in the DSW. On EXIT for any reason (including task abort), the Executive frees task resources as follows:

1 Detaches all attached devices.

2 Flushes the Asynchronous System Trap (AST) queue.

3 Flushes the clock queue for outstanding Mark Time requests for the task.

4 Flushes the receiver queue (unless you have built the task with the 'do not flush receive queues' attribute).

5 Closes and locks all open files as appropriate.

6 Flushes I/O queues.

7 Detaches all attached regions.

8 Frees task's memory, if the task is not fixed.

If the Executive rejects a directive, it usually does not clear or set any specified event flag. Thus, the task can wait forever if it indiscriminately executes a WAITFOR directive corresponding to a previously issued MARK TIME directive that the Executive has rejected. You must always take care to determine whether a directive has successfully completed.

1.3 Conventions

The following conventions and assumptions are standard for all directives:

1 Unless a decimal point follows a number (.), the system assumes the number to be octal.

In FORTRAN programs, use INTEGER*2 type unless the directive description states otherwise.

2 In MACRO-11 programs, task and partition names can be from 1 to 6 characters long and are represented as two words in Radix-50 form.

In FORTRAN programs, specify task and partition names by a variable of type REAL (single precision) that contains the task or partition name in Radix-50 form. To establish Radix-50 representation, either use the DATA statement at compile time, or use the IRAD50 subprogram or RAD50 function at run time.

3 Device names are 2 characters long and represented by one word in ASCII code.

4 In the directive descriptions, square brackets ([]) enclose optional parameters or arguments. To omit optional items, either use an empty (null) field in the parameter list, or omit a trailing optional parameter.

5 Logical Unit Numbers (LUNs) can range from 1 to 255(10).

6 Event flag numbers range from 1 to 64(10). Numbers from 1 to 32(10) denote local flags. Numbers from 33 to 64 denote global flags, common to all tasks.

1.4 Error Returns

Directive rejections are divided into two classes: those where a programmed recovery is likely, and those where it is unlikely. The error code, always negative, returns in the DSW. The DSW is located at symbolic address $DSW. Rejections with expected programmed recoveries have values ranging from -1 to -19. Error codes indicating errors for which programmed recoveries are not feasible are in the range -20 to -99. The Task Builder resolves the address of $DSW. Users addressing the DSW with a physical address are not guaranteed compatibility with RSX-11M or future IAS releases.

The individual directive descriptions contain lists of the error codes that the system can return. The symbols have the form IS.xxx for success and IE.xxx for error. Use these symbols for testing the DSW in user programs.

1.5 Using the Directive Macros

Issuing a directive requires supplying the system with a directive code and parameters (called the Directive Parameter Block) and issuing an EMT 377 instruction.

Formation of the Directive Parameter Block (DPB) can be done in two ways:

1 Dynamically on the stack at run time.

2 At assembly time.

The first method requires the use of the $S form of the directive and always generates the EMT 377 code (see Section "$S Form").

The second method requires that the programmer create the DPB as a block of data, generate code which supplies the DPB address to the Executive, and issues an EMT 377 when the directive is required to execute. This can be done in two ways:

1 Forming the DPB and the code using separate macros. The DPB is formed by using the $ form of the appropriate macro (see Section "$ Form"). The program then invokes the directive by means of a DIR$ macro (see Section 1.5.2) that specifies the address of the DPB.

2 Forming the DPB and code using a single macro. This is done by using the $C form of the macro (see Section "$C Form"). Because the DPB is generated in a separate PSECT (program section), the code and DPB are physically separated at link time.

Figure 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

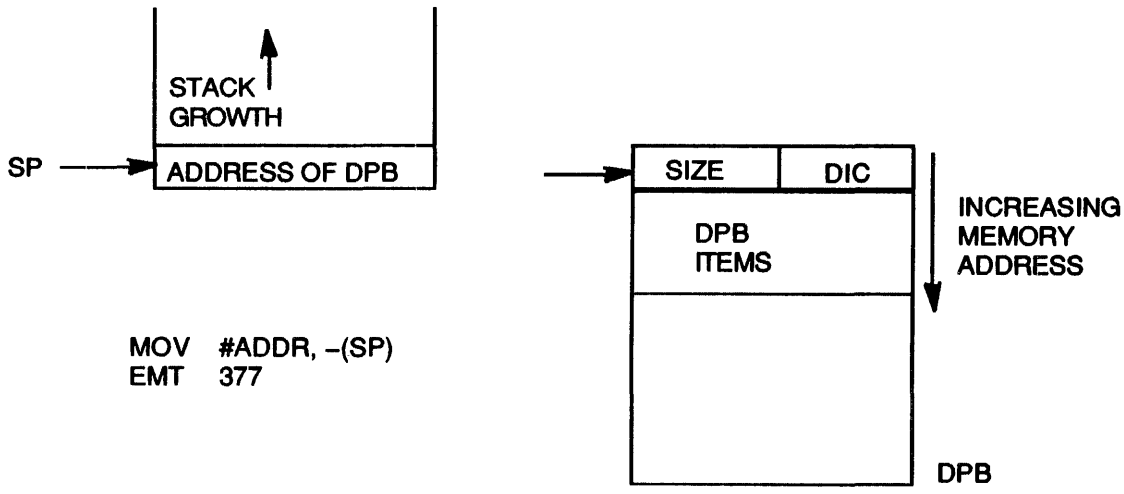
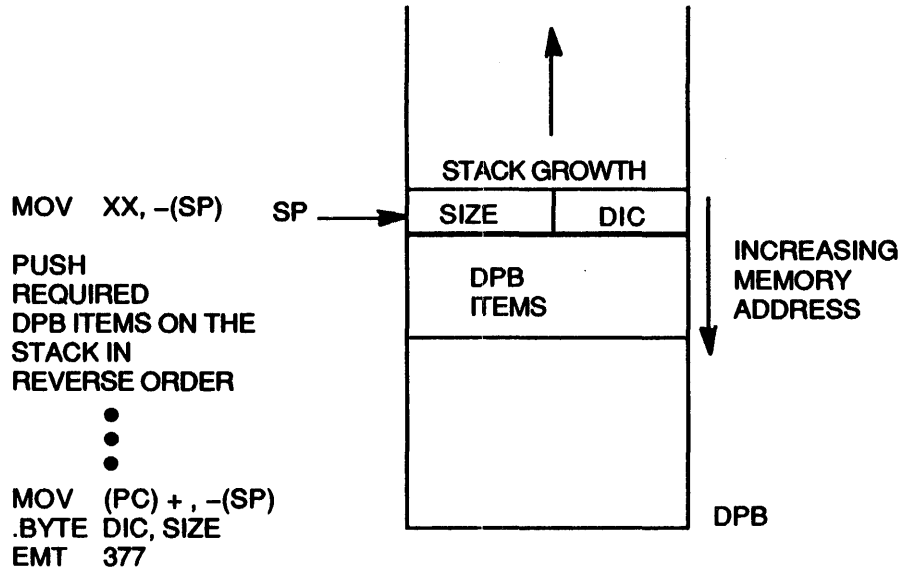


Figure 1–2 Directive Parameter Block (DPB) on the Stack



1.5.1 Macro Name Conventions

You issue directives by including appropriate macro calls in the program. The System Macro Library contains the macros that generate system directives. To make the macros available to a program, the programmer issues the .MCALL directive. The .MCALL arguments are all the directive macros used in the program. For example:

```
;
;CALLING DIRECTIVES FROM THE SYSTEM MACRO LIBRARY
;AND INVOKING THEM
;
  .MCALL MRKT$S,WTSE$S
    .
    .
  Additional .MCALLs or code
     .
     .
  MRKT$S #1,#1,#2,,ERR ;MARK TIME FOR 1 SECOND
  WTSE$S #1     ;WAIT FOR MARK TIME TO COMPLETE
     .
     .
```

Macro names consist of up to four letters, followed by a dollar ($) sign and, optionally, one letter. The optional letter specifies which of three possible expansions of the macro you want.

Examples of each form are given in Section 1.5.5.

$ Form

The $ form (omission of the optional letter) is useful for a directive operation that you want to issue several times from different locations in a non re-entrant program. This form produces only the directive's DPB, and must be issued from a data section of the program. The code for actually executing a directive that is in the $ form is produced by a special macro, DIR$ (described in Section 1.5.3).

Because execution of the directive is separate from the creation of the directive's DPB:

1 You need issue a $ form of a given directive only once (to produce its DPB).

2 You can issue a DIR$ macro associated with a given directive several times, without incurring the overhead of generating a DPB each time you issue it.

  When issuing the $ form of macro call, specifying the generation of a DPB at assembly time, it is assumed that the parameters you need for DPB constructions are valid expressions to be used in MACRO-11 data storage directives (for example, .BYTE, .WORD, .RAD50). See Section 1.5.5.

NOTE: Because space will not have been reserved in the DPB, parameters not specified at assembly time cannot later be specified at run time, despite the offsets to the parameters being defined.

$C Form

Programmers use the $C form when a directive is only issued once, and the program segment does not need to be re-entrant, and the directive arguments are known at assembly time. The $C form eliminates the need to push the DPB (created at assembly time) onto the stack at run time. Other parts of the program, however, cannot access the DPB because the DPB address is unknown. (Note, that in the $C form macro expansion of Section 1.5.5, the DPB address $$$ is redefined by the new value of the assembler's location counter each time you issue an additional $C directive.)

The $C form generates a DPB in a separate PSECT (program section) called $DPB$$. The DPB is followed by a return to the original PSECT, an instruction to push the DPB address onto the stack, and an EMT 377. To ensure that the correct program section is re-entered, you must specify its name in the argument list immediately following the required DPB parameters. If you do not specify the argument, the system assumes the blank (un-named) program section.

The $C form also accepts an optional final argument which specifies a routine to be called if an error occurs during the execution of the directive (see Section 1.5.3).

When issuing the $C form of macro call, specifying the generation of a DPB at assembly time, it is assumed that the parameters required for DPB construction are valid expressions to be used in MACRO-11 data storage directives (for example, .BYTE, .WORD, .RAD50). See Section 1.5.5.

$S Form

If the optional letter is S, the macro produces code to push a DPB onto the stack, followed by an EMT 377. In this case the parameters must be valid source operands to be placed directly in MOV type instructions. Compare Example 3 in Section 1.5.5 with Examples 1 and 2.

Also, where the macro involves a Radix-50 string (for example a task name, a common block name), you specify the appropriate parameter differently in the $S form from the $ or $C form. In this case, it must be the address of a two-word buffer containing the required Radix-50 string. For example, to get details of the common block SYSRES into a user's buffer the programmer can write:

```
  .PSECT DATA,D
BUFFER: .BLKW 8.
DPB: GCOM$ SYSRES,BUFFER

  .PSECT
  DIR$ #DPB

 or

.PSECT DATA,D
BUFFER: .BLKW 8.

  .PSECT
  GCOM$C SYSRES,BUFFER
```

But with the $S form, the Radix-50 parameter must be specified as the address of a two-word buffer containing the Radix-50 string, for example:

```
   .PSECT DATA,D
SYSBUF: .RAD50 /SYSRES/
BUFFER: .BLKW 8.

   .PSECT
   GCOM$S #SYSBUF,#BUFFER
```

The $S form, like the $C, accepts an optional final argument which specifies a routine to be called if an error occurs during the execution of the directive (see Section 1.5.3 below).

Only the $S form produces the DPB at run time. The other two forms produce the DPB at assembly time. Programs that need to be re-entrant must use the $S form. See Section 1.5.5.

Note that you must not use the stack pointer (SP) to address the parameters. Subroutines or macro calls can use SP for temporary storage, thereby destroying the positional relationship between the stack pointer and the parameters.

1.5.2 Predefined DPB

You can predefine a DPB by using the $ and $C forms of a macro. A predefined DPB is very useful if you want to perform the same directive operation several times. Also, with the $ form, you can modify the individual parameters later, for instance when you use the same directive many times with varying parameters.

If you have a predefined DPB, for example if you have used the $ form of the macro, and want to avoid creating another copy of the same DPB, use the DIR$ macro. This macro pushes the DPB address onto the stack and issues an EMT 377.

Macro call:

```
DIR$  adr,err
```

adr and err are optional

- adr - is the address of the DPB. If you do not specify this address, the DPB, or its address must be on the stack. If you use either the $ or $C form of the macro, the address must be on the stack. If you use the $S form, the DPB must be on the stack.)
- err - is the address of the error return. If you do not specify this error, any error simply results in the C bit being set in the Processor Status word.

NOTE: DIR$ is not a "$ form macro," and does not behave as one. There are no variations in the spelling of this macro.

1.5.3 Optional Final Argument

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

Macro call:

```
DIR$   #DPB,ERROR
```

Macro expansion:

```
MOV  #DPB,-(SP)
EMT 377
BCC  .+6
JSR  PC,ERROR
```

This argument is ignored and causes an assembly error if it is used in the $ form, in which the user specifies the generation of the DPB only.

1.5.4 Symbolic Offsets

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

Because the offsets are defined symbolically, you can refer to or modify DPB elements without knowing the offset values. Symbolic offsets also make it unnecessary to rewrite programs to accommodate changes in DPB specifications.

All $ and $C forms of macros that generate DPBs longer than one word, generate local offsets. All informational directives (see Chapter 3, Section 3.2) including the $S form, generate local symbolic offsets for the parameter block returned.

If you invoke either the $ or the $C form of a macro and the symbol $$$GLB has been defined in the program (for example $$$GLB=0), the macro generates the symbolic offsets as global symbols and does not generate the DPB itself. This allows one module to use a DPB defined in another module. $$$GLB has no effect on the expansion of $S macros.

1.5.5 Examples of Macro Calls and Corresponding Expansions

1 The $ form

Call:

MT:

```
MRKT$ 1,5,2,MTRAP ; GENERATE DPB ONLY IN CURRENT PSECT
```

Expansion:

```
.BYTE 23.,5  ; "MARK-TIME" DIC & DPB SIZE
.WORD 1   ; EVENT FLAG NUMBER
.WORD 5   ; TIME INTERVAL MAGNITUDE (5)
.WORD 2   ; TIME INTERVAL UNIT (SECONDS)
.WORD MTRAP  ; AST ENTRY POINT
```

2 The $C form

Call:

```
MRKT$C 1,5,2,MTRAP,PROG1,ERR ;GENERATE DPB IN SEPARATE PSECT
```

Expansion:

```
.PSECT $DPB$$
$$$=.     ; DEFINE TEMPORARY SYMBOL
.BYTE 23.,5  ; "MARK-TIME" DIC & DPB SIZE
.WORD 1   ; EVENT FLAG NUMBER
.WORD 5   ; TIME INTERVAL MAGNITUDE
.WORD 2   ; TIME INTERVAL UNIT
.WORD MTRAP  ; AST ENTRY POINT
.PSECT PROG1  ; RETURN TO THE ORIGINAL PSECT
MOV  #$$$,-(SP) ; PUSH DPB ADDRESS ON STACK
EMT  377   ; TRAP TO THE EXECUTIVE
BCC  .+6   ; BRANCH ON DIRECTIVE ACCEPTANCE
JSR  PC,ERR  ; ELSE, CALL ERROR SERVICE ROUTINE
```

3 The $S form

Call:

```
MRKT$S #1,#5,#2,R2,ERR ;PUSH DPB ON STACK
```

Expansion:

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

4 The DIR$ macro

Call:

```
DIR$  #MT,ERR  ; DPB ALREADY DEFINED. DPB ADDRESS MT.
```

Expansion:

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

1.5.6 Testing the Directive Status Word

It is often convenient to test the C condition code for success. If the C condition code is set, you can test individual error returns. For example:

```
BCC SUCCES    ; SUCCESS IF (C) CLEAR
CMPB @#$DSW, #IE.UPN ; NOT ENOUGH POOL NODES?
BNE NONODE
CMPB @#$DSW, #IE.LUN ; UNASSIGNED LUN?
BNE NOLUN
```

and so on.

The symbol $DSW is resolved by the Task Builder. If you address the DSW with a physical address, you are not guaranteed compatibility with RSX-11M or future IAS releases.

1.6 FORTRAN Subroutines

IAS provides a set of subroutines designed to be used by FORTRAN programs for performing process control and IAS system directive operations. These subroutines fall into three basic groups:

1 Subroutines based on the Instrument Standard of America (ISA), Standard ISA 62.1. These subroutines are included in the subroutine descriptions associated with the macro calls. See Chapter 4 for details.

2 Subroutines designed to use and control specific process control interface devices, supplied by Digital and supported by the IAS operating system.

3 Subroutines for using the AFC-11 A/D Converter, the AD01-D A/D Converter, and the UDC-11 Universal Digital Controller are described in the *IAS FORTRAN Special Subroutines Reference Manual*.

4 Subroutines for performing IAS system directive operations. In general, one subroutine is available for each directive.

All of the subroutines described in this manual can be called by FORTRAN programs compiled by either of the FORTRAN compilers supplied for use with IAS (that is, the FORTRAN IV compiler and the FORTRAN IV PLUS compiler).

Equivalent subroutines can be called by COBOL, BASIC and CORAL programs (see the appropriate language manual).

These subroutines can also be called from programs written in the MACRO-11 assembly language by using PDP-11 FORTRAN calling sequence conventions described in the *IAS/RSX-11 FORTRAN IV User's Guide* and in the *FORTRAN IV-PLUS User's Guide*.

For categories 1 and 3, this manual gives the FORTRAN call after describing the macro call.

1.7 Subroutine Usage

The IAS System Library includes all the subroutines described in this manual. When linking a task for execution, after ascertaining that the specified routine does not already exist (that is, is user defined), the Task Builder automatically searches the System Library.

To use one of these routines, the programmer must include the appropriate CALL statement in the FORTRAN program. The subroutine is selected by the Task Builder and automatically included in the task image during standard linking procedures.

Optional Arguments

Many of the subroutines described in this manual have optional arguments. If you omit an optional argument, you must keep the comma following it if you specify further arguments in the list. You do not need to keep the comma if you omit one or more arguments at the end of the list. For example, the format of a call to SUB could be the following:

```
CALL SUB (AA,[BB],[CC],DD,[EE],[FF])
```

Then the BB, CC, EE and FF arguments can be left unspecified, as in

```
CALL SUB (AA,,,DD,,)
```

or

```
CALL SUB (AA,,,DD)
```

In some cases, a subroutine will use a default value for an unspecified optional argument. Each subroutine description notes such default values.

Task Name Arguments

In these subroutines, task names can be up to six characters long. Characters permitted in a task name are the letters A through Z, the numerals 0 through 9 and the special characters dollar sign ($) and period (.). Task names are stored in Radix-50 representation, which permits up to three characters from the above set to be encoded in one PDP-11 word. (Radix-50 is described in detail in the *IAS FORTRAN IV Language Reference Manual* and the *FORTRAN IV-PLUS User's Guide*.)

In the subroutine calls, a task name is defined as a variable of type REAL which contains the task name in Radix-50 representation. This value can be defined at program compilation time by a DATA statement, which gives the real variable an initial value (a Radix-50 constant).

For example, if a task named TNAME is to be used in a system directive call, the task name could be defined and used as follows:

```
DATA TNAME/5RTNAME/
...
CALL REQUES (TNAME)
```

You can also define task names during execution by using the IRAD50 subroutine or the RAD50 function as described in the *IAS/RSX-11 FORTRAN IV User's Guide* and *FORTRAN IV-PLUS User's Guide*.

Integer Arguments

Many of the arguments used to communicate a value to system subroutines or to receive values from these routines are described as Integer Arguments. All of the subroutines described in this manual assume that integer arguments are, specifically, INTEGER*2 type arguments.

Both the FORTRAN IV and FORTRAN IV-PLUS systems normally treat an integer variable as one PDP-11 storage word, provided that its value is within the range -32768 to +32767. However, if you use the /I4 compile-time option, you must take particular care to ensure that all the integer arguments you use in these subroutines are explicitly specified to be of type INTEGER*2.

GETADR Subroutine

Some subroutines contain an argument in the subroutine call described as an integer array containing some values which are the addresses of other variables or arrays. The FORTRAN language does not provide a means of assigning such an address as a value. This capability is provided by the GETADR subroutine, described below.

Calling Sequence:

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

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

Example:

```
DIMENSION IBUF(80),IOSB(2),IPARAM(6)
.
.
.
CALL GETADR (IPARAM(1),IBUF(1))
IPARAM(2)=80
CALL QIO (IREAD,LUN,IEFLAG,IOSB,IPARAM,IDSW)
.
.
.
```

In this example, CALL GETADR enables the programmer to specify a buffer address in the CALL QIO directive.

1.7.1 Subroutine Categories

This section contains a list of the FORTRAN subroutines associated with system directives (see Chapter 4 of this manual for detailed descriptions).

Some subroutines, notably MARK TIME (CALL MARK), permit both the standard FORTRAN-IV subroutine call and the ISA standard call. Other directives, however, are not available to FORTRAN tasks, [for example, Specify Floating Point Exception AST (SFPA$) and Specify SST Vector Table For Task (SVTK$)].

The subroutines GETADR and MNLOAD are specialized subroutines and are not associated with any one particular directive. These subroutines are described in the *IAS FORTRAN Special Subroutines Reference Manual*.

Table 1–1 Using System Directives

Directive	MACRO Call	FORTRAN Subroutine
ABORT TASK	ABRT$	CALL ABORT
ALTER PRIORITY	ALTP$	CALL ALTPRI
ASSIGN LUN	ALUNS	CALL ASNLUN
AST SERVICE EXIT	ASTX$	Not available
ATTACH REGION	ATRG$	CALL ATRG
CLEAR EVENT FLAG	CLEF$	CALL CLREF
CANCEL MARK TIME REQUESTS	CMKT$	CALL CANMT
CANCEL MARK TIME AST REQUESTS	CMTA$	Not available
CANCEL SCHEDULED REQUESTS	CSRQ$	CALL CANALL (for indicated task) CALL CANOBY (for indicated task but made by another task)
CONNECT TO TASK	CNCT$	CALL CNCT
CREATE ADDRESS WINDOW	CRAW$	CALL CRAW
CREATE REGION	CRRG$	CALL CRRG
DECLARE SIGNIFICANT EVENT	DECL$	CALL DECLAR
DISABLE	DSBL$	CALL DISABL
DISABLE CHECKPOINTING	DSCP$	CALL DISCKP
DETACH REGION	DTRG$	CALL DTRG
ELIMINATE ADDRESS WINDOW	ELAW$	CALL ELAW
EMIT STATUS	EMST$	CALL EMST
ENABLE AST RECOGNITION	ENAR$	CALL ENASTR
EXTEND TASK	EXTK$	CALL EXTTSK
ENABLE CHECKPOINTING	ENCP$	CALL ENACKP
EXECUTE	EXEC$	CALL EXECUT
EXITIF	EXIF$	CALL EXITIF
TASK EXIT	EXIT$	CALL EXIT
EXIT WITH STATUS	EXST$	Not available

Table 1–1 (Cont.) Using System Directives

Directive	MACRO Call	FORTRAN Subroutine
FIX	FIX$	CALL FIXMEM
GET COMMON BLOCK PARAMETERS	GCOM$	CALL GETCMN
GET LUN INFORMATION	GLUN$	CALL GETLUN
GET MAPPING CONTEXT	GMCX$	CALL GMCX
GET MCR COMMAND LINE	GMCR$	CALL GETMCR
GET PARTITION PARAMETERS	GPRT$	CALL GETPAR
GET REGION PARAMETERS	GREG$	CALL GETREG
GET SENSE SWITCHES	GSSW$	CALL READSW CALL SSWTCH
GET TIME PARAMETERS	GTIM$	Several subroutines available (see the *IAS/RSX-11 FORTRAN IV User's Guide* or the *FORTRAN IV-PLUS User's Guide*)
GET TASK PARAMETERS	GTSK$	CALL GETTSK
INHIBIT AST RECOGNITION	IHAR$	CALL INASTR
MAP ADDRESS WINDOW	MAP$	CALL MAP
MARK TIME	MRKT$	CALL MARK CALL WAIT (ISA Standard call)
QUEUE I/O REQUEST	QIO$	CALL QIO
QUEUE I/O REQUEST AND WAIT	QIOW$	CALL QIOW
READ ALL FLAGS	RDAF$	Only a single event flag may be read
READ EVENT FLAG	RDEF$	CALL READEF
RECEIVE BY REFERENCE	RREF$	CALL RREF
REQUEST	RQST$	CALL REQUES
RESUME	RSUM$	CALL RESUME
RUN	RUN$	CALL RUN CALL START (ISA standard call)
SCHEDULE	SCHD$	CALL TRNON (ISA standard call)
SET EVENT FLAG	SETF$	CALL SETEF
SPECIFY FLOATING POINT EXCEPTION AST	SFPA$	Not available
SUSPEND	SPND$	CALL SUSPND
SPECIFY POWER RECOVERY AST	SPRA$	EXTERNAL SUBNAM CALL PWRUP (SUBNAM) (to establish an AST Service routine) CALL PWRUP (to remove an AST Service routine)
SPAWN	SPWN$	CALL SPAWN
SPECIFY RECEIVE DATA AST	SRDA$	Not available
SEND BY REFERENCE	SREF$	CALL SRRF

Table 1–1 (Cont.) Using System Directives

Directive	MACRO Call	FORTRAN Subroutine
SEND BY REFERENCE AND REQUEST OR RESUME	SRFR$	CALL SRFR
SPECIFY RECEIVE BY REFERENCE AST	SRRA$	Not available
SPECIFY SST VECTOR TABLE FOR DEBUGGING AID	SVDB$	Not available
STOP FOR LOGICAL OR OF EVENT FLAGS	STLO$	CALL STOPOR
STOP FOR SINGLE EVENT FLAG	STSE$	CALL STOPFR
STOP	STOP$	CALL STOPTK
SPECIFY SST VECTOR TABLE FOR TASK	SVTK$	Not available
SYNCHRONIZE	SYNC$	CALL SYNC
UNFIX	UFIX$	CALL UNFIX
UNSTOP TASK	USTP$	CALL UNSTOP
UNMAP ADDRESS WINDOW	UMAP$	CALL UNMAP
RECEIVE DATA	VRCD$/RCVD$	CALL VRECEV/RECEIV
RECEIVE DATA OR SUSPEND	VRCS$/RCVS$	CALL VRECSP/RECOSP
RECEIVE DATA OR STOP	VRCT$/RCST$	CALL VRECST /RECOST
RECEIVE DATA OR EXIT	VRCX$/RCVX$	CALL VRECEX/RECOEX
SEND DATA	VSDA$/SDAT$	CALL VSEND/SEND
SEND DATA AND REQUEST OR RESUME RECEIVER	VSDR$/SDRQ$	CALL VSNDRR/SNDROR
WAIT FOR SIGNIFICANT EVENT	WSIG$	CALL WFSNE
WAIT FOR LOGICAL OR OF FLAGS	WTLO$	CALL WFLOR
WAIT FOR SINGLE EVENT FLAG	WTSE$	CALL WAITFR

1.7.2 Error Conditions

Each subroutine provides a means whereby the calling program can determine whether the requested operation was successfully performed or whether an error condition was detected. This is provided by means of an (optional) argument (ids), the value of which is set by the subroutine. In the case of an error, the value will also include the type of error detected.

In addition, two types of error are reported by means of the FORTRAN Object Time System diagnostic messages described in the *IAS/RSX-11 FORTRAN IV Users Guide* or the *FORTRAN IV-PLUS User's Guide*. Both of these errors result in the termination of the task. The error conditions are:

1 "SYSTEM DIRECTIVE: MISSING ARGUMENT(S)" This message indicates one or more necessary arguments were missing from a call to a system directive subroutine. (OTS error number 100).

2 "SYSTEM DIRECTIVE: INVALID EVENT FLAG NUMBER" This message indicates an event flag number in a system directive call was not in the range 1 to 64. (OTS error number 101).

1.8 System Clocks

In many of the macro calls and subroutine calls described, you must specify time intervals and the units to measure as arguments. You can specify time units as hours, minutes, seconds, milliseconds (ISA calls only), or basic counts (ticks) of the system clock.

IAS supports two system clocks for scheduling purposes; the KW11-L and the KW11-P. Although you can specify all time intervals in milliseconds in ISA-standard calls, scheduling precision can be limited by the resolution of the system clock.

The KW11-L is a line frequency clock; maximum resolution is 16.7 milliseconds at 60 Hz line frequency and 20 milliseconds at 50 Hz line frequency.

The KW11-P is a programmable crystal-controlled clock. The clock rate is selected at System Generation, therefore, scheduling accuracy depends on the individual installation's particular clock rate.

1.9 Use of the System Node Pool

Many of the system directives require space to be allocated from the system node pool. This is indicated in the descriptions of the individual directives.

Normally, nodes are charged against the issuing task's node pool allocation, as specified when the task was built or overidden when it was installed. This provides a protection mechanism to stop a single task from picking all the nodes in the pool and hence interfering with the progress of other tasks. Node pool accounting is fully described in the *IAS Executive Facilities Reference Manual.*

It may not be possible to allocate the nodes required for the successful execution of a directive. This can be for one of two reasons:

1 The task has used its entire allocation of nodes. In this case, although there may be sufficient nodes in the system pool, no more nodes can be allocated to this task. The nodes will be available when the task releases some nodes which are currently allocated (for example, because of an I/O completion). Note also that for multiuser tasks, nodes can become available when another invocation of the task releases nodes.

2 The system node pool does not contain enough free nodes. This can occur if the system is temporarily overloaded, for example because a large number of events requiring space from the node pool have occurred simultaneously. In this case, space normally becomes available within a short period of time, as the overload peak passes. This condition only persists if the system is grossly overloaded.

Whatever the reason for failing to obtain nodes, the Executive normally stalls the task and does not allow the directive to complete until sufficient nodes have been obtained. Exceptionally, an error return can be given to the task if:

1 The task was built with the /NOWAIT_FOR_NODES (/-WN) attribute. In this case, the return will be immediate.

2 After a certain period (normally 500 clock ticks) it was impossible to find sufficient nodes.

The Executive normally returns an error status in the DSW of IE.UPN (-01) if insufficient pool nodes are available. A status of IE.UNS (-04) is returned if one of the SEND DATA directives fails to send the data because of a lack of nodes.

2 Memory Management Directives

This chapter describes the concepts of extended logical address space, regions, and virtual address windows. The chapter also introduces the related memory management directives.

2.1 Addressing Capabilities of an IAS Task

An IAS task cannot explicitly refer to a location with an address greater than 32,768 words, commonly called 32K words. The 16-bit word size of the PDP-11 computers imposes this restriction on a task's addressing capability. To avoid limiting the size of a task to its addressing capability, IAS allows tasks to be overlaid. An overlaid task is divided into segments: a single root segment, which is always in memory when the task is running, and any number of segments, which can be loaded into memory as required. Unless an IAS task uses the memory management directives described in this chapter, the combined size of the task segments concurrently in memory cannot exceed 32K words.

When resident task segments cannot exceed 32K words, a task requiring large amounts of data must access data residing on disk. Data is disk-based because of limited memory space and also because transmission of large amounts of data between tasks is only practical by means of disks. A task that has disk overlays or a task that needs to access or transfer large amounts of data incurs a considerable amount of transfer activity over and above that caused by the task's function.

Task execution could obviously be faster if all or a greater portion of the task were simultaneously resident in memory. IAS includes a group of memory management directives that provide the task with this capability. The directives overcome the 32K word addressing restriction by allowing the task to change dynamically the physical memory referred to by a range of addresses. With these directives, a task can increase its execution speed by reducing its disk I/O requirements, at the expense of increased memory requirements.

A task that needs to be overlaid can be built using memory resident overlays. These use the memory management directives, although this is transparent to the user. See the *IAS Task Builder Reference Manual* for details.

2.1.1 Address Mapping

Mapping is the process by which a 16-bit task address (called a Virtual Address) is translated into an 18 or 22-bit physical memory address. (16 bits allow 64K bytes to be addressed, although the actual number of bits required depends on the processor.) The 32K words of task addressable space are divided into eight segments of 4K words. Corresponding with these segments and included within the mapping hardware are eight Active Page Registers (APRs). An APR is selected according to the high-order 3-bits of the virtual address as shown below.

Address Range	APR #
000000-017777	0
020000-037777	1
040000-057777	2
060000-077777	3
100000-117777	4
120000-137777	5
140000-157777	6
160000-177777	7

An APR consists of two component registers:

— a Page Address Register

— a Page Descriptor Register

Page Address Register (PAR)

The PAR contains a physical memory address divided by 64. When six zero bits are appended to the right hand end this can be considered as a 22-bit value. For example:

- PAR #3 contains 2437
- Thus, 2437 corresponds to a Physical Memory Address of 243700.

The physical memory address to which the PAR corresponds is known as the Base Address of the segment and must lie on a 64-byte (32-word) boundary.

You obtain a complete physical memory address from a virtual address by using the high order 3-bits of the virtual address to select the APR (as explained earlier). Once you have obtained the base address, from the associated PAR, the value of the remaining low-order 13-bits of virtual address is added. For example:

- A virtual address of 073432 causes APR #3 to be selected.
- The PAR of APR #3 contains 2437
- Thus the corresponding physical address is: 243700 + 013432 = 257332.

PAGE DESCRIPTOR REGISTER (PDR)

The PDR contains the access rights and the size of the segment mapped from 0 to 4K words in 32-word steps. The size (important for memory protection purposes) means that although a complete 4K words of address space can be mapped per APR, the memory management software can limit access to a smaller number of words (for example, 3K words).

Any attempt to exceed the permitted range gives rise to a memory protect violation trap. This trap will also occur if modification is attempted where write access is not permitted.

NOTE: Although virtual addresses run sequentially from 0 through 177777, there is no restriction on APR contents. Thus, at each 4K word (8K byte) boundary, the task can be mapped to a completely different part of physical memory (either above or below the previous segment). This is the basic feature upon which memory management directives rely.

See the *PDP-11 Processor Handbook* for a full description of memory management hardware and its operations.

2.1.2 Virtual and Logical Address Space

The two concepts defined below, virtual address space and logical address space, provide a basis for understanding the functions performed by the memory management directives:

1 Virtual Address Space - A task's virtual address space corresponds to the 32K word address range imposed by the PDP-11's architecture. The task can divide its virtual address space into segments called virtual address windows.

2 Logical Address Space - A task's logical address space is the total amount of physical memory to which the task has access rights. The task can divide its logical address space into various areas called regions. Each region resides in a contiguous block of memory.

If the capabilities supplied by the IAS memory management directives were not available, a task's virtual address space and logical address space would directly correspond; a single virtual address would always correspond to the same logical location. Both types of address space would have a maximum size of 32K words. However, the ability of the memory management directives to assign or map a range of virtual addresses (a window) to different logical areas (regions) enables you to extend a task's logical address space beyond 32K words.

2.2 Virtual Address Windows

In order to manipulate the mapping of virtual addresses to various logical areas, you must first divide a task's 32K words of virtual address space into parts called virtual address windows. Each window encompasses a contiguous range of virtual addresses, which must begin on a 4K-word boundary (that is, the first address must be a multiple of 4K words). The number of windows defined by a task can vary from 1 to 8. the size of each window can range from a minimum of 32 words to a maximum of 32K minus 32 words.

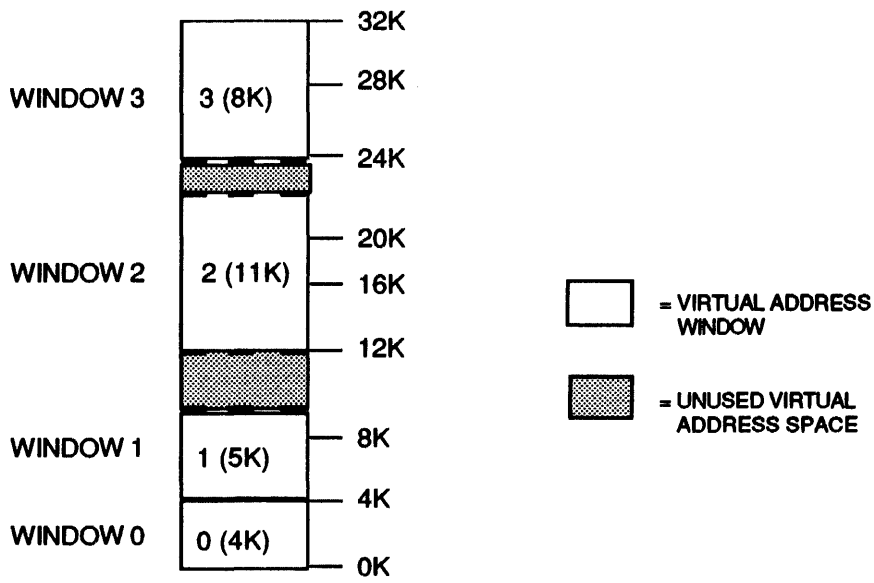
As there are only eight APRs available, (see Section 2.1.1) the number of windows available and the size of each window are interdependent. For each window, each 4K words (or part thereof) requires one APR. All APRs for a window are allocated consecutively to allow contiguous virtual addresses throughout. Therefore, a window 5K words in length requires two consective APRs, in this case the upper 3K words of address space in the second APR are inaccessible to the task.

A window's identification is a number from 0 to 7, which is an index to the window's corresponding window block. The Executive uses window blocks to identify and describe each currently existing window. The address window identified by 0 is the window that always maps the task's root segment. The Task Builder automatically creates window 0, which is mapped by the Executive and cannot be specified in any directive.

Figure 2–1 shows the virtual address space of a task divided into four address windows (windows 0, 1, 2, and 3). The shaded areas indicate portions of the address space that are not included in any window (9K to 12K and 23K to 24K). Addresses that fall within the ranges corresponding to the shaded areas cannot be used.

When a task uses memory management directives, the Executive views the relationship between the task's virtual and logical address space in terms of windows and regions. Unless a virtual address is part of an existing address window, the address does not refer to anything. Similarly, a window can be mapped only to an area that is all or part of an existing region within the task's logical address space.

Figure 2–1 Virtural Address Windows



Once a task has defined the necessary windows and regions, the task can issue memory management directives to perform operations such as the following:

1 Map a window to all or part of a region

2 Unmap a window from one region in order to map it to another region, or

3 Unmap a window from one part of a region in order to map it to another part of the same region.

2.3 Regions

The current window-to-region mapping context determines the part of a task's logical address space that the task can access at one time. A task's logical address space can consist of various types of region:

1 Resident Libraries - These are pure and are never written back to disk. They are loaded from the installed task image file.

2 Common Areas - These are loaded from and swapped to and from the installed task image file.

3 Installed Regions - These are initially loaded from the installed task image file and subsequently swapped to and from the swap files.

4 Dynamic Regions - A dynamic region is a region created dynamically at run time by issuing the memory management directives.

Tasks refer to a region with a region ID returned to the task by the Executive. Region IDs are the attachment descriptor number in the task header and are always a number between 1 and 255. If a task has resident overlays, separate regions are allocated for the task itself, its read/write resident overlays and its read-only overlays. Region IDs for such fixed attachments are allocated in the following order:

1 Read/write resident overlays

2 Read-only resident overlays

3 Libraries and common areas.

Figure 2–2 Logical Address Space

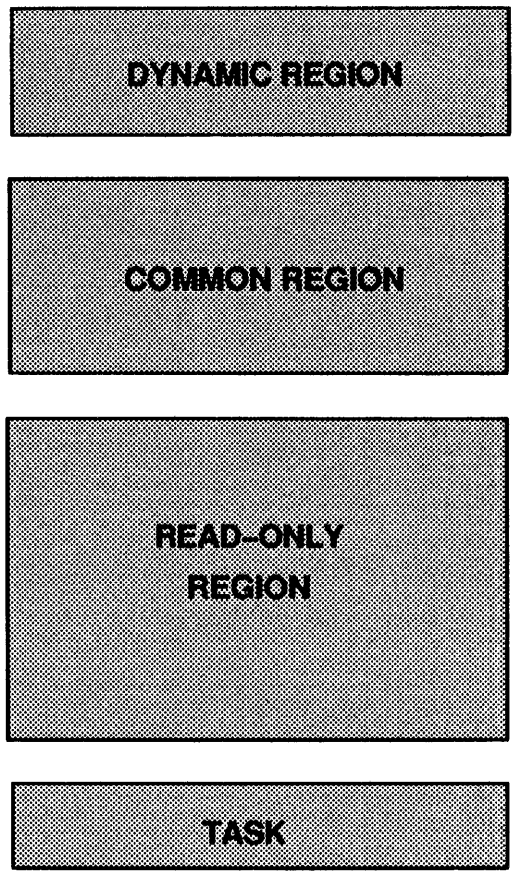
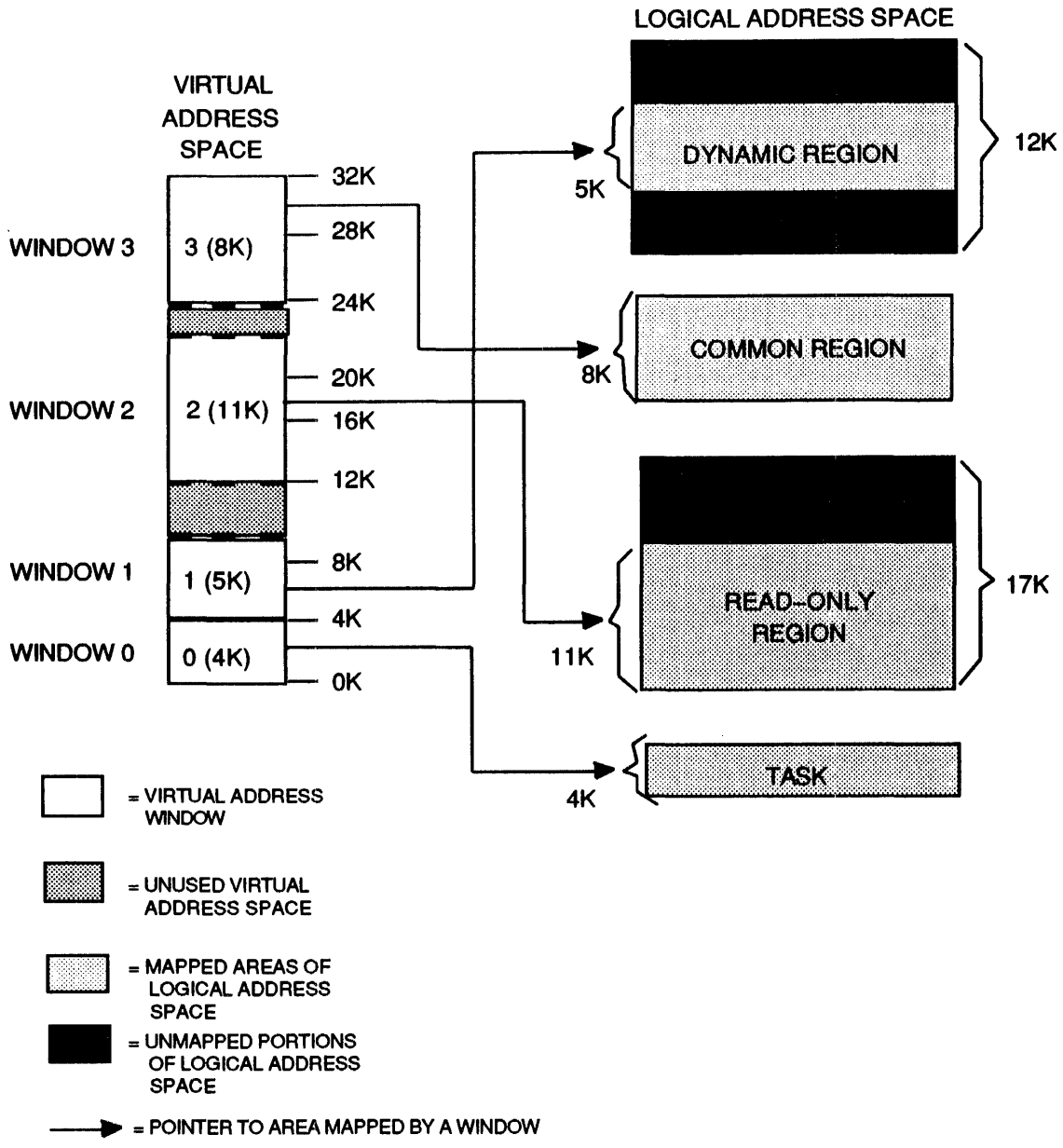


Figure 2–2 shows a sample collection of regions that could comprise a task's logical address space at a given time. (A task's logical address space can enlarge or contract dynamically.) The header and root segment are always part of the task region. Because a region resides in a contiguous area of memory, each region is shown as a separate block.

Figure 2–3 illustrates a possible mapping relationship between the windows and regions shown in the first two figures.

Figure 2–3 Mapping Windows to Regions



2.3.1 Shared Regions

Address mapping not only extends a task's logical address space beyond 32K words; it also allows the space to extend to regions that have not been linked to the task at task build. one result is an increased potential for task interaction by means of shared regions.

For example, a task can create a dynamic region to contain large amounts of data; any number of tasks can then access that data by mapping to the region. Another result is the ability of tasks to use a greater number of common routines; tasks can map to required routines at runtime, rather than link to them at task build.

2.3.2 Attaching to Regions

Attaching is the means by which a region becomes part of a task's logical address space. A task can map only to a region that is part of the task's logical address space. There are three ways to attach a region to a task:

1 All regions that are linked to a task at task build are automatically attached.

2 A task can issue a directive to attach itself to a named common region or a named dynamic region.

3 A task can request the Executive to attach any region within its own logical address space (other than its task region) to another specified task.

Attaching identifies a task as a user of a region and prevents the system from deleting a region until all user tasks have been detached.

For each region to which a task is attached, the executive requires a data structure called an Attachment Descriptor Block (ADB). These are created automatically for regions linked to a task at build time, and for the two possible resident overlay regions (read-only and read/write). You must use the ATRG task builder option to create additional ADBs needed for dynamically attached regions. See the *IAS Task Builder Reference Manual* for details of the ATRG option.

2.3.3 Region Protection

A task cannnot indiscriminately attach to any region. The following criteria determine how tasks can attach to regions outside their logical address space:

- Each region has a protection mask to prevent unauthorized access. The mask indicates the types of access (read, write, extend, delete) allowed for each category of user (system, owner, group, world). The Executive checks that the requesting task's User Identification Code (UIC) allows it to make the attempted access. The attempt fails if the protection mask denies that task the access it wants.
- When a task creates a dynamic region, it may or may not give that region a name. If it does not give a name, other tasks can attach to the region only if its identity is sent to them by the creating task, using one of the SEND BY REFERENCE directives (see Chapter 3, Section 3.7). If it gives a name, it can be made global or terminal-sensitive. If the name is global, any task may attach to it as long as it knows the name and there is no protection violation. If the name is terminal-sensitive, any task running on the same terminal as the creating task may attach to the region, as long as there is no protection violation.
- Any task can issue a SEND BY REFERENCE directive to attach any region (except the task region) to another task. The reference sent includes the access rights with which the receiving task attaches to the region. The sending task can only grant access rights that it has itself.
- Any task can map to a named common region as long as there is no protection violation.

2.4 Directive Summary

The following memory management directives are available:

- CREATE REGION (CRRG$)
- ATTACH REGION (ATRG$)
- DETACH REGION (DTRG$)
- CREATE ADDRESS WINDOW (CRAW$)
- ELIMINATE ADDRESS WINDON (ELAW$)
- MAP ADDRESS WINDOW (MAP$)
- UNMAP ADDRESS WINDOW (UMAP$)
- SEND BY REFERENCE (SREF$)
- SEND BY REFERENCE AND REQUEST OR RESUME (SRFR$)
- RECEIVE BY REFERENCE (RREF$)
- GET MAPPING CONTEXT (GMCX$)
- GET REGION PARAMETERS (GREG$)

You cannot issue the CREATE REGION, ATTACH REGION, or DETACH REGION directives without special privilege. See Chapter 3, Section 3.7 for brief descriptions of the function and use of each directive. See Chapter 4 for detailed descriptions.

2.5 User Data Structures

Most memory management directives are individually capable of performing a number of separate actions. For example, a single CREATE ADDRESS WINDOW directive can unmap and eliminate up to seven conflicting address windows, create a new window, then map the new window to a specified region. The complexity of the directives requires a special means of communication between the user task and the Executive. The communication is achieved through data structures that:

1 Allow the task to specify which directive options it wants the Executive to perform, and

2 Permit the Executive to provide the task with details about the outcome of the requested actions.

There are two types of user data structures that correspond to the two key elements (regions and address windows) manipulated by the directives. The structures are called:

1 The Region Definition Block (RDB)

2 The Window Definition Block (WDB)

Every memory management directive (except GET REGION PARAMETERS) uses one of these structures as its communications area between the task and the Executive. Each directive issued includes in the Directive Parameter Block (DPB) a pointer to the appropriate definition block. Values assigned by the task to offsets within an RDB or a WDB define or modify the directive operation. After the Executive has carried out the specified operation, it assigns values to various locations within the block to describe the actions taken and to provide the task with information useful for subsequent operations.

2.5.1 Region Definition Block (RDB)

Figure 2–4 illustrates the format of an RDB. In addition to the symbolic offsets defined in the diagram, the region status word, R.GSTS, contains defined bits that can be set or cleared by the Executive or the task. (IAS reserves undefined bits for future expansion.) The defined bits are:

Bit			Definition
RS.CRR	=	100000	Region was successfully created.
RS.UNM	=	40000	One or more regions were unmapped on a detach.
RS.TSK	=	2000	Reserved.
RS.TIS	=	1000	Region is terminal sensitive.
RS.NPL	=	400	Reserved.
RS.MDL	=	200	Mark region for deletion on last detach.
RS.NDL	=	100	Created region is not to be marked for deletion on last detach.
RS.ATT	=	40	Attach to created region.
RS.CON	=	20	Reserved.
RS.NEX	=	20	Reserved.
RS.DEL	=	10	Delete access desired on attach.
RS.EXT	=	4	Extend access desired on attach.
RS.WRT	=	2	Write access desired on attach.
RS.RED	=	1	Read access desired on attach.

The three memory management directives that require a pointer to an RDB are:

- CREATE REGION (CRRG$)
- ATTACH REGION (ATRG$)
- DETACH REGION (DTRG$)

When a task issues one of these directives, the Executive clears the four high order bits in the region status word of the appropriate RDB. After completing the directive operation, the Executive sets the RS.CRR and/or RS.UNM bit to indicate to the task what actions were taken. The Executive never modifies the other bits.

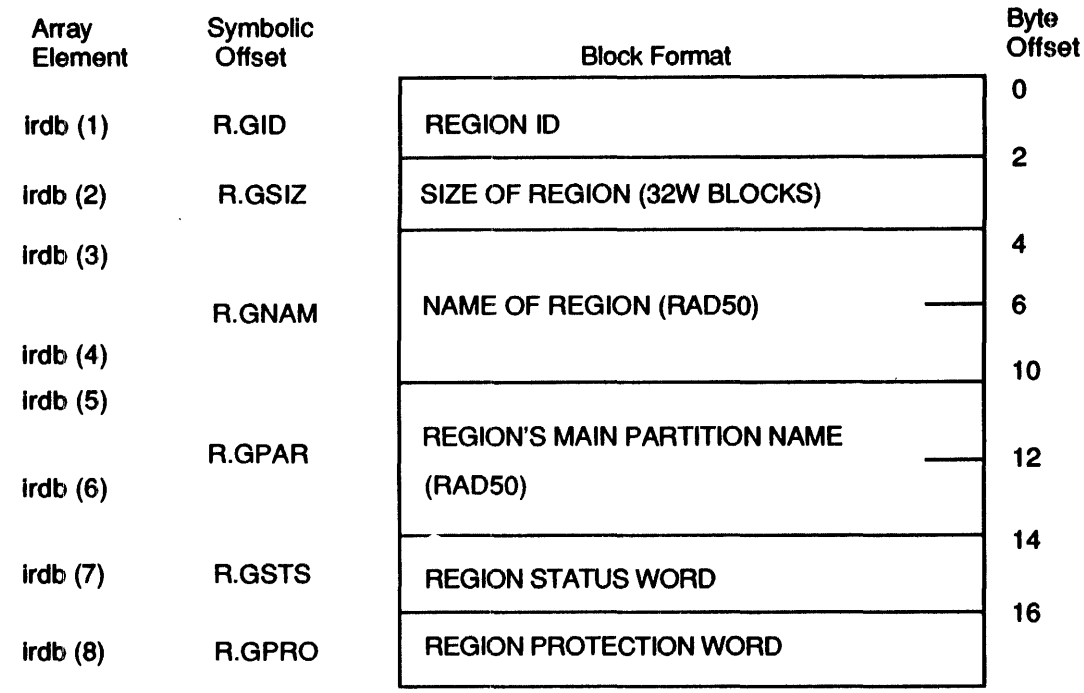
Using Macros to Generate an RDB

IAS provides two macros, RDBDF$ and RDBBK$, to generate and define an RDB. RDBDF$ defines the offsets and status word bits for a region definition block; RDBBK$ then creates the actual region definition block. The format of RDBDF$ is:

```
RDBDF$
```

Since RDBBK$ automatically invokes RDBDF$, the programmer need only specify RDBBK$ in a module that creates an RDB. The format of the call to RDBBK$ is:

Figure 2–4 Region Definition Block




```
RDBBK$  siz,nam,par,sts,pro
```

where:

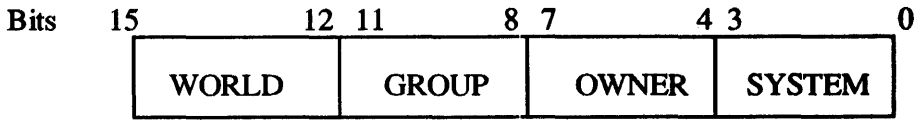
- siz - is the region size in 32-word blocks
- nam - is the region name (Radix-50)
- par - is the name of the partition in which to create the region (Radix-50)
- sts - is the region status word bit definitions
- pro - is the region's default protection word

The argument "sts" sets specified bits in the status word R.GSTS. The argument normally has the following format:

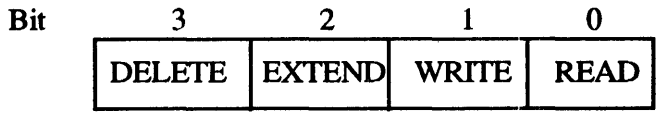
```
<bit1[!...!bitn]>
```

where "bit" is a defined bit to be set.

The argument pro is an octal number. The 16-bit binary equivalent specifies the region's default protection as follows:



Each of the four categories above has four bits; each bit represents a type of access:



A bit value of zero (0) indicates that the respective type of access is to be allowed; a bit value of one (1) indicates that the respective type of access is to be denied.

The macro call:

```
RDBBK$  102.,ALPHA,GEN,<RS.NDL!RS.ATT!RS.WRT!RS.RED>,167000
```

expands to:

```
.WORD   0
.WORD   102.
.RAD50   /ALPHA/
.RAD50   /GEN/
.WORD   0
.WORD   <RS.NDL!RS.ATT!RS.WRT!RS.RED>
.WORD   167000
```

If a CREATE REGION directive points to the RDB defined by the macro call expansion above, the Executive creates a region 102 (decimal) 32-word blocks in length, named ALPHA, in a partition named GEN. The defined bits specified in the sts argument inform the Executive:

1 Not to mark the region for deletion on the last detach,

2 To attach region ALPHA to the task issuing the directive macro call, and

3 To grant read and write access to the attached task.

The protection word specified as 167000 (octal) assigns the protection mask to the region. The octal number, which has a binary equivalent of 1110111000000000, grants all types of access to system and owner tasks, and read access only to group and world tasks.

If no protection word is specified, the Executive treats this as a value of zero (that is, unrestricted access for all categories of user).

If the CREATE REGION directive is successful, the Executive returns to the issuing task the region ID in the symbolic offset R.GID and sets the defined bit RS.CRR in the status word R.GSTS.

Using FORTRAN to Generate an RDB

FORTRAN programmers must create an 8-word, single-precision integer array as the RDB to be supplied in the subroutine calls:

- CALL ATRG - (ATTACH REGION directive)
- CALL CRRG - (CREATE REGION directive)
- CALL DTRG - (DETACH REGION directive)

(See the *PDP-11 FORTRAN Language Reference Manual* for information on the creation of arrays.) An RDB array has the following format:

Word	Contents
irdb(1)	Region ID
irdb(2)	Size of the region in 32-word blocks
irdb(3) irdb(4)	Region name (2 words in Radix-50 format)
irdb(5) irdb(6)	Name of the partition that contains the region (2 words in Radix-50 format)
irdb(7)	Region status word (see paragraph below)
irdb(8)	Region protection code

The FORTRAN programmer modifies the region status word, irdb(7), by setting or clearing the appropriate bits. See the list in Section 2.5.1 that describes the defined bits. The bit values are listed alongside the symbolic offsets.

Note that Hollerith text strings can be converted to Radix-50 values by calls to IRAD50 (see the appropriate *IAS FORTRAN User's Guide*).

2.5.2 Window Definition Block (WDB)

Figure 2–5 illustrates the format of the WDB. The block consists of a number of symbolic offsets. One of the offsets is the window status word, W.NSTS, which contains defined bits that the Executive or the task can set or clear. (IAS reserves all undefined bits for future expansion.) The defined bits are:

Bit			Definition
WS.CRW	=	100000	Address window was successfully created.
WS.UNM	=	40000	One or more windows were unmapped by a CREATE ADDRESS WINDOW, MAP ADDRESS WINDOW, or UNMAP ADDRESS WINDOW directive.
WS.ELW	=	20000	One or more windows were eliminated in a CREATE ADDRESS WINDOW or ELIMINATE ADDRESS WINDOW directive.
WS.RRF	=	10000	Reference was successfully received.
WS.RST	=	2000	Stop task if no references to receive.
WS.RSU	=	1000	Suspend task if no references to receive.

Bit			Definition
WS.64B	=	400	Defines the task's permitted alignment boundaries: 0 for 256-word (512-byte) alignment, 1 for 32-word (64-byte) alignment.
WS.MAP	=	200	Window is to be mapped in a CREATE ADDRESS WINDOW or RECEIVE BY REFERENCE directive.
WS.RCX	=	100	Exit if no references to receive.
WS.CON	=	20	Reserved.
WS.DEL	=	10	Send with delete access.
WS.EXT	=	4	Send with extend access.
WS.WRT	=	2	Send with write access or map with write access.
WS.RED	=	1	Send with read access.

Figure 2–5 Window Definition Block

Array Element	Symbolic Offset	Block Format	Byte Offset
iwdb (1)	W.NID W.NAPR	BASE.APR | WINDOW ID	0 2
iwdb (2)	W.NBAS	VIRTUAL BASE ADDRESS (BYTES)	4
iwdb (3)	W.NSIZ	WINDOW SIZE (32W BLOCKS)	6
iwdb (4)	W.NRID	REGION ID	10
iwdb (5)	W.NOFF	OFFSET IN REGION (32W BLOCKS)	12
iwdb (6)	W.NLEN	LENGTH TO MAP (32W BLOCKS)	14
iwdb (7)	W.NSTS	WINDOW STATUS WORD	16
iwdb (8)	W.NSRB	SEND/RECEIVE BUFFER ADDRESS (BYTES)	

The following directives need a pointer to a WDB:

- CREATE ADDRESS WINDOW (CRAW$)
- ELIMINATE ADDRESS WINDOW (ELAW$)
- MAP ADDRESS WINDOW (MAP$)
- UNMAP ADDRESS WINDOW (UMAP$)

- SEND BY REFERENCE (SREF$)
- SEND BY REFERENCE AND REQUEST OR RESUME (SRFR$)
- RECEIVE BY REFERENCE (RREF$)

When a task issues one of these directives, the Executive clears the four high order bits in the window status word of the appropriate WDB. The Executive can then set any of these bits after completing the directive operation to tell the task what actions were taken. The Executive never modifies the other bits.

Using Macros to Generate a WDB

IAS provides two macros, WDBDF$ and WDBBK$, to generate and define a WDB. WDBDF$ defines the offsets and status word bits for a window definition block; WDBBK$ then creates the actual window definition block. The format of WDBDF$ is:

```
WDBDF$
```

Since WDBBK$ automatically invokes WDBDF$, the programmer need only specify WDBBK$ in a module that generates a WDB. The format of the call to WDBBK$ is:

```
WDBBK$  apr,siz,rid,off,len,sts,srb
```

where:

- apr - is a number from 0 to 7 that specifies the window's base Active Page Register (APR). The APR determines the 4K word boundary on which the window is to begin. APR 0 corresponds to virtual address 0, APR 1 to 4K, APR 2 to 8K, and so on. (See Section 2.1.1.)
- siz - is the size of the window in 32-word blocks.
- rid - is a region ID
- off - is the offset within the region to be mapped in 32-word blocks
- len - is the length within region to be mapped, in 32-word blocks
- sts - is the window status word bit definitions
- srb - is a send/receive buffer virtual address

The argument "sts" sets specified bits in the status word W.NSTS. The argument normally has the following format:

```
<bit1[!...!bitn]>
```

where "bit" is a defined bit to be set.

The macro call:

```
WDBBK$  5,76.,0,50.,,<WS.MAP!WS.WRT>
```

expands to:

```
.BYTE   0,5
.WORD   0
.WORD   76.
.WORD   0
.WORD   50.
.WORD   0
.WORD   <WS.MAP!WS.WRT>
.WORD   0
```

If a CREATE ADDRESS WINDOW directive points to the WDB defined by the macro call expanded above, the Executive:

1 Creates a window 76 (decimal) 32-word blocks in length, beginning at APR 5 (virtual address 20K or 120000 octal).

2 Maps the window with write access (<WS.MAP!WS.WRT>) to the issuing task's task region (because the macro call specified 0 for the region ID).

3 Starts to map 50 (decimal) blocks from the base of the region and maps an area either equal to the length of the window (76 [decimal] blocks) or the length remaining in the region, whichever is smaller (because the macro call defaulted the len argument).

4 Returns values to the symbolic offsets W.NID (the window's ID) and W.NBAS (the window's virtual base address).

Using FORTRAN to Generate a WDB

FORTRAN programmers must create an 8-word, single-precision integer array as the WDB to be supplied in the subroutine calls:

- CALL CRAW - (CREATE ADDRESS WINDOW directive)
- CALL ELAW - (ELIMINATE ADDRESS WINDOW directive)
- CALL MAP - (MAP ADDRESS WINDOW directive)
- CALL UNMAP - (UNMAP ADDRESS WINDOW directive)
- CALL SREF - (SEND BY REFERENCE directive)
- CALL RREF - (RECEIVE BY REFERENCE directive)

(See the *PDP-11 FORTRAN Language Reference Manual* for information on the creation of arrays.) A WDB array has the following format:

Word	Contents
iwdb(1)	Bits 0 to 7 contain the window ID; bits 8 to 15 contain the window's base APR
iwdb(2)	Base virtual address of the window
iwdb(3)	Size of the window in 32-word blocks
iwdb(4)	Region ID
iwdb(5)	Offset length within the region at which map begins, in 32-word blocks.
iwdb(6)	Length mapped within the region in 32-word blocks.
iwdb(7)	Window status word (see paragraph immediately below)
iwdb(8)	Address of send/receive buffer

The FORTRAN programmer modifies the window status word, iwdb(7), by setting or clearing the appropriate bits. See the list above in Section 2.5.2 that describes the defined bits. The bit values are listed alongside the symbolic offsets.

Notes:

1 The contents of bits 8 to 15 of iwdb(1) must normally be set without destroying the value in bits 0 to 7 for any directive other than CREATE ADDRESS WINDOW.

2 A call to GETADR can be used to set up the address of the SEND/RECEIVE buffer. For example:

3 CALL GETADR(IWDB,,,,,,,,IRCVB)

4 This call places the address of buffer IRCVB in array element 8. The remaining elements are unchanged. The subroutines SREF and RREF also set up this value.

2.5.3 Assigned Values or Settings

The exact values or settings assigned to individual fields within the RDB or the WDB vary according to each directive. Fields that are not needed as input can have any value when you issue the directive. Chapter 4 describes which offsets and settings are relevant for each memory management directive. The values that the task assigns are called input parameters, those that the Executive assigns are called output parameters.

2.6 Executive Privileged Tasks

All Executive privileged tasks map to SCOM and the External page. The system dedicates APRs 4, 5, 6 and 7 to this mapping. A privileged task can issue memory management directives to remap any number of these APRs to regions. You must take care when using the directives in this way; such remapping can cause obscure bugs to occur. When a directive unmaps a window that formerly mapped SCOM or the External page, the Executive restores the former mapping.

3 System Directive Categories

This chapter groups the directives by function and gives a brief description of each. See Chapter 4 for detailed descriptions of each system directive, presented alphabetically according to the directive macro calls.

The system directives are grouped into the following categories:

1 Task execution control directives

2 Information directives

3 Event-associated directives

4 Trap-associated directives

5 I/O and intertask communications-related directives

6 Task status control directives

7 Memory management directives

Note that in the following tables those directives flagged (R) require real-time directive privilege and those marked (M) require memory management directive privilege for timesharing tasks run in a timesharing IAS system. If a directive is not flagged as such, then you need no privilege for issuing that directive.

On a multi-user system, or for real-time tasks in a timesharing system, you do not need these privileges.

3.1 Task Execution Control Directives

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

Table 3–1 Task Execution Control Directives

Directive	MACRO Call	Function
REQUEST (R)	RQST$	Runs a task contingent upon priority and memory availability. If memory is not available, the request is queued and the task is run as soon as possible.
SPAWN (R)	SPWN$	Requests a specified task for execution, optionally establishing exit events and supplying a command line.
EXECUTE (R)	EXEC$	Executes a task only if sufficient memory is available at the time involved. If the task cannot be run immediately, the request is rejected.
SCHEDULE (R)	SCHD$	Requests a task using the RQST$ directive at a specific time and, optionally, repeats the request periodically (for example, the task can be scheduled to run at 09.35 a.m.)

Table 3–1 (Cont.) Task Execution Control Directives

Directive	MACRO Call	Function
RUN (R)	RUN$	Requests a task using the RQST$ directive at a specified interval from the current time and, optionally, repeats the request periodically.
SYNCHRONIZE (R)	SYNC$	Requests a task using the RQST$ directive at a specific interval from a specified future time and, optionally, repeats the request periodically.
CANCEL SCHEDULED REQUESTS (R)	CSRQ$	Cancels scheduled requests for task execution.
CANCEL MARK TIME REQUESTS	CMKT$	Cancels all MARK TIME requests that have been made by the issuing task.
SUSPEND	SPND$	Suspends execution of the task issuing the suspend.
RESUME (R)	RSUM$	Resumes the execution of a task that has suspended itself.
TASK EXIT	EXIT$	Terminates execution of the issuing task.
TASK EXIT WITH STATUS	EXST$	Terminates execution of the issuing task and returns status information.
ABORT TASK (R)	ABRT$	Terminates execution of the indicated task.
EXTEND TASK	EXTK$	Instructs the system to modify the size of the issuing task by a positive or negative increment of 32-word blocks.

3.2 Informational Directives

Several informational directives provide the issuing task with data retained by the system. These directives provide the time of day, the task parameters, the console switch settings and partition or region parameters.

Table 3–2 Informational Directives

Directive	MACRO Call	Function
GET PARTITION PARAMETERS	GPRT$	Fills an indicated 3-word buffer with information for a specific partition.
GET SENSE SWITCHES	GSSW$	Obtains the status of the console sense and stores it in the issuing task's directive status word.
GET TIME PARAMETERS	GTIM$	Fills an indicated 8-word buffer with current time and date information.
GET COMMON BLOCK PARAMETERS	GCOM$	Fills an indicated 8-word buffer with information for a specific common block.
GET TASK PARAMETERS	GTSK$	Fills an indicated 16-word buffer with information about the task issuing the directive.

3.3 Event-Associated Directives

The event and event flag directives are the means provided in the system for inter-task and intra-task synchronization and signalling. You must use these directives carefully because software faults resulting from erroneous signalling and synchronization are often obscure and difficult to isolate. See the *IAS Executive Facilities Reference Manual* for a description of events and event flags.

Table 3–3 Event-associated Directives

Directive	MACRO Call	Function
CONNECT TO TASK	CNCT$	Synchronizes the issuing task with the exit or emit status of an active task.
EMIT STATUS	EMST$	Returns a specified 16-bit value to the the specified connected task and disconnects from the parent task.
DECLARE SIGNIFICANT EVENT (R)	DECL$	Declares a significant event and, optionally, sets an event flag and reports the status of the flag before it was set.
WAIT FOR SIGNIFICANT EVENT	WSIG$	Suspends execution of the issuing task until the next significant event occurs.
SET EVENT FLAG	SETF$	Sets an indicated event flag and reports the status of the flag before it was set. SETF$ does not cause a significant event to occur.
CLEAR EVENT FLAG	CLEF$	Clears an indicated event flag and reports the flag's status before clearing. CLEF$ does not cause a significant event to occur.
READ EVENT FLAG	RDEF$	Reads a specified event flag and indicates by the return code in the directive status word whether the flag is set or cleared.
READ ALL EVENT FLAGS	RDAF$	Reads all 64 event flags and records their status by setting or clearing corresponding bits in a 64-bit (4-word) buffer.
WAIT FOR SINGLE EVENT FLAG	WTSE$	Suspend execution of the issuing task until the indicated event flag is set.
WAIT FOR LOGICAL OR OF EVENT FLAG	WTLO$	Suspends execution of the issuing task until an indicated event flag in one of a number of flags is set.
STOP FOR SINGLE EVENT FLAG	STSE$	Stops execution of the issuing task until the indicated event flag is set.
STOP FOR LOGICAL OR OF EVENT FLAGS	STLO$	Stops execution of the issuing task until and indicated event flag in one of a number of flags is set.
STOP	STOP$	Stops the issuing task.
UNSTOP	USTP$	Unstops a task which has been stopped via either a STOP or RECEIVE OR STOP directive
EXITIF	EXIF$	Terminates the execution of the issuing task if an indicated event flag is not set.
MARK TIME	MRKT$	Declares an event after an indicated time interval starting from when the directive is issued. An event flag can be set or an AST routine entered when the event occurs.
CANCEL MARK TIME AST REQUESTS	CMTA$	Cancels any unserviced mark time AST requests for the issuing task.

3.4 Trap-Associated Directives

The trap-associated directives provide you with the same facilities inherent in the PDP-11 hardware trap system. These directives allow transfers of control (software interrupts) to the executing tasks. See the *IAS Executive Facilities Reference Manual* for a description of system traps.

Table 3–4 Trap-associated Directives

Directive	MACRO Call	Function
AST SERVICE EXIT	ASTX$	Terminates execution of an asynchronous system trap service routine.
SPECIFY SST VECTOR TABLE FOR TASK	SVTK$	Specifies the address of a table containing the addresses of synchronous system trap service routines.
INHIBIT AST RECOGNITION	IHAR$	Inhibits recognition of ASTs for the issuing task.
ENABLE AST RECOGNITION	ENAR$	Enables recognition of ASTs for the issuing task.
SPECIFY POWER RECOVERY AST	SPRA$	Informs the system whether or not power recovery ASTs are desired for the issuing task. If desired, this directive indicates where control is to be transferred when the AST occurs.
SPECIFY FLOATING POINT AST	SFPA$	Informs the system whether or not PDP-11/70 or PDP-11/45 floating point exception ASTs are desired for the task. If desired, the directive indicates where control is to be transferred for the AST.
SPECIFY RECEIVE DATA AST	SRDA$	Allows the programmer to specify the AST service routine which is to be executed when any data is sent to this task.
SPECIFY RECEIVE-BY-REFERENCE AST	SRRA$	Allows the programmer to specify the AST service routine which is to be executed when any references are sent to the task.
SPECIFY SST VECTOR TABLE FOR DEBUGGING AID	SVDB$	Specifies the virtual address of a table of synchronous system trap service routine entry points for use by ODT or other debugging aids. SVDB$ takes precedence over SVTK$ (see above).

3.5 I/O and Intertask Communications-Related Directives

The I/O and intertask communications-related directives allow tasks to access I/O devices at the device handler level, to communicate with other tasks in the system and to retrieve command lines sent by MCR to the task.

Table 3–5 I/O and Intertask Communications-Related Directives

Directive	MACRO Call	Function
GET LUN INFORMATION	GLUN$	Fills a 6-word buffer with information about a physical device unit to which the logical unit number is assigned.
GET MCR COMMAND LINE	GMCR$	Transfers an 80-byte command line to the issuing task.
QUEUE I/O	QIO$	Places an I/O request for an indicated device in a priority-ordered queue of requests for that unit.
QUEUE I/O AND WAIT	QIOW$	Performs the functions of QIO$ (see above) and waits for the event flag specified in the QIO to become set (see WTSE$, Table 3–3).
ASSIGN LUN	ALUN$	Assigns a Logical Unit Number (LUN) to a physical device unit.
SEND DATA	VSDA$/SDAT$	VSDA$ queues a variable-length data block by priority for a task to receive. SDAT$ sends a 13-word data block.
SEND DATA AND RESUME OR REQUEST RECEIVER	VSDR$/SDRQ$	VSDR$ queues a variable length data block by priority for a task to receive and resumes or requests execution of the receiving task. SDRQ$ sends a 13-word data block. The issuing task needs to be real-time privileged unless the receiver is built to receive from any task.
RECEIVE DATA	VRCD$/RCVD$	VRCD$ receives a variable-length data block that was queued by another task. RCVD$ receives a 13-word data block.
RECEIVE DATA OR EXIT	VRCX$/RCVX$	VRCX$ receives a variable-length data block that has been queued by another task or exits if none is queued. RCVX$ receives a 13-word data block or exits.
RECEIVE DATA OR SUSPEND (R)	VRCS$/RCVS$	VRCS$ receives a variable-length data block if one is queued for the task or suspends the task. RCVS$ receives a 13-word data block or suspends.
RECEIVE DATA OR STOP	VRCT$/RCST$	VRCT$ attempts to dequeue a send data packet from the specified task (or any task). RCST$ receives a 13-word data block or stops.

3.6 Task Status Control Directives

The task status control directives are used for fixing and unfixing tasks, disabling and enabling tasks, disabling and enabling checkpointing of tasks and altering a task's priority.

Table 3–6 Task Status Control Directives

Directive	MACRO Call	Function
FIX-IN-MEMORY (R)	FIX$	Fixes in memory (makes permanently resident) an inactive, installed task.
UNFIX (R)	UFIX$	Reverses a FIX$ directive and frees the memory allocated to the task.
DISABLE (R)	DSBL$	Rejects future attempts to execute or fix an indicated task using any of the following directives: RQST$, EXEC$, SCHD$, RUN$, SYNC$, FIX$, SPWN$.
ENABLE (R)	ENBL$	Instructs the system to make an indicated disabled task runnable (that is, reverses the DSBL$ directive).
DISABLE (R) CHECKPOINTING	DSCP$	Instructs the system to make the issuing task non-checkpointable.
ENABLE (R) CHECKPOINTING	ENCP$	Makes the issuing task checkpointable if checkpointing for the task was previously disabled.
ALTER PRIORITY (R)	ALTP$	Alters the priority of a specified active task to the new priority indicated in the directive.

3.7 Memory Management Directives

The memory management directives allow a task to manipulate its virtual and logical address space and to set up and control dynamically the window-to-region mapping assignments. The directives also provide the means by which tasks can share and pass references to data and routines. See Chapter 2 for a detailed description of memory management in the context of system directives. Note also that those directives flagged (M) require memory management privilege.

Table 3–7 Memory Management Directives

Directive	MACRO Call	Function
CREATE REGION (M)	CRRG$	Creates a dynamic region in a system- controlled partition and attaches system-controlled partition and attaches it to the issuing task.
ATTACH REGION (M)	ATRG$	Attaches the issuing task to a common region or to a named dynamic region.
DETACH REGION (M)	DTRG$	Detaches the issuing task from a specified region. Any of the task's address windows that are mapped to the region are automatically unmapped.
CREATE ADDRESS WINDOW	CRAW$	Creates an address window, establishes its virtual address size and optionally maps the window. Any other windows that overlap with the range of addresses of the new window are first unmapped, if necessary, and then eliminated.
ELIMINATE ADDRESS WINDOW	ELAW$	Eliminates an existing address window, unmapping it first, if necessary.
MAP ADDRESS WINDOW	MAP$	Maps an existing window to a specified offset and length within a region attached to the issuing task. If the window is already mapped elsewhere, the Executive unmaps it before carrying out the map assignment described in the directive.

Table 3–7 (Cont.) Memory Management Directives

Directive	MACRO Call	Function
UNMAP ADDRESS WINDOW	UMAP$	Unmaps a specified window. After the window has been unmapped, the corresponding virtual address range cannot be referenced until the task issues another mapping directive.
SEND BY REFERENCE	SREF$	Inserts a packet containing a reference to a region into the receive queue of a specified task. The receiver task is automatically attached to the region to which it refers.
SEND BY REFERENCE AND REQUEST OR RESUME	SRFR$	Inserts a packet containing a reference to a region into the receive queue of a specified task and requests or resumes the task if it was inactive or suspended when the reference was sent.
RECEIVE BY REFERENCE	RREF$	Causes the Executive to dequeue the next receive-by-reference packet in the receive-by-reference queue of the issuing task. Optionally, the directive can map a window to the referenced region, or cause the task to exit, be suspended or be stopped if the queue does not contain a receive-by-reference packet.
GET MAPPING CONTEXT	GMCX$	Causes the Executive to return to the issuing task a description of the current window-to-region mapping assignments. The description is in a form that enables the user to restore the mapping context by a series of CREATE ADDRESS WINDOW directives.
GET REGION PARAMETERS	GREG$	Causes the Executive to supply the issuing task with information about a specified region.

4 System Directive Descriptions

This chapter provides a detailed description of the system directives.

Each directive description consists of an explanation of the directive's function and use, the names of the corresponding macro and FORTRAN calls, the associated parameters, and possible return values of the Directive Status Word (DSW).

The descriptions show only the $ form of the macro name, although the $C and $S forms are also available, unless otherwise specified. The $S form is recommended for directives which take no parameters and generate a single word DPB; for example: ASTX$S, EXIT$S.

In addition to the directive macros themselves, the DIR$ macro also allows the programmer to execute a directive by means of a predefined DPB (see Chapter 1, Section 1.5.2 for details).

4.1 Directive Privilege

There are two categories of directive privileges:

1 Real-time directive privilege

2 Memory management directive privilege

In the individual descriptions for each directive, those directives marked (R) alongside the heading need real-time privilege and those marked (M) need memory management privilege. Some directives, however, do not need any special privilege. All real-time tasks and all tasks in a multi-user system are granted both sets of privileges and can, therefore, issue any directive. Timesharing tasks, however, must have the appropriate privilege to perform the corresponding directives. The system manager normally allocates such privilege to each individual user.

If a timesharing task attempts to issue a directive without having been granted the necessary privilege, the system returns an error code of IE.AST in the DSW.

Some directives need real-time directive privilege for the cases where a global event flag is to be set or cleared. Reading a global event flag does not need any directive privilege.

Any task that has executive privilege (see Section 4.2, below) can issue any directives, irrespective of other privileges.

4.2 Executive Privilege

In IAS, executive privilege is defined as the linking of a task to SCOM and to the External Page by means of the LINK/PRIVILEGED qualifier or the /PR switch to Task Builder. See the *IAS Task Builder Reference Manual* for further details of the LINK and Task Builder commands. See the *IAS Executive Facilities Reference Manual* for information regarding SCOM and the External Page.

Executive privileged tasks can only be run by users with the required privilege assigned, except in the case of system tasks installed by the system manager (where no privilege is required).

In connection with system directives, executive privilege is required when:

1 Issuing a QUEUE I/O (QIO$) or QUEUE I/O AND WAIT (QIOW$) directive to read or write logical blocks to a device that has:

   a. A volume mounted as a Files-11 directory volume.

   b. A volume that is mountable, but is not currently mounted.

2 Issuing an EXECUTE (EXEC$), REQUEST (RQST$), RUN (RUN$) or SYNCHRONIZE (SYNC$) directive by which another task is run under a changed UIC (see Section 4.3, below).

4.3 Task UIC

Normally, tasks are run under the (default) UIC indicated in the task's header. The default can be set when a task is linked or installed. If no default is specified, the task is run under the terminal user's own UIC.

A task must be executive privileged if it needs to issue a directive that will cause another task to run under a UIC different from its default UIC. See Section 4.2.

If a task is run using the Timesharing Control Services (TCS), the UIC of the terminal is always used. Refer to the IAS Guide to Writing a Command Language Interpreter manual for further information.

4.4 Ti Indicator

The Terminal Interface Input device (TI:) is the device for which a task was invoked. The TI is normally the terminal at which you typed the command that resulted in the task being activated. When you use the TI indicator in a directive parameter list, the TI is the PUD address of the device for which the task was activated. This facility provides identification of a specific copy of a multiuser task, provided there is only one copy of the task invoked for a particular device.

4.5 System Directive Descriptions

Each directive description includes most or all of the following elements:

Name:

A description of the function of the directive.

Macro Call:

The macro call is shown, each parameter is defined, and the defaults for optional parameters are given in parentheses following the definition of the parameter. Since zero is supplied for most defaulted parameters, only non-zero default values are shown. Note also that square brackets ([]) denote optional arguments.

Local Symbol Definitions:

Macro expansions usually generate local symbol definitions with an assigned value equal to the byte offset from the start of the DPB to the respective DPB element. There is a list of these symbols. The length in bytes of the datum pointed to by the symbol appears in parentheses following the symbol's description.

Thus:

- A.BTTN - (length 4 bytes) Task name in Radix-50

defines A.BTTN as pointing to a task name in the Abort Task DPB; the task name datum has a length of 4 bytes.

Definition Block Parameters:

These parameters are given only in the memory management directive descriptions. This section describes all the relevant input and output parameters in the region or window definition block (see Chapter 2, Section 2.5).

DSW Return Codes:

A list of all valid return codes.

Macro Expansion:

The $ form of the macro is expanded. The $S and $C forms are normally also available, except where otherwise indicated in the directive description.

FORTRAN Call:

The FORTRAN subroutine call is shown, and each parameter is defined. As for the macro calls, square brackets ([]) denote all optional arguments.

ABRT$

The ABORT TASK directive terminates the execution of the indicated task. Termination information is printed on the terminal used to invoke the task. A task can abort any task. If the task being aborted is a multi-user task, it is aborted only if its TI matches that of the task issuing the ABORT directive.

MACRO CALL

```
ABRT$ tsk
```

where:

- tsk - is the name of the task to be aborted.

LOCAL SYMBOL DEFINITIONS

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

- A.BTTN - (Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IS.SPD	+2	Task aborted before execution commenced
IE.INS	-02	Task not installed
IE.ACT	-07	Task not active
IE.ITS	-08	Task loading or exiting
IE.CKP	-10	Task is not abortable
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- tsk - is a two-word 1- to 6- character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

ALTP$

The ALTER PRIORITY directive alters the priority of the specified active task to the new priority indicated in the directive. If the task is multi-user, its priority is altered only if its TI matches that of the calling task. The Executive declares a significant event.

MACRO CALL

```
ALTP$  [tsk][,pri]
```

where:

- tsk - is the name of the task whose priority is to be changed. If you do not specify the task, the calling task's priority is changed.
- pri - is the new priority (in the range 1 through 250) for the task. If you do not specify pri, the new priority is the priority specified at link or installation time. If you did not specify a priority during linking or installation, a system default of 50 (decimal) is used.

LOCAL SYMBOL DEFINITIONS

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

- A.LTTN - (Length 4 bytes) Task name in Radix-50
- A.LTPR - (2) Priority

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.ACT	-07	Task not active
IE.ITS	-08	Task loading or exiting
IE.PRI	-16	Directive privilege violation
IE.IPR	-95	Invalid priority specified (<0 or >250)
IE.ILU	-96	Task under scheduler control
IE.ADP	-98	Part of DPB is out of issuing task's address space

Code	Value Returned	Explanation
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
ALTP$   TASK10,100
.BYTE   9.,4          ;ALTP$ MACRO DIC, DPB SIZE = 4 WORDS
.RAD50  /TASK10/      ;TASK 'TASK10'
.WORD   100           ;NEW PRIORITY
```

FORTRAN CALL

```
CALL ALTPRI ([tsk],[ipri][,ids])
```

where:

- tsk - is a 2-word Radix-50 name of the task whose priority is to be altered. If you omit this argument, the default is the calling task.
- ipri - is a 1-word integer value for the new priority (in the range 1 through 250) for the task. If you do not specify ipri, the new priority is the priority specified at link or installation time. If you did not specify a priority during linking or installation, a system default of 50 (decimal) is used.
- ids - is a 1-word integer variable for the directive status. If you omit this argument, status information is not provided.

ALUN$

The ASSIGN LUN directive assigns a Logical Unit Number (LUN) to a physical device unit. You can subsequently reassign LUNs if necessary by issuing further assigns for the same logical unit to a different physical device. On successful reassignment, all I/O requests for the issuing task in the previous device queue are cancelled. You can also use this directive to deassign a LUN if you do not specify a physical device in the request.

MACRO CALL

```
ALUN$  lun[,dev,unt]
```

where:

- lun - is a logical unit number
- dev - is a physical device name (two characters)
- unt - is a physical device unit number

  NOTE: If you omit dev and unt, the LUN is deassigned. Future attempts to use the LUN will fail with the error "unassigned LUN" (IE.LUN). The LUN may have previously been assigned by means of an ASSIGN LUN directive, or by means of default or specific assignment at task build time.

LOCAL SYMBOL DEFINITIONS

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

- A.LULU - (Length 2 bytes) Logical unit number
- A.LUNA - (2) Physical device name
- A.LUNU - (2) Physical device unit number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.LNL	-90	LUN usage interlocked (LUN is already assigned to a device and a file is currently open on that device for the specified LUN or the device is attached to the issuing task)
IE.IDU	-92	Invalid device and/or unit
IE.ILU	-96	Invalid logical unit number
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

```
CALL ASNLUN (ilun,idev,iunt[,ids])
```

where:

- ilun - is an integer containing a Logical Unit Number.
- idev - is an integer (format: 1A2) containing a device name.
- iunt - is an integer containing a device unit number.
- ids - is an integer variable to receive the Directive Status Word.

ASTX$

The AST SERVICE EXIT directive terminates execution of an asynchronous system trap service routine. See the *IAS Executive Facilities Reference Manual* for a description of ASTs.

On completion of the ASTX$, if another AST is queued, and ASTs are not inhibited, the next AST is immediately executed. Otherwise, the task's pre-AST state is restored.

NOTE: An AST routine may not be executed immediately the AST occurs if, for example, a higher priority process is running. In this case, one or more other ASTs can occur before the first AST routine is executed. This can occasionally cause synchronization problems if the AST routines are interdependent.

When an AST service routine is entered, the stack contains certain information. Those portions of control areas that can be used to effect requests from AST service routines are saved on the stack. The following information is always put on the user stack:

- SP+14+x - Event flag mask word for flags 1-16
- SP+12+x - Event flag mask word for flags 17-32
- SP+10+x - Event flag mask word for flags 33-48
- SP+06+x - Event flag mask word for flags 49-64
- SP+04+x - The pre-AST task's processor status (PS)
- SP+02+x - The pre-AST task's program counter (PC)
- SP+x - The pre-AST directive status word ($DSW)
- SP+x-2 to SP+00 - Additional information, if appropriate.

x here depends on the amount of additional information.

The stack can contain additional information as follows:

1 For power recovery ASTs, receive ASTs or receive-by-reference ASTs, no additional information is added.

2 For I/O completion, the stack additionally contains the address of the I/O status block.

3 For Mark-time ASTs, the stack additionally contains the event flag number.

4 For an 11/45 or 11/70 floating point exception, the stack additionally contains the exception code, and the exception address.

5 For a spawn AST, the stack additionally contains the address of the exit status block.

Before issuing the AST SERVICE EXIT directive, the AST service routine must remove any information on the stack that is additional to the first seven words shown above. The following example shows how this might be done when an AST routine is entered on a clock interrupt after the specified time of one minute.

```
;MAIN BODY OF PROGRAM
START:  .               ;PROCESS
        .
        .
        MRKT$S  ,#1,#3,#ASTSER
        .               ;PROCESS
        .
        EXIT$S          ;EXIT FROM TASK
;AST SERVICE
ASTSER: .
        .
        .
        .
        .
        TST (SP)+       ;REMOVE THE EVENT FLAG NUMBER
                        ;(THE SINGLE ADDITIONAL WORD)
        ASTX$S          ;AST EXIT MACRO
```

MACRO CALL

```
ASTX$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.AST	-80	Directive not issued from an AST service routine
IE.ADP	-98	Part of DPB or table is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

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.

ATRG$

The ATTACH REGION directive attaches the issuing task to a common region or to a named dynamic region. (You can not attached any other type of region to the task by means of this directive.) The Executive checks the desired access specified in the region status word against the owner UIC and the protection word of the region. If there is no protection violation, you are granted the access you need. If the region is successfully attached to the task, the Executive returns a 16-bit region ID (in R.GID), which the task uses in subsequent mapping directives.

You can also use this directive to determine the ID of a region already attached to the task. In this case, the task specifies the name of the attached region in R.GNAM and clears all four bits described below in the region status word R.GSTS. When the Executive processes the directive, it checks that the named region is attached. If the region is attached to the issuing task, the Executive returns the region ID, as well as the region size, for the task's first attachment to the region. You may wish to use the ATTACH REGION directive in this way to determine the region ID of a common block attached to the task at task build.

For this directive to succeed, there must be an Attachment Descriptor Block (ADB) available in the task header. These blocks are allocated by the ATRG Task Builder option (see Chapter 2, Section 2.3.2).

MACRO CALL

```
ATRG$   rdb
```

where:

- rdb - is the region definition block address

REGION DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
irdb(3)(4)	R.GNAM	Name of the region to be attached
irdb(7)	R.GSTS	Bit settings in the region status word (specifying desired access to the region):
		RS.RED - 1 if read access is desired
		RS.WRT - 1 if write access is desired

Input Parameters		
Array Element	**Offset**	
		RS.EXT - 1 if extend access is desired RS.DEL - 1 if delete access is desired Clear all four bits to request the region ID of the named region if it is already attached to the issuing task.

Output Parameters		
Array Element	**Offset**	
irdb(1)	R.GID	ID assigned to the region
irdb(2)	R.GSIZ	Size in 32-word blocks of the attached region

LOCAL SYMBOL DEFINITIONS

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

- A.TRBA - (Length 2 bytes) Region definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.PRI	-16	Privilege violation
IE.WOV	-85	No attachment descriptors available in task header
IE.PNS	-94	The specified region name does not exist
IE.ADP	-98	Part of the DPB or RDB is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
ATRG$     RDBADR
.BYTE     57.,2      ;ATRG$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     RDBADR     ;RDB ADDRESS
```

FORTRAN CALL

```
CALL ATRG (irdb[,ids])
```

where:

- irdb - is an 8-word integer array containing a region definition block
- ids - is an integer variable to receive the Directive Status Word

CLEF$

The CLEAR EVENT FLAG directive clears a specified event flag and reports the flag's status in the DSW before clearing. Clearing an event flag does not cause a significant event to occur. A task that does not have real-time directive privilege can clear only local event flags (1-32).

MACRO CALL

```
CLEF$  efn
```

where:

- efn - is an event flag number

LOCAL SYMBOL DEFINITIONS

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

- C.LEEF - (Length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.CLR	+0	Flag was already clear
IS.SET	+2	Flag was set
IE.PRI	-16	Privileged function (global flag cannot be cleared by a task that does not have real-time directive privilege).
IE.IEF	-97	Invalid event flag number (event flag number <1 or > 64).
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
CLEF$  1
.BYTE  31.,2      ;CLEF$ MACRO DIC,DPB=2 WORDS
.WORD  1          ;EVENT FLAG NUMBER 1
```

FORTRAN CALL

```
CALL CLREF (iefn[,ids])
```

where:

- iefn - is an integer containing an Event Flag Number.
- ids - is an integer variable to receive the Directive Status Word.

CMKT$

The CANCEL MARK TIME REQUESTS directive cancels one or more MARK TIME requests that have been made by the issuing task. If no parameters are supplied with the macro call, all MARK TIME requests made by the issuing task are cancelled. You specify parameters to cancel either those mark time requests that set an indicated event flag, or those that cause an AST at a specified location, or both.

Note that this directive has no effect on MARK TIME requests which have come due but have not yet notified the task. In particular, if the original request specified an AST entry point, but the AST has not been serviced yet (for example, because ASTs have been inhibited) this AST will still occur.

Refer also to the CANCEL MARK TIME AST REQUESTS (CMTA$) directive.

MACRO CALL

```
CMKT$ [efn][,ast]
```

where:

- efn - is an event flag number (0 implies no event flag)
- ast - is an AST service routine entry address

LOCAL SYMBOL DEFINITIONS

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

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

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.LNL	-90	EFN or AST entry implied by syntax but missing (for example, CMKT$ 5, or CMKT$,).
IE.IEF	-97	Invalid event flag number (event flag number <1 or >64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
 CMKT$    5,CMTAST ;CMKT MACRO DIC;DPB=3 WORDS
.BYTE    27.,3
.WORD    5          ;EVENT FLAG NUMBER 5
.WORD    CMTAST     ;AST SERVICE ROUTINE
```

FORTRAN CALL

```
CALL CANMT ([iefn][,ids])
```

where:

- iefn - is an integer containing an Event Flag number.
- ids - is an integer variable to receive the Directive Status Word.

  When an event flag number is specified, MARK TIME requests (made by the issuing task) to set that event flag are cancelled.

  When no event flag is specified, all MARK TIME requests (made by the issuing task) are cancelled.

CMTA$

The CANCEL MARK TIME AST REQUESTS directive instructs the system to cancel Mark Time requests that have been made by the issuing task. In addition, the directive allows the cancellation of pending ASTs which have not yet been serviced but for which the corresponding MARK TIME has come due. If you do not supply any parameters with the macro call, all MARK TIME requests made by the issuing task are cancelled. You can specify parameters to cancel either those mark time requests that set an indicated event flag, or those that cause an AST at a specified location, or both.

MACRO CALL

```
CMTA$ [efn],[ast][,flg]
```

where:

- efn - is an event flag number (0 implies no event flag)
- ast - is an AST service routine entry address
- flg - is a flag for the cancellation of unserviced ASTs (0 means do not cancel unserviced ASTs, < 0 means also cancel unserviced ASTs).

LOCAL SYMBOL DEFINITIONS

The following symbols are defined with the assigned values equal to the byte offset from the start of the DPB to the DPB elements:

- C.MKEF - (Length 2 bytes) Event flag number
- C.MKAE - (2) AST service routine entry address
- C.MKFL - (2) Cancel unserviced ASTs flag

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UNL	-90	EFN or AST entry implied by syntax but missing (for example, CMTA$ 5, or CMTA$,).
IE.IEF	-97	Invalid event flag number (event flag number <1 or>64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
CMTA$   EFN,AST,FLG
.BYTE   27.,4       ;CMTA$ MACRO DIC,DPB SIZE = 4 WORDS
.WORD   EFN         ;EVENT FLAG NUMBER
.WORD   AST         ;AST SERVICE ROUTINE
.WORD   FLG         ;UNSERVICED AST FLAG
```

FORTRAN CALL

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

CNCT$

The CONNECT TO TASK directive synchronizes the task issuing the directive with the exit or emit status of another task (offsppring that is already active. Execution of this directive creates an STL for the connector pair and increments the spawned task count in the issuing task's header. The optional exit AST routine is called when the offspring exits or emits status with the address of the associated exit status block on the stack. This directive should not be issued to connect to Command Line Interpreter (CLI) tasks; it is illegal to connect to a CLI task.

MACRO CALL

```
CNCT$ tname, [efn], [east], [esb]
```

where:

- tnam - is the radix-50 name of the offspring to be connected.
- efn - is the event flag to be cleared on issuance and set when the offspring task exits or emits status.
- east - is the address of an ast routine to be called when the offspring task exits or emits status.
- esb - is the address of a one-word status block to be written when the offspring task exits or emits status.

LOCAL SYMBOL DEFINITIONS

The following symbols are defined with the assigned values equal to the byte offset from the start of the DPB to the DPB elements:

- C.NCTN - (Length 4 bytes) Task name
- C.NCEF - (Length 2 bytes) Event flag
- C.NCEA - (Length 2 bytes) AST routine address
- C.NCES - (Length 2 bytes) Exit status block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient dynamic memory to allocate an offspring control block.
IE.INS	-02	The specified task was a command line interpreter.
IE.ACT	-07	The specified task was not active.
IE.IEF	-97	Invalid event flag number (EFN<0, or EFN>96 IF GROUP FLOBAL EVENT FLAGS EXIST FOR THE TASK'S GROUP; OR EFN>64 if not).
IE.ADP	-98	Part of the DPB or exit status block is not in the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
CNCT$ TNAM,EFN,AST,ESB
.BYTE   143.,6     ;CNCMT$ MACRO DIC,DPB SIZE = 6 WORDS
.RAD50  TNAM       ;OFFSPRING TASK NAME
.BYTE   EFN        ;EVENT FLAG NUMBER
.BYTE   16.        ;EXIT STATUS BLOCK CONSTANT
.WORD   AST        ;AST SERVICE ROUTINE
.WORD   ESB        ;EXIT STATUS BLOCK ADDRESS
```

FORTRAN CALL

```
CALL CNCT(rtname,[iefn],[iast],[iesb],[iparm][,ids])
```

where:

- rtname - is a single-precision, floating-point variable containing the offspring task name in radix-50 format.
- iefn - is the event flag to be set when the offspring task exits or emits status.
- iast - is the name of an AST to be called when the offspring task exits or emits status.
- iesb - is the name of a status block to be written when the offspring task exits or emits status.
- iparm - is the name of a word to receive the status block address when an AST occurs.
- ids - is an integer to receive the Directive Status Word.

CRAW$

The CREATE ADDRESS WINDOW directive creates a new virtual address window by establishing its virtual address base and size. Any existing windows that overlap the specified range of virtual addresses are unmapped, if necessary, and then eliminated. If the window is successfully created, the Executive returns an 8-bit window ID to the task. (The 8-bit window ID returned to the task is a number from 1 to 7.)

If WS.MAP in the window status word is set, the Executive proceeds to map the window according to the window definition block input parameters.

A task can specify any length for the mapping assignment that is less than or equal to both:

1 The window size specified when the window was created

2 The length remaining between the specified offset within the region and the end of the region.

If W.NLEN is set to 0, the length defaults to either the window size or the length remaining in the region, whichever is smaller. (Since the Executive returns the actual length mapped as an output parameter, the task must clear that offset before issuing the directive each time it wants to default the length of the map.)

The values that can be assigned to W.NOFF depend on the setting of bit WS.64B in the window status word (W.NSTS):

1 If WS.64B = 0, the offset specified in W.NOFF must represent a multiple of 256 words (512 bytes). Because the value of W.NOFF is expressed in units of 32-word blocks, the value must be a multiple of 8.

2 If WS.64B = 1, the task can align on 32-word boundaries; the programmer can therefore specify any offset within the region.

NOTE: Applications dependent on 32-word or 64-byte alignment (WS.64B = 1) may not be compatible with future software products. To avoid future incompatibility, programmers should write applications adaptable to either alignment requirement. The bit setting of WS.64B could be a parameter chosen at assembly (by means of a prefix file), at task build (as input to the GBLDEF option), or at runtime (by means of command input).

MACRO CALL

```
CRAW$    wdb
```

where:

- wdb - is the window definition block address

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
iwdb(1),bits 8-15	W.NAPR	Base APR of the address window to be created
iwdb(3)	W.NSIZ	Desired size, in 32-word blocks, of the address window
iwdb(4)	W.NRID	ID of the region to which the new window is to be mapped, or 0 for task region (to be specified only if WS.MAP=1)
iwdb(5)	W.NOFF	Offset in 32-word blocks from the start of the region at which the window is to start mapping (to be specified only if WS.MAP=1). Note that if WS.64B in the window status word equals 0, the value specified must be a multiple of 8.
iwdb(6)	W.NLEN	Length in 32-word blocks to be mapped, or 0 if the length is to default to either the size of the window or the space remaining in the region, whichever is smaller (to be specified only if WS.MAP=1)
iwdb(7)	W.NSTS	Bit settings in the window status word: WS.MAP - 1 if the new window is to be mapped WS.WRT - 1 if the mapping assignment is to occur with write access WS.64B - 0 for 256-word (512-byte) alignment, or 1 for 32-word (64-byte) alignment

Output Parameters

Array Element	Offset	
iwdb(1),bits 0-7	W.NID	ID assigned to the window
iwdb(2)	W.NBAS	Virtual address base of the new window
iwdb(6)	W.NLEN	Length, in 32-word blocks, actually mapped by the window
iwdb(7)	W.NSTS	Bit settings in the window status word: WS.CRW - 1 if the address window was successfully created WS.ELW - 1 if any address windows were eliminated WS.UNM - 1 if any address windows were unmapped

This directive will fail if a conflicting address window is mapped and the task has I/O in progress.

LOCAL SYMBOL DEFINITIONS

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

- C.RABA - (Length 2 bytes) Window definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.PRI	-16	Requested access denied at mapping stage
IE.IOP	-83	Conflicting window has I/O in progress
IE.ALG	-84	Window conficts with window 0 or window start APR and length are inconsistent; or WS.64B = 0 and the value of W.NOFF is not a multiple of 8.
IE.NVR	-86	Invalid region ID
IE.ADP	-98	Part of the DPB or WDB is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
CRAW$     WDBADR
.BYTE     117.,2      ;CRAW$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     WDBADR      ;WDB ADDRESS
```

FORTRAN CALL

```
CALL CRAW (iwdb[,ids])
```

where:

- iwdb - is an 8-word integer array containing a window definition block
- ids - is an integer variable to receive the Directive Status Word.

CRRG$

The **CREATE REGION** directive creates a dynamic region in a system-controlled partition and optionally attaches it to the issuing task.

If RS.ATT is set in the region status word, the Executive attempts to attach the task to the newly created region. No access protection checking is done when attaching the task. If no region name is specified, you must set RS.ATT. See the description of the ATTACH REGION directive.

By default, the Executive automatically marks a dynamically created region for deletion when the last task detaches from it. To override this default condition, you can set RS.NDL in the region status word as an input parameter. Note that programmers must take great care when overriding the delete-on-last-detach option. Uncontrolled creation of regions which are not subsequently deleted would seriously deplete system resources (in particular, swap space and the node pool).

If the region is not given a name, the Executive effectively ignores the state of RS.NDL. All unnamed regions are deleted when the last task detaches from them.

If a name is specified and another region exists with the same name, a new region is not created but the directive does not fail. RS.CRR is cleared in the RDB upon completion. If RS.ATT is set, the task is attached to the existing region, subject to it having the appropriate access rights.

Exceptionally, the directive will fail if the region being created has a global name (RS.TIS not set) and another region exists with the same name, created locally (RS.TIS set) at another terminal. In this case an error status of IE.PNS is returned.

The Executive returns an error if there is not enough space to contain the region in the specified partition.

This directive requires three nodes from the system node pool. These nodes are not charged to the requesting task. They are returned to the pool when the region is deleted.

In order to prevent the creation of common blocks that are not easily deleted, the system and owner categories are always forced to have delete access, regardless of the value actually specified in the protection word.

MACRO CALL

```
CRRG$   rdb
```

where:

- rdb - is the region definition block address

REGION DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
irdb(2)	R.GSIZ	Size, in 32-word blocks, of the region to be created
irdb(3)(4)	R.GNAM	Name of the region to be created, or 0 for no name
irdb(5)(6)	R.GPAR	Name of the system-controlled partition in which the region is to be allocated, or 0 for the partition in which the task is running
irdb(7)	R.GSTS	Bit settings in the region status word:
		RS.NDL - 1 if the region should not be deleted on last detach
		RS.ATT - 1 if created region should be attached
		RS.RED - 1 if read access is desired on attach
		RS.WRT - 1 if write access is desired on attach
		RS.EXT - 1 if extend access is desired on attach
		RS.DEL - 1 if delete access is desired on attach
		RS.TIS - 1 if a region name is to be local to the terminal on which the task is running
irdb(8)	R.GPRO	Protection word for the region (DEWR,DEWR,DEWR,DEWR)

Output Parameters

Array Element	Offset	
irdb(1)	R.GID	ID assigned to the created region (returned if RS.ATT=1)
irdb(2)	R.GSIZ	Size in 32-word blocks of the attached region (returned if RS.ATT=1)
irdb(7)	R.GSTS	Bit settings in region status word:
		RS.CRR - 1 if region was successfully created

LOCAL SYMBOL DEFINITIONS

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

- C.RRBA - (Length 2 bytes) Region definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.UPN	-01	No nodes available for region descriptor (GCD) node
IE.PTS	-03	Zero size specified or specified size larger than partition
IE.PRI	-16	Attach failed because desired access was not allowed.
IE.NSW	-18	No swap space available for region.
IE.WOV	-85	No attachment descriptors available in task header
IE.PNS	-94	Specified partition in which the region was to be allocated does not exist; or no region name was specified and RS.ATT = 0; or a global region was to be allocated (RS.TIS not set) when a local region (RS.TIS set) of the same name already exists.
IE.ADP	-98	Part of the DPB or RDB is out of issuing task's address space
IE.SDP	-99	DIC or RDB size is invalid

MACRO EXPANSION

```
CRRG$     RDBADR
.BYTE     55.,2       ;CRRG$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD     RDBADR      ;RDB ADDRESS
```

FORTRAN CALL

```
CALL CRRG (irdb[,ids])
```

where:

- irdb - is an 8-word integer array containing a region definition block
- ids - is an integer variable to receive the Directive Status Word

CSRQ$

The CANCEL SCHEDULED REQUESTS directive cancels scheduled requests for task executions. Either all requests to run a specified task can be cancelled, or only those issued for a specified task by another specified task. If the requests to be cancelled are for a multi-user task, they are cancelled only if their TI matches that of the scheduling task.

MACRO CALL

```
CSRQ$  ttsk[,rtsk]
```

where:

- ttsk - is the scheduled (requested) task name
- rtsk - is the task name of the schedule requestor. If "rtsk" is omitted, all requests for "ttsk" are cancelled.

LOCAL SYMBOL DEFINITIONS

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

- C.SRTN - (Length 4 bytes) Target task name in Radix-50
- C.SRRN - (4) Requestor task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.AST	-80	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

or

```
CSRQ$  ALPHA,BETA
.BYTE  25.,5     ;CSRQ$ MACRO DIC, DPB SIZE=5 WORDS
.RAD50 /ALPHA/   ;REQUESTED TASK 'ALPHA'
.RAD50 /BETA/    ;REQUESTOR TASK 'BETA'
```

FORTRAN CALL

Subroutine to issue a CANCEL SCHEDULED REQUESTS directive to cancel all scheduled requests for an indicated task:

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

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

Subroutine to issue a CANCEL SCHEDULED REQUESTS directive to cancel only scheduled requests for an indicated task made by another indicated task:

```
CALL CANOBY (schled,[schler][,ids])
```

where:

- schled - is the task name (Radix-50) of the scheduled task.
- schler - is the task name (Radix-50) of the scheduler task.
- ids - is an integer variable to receive the Directive Status Word.

When a scheduler task is not specified, the issuing task is taken as the scheduler.

DECL$

The DECLARE SIGNIFICANT EVENT directive declares a significant event and, optionally, sets an event flag and reports its state before it was set. Declaring a significant event causes the Executive to scan the list of active tasks and possibly reschedule them. The directive performs four functions:

1 Tests the event flag (if specified)

2 Sets the event flag (if specified)

3 Declares an event

4 Reports in the task's DSW the event flag's polarity prior to being set.

See the *IAS Executive Facilities Reference Manual* for details of events and event flags.

MACRO CALL

```
DECL$   [efn]
```

where:

- efn - is an event flag number (an event flag number of 0 implies no event flag number)

LOCAL SYMBOL DEFINITIONS

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

- D.CLEF - (Length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Declaration successful, no event flag specified
IS.CLR	+0	Specified flag was previously cleared
IS.SET	+2	Specified flag was previously set
IE.PRI	-16	Significant event cannot be declared because of directive privilege violation
IE.IEF	-97	Event flag number is invalid (event flag number <0 or >64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
DECL$  40
.BYTE  35.,2     ;DECL$ MACRO DIC ; DPB SIZE=2 WORDS
.WORD  40        ;EVENT FLAG 40

     or

DECL$
.BYTE 35.,1      ;DECL$ MACRO DIC ; DPB SIZE=1 WORD
```

FORTRAN CALL

```
CALL DECLAR ([iefn][,ids])
```

where:

- iefn - is an integer containing an Event Flag Number.
- ids - is an integer variable to receive the Directive Status Word.

DSBL$

The DISABLE directive instructs the system to reject future attempts to run or fix an indicated task (REQUEST, EXECUTE, SCHEDULE, SPAWN, RUN, SYNCHRONIZE, and FIX-IN-MEMORY directives). Along with ENABLE, you can use it to temporarily disallow a task's execution without removing it from the system. Any versions of the task which may already be active are unaffected by this directive.

MACRO CALL

```
DSBL$  tsk
```

where:

- tsk - is the name of the task to be disabled.

LOCAL SYMBOL DEFINITIONS

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

- D.SBTN - (Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.ITS	-08	Task is already disabled
IE.CKP	-10	Task is not to be disabled
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
DSBL$   MART
.BYTE   91.,3    ;DSBL$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50  /MART/   ;TASK 'MART'
```

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character receiver-task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

DSCP$

The DISABLE CHECKPOINTING directive makes the issuing task temporarily non-checkpointable. When a checkpointable task's execution is started, checkpointing is not disabled (that is, the task can be checkpointed).

You can, for example, use this directive to prevent a task from being checkpointed while it performs some time-critical activity.

This directive has no effect upon a task running under the control of the IAS Scheduler.

MACRO CALL

```
DSCP$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ITS	-08	Task checkpointing already disabled
IE.CKP	-10	Issuing task not checkpointable
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
DSCP$
.BYTE  95.,1    ;DSCP$ MACRO DIC, DPB LENGTH=1 WORD
```

FORTRAN CALL

```
CALL DISCKP
```

DTRG$

The DETACH REGION directive detaches the issuing task from a specified, previously attached region. Any of the task's windows that are currently mapped to the region are automatically unmapped.

If RS.MDL is set in the region status word when you issue the directive, the task marks the region for deletion on the last detach. A task must be attached with delete access to mark a region for deletion.

This directive fails if any windows are mapped to the region and the task has I/O in progress.

MACRO CALL

```
DTRG$   rdb
```

where:

- rdb - is the region definition block address

REGION DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
irdb(1)	R.GID	ID of the region to be detached
irdb(7)	R.GSTS	Bit settings in the region status word:
		RS.MDL - 1 if the region should be marked for deletion when the last task detaches from it

Output Parameters

Array Element	Offset	
irdb(7)	R.GSTS	Bit settings in the region status word:
		RS.UNM - 1 if any windows were unmapped

LOCAL SYMBOL DEFINITIONS

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

- D.TRBA - (Length 2 bytes) Region definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.PRI	-16	The task, which is not attached with delete access, has attempted to mark the region for deletion on the last detach.
IE.IOP	-83	Task has I/O in progress
IE.NVR	-86	The task specified an invalid region ID or attempted to detach a permanently attached region
IE.ADP	-98	Part of the DPD or RDB is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
DTRG$     RDBADR
.BYTE     59.,2     ;DTRG$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     RDBADR    ;RDB ADDRESS
```

FORTRAN CALL

```
CALL DTRG (irdb[,ids])
```

where:

- irdb - is an 8-word integer array containing a region definition block
- ids - is an integer variable to receive the Directive Status Word

ELAW$

The ELIMINATE ADDRESS WINDOW directive deletes an existing address window, unmapping it first if necessary. Subsequent use of the eliminated window's ID is invalid.

This directive fails if the window is currently mapped and the task has I/O in progress.

MACRO CALL

```
ELAW$   wdb
```

where:

- wdb - is the window definition block address

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
iwdb(1) bits 0-7	W.NID	ID of the address window to be eliminated

Output Parameters

Array Element	Offset	
iwdb(7)	W.NSTS	Bit settings in the window status word: WS.ELW - 1 if the address window was successfully eliminated WS.UNM - 1 if the address window was unmapped

LOCAL SYMBOL DEFINITIONS

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

- E.LABA - (Length 2 bytes) Window definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.IOP	-83	Task has I/O in progress
IE.NVW	-87	Invalid address window ID
IE.ADP	-98	Part of the DPB or WDB is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
ELAW$     WDBADR
.BYTE     119.,2      ;ELAW$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     WDBADR      ;WDB ADDRESS
```

FORTRAN CALL

```
CALL ELAW (iwdb[,ids])
```

where:

- iwdb - is an 8-word integer array comprising a window definition block
- ids - is an integer variable to receive the Directive Status Word

EMST$$

The EMIT STATUS directive returns the specified 16-bit quantity to the specified connected task. It sets an event flag or declares an AST if previously specified by the connected task in a SPAWN or a CONNECT directive. In any case, whenever status is emitted to one or more tasks, those tasks no longer remain connected to the task issuing the Emit Status directive.

MACRO CALL

```
EMST$ [tname],status
```

where:

- tname - is the name of a task connected to the issuing task to which the status is to be emitted. the default is all.
- status - is a 16-bit quantity to be returned to the connected task.

LOCAL SYMBOL DEFINITIONS

The following symbols are defined with the assigned values equal to the byte offset from the start of the DPB to the DPB elements:

- E.MSTN - (Length 4 bytes) Task name
- E.MSST - (Length 2 bytes) Status to be returned
- C.MKFL - (Length 2 bytes) Cancel unserviced ASTs flag

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ITS	-08	The specified task is not connected to the issuing task.
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
EMST$ tname,status
.BYTE   147.,4     ;EMST$ MACRO DIC,DPB SIZE = 4 WORDS
.WORD   STATUS     ;VALUE OF STATUS TO BE RETURNED
```

FORTRAN CALL

```
CALL EMST([RTNAME], STATUS, [IDS])
```

where:

- tname - is the name of a task connected to the issuing task to which the status is to be emitted. The default is all.
- status - is a 16-bit quantity to be returned to the connected task.
- ids - is an integer to receive the directive status word

ENAR$

The ENABLE AST RECOGNITION directive allows recognition of asynchronous system traps for the issuing task (that is, reverses an INHIBIT AST RECOGNITION directive). ASTs that have occurred while recognition was inhibited are initiated as soon as the AST recognition is enabled. (AST recognition cannot be enabled while the task is in an AST service routine.) See the *IAS Executive Facilities Reference Manual* for a description of ASTs.

MACRO CALL

```
ENAR$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ITS	-08	AST recognition not inhibited
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
ENAR$
.BYTE     101.,1
```

FORTRAN CALL

```
CALL ENASTR
```

ENBL$

The ENABLE directive instructs the system to make a disabled task runnable, that is, to reverse the effect of a DISABLE directive.

MACRO CALL

```
ENBL$  tsk
```

where:

- tsk - is the name of the task to be enabled

LOCAL SYMBOL DEFINITIONS

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

- E.NBTN - (Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.ITS	-08	Task is not disabled
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
ENBL$   LOL09
.BYTE   93.,3   ;ENBL$ MACRO DIC, DPB LENGTH=3 WORDS
.RAD50  /LOL09/ ;TASK 'LOL09'
```

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character receiver task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

ENCP$

The ENABLE CHECKPOINTING directive reverses the effect of the DISABLE CHECKPOINTING (DSCP$) directive (that is, it allows the task to be checkpointed again). The directive is only valid for a task that was built checkpointable and you cannot use it to make a non-checkpointable task checkpointable.

MACRO CALL

```
ENCP$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ITS	-08	Checkpointing not disabled
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
ENCP$
.BYTE  97.,1    ;ENCP$ MACRO DIC, DPB SIZE=1 WORD
```

FORTRAN CALL

```
CALL ENACKP
```

EXEC$

The **EXECUTE** directive activates a task only if the memory required for its execution is presently available.

If sufficient memory cannot be immediately found for the task and any associated SGAs without having to checkpoint any other tasks, the directive fails. Thus, upon completion of this directive, the requested task has all memory allocated and will start to run (subject to its priority) as soon as it has been loaded.

Note that it may take a significant amount of time for this directive to be completed. This is because the requested task must be loaded before it can be known how much memory is required for its associated SGAs (if any). While the requested task is being loaded, the task which issued the directive is effectively suspended and cannot run. In particular, ASTs will not be serviced while completion of the directive is pending.

This directive requires either three or four nodes from the system node pool, which are charged to the issuing task. The nodes are released when the executed task exits.

This directive can specify a partition name to override the task's default partition and a priority to override the task's default priority. If the issuing task is executive privileged, the directive can specify a UIC to override the task's default UIC. If the issuing task is non-privileged, the directive can specify a UIC under which the task will be run, provided that the UIC is:

1 The UIC of the executing task, or

2 The logged-in UIC for the terminal.

MACRO CALL

```
EXEC$  tsk,[prt],[pri][,ugc,umc]
```

where:

- tsk - is the task name
- prt - is the partition name
- pri - is the priority
- ugc - is the UIC group code
- umc - is the UIC member code

  A partition cannot be specified for a multi-user task; that is, the task must be requested to execute in its default partition.

LOCAL SYMBOL DEFINITIONS

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

- E.XCTN - (Length 4 bytes) Task name in Radix-50
- E.XCPN - (4) Partition name
- E.XCPR - (2) Priority
- E.XCGC - (1) UIC group name
- E.XCPC - (1) UIC member code

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.INS	-02	Task not installed
IE.PTS	-03	No memory for execution
IE.HWR	-06	Handler task not resident to load task
IE.ACT	-07	Task is active
IE.ITS	-08	Task is disabled
IE.PRI	-16	Directive privilege violation
IE.IUI	-91	Invalid UIC. Executive privilege is required to cause another task to execute under a changed UIC
IE.PNS	-94	Partition not in system
IE.IPR	-95	Invalid priority specified(<0 or >250)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
EXEC$   LEEDSU,PART,30,200,200
.BYTE   13.,7         ;EXEC$ MACRO DIC, DPB SIZE=7 WORDS
.RAD50  /LEEDSU/      ;TASK 'LEEDSU'
.RAD50  /PART/        ;PARTITION 'PART'
.WORD   30            ;PRIORITY 30
.BYTE   200,200       ;UIC [200,200]
```

FORTRAN CALL

```
CALL EXECUT (tsk,[iop][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (1st Half)
  - — iop(2) - Radix-50 partition name (2nd Half)
  - — iop(3) - Run priority
  - — iop(4) - UIC (User Identification Code). High byte=group code, low byte=member code.

  NOTE: The iop arguments (1), (2), (3) and (4) default to zero when none specified.

- ids - is an integer to receive the Directive Status Word.

EXIF$

The EXITIF directive terminates the execution of the issuing task if an indicated event flag is NOT set. Control is returned if the specified event flag is set. If the exit is taken, a significant event is declared.

The EXITIF directive is useful in avoiding a possible "race condition" that can occur between tasks communicating by means of global event flags. The race condition occurs when one task tests an event flag and finds the flag clear, but before the task can EXIT the other task sets the global flag. Since the first task is in the process of exiting, the event flag is not recognized. This condition can be avoided if the task executes an EXITIF specifying the same common event flag.

If the exit is taken, task resources are freed, in particular:

1 All attached devices are detached

2 The Asynchronous System Trap (AST) queue is flushed

3 The clock queue is flushed for outstanding Mark Time requests for the task

4 The receive and receive-by-reference queues are flushed (unless the task has been built with the "do not flush receive queues" attribute)

5 All open files are closed, and those opened for write left in a locked state

6 I/O queues are flushed

7 All attached regions are detached, and

8 If the task is not fixed, its memory is freed.

If the task exits, the Executive declares an event.

MACRO CALL

```
EXIF$  efn
```

where:

- efn - is an event flag number

LOCAL SYMBOL DEFINITIONS

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

- E.XFEF - (length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SET	+2	Indicated event flag set, task not exited
IE.IEF	-97	No event flag specified in mask word(s), or invalid event flag number (event flag number <1 or >64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
EXIF$  40
.BYTE  53.,2        ;EXIF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD  40           ;EVENT FLAG NUMBER 40
```

FORTRAN CALL

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

where:

- iefn - is an integer containing an Event Flag Number.
- ids - is an integer variable to receive the Directive Status Word.

EXIT$

The **TASK EXIT** directive terminates the execution of the issuing task. If the exit is taken, the Executive declares a significant event.

A return to the task occurs if (and only if) the directive is rejected. On EXIT, the Executive frees task resources; in particular:

1 All attached devices are detached

2 The Asynchronous System Trap (AST) queue is flushed

3 The clock queue is flushed for outstanding Mark Time requests for the task

4 The receive and receive-by-reference queues are flushed (unless the task has been built with the "do not flush receive queues" attribute)

5 All open files are closed, and those opened for write are left in a locked state

6 I/O queues are flushed

7 All attached regions are detached, and

8 If the task is not fixed, its memory is freed.

MACRO CALL

```
EXIT$
```

DSW RETURN CODES

The DSW Return Codes can be tested only if the EXIT$ directive fails.

Code	Value Returned	Explanation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

```
STOP
```

or

```
CALL EXIT
```

Note that either the STOP statement or CALL EXIT subroutine is used to terminate a task.

EXST$

The **TASK EXIT WITH STATUS INDICATION** directive terminates the execution of the issuing task and enables it to return a success or failure indication to the system. The status returned by EXST$ can be used by the ON command in an indirect or batch command file to take steps conditional on the status. See the *IAS PDS User's Guide* for details of the ON command.

For a real-time task initiated by the SPAWN directive, the status is returned to the requesting task via its exit status block.

The status given must be one of the four values EX$SUC, EX$WAR, EX$ERR and EX$SEV, which are defined by all the different forms of the EXST$ macro. They should be used as follows:

- EX$SUC - program has succeeded. All the results (for example, output files) will be as expected.)
- EX$WAR - program has succeeded but diagnostic errors have occurred. The results have been generated but may not be as expected.)
- EX$ERR - program has succeeded but errors have occurred. The results have been generated but probably will not be as expected.
- EX$SEV - program has failed, that is, fatal errors have occurred. The results have not been generated. This status is also given if the task is aborted for any reason.)

If the task exits, the Executive declares a signifcant event.

When a task exits, the Executive frees task resources; in particular:

1 All attached devices are detached

2 The Asynchronous System Trap (AST) queue is flushed

3 The clock queue is flushed for outstanding Mark Time requests for the task

4 The receive and receive-by-reference queues are flushed (unless the task has been built with the 'do not flush receive queues' attribute)

5 All open files are closed and those opened for write left in a locked state

6 I/O queues are flushed

7 All attached regions are detached, and

8 If the task is not fixed, its memory is freed.

MACRO CALL

```
EXST$ stat
```

where:

- stat - is one of the status values defined above.

LOCAL SYMBOL DEFINITIONS

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

- E.XSTS - (length 2 bytes) Exit status

DSW RETURN CODES

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

Code	Value Returned	Explanation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
EXST$   EX$SUC
.BYTE   29.,2       ;  EXST$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   EX$SUC      ;  STATUS
```

FORTRAN CALL

Not implemented.

EXTK$

The **EXTEND TASK** directive instructs the system to modify the size of the read/write root segment of the issuing task by a positive or negative increment of 32-word blocks. If the directive does not specify an increment value, the Executive makes the issuing task equal in size to the installed task size. A real-time task must be checkpointable to modify its size. If necessary, the Executive will checkpoint the task, returning it to memory after modifying the size as directed.

If the task is still performing I/O and an attempt is made to reduce its size, the directive fails. Failure also occurs if the directive is issued by a scheduler-controlled task and there is not enough additional space in a swap file.

The EXTEND TASK directive limits the size to which a task can extend itself as follows:

1 No task can extend itself beyond any of the following:

- The maximum size set by the SCI command SET EXTENDED_TASK_SIZE/MAXIMUM.
- The maximum size set by the MCR command SET /MAXEXT.
- The maximum extension set by the task builder keyword MAXEXT=.
- The size of the partition in which the task is running.
- See the *IAS System Management Guide*, the *IAS MCR User's Guide* and the *IAS Task Builder Reference Manual* for more details.

2 The availability of consecutive APRs determines the limit of a task's extension. Eight APRs, each covering 4K words, map the 32K words of address space. Each region mapped by the task (including task read-only areas and memory-resident overlays) needs an APR for each 4K (or part 4K) words. APRs for read-only areas, memory-resident overlays and position-independent libraries (such as SYSRES) are, by default, allocated by the task builder at the high end of the task's virtual address space.

MACRO CALL

```
EXTK$ [inc]
```

where:

- inc is a positive or negative number equal to the number of 32-word blocks by which the task size is to be extended or reduced. The default value of zero causes the task to revert to its installed size.

LOCAL SYMBOL DEFINITIONS

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

- E.XTIN - (Length 2 bytes) Extend increment

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.NSW	-18	There is insufficient swap space for a scheduler controlled task or a real-time task that has to be checkpointed.
IE.ALG	-84	The issuing task attempted one of the following: 1 To reduce its size to less than that of its task header plus 32-words. 2 To increase its size beyond 32K words. 3 To increase its size beyond the maximum set by the SCI SET EXTENDED_TASK_SIZE/MAXIMUM or MCR SET /MAXEXT command. 4 To increase its size in excess of the limit set by the task builder MAXEXT option. 5 To increase its size so that one virtual address window would overlap another.
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB space is invalid.
IE.CKP	-10	Task not checkpointable, or checkpointing is disabled. (Real-time tasks only.)
IE.IOP	-83	Task with I/O in progress is trying to reduce its size.

MACRO EXPANSION

```
EXTK$    40
.BYTE    89.,3     ;EXTK$ MACRO DIC,DPB SIZE=3 WORDS
.WORD    40        ;EXTEND INCREMENT, 40 (8) BLOCKS (of 32 words each)
.WORD    0         ;RESERVED WORD
```

FORTRAN CALL

```
CALL EXTTSK (inc[,ids])
```

where:

- inc - is a positive or negative number equal to the number of 32-word blocks by which the task is to be extended or reduced. A value of zero causes the task to revert to its installed size.
- ids - is an integer variable to receive the Directive Status Word.

FIX$

The FIX-IN-MEMORY directive fixes an inactive, installed task in memory. Once fixed in memory, it does not relinquish its memory space until removed by the UNFIX directive. Only tasks built as non-checkpointable may be fixed.

This directive is particularly useful when the speed of task execution is critical, or when execution is frequently requested. A fixed task does not have to be reloaded from disk every time it is requested. Further, there is no need to find space in memory, by checkpointing other task's, to load it. See the *IAS Executive Facilities Reference Manual* for further details of fixed tasks. For this directive to succeed, the specified task must have been built so that it can be fixed and checkpointed.

Make sure there is task code to reset variable data areas to their initial values when executing a fixed task. This is necessary because the values will have been modified by the previous execution. Fixing a task will not stop it being shuffled in memory.

This directive requires three nodes from the system node pool. These nodes are charged to the issuing task and are released when the task is unfixed.

MACRO CALL

```
FIX$  tsk
```

where:

- tsk - is the name of the task to be fixed in memory

LOCAL SYMBOL DEFINITIONS

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

- F.IXTN - (Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.INS	-02	Task not installed
IE.PTS	-03	Partition too small for task
IE.HWR	-06	Handler task not resident to load task
IE.ACT	-07	Task is active
IE.ITS	-08	Task is disabled
IE.FIX	-09	Task is already fixed
IE.CKP	-10	Task not fixable
IE.TCH	-11	Task is checkpointable
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
FIX$    TASK2
.BYTE   85.,3    ;FIX$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50  /TASK2/ ;TASK 'TASK2'
```

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

GCOM$

The GET COMMON BLOCK PARAMETERS directive fills an indicated 8-word buffer with parameters giving information in a named SGA. The SGA can be a global library or global common area.

The 8-word buffer is filled as follows:

- WD. 00 - Base address of common block (SGA) in 32-word blocks
- WD. 01 - Size of common block in 32-word blocks
- WD. 02 - Creation year
- WD. 03 - Creation month (low byte) and day (high byte)
- WD. 04 - Global Common Directory status (low byte) and Active Page Register number (high byte)
- WD. 05 - User Identification Code (UIC). High byte = group code, low byte = member code.
- WD. 06 - Task Partition Directory address
- WD. 07 - Common block flags word

MACRO CALL

```
GCOM$  blk,buf
```

where:

- blk - is the name of the common block
- buf - is an address of the 8-word buffer

LOCAL SYMBOL DEFINITIONS

The following symbols are locally defined with their assigned value equal to the byte offset from the start of the DPB to the respective DPB element. The length of the element in bytes is in round brackets.

- G.COBN - (Lenth 4 bytes) Common block (SGA) name
- G.COBA - (2) Buffer address (see below)

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

- G.COBB - (2) Common block base address
- G.COBS - (2) Common block size
- G.COYR - (2) Creation year

- G.COMO - (1) Creation month
- G.CODA - (1) Creation day
- G.COST - (1) Global Common Directory (GCD) status byte
- G.COSA - (1) Starting Active Page Register (APR)
- G.COUI - (2) User Identification Code (UIC)
- G.COTP - (2) Task Partition Directory address
- G.COFW - (2) Common block flags word address

The following bits are defined for the flags word:

Symbol	Bit	Meaning When Set
GF.SG	0	SGA flag (set when region must be loaded from task image file).
GF.LI	1	1 = library, 0 = common area.
GF.RI	2	Library relocatability indicator (set for position-independent code).
	3	(Reserved).
GF.FT	4	Region not yet loaded - do not read from swap file.
GF.PA	5	Region is task's pure area.
GF.IR	6	Region is installed region.
GF.DE	7	Region is marked for delete.
GF.TI	8	Region's name is TI dependent.
GF.RW	9	Region is task's read/write resident overlay region.
GF.PS	10	Region has permanently allocated swap space.

The following values are defined for the status byte:

Symbol	Meaning
GS.NUL	Global area not in use
GS.LRQ	Load request queued
GS.LRS	Load request succeeded
GS.LRF	Load request failed
GS.RRF	Record request queued
GS.RRS	Record request succeeded
GS.RRF	Record request failed

The symbols beginning GF. and GS. (above) may be defined in a user task by including the definition file [1,1]EXEC.STB/SS when the task is built.

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Indicated SGA not in system
IE.ADP	-98	Part of DPB or buffer is out of issuing task's address space

MACRO EXPANSION

```
GCOM$     SYSRES,COMBUF
.BYTE     67.,4      ;GCOM$ MACRO DIC,DPB SIZE =4 WORDS
.RAD50    /SYSRES/   ;COMMON BLOCK NAME
.WORD     COMBUF     ;ADDRESS OF 8-WORD BUFFER
```

FORTRAN CALL

```
CALL GETCMN (cmn,ibuf[,ids])
```

where:

- cmn - is a 1- to 6- character SGA name in Radix-50 form.
- ibuf - is an 8-word integer array to receive Common Block Parameters.
- ids - is an integer variable to receive the Directive Status Word.

GLUN$

Use the GET LUN INFORMATION directive to obtain information about the device to which a logical unit is assigned. A specified 6-word buffer is filled as described below. The device to which the LUN is assigned may have been redirected to another device. In this case, the information returned will refer to the device to which I/O requests would actually be queued.

The buffer is filled as follows:

- WD.00 - Name of Assigned Device (2 ASCII characters)
- WD.01 - Unit Number of Assigned Device and flags byte
- WD.02 - First Device Characteristics Word
  - — UC.REC - Bit 0 - Record Oriented Device (1=yes)
  - — UC.CCL - Bit 1 - Carriage Control Device (1=yes)
  - — UC.TTY - Bit 2 - Terminal Device (1=yes)
  - — UC.DIR - Bit 3 - Directory Device (1=yes)
  - — UC.SDI - Bit 4 - Single Directory Device (1=yes)
  - — UC.SQD - Bit 5 - Sequential Device (1=yes)
  - — UC.IAS - Bit 6 - Interactive IAS terminal (1=yes)
  - — UC.IEX - Bit 7 - IAS exclusive device (1=yes)
  - — UC.INB - Bit 8 - Intermediate buffered (1=yes)
  - — UC.SWL - Bit 9 - Software write locked (1=yes)
  - — UC.ISP - Bit 10 - Input spooled (1=yes)
  - — UC.OSP - Bit 11 - Output spooled (1=yes)
  - — UC.PSE - Bit 12 - Pseudo Device (1=yes)
  - — UC.COM - Bit 13 - Device mountable as a Communications Channel (1=yes)
  - — UC.F11 - Bit 14 - Device mountable as a Files-11 device (1=yes)
  - — UC.MNT - Bit 15 - Device mountable (1=yes)
- WD.03 - Second Device Characteristics Word (See the *IAS Device Handlers Reference Manual*)
- WD.04 - Third Device Characteristics Word (See the *IAS Device Handlers Reference Manual*).
- WD.05 - Standard device buffer size

The second and third Device Characteristic Words are specific to the device handler.

If the device to which the LUN is assigned is set output spooled, or if this device is redirected to a spooled device, the buffer is filled as follows:

- WD.00 - Name of assigned device
- WD.01 - Unit number of assigned device, and flags byte for spooler temporary device (SP0).
- WD.02 - First device characteristics word for spooler temporary device, with UC.OSP set

- WD.03 - Second device characteristics word for spooler temporary device.
- WD.04 - Standard buffer size for assigned device
- WD.05 - Standard buffer size for spooler temporary device.

MACRO CALL

```
GLUN$  lun,buf
```

where:

- lun - is a logical unit number
- buf - is the address of a 6-word buffer which holds LUN information

LOCAL SYMBOL DEFINITIONS

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

- G.LULU - (Length 2 bytes) Logical unit number
- G.LUBA - (2) Buffer address (see below)

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

- G.LUNA - (2) Name of assigned device
- G.LUNU - (1) Unit number of assigned device
- G.LUFB - (1) Flags byte
- G.LUCW - (8) Device characteristics word

The following bits are defined for the flags byte:

Symbol	Bit	Meaning if set
UF.OFL	5	Device is off-line
UF.TL	6	Handler task recognizes load and record
UF.RH	7	Handler task is declared resident

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ULN	-05	Unassigned LUN
IE.ILU	-96	Invalid logical unit number
IE.ADP	-98	Part of DPB or buffer is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

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
```

FORTRAN CALL

```
CALL GETLUN (ilun,idata[,ids])
```

where:

- ilun - is an integer containing a Logical Unit Number.
- idata - is a 6-word integer array to receive LUN information.
- ids is an integer variable to receive the Directive Status Word.

GMCR$

The GET MCR COMMAND LINE directive instructs the system to transfer an 80-byte command line to the issuing task.

A command line can be present in the following circumstances:

1 The task was requested by the MCR dispatcher.

2 The task was initiated by another task by means of the SPAWN TASK (SPWN$) directive.

3 The task was initiated by a timesharing task using the RUN$T macro of TCS (see the *IAS Guide to Writing Command a Language Interpreter.*

The command line is copied into the DPB itself and there is, therefore, no $S or $C form of the macro.

The length of the command line is returned in the task's Directive Status Word (DSW). The line will also be terminated, usually with a carriage return (octal code 15), but possibly with an escape (octal code 33). The count does not include this terminator.

MACRO CALL

```
GMCR$
```

LOCAL SYMBOL DEFINITIONS

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

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

DSW RETURN CODES

Code	Value Returned	Explanation
+n		Successful completion; n is the number of data bytes transferred (excluding the termination character). The termination character is, however, in the buffer.
IE.AST	-80	No command line exists for the issuing task or the command line has already been read by a previous MCR COMMAND LINE directive.
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- buf - is an 80-byte array to receive the command line
- ids - is an integer variable to receive the Directive Status word

GMCX$

The GET MAPPING CONTEXT directive causes the Executive to return a description of the current window-to-region mapping assignments. The returned description is in a form that enables you to restore the mapping context described by a series of CREATE ADDRESS WINDOW directives. The macro argument specifies the address of a vector that contains one 8-word window definition block (WDB) for each currently valid window, plus a terminator word. Thus, the maximum space required for the vector is 8*8+1=65 words.

For each valid window the Executive fills in an 8-word WDB in the vector as follows:

1 If window is currently unmapped, the Executive fills in the offsets W.NID, W.NAPR, W.NBAS, and W.NSIZ with information sufficient to recreate the window. The window status word W.NSTS is cleared.

2 If a window is currently mapped, the Executive fills in the offsets W.NAPR, W.NBAS, W.NSIZ, W.NRID, W.NOFF, W.NLEN, and W.NSTS with information sufficient to create and map the address window. WS.MAP is set in the status word (W.NSTS), and if the window is mapped with write access, the bit WS.WRT is set as well.

3 For window zero, (task window), the Executive fills in the offsets as for an unmapped window, plus the mapped length W.NLEN. The window status word, W.NSTS, is left clear.

Note that, in all cases, the word at offset W.NSRB is cleared

The terminator word, which follows the last WDB filled in, is set negative. Because the first word of a WDB is positive, a "TST" instruction can be used to detect the last WDB in the vector.

When you use CREATE ADDRESS WINDOW (CRAW$) directives to restore the mapping context, there is no guarantee that the same address window IDs will be used. You must, therefore, be careful to use the latest window IDs returned from the CREATE ADDRESS WINDOW directives.

For compatibility with other operating systems, the first WDB in the vector always describes window zero (that is, the task window). This window cannot be unmapped, eliminated or re-created. The CREATE ADDRESS WINDOW directive will fail when applied to this WDB, but the error should be ignored.

MACRO CALL

```
GMCX$    vec
```

where:

- vec - is the address of a vector of up to 8 window definition blocks, each of 8 words, followed by a terminator word.

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

NONE

Output Parameters

Array Element	Offset	
iwdb(1)	W.NID	ID of address window bits 0-7
iwdb(1)	W.NAPR	Base APR of the window bits 8-15
iwdb(2)	W.NBAS	Base virtual address of the window
iwdb(3)	W.NSIZ	Size, in 32-word blocks, of the window
iwdb(4)	W.NRID	ID of the mapped region, or no change if the window is unmapped or is window zero
iwdb(5)	W.NOFF	Offset, in 32-word blocks, from the start of the region at which mapping begins, or no change if the window is unmapped or is window zero
iwdb(6)	W.NLEN	Length, in 32-word blocks, of the area currently mapped within the region, or no change if the window is unmapped
iwdb(7)	W.NSTS	Bit settings in the window status word (all 0 if the window is not mapped or is window zero):
		WS.MAP - 1 if the window is mapped
		WS.WRT - 1 if the window is mapped with write access
		WS.64B - 1 if the window is not aligned on a 256 word boundary
iwdb(8)	W.NSRB	Address of Send/Receive by Reference buffer. (This directive always clears the Send/Receive by Reference buffer)

Note that the length mapped (W.NLEN) can be less than the size of the window (W.NSIZ) if the area from W.NOFF to the end of the partition is smaller than the window size.

LOCAL SYMBOL DEFINITIONS

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

- G.MCVA - (Length 2 bytes) Address of the vector (wvec) containing the window definition blocks and terminator word

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.ADP	-98	Address check of the DPB or the vector (wvec) failed
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
GMCX$     VECADR
.BYTE     113.,2     ;GMCX$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     VECADR     ;WDB VECTOR ADDRESS
```

FORTRAN CALL

```
CALL GMCX (imcx[,ids])
```

where:

- imcx - is an integer array to receive the mapping context. The size of the array is 8*n+1 where n is the number of window blocks in the task's header. The maximum size is 8*8*+1=65 words.
- ids - is an integer variable to receive the Directive Status Word.

GPRT$

The GET PARTITION PARAMETERS directive fills a specified 3-word buffer with information about a specified partition. If you do not specify a partition name, the partition in which the issuing task is running is assumed.

The 3-word buffer is filled as follows:

- WD. 00 - Base address of partition in 32-word blocks
- WD. 01 - Size of partition in 32-word blocks
- WD. 02 - partition flags byte

This directive indicates success by returning a value of zero in the Directive Status Word (DSW) rather than IS.SUC.

MACRO CALL

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

where:

- prt - is the partition name
- buf - is the address of the buffer

LOCAL SYMBOL DEFINITIONS

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

- G.PRPN - (Length 4 bytes) Partition name in Radix-50
- G.PRBA - (2) Buffer address

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

- G.PRPB - (2) Partition base address
- G.PRPS - (2) Partition size
- G.PRFW - (2) Partition flags word

The following bits are defined for the flags word:

Symbol	Bit	Meaning When Set
TF.UC	0	User controlled partition
TF.OU	1	Occupied user-controlled partition
	2	Reserved
	3	Reserved
TF.IA	4	Timesharing partition
TF.SG	5	(Used by system generation)

DSW RETURN CODES

Successful completion is indicated by carry clear. The Directive Status Word (DSW) is set to zero in this case, rather than IS.SUC. Unsuccesful completion is indicated by carry set and one of the following codes in the DSW:

Code	Value Returned	Explanation
IE.INS	-02	Indicated partition not in system
IE.ADP	-98	Part of DPB or buffer is out of task's address space
IE.SDP	-99	DIC or DPB size is invalid

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
```

FORTRAN CALL

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

where:

- prt - is a 2-word, 1 to 6 character partition name in Radix-50 form.
- ibuf - is a 3-word integer array to receive partition parameters.
- ids - is an integer variable to receive the Directive Status Word.

GREG$

The GET REGION PARAMETERS directive instructs the Executive to fill an indicated 3-word buffer with information about the specified region, to which the task must be attached.

MACRO CALL

```
GREG$   rid,buf
```

where:

- rid - is the region ID
- buf - is the address of a 3-word buffer

BUFFER FORMAT

- WD.0 - Region base address expressed as a multiple of 32 words (regions are always aligned on 32-word boundaries). Thus, a region starting at 1000(8) will have 10(8) returned in this word.
- WD.1 - Region size expressed as a multiple of 32-words.
- WD.2 - Region flags word.

LOCAL SYMBOL DEFINITIONS

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

- G.RGID - (Length 2 bytes) Region ID
- G.RGBA - (length 2 bytes) Buffer address

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

- G.RGRB - Region base address expressed as an absolute 32-word block number (2)
- G.RGRS - Region size expressed as a multiple of 32-word blocks (2)
- G.RGFW - Region flags word (2)

The following bits are defined for the flags word:

Symbol	Bit	Meaning When Set
GF.SG	0	Region must be loaded from task image file
GF.LI	1	Region is a library (0=common)
GF.RI	2	Library is position independent
	3	Reserved
GF.FT	4	Region not yet loaded
GF.PA	5	Region is task's pure area
GF.IR	6	Region is "installed region"
GF.DE	7	Region is marked for delete
GF.TI	8	Region's name is TI dependent
GF.RW	9	Region is task's RW resident overlay region
GF.PS	10	Region has permanently allocated swap space

DSW RETURN CODES

Successful completion is indicated by carry clear. The Directive Status Word (DSW) is set to zero in this case, rather than IS.SUC. Unsuccessful completion is indicated by carry set and one of the following codes in the DSW:

Code	Value Returned	Explanation
IE.NVR	-86	Invalid region ID
IE.ADP	-98	Part of the DPB or buffer is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
GREG$     RID,DATBUF
.BYTE     65.,4       ;GREG$ MACRO DIC,DPB SIZE=4 WORDS
.WORD     0           ;WORD THAT DISTINGUISHES GREG$
                      ;FROM GPRT$
.WORD     RID         ;REGION ID
.WORD     DATBUF      ;ADDRESS OF 3-WORD BUFFER
```

FORTRAN CALL

```
CALL GETREG (rid,buf[,ids])
```

where:

- rid - is the region id
- buf - is a 3-word integer array to receive region parameters
- ids - is an integer variable to receive the Directive Status Word.

GSSW$

The GET SENSE SWITCHES directive instructs the system to get the status of the console sense switches and store the value in the issuing task's Directive Status Word. For processors which do not have console sense switches, this will be the value set by using the SET/SWR (MCR) or SET SWITCH_REGISTER (SCI) command. See the *IAS MCR User's Guide* for details of the SET /SWR command or the *IAS System Management Guide* for details of the SET SWITCH_REGISTER command. The presence or absence of a hardware switch register is determined when the system is bootstrapped.

MACRO CALL

```
GSSW$
```

DSW RETURN CODES

Successful completion is indicated if the carry condition code is clear. Switch values will be found in the DSW. Unsuccessful completion is indicated by the carry condition code set and one of the following codes in the DSW:

Code	Value Returned	Explanation
IE.SDP	-98	Part of DPB is out of issuing task's address space
IE.ADP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
GSSW$
.BYTE  125.,1   ;GSSW$ MACRO DIC, DPB SIZE=1 WORD
```

FORTRAN CALL

```
CALL READSW
     or
CALL SSWTCH
```

See READSW (Read Sense Switches) and SSWTCH (Test a Sense Switch) Subroutine calls as described in the *IAS FORTRAN Special Subroutines Reference Manual.*

GTIM$

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

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

MACRO CALL

```
GTIM$  buf
```

where:

- buf - is the address of an 8-word buffer

LOCAL SYMBOL DEFINITIONS

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

- G.TIBA - (Length 2 bytes) Buffer address (see below)

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

- G.TIYR - (2) Year
- G.TIMO - (2) Month
- G.TIDA - (2) Day
- G.TIHR - (2) Hour
- G.TIMI - (2) Minute
- G.TISC - (2) Second
- G.TICT - (2) Clock tick

- G.TICP - (2) Clock ticks per second

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ADP	-98	Part of DPB or buffer is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

FORTRAN IV provides several subroutines for obtaining the time in a number of formats. See the *IAS/RSX-11 Fortran IV User's Guide* or the *FORTRAN IV-PLUS User's Guide.*

GTSK$

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

The 16-word buffer is filled as follows:

- WD. 00 - Issuing task's name (first half) (Radix-50)
- WD. 01 - Issuing task's name (second half) (Radix-50)
- WD. 02 - Partition name (first half) (Radix-50)
- WD. 03 - Partition name (second half) (Radix-50)
- WD. 04 - Name of requestor (first half) (Radix-50)
- WD. 05 - Name of requestor (second half) (Radix-50)
- WD. 06 - Run priority
- WD. 07 - Default UFD for file access. For a real-time task, this is always the same as the task UIC (WD.17). For a timesharing task (a task run using PDS or the Timesharing Control Services, this is the default UFD set by the SET DEFAULT command.
- WD. 10 - Number of logical I/O units (LUNs)
- WD. 11 - Machine type indicator (for example, 60. for PDP-11/60, 70. for PDP-11/70)
- WD. 12 - System Task Directory (STD) flag word
- WD. 13 - Address of task SST vector tables, or zero if word 14 is zero
- WD. 14 - Size of task SST vector table (in words), or zero if none specified
- WD. 15 - Task size. This is the address of the first byte above the task's main read/write area (that is, the first byte which will cause a memory management violation if an attempt is made to access that byte). This value takes no account of the task's read-only area, of shareable libraries and common areas, of dynamically mapped address windows or of Task Builder allocated resident overlays.
- The value reflects any alteration of the read/write area size resulting from use of the Extend Task directive.
- WD. 16 - System identification (see below, G.TSSY)
- WD. 17 - User Identification Code. High byte = group code, low byte = member code. This is the UIC which will be used, for instance, to determine the task's access to files, regions. Note that it is not affected by the SET DEFAULT command command for a timesharing task.

MACRO CALL

```
GTSK$  buf
```

where:

- buf - is the address of a 16-word buffer

LOCAL SYMBOL DEFINITIONS

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

- G.TSBA - (Length 2 bytes) Buffer Address (see below)

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

- G.TSTN - (4) Task name
- G.TSPN - (4) Partition name
- G.TSRN - (4) Name of task's requester
- G.TSPR - (2) Priority
- G.TSPC - (1) Default UFD Member code
- G.TSGC - (1) Default UFD Group code
- G.TSNL - (2) Number of logical units
- G.TSMT - (2) Machine type
- G.TSFW - (2) System Task Directory (STD) flags word
- G.TSVA - (2) Task's SST vector address
- G.TSVL - (2) Task's SST vector (word) length
- G.TSTS - (2) Task size (in bytes)
- G.TSSY - (2) System identification (see below)
- G.TSDU - (2) User Identification Code

The following bits are defined for the System Task Directory Flags Word (G.TSFW):

Symbol	Bit	Meaning When Set
SF.FX	1	Task is fixed in memory
SF.RM	2	STD entry is to be removed

Symbol	Bit	Meaning When Set
SF.TD	3	Task is disabled
SF.BF	4	Task is being fixed in memory
SF.XT	5	Task is to be removed on Exit
SF.MU	6	Task is multiuser
SF.PT	7	Task is privileged
SF.NT	8	Network attribute bit
SF.R1	9	Task does not have directive privilege
SF.XS	10	Task cannot be sent data or references
SF.XA	11	Task cannot be aborted
SF.XD	12	Task cannot be disabled
SF.XF	13	Task cannot be fixed in memory
SF.XC	14	Task cannot be checkpointed.
SF.SR	15	Task can be requested/resumed by SFFR$, VSDR$ or SDRQ$ directives issued by any task.

The following bits are defined for the System Identification (G.TSSY):

- 1 for RSX-11M
- 2 for RSX-11S
- 3 for IAS
- 4 for RSTS
- 5 for VAX/VMS
- 6 for RSX-11M-PLUS

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ADP	-98	Part of DPB or buffer is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- idata - is a 16-word integer array to receive task parameters.
- ids - is an integer variable to receive the Directive Status Word.

IHAR$

The INHIBIT AST RECOGNITION directive inhibits recognition of asynchronous system traps for the issuing task. The ASTs are queued as they occur and are effected when AST recognition is re-enabled (ENAR$). AST recognition is inhibited whenever an AST service routine is executing. ASTs are described in the IAS Executive Facilities Reference Manual.

It is only the recognition of ASTs which is inhibited. The ASTs are still queued by the system. They are queued on a First in/first out (FIFO) basis and occur in that order when AST recognition is re-enabled.

MACRO CALL

```
IHAR$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ITS	-08	AST recognition already inhibited
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
IHAR$
.BYTE   99.,1  ;IHAR$ MACRO DIC,DPB SIZE=1 WORD
```

FORTRAN CALL

```
CALL INASTR
```

MAP$

The MAP ADDRESS WINDOW directive maps an existing window onto an attached region. The mapping begins at a specified offset from the start of the region. If the window is already mapped elsewhere, the Executive unmaps it before carrying out the mapping assignment described in the directive.

For the mapping assignment, a task can specify any length that is less than or equal to both:

1 The window size specified when the window was created

2 The length remaining between the specified offset within the region and the end of the region.

A task must be attached with write access to a region in order to map to it with write access. To map to a region with read-only access, the task must be attached with either read or write access.

If W.NLEN is set to 0, the length defaults to either the window size or the length remaining in the region, whichever is smaller. Because the Executive returns the actual length mapped as an output parameter in W.NLEN, the task must clear that parameter in the WDB before issuing the directive each time it wants to default the length of the map.

The values that can be assigned to W.NOFF depend on the setting of bit WS.64B in the window status word (W.NSTS):

1 If WS.64B = 0, the offset specified in W.NOFF must represent a multiple of 256 words (512 bytes). Because the value of W.NOFF is expressed in units of 32-word blocks, the value must be a multiple of 8.

2 If WS.64B = 1, the task can align on 32-word boundaries; the programmer can therefore specify any offset within the region.

NOTE: Applications dependent on 32-word or 64-byte alignment (WS.64B = 1) may not be compatible with future software products. To avoid future incompatibility, programmers should write applications adaptable to either alignment requirement. The bit setting of WS.64B could be a parameter chosen at assembly (by means of a prefix file), at task build (as input to the GBLDEF option), or at runtime (by means of command input).

This directive will fail if the window is already mapped and the task has I/O in progress.

MACRO CALL

```
MAP$    wdb
```

where:

- wdb - is the window definition block address

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
iwdb(1)	W.NID	ID of the window to be mapped bits 0-7
iwdb(4)	W.NRID	ID of the region to which the window is to be mapped.
iwdb(5)	W.NOFF	Offset, in 32-word blocks, within the region at which mapping is to begin. Note that if WS.64B in the window status word equals 0, the value specified must be a multiple of 8.
iwdb(6)	W.NLEN	Length, in 32-word blocks, within the region to be mapped, or 0 if the length is to default to either the size of the window or the space remaining in the region from the specified offset, whichever is smaller
iwdb(7)	W.NSTS	Bit settings in the window status word: WS.WRT - 1 if write access is desired WS.64B - 0 for 256-word (512-byte) alignment, or 1 for 32-word (64-byte) alignment.

Output Parameters

Array Element	Offset	
iwdb(6)	W.NLEN	Length of the area within the region actually mapped by the window
iwdb(7)	W.NSTS	Bit settings in the window status word: WS.UNM - 1 if the window was unmapped first

LOCAL SYMBOL DEFINITIONS

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

- M.APBA - (Length 2 bytes) Window definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.PRI	-16	Privilege violation
IE.IOP	-83	Task has I/O in progress
IE.ALG	-84	Task specified an invalid region offset and length combination in the window definition block parameters; or WS.64B = 0 and the value of W.NOFF is not a multiple of 8.
IE.NVR	-86	Invalid region ID
IE.NVW	-87	Invalid address window ID
IE.ADP	-98	Part of the DPB or WDB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
MAP$     WDBADR
.BYTE    121.,2      ;MAP$ MACRO DIC, DPB SIZE=2 WORDS
.WORD    WDBADR      ;WDB ADDRESS
```

FORTRAN CALL

```
CALL MAP (iwdb[,ids])
```

where:

- iwdb - is an 8-word integer array containing a window definition block
- ids - is an integer variable to receive the Directive Status Word

MRKT$

The **MARK TIME** directive declares a significant event after an indicated time interval. The interval begins when you issue the directive. If you specify an event flag, it is cleared when you issue the directive and set at the time of the significant event. If you specify an AST service entry point, an asynchronous system trap is queued at the time of the significant event. If you specify neither an event flag number nor an AST service entry point, the significant event is still declared after the indicated time interval.

If an AST entry point address is specified in the DPB, the AST routine is entered (when the indicated time interval has elapsed) 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 (ASTX$) directive is executed.

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 after the MARK TIME directive is rejected, then the task may wait for ever. You must always make sure that the directive has successfully completed.

A task must have real-time directive privilege to issue a MARK TIME directive specifying a global event flag.

If you issue a MARK TIME directive with no AST entry address and the task needs to know of the event, the task should issue a WAITFOR directive specifying the same event flag as for the MARK TIME. ASTs are described in the IAS Executive Facilities Reference Manual.

This directive requires two nodes from the system node pool. These nodes are charged to the issuing task and are released when the Mark Time becomes due.

MACRO CALL

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

where:

- efn - is an event flag number (0 implies no event flag)
- tmg - is a time interval magnitude; that is, how many time interval units

- tnt - is a time interval unit
  - 1 = clock ticks
  - 2 = seconds
  - 3 = minutes
  - 4 = hours
- ast - is an AST entry address

LOCAL SYMBOL DEFINITIONS

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

- M.KTEF - (Length 2 bytes) Event flag number
- M.KTMG - (2) Time interval magnitude
- M.KTUN - (2) Time interval unit
- M.KTAE - (2) AST entry address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Unavailable pool node
IE.ITI	-93	Invalid time specified
IE.IEF	-97	Invalid event flag number (event flag number <0 or >64)
IE.PRI	-16	Privileged function (global flag cannot be set or cleared by a task that does not have real-time directive privilege).
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
MRKT$   24,30,2,MRKAST
.BYTE   23.,5          ;MRKT$ MACRO DIC, DPB SIZE=5 WORDS
.WORD   24             ;EVENT FLAG NUMBER 24
.WORD   30             ;TIME MAGNITUDE=30
.WORD   2              ;TIME UNIT=SECONDS
.WORD   MRKAST         ;ADDRESS OF MARK TIME AST ROUTINE
```

FORTRAN CALL

```
CALL MARK (iefn,idm,idu[,ids])
```

where:

- iefn - is an integer containing an Event Flag Number.
- idm - is an integer containing the time interval magnitude.
- idu - is an integer containing the time interval units (1-4)
- ids - is an integer variable to receive the Directive Status Word.

The time interval magnitude can be set to a maximum of 24 hours.

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

A task can relinquish control of the system for a specified length of time by means of the WAIT call. After expiration of the specified delay, the subroutine resumes execution of the requesting task at the first execution statement following the WAIT call.

The form of the call is:

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

where:

- tmg - specifies the length of time, in units of tnt, to delay before continuing execution in the requesting program. If the value is zero or negative no delay will occur.
- tnt - specifies units of time as follows:
  - — 0 - basic counts of the system's clock
  - — 1 - Milliseconds (to the nearest clock tick)
  - — 2 - Seconds
  - — 3 - Minutes
  - — 4 - Hours
- ids - is set on return to the calling program to indicate the disposition of the request as follows:

The routine executes a MARK TIME directive followed, if successful, by a WAITFOR using Event Flag 29. An error return indicates that the MARK TIME Directive failed. Subtract one from the status variable and negate to obtain the standard IAS error code.

QIO$

The **QUEUE I/O** directive places an I/O request for an indicated device in a queue of priority-ordered requests for that device unit. The device is indicated by specifying a Logical Unit Number (LUN) assigned to the device.

A significant event is declared upon completion of an I/O request. If you specify an event flag in the QIO request, it is cleared when the request is queued, and set at the occurrence of the event. The I/O Status block is also cleared when the request is queued and contains the final I/O status when the I/O request is completed.

If you specify an AST service routine address, the AST occurs upon I/O completion with the task's PS, PC, Directive Status Word (DSW) and the address of the I/O status block pushed onto the task's (user) stack. The service routine is entered with the 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 in the QIO directive
- 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 is executed.

NOTE: You cannot use the contents of the I/O status block as an indication of I/O completion. The only legal ways of determining I/O completion are the setting of the associated event flag (if any), and/or the occurrence of the associated AST.

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 after the QIO directive was rejected, then the task may wait forever. You must always make sure that the directive has successfully completed.

The QUEUE I/O AND WAIT (QIOW$) directive should be used rather than QUEUE I/O followed by a WAITFOR whenever it is simply required to perform I/O and wait until it is complete. Use of QIOW$ enables the Executive to know why it is waiting and therefore to make better use of the processor. Also, you only need a single system directive.

Three error indicators are used in conjunction with QIO$:

1 The C bit

2 The Directive Status Word

3 The I/O status block

The programmer should check the C bit and the DSW immediately after the macro call. The C bit is set to indicate that the format of the macro call was incorrect or that the I/O request was not queued to the handler. The DSW can be examined to determine the reason for rejection. If the C bit is clear, the format of the macro call is correct and the request has been queued.

The I/O status block can be tested upon I/O completion to determine the success or failure of the I/O operation. The low byte of the first word is set as follows:

- positive - operation was successful. Normally only the value IS.SUC is used, but some handlers may use other values.
- negative - operation failed. The actual code used indicates why the operation failed.
- zero - the I/O status block is set to zero when the QIO directive is issued and remains zero until I/O completion occurs. Thus a value of zero indicates that the operation has not yet been completed.

The second word of the I/O status block is used for transfer (read or write) requests to contain the number of bytes actually transferred. The high byte of the first word, and the second word for non-transfer requests, may be used in a handler-dependent way.

For a full description of the use of the I/O status block see the *IAS Device Handlers Reference Manual.*

If an illegal I/O status block is specified (that is, the address is odd, or one or both words cannot be written to), the directive will fail with a status of IE.ADP. However a task may issue a QIO$ directive with a valid I/O status block address and then use the memory management directives to make the address invalid. In this case, the task will be aborted by the Executive when the I/O operation is completed. The termination message for the task will be "INVALID STATUS BLOCK".

This directive requires three nodes from the system node pool. These nodes are charged to the issuing task and released when the I/O request is completed.

MACRO CALL

```
QIO$ fnc,lun,[efn],[pri],[iosb],[ast][,prm]
```

where:

- fnc - is an I/O function code

  NOTE: If the function is read/write logical block to a device that has a volume mounted as a Files-11 directory volume, the issuing task must have executive privilege (LINK/PRIVILEGED or /PR switch).

- lun - is a logical unit number
- efn - is the event flag number (0 implies no event flag). It can be either global or local but to specify a global efn in a QIO macro the issuing task must have directive privilege.
- pri - is the priority
- iosb - is an address for the 2-word I/O status block
- ast - is an address for the I/O done AST service routine entry point

- prm - is a parameter list of the form <P1,....,P6>. The parameters required for each device are listed in the corresponding chapter of the IAS Device Handlers Reference Manual.

LOCAL SYMBOL DEFINITIONS

On the first call of this macro the following symbols are locally defined with their assigned values equal to the byte offsets from the start of the DPB to the respective DPB elements:

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

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.ULN	-05	Unassigned LUN
IE.HWR	-06	Handler task not resident
IE.PRI	-16	Global event flag cannot be specified because of directive privilege violation
IE.IPR	-95	Invalid Priority (>250)
IE.ILU	-96	Invalid LUN
IE.IEF	-97	Invalid event flag number (<0 or > 64)
IE.ADP	-98	Part of DPB or I/O status block is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
QIO$      IO.WVB,3,5,50,IOSB,QIOAST,<BUF,SIZ,40>
.BYTE     1,$$$ARG          ;QIO$ MACRO DIC, DPB SIZE=VARIABLE (6-12)
.WORD     IO.WVB            ;I/O FUNCTION CODE=WRITE VIRTUAL BLOCK
.WORD     3                 ;LUN 3
.BYTE     5,50              ;EVENT FLAG 5, PRIORITY 50
.WORD     IOSB              ;ADDRESS OF I/O STATUS BLOCK
.WORD     QIOAST            ;ADDRESS OF I/O DONE AST ENTRY POINT
.WORD     BUF               ;PARAMETER WORD 1
.WORD     SIZ               ;PARAMETER WORD 2
.WORD     40                ;PARAMETER WORD 3
```

FORTRAN CALL

```
CALL QIO (ifnc,ilun,[iefn],[ipri],[istat],[iprm][,ids])
```

where:

- ifnc - is an integer specifying the device function code.
- ilun - is an integer specifying device Logical Unit Number.
- iefn - is an integer specifying Event Flag Number.
- ipri - is an integer specifying QIO request priority.
- istat - is a 2-word integer array to receive device status.
- iprm - is a 6-word integer array containing device dependent parameters to be placed in parameter words 1-6 of the Directive Parameter Block (DPB).
- ids - is an integer variable to receive the Directive Status Word.

The subroutine GETADR can be called to insert addresses in selected elements of the iprm array. (See the *IAS FORTRAN Special Subroutines Reference Manual*).

QIOW$

The **QUEUE I/O AND WAIT** directive performs the functions of both QUEUE I/O (QIO$) and WAIT FOR SINGLE EVENT FLAG (WTSE$). The format of the call and other related information is identical to that of QIO$. If event flag 0 is specified, or the cfn parameter is omitted, the queue I/O is performed, but the wait is not performed.

The programmer should check the C bit and DSW immediately after the macro call (see QIO$).

MACRO CALL

QIOW$ (parameters as for QIO$)

DSW RETURN CODES

All those returned for QIO$.

MACRO EXPANSION

```
QIOW$   IO.RLB,5,3,200,IOSB1,RDAST,<INBUF,SIZE>
.BYTE   3,$$$ARG ;QIOW$ MACRO DIC, DPB SIZE=VARIABLE (6-12)
.WORD   IO.RLB   ;I/O FUNCTION CODE=READ LOGICAL BLOCK
.WORD   5        ;LUN=5
.BYTE   3,200    ;EVENT FLAG=3, PRIORITY=200
.WORD   IOSB1    ;ADDRESS OF I/O STATUS BLOCK
.WORD   RDAST    ;ADDRESS OF I/O DONE AST ENTRY POINT
.WORD   INBUF    ;PARAMETER WORD 1
.WORD   SIZE     ;PARAMETER WORD 2
```

FORTRAN CALL

```
CALL WTQIO (same parameters as QIO)
```

See under the QIO$ directive for a list of argument descriptions.

RDAF$

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

- WD - 00 Task Local Flags 1-16
- WD - 01 Task Local Flags 17-32
- WD - 02 Global Flags 33-48
- WD - 03 Global Flags 49-64

MACRO CALL

```
RDAF$  buf
```

where:

- buf - is an address of a 4-word buffer

LOCAL SYMBOL DEFINITIONS

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

- R.DABA - (Length 2 bytes) Buffer address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ADP	-98	Part of DPB or buffer is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

Only a single event flag can be read (see READ EVENT FLAG [RDEF$]).

RDEF$

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

MACRO CALL

```
RDEF$  efn
```

where:

- efn - is an event flag number

LOCAL SYMBOL DEFINITIONS

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

- R.DEEF - (Length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.CLR	+0	Flag was clear
IS.SET	+2	Flag was set
IE.IEF	-97	Invalid event flag number (event flag number <1 or >64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- iefn - is an integer containing an Event Flag Number.
- ids - is an integer variable to receive the Directive Status Word.

RZST$

The REQUEST directive activates a task. The task is initiated and subsequently run contingent upon priority and memory availability. The requested task must be installed in the system. If the task cannot run immediately, the request is queued so that the task executes when sufficient memory is available.

A real-time task is loaded into a partition if it is of sufficiently high priority and there is sufficient memory available. If the task cannot be loaded it is queued in a list of tasks waiting for memory and runs when memory becomes free. When a task requests memory space, checkpointing can occur. If a checkpointable task of lower priority than the requesting task is currently resident, it will be checkpointed onto a checkpoint file and the requesting task will be loaded. Timesharing tasks of lower priority (as is normally the case) will also be swapped out of memory. See the *IAS Executive Facilities Reference Manual* for details of checkpointing.

This directive can specify a partition to override the task's default partition and a priority to override the task's default execution priority. If the issuing task is executive privileged, this directive can specify a UIC to override the task's default execution UIC. If the issuing task is non-privileged, this directive can specify a UIC under which the task will be run, providing that the UIC is:

1 The UIC of the requesting task, or

2 The logged-in UIC for the terminal.

The terminal identification (TI) of the requested task is always the same as the TI of the task issuing the REQUEST directive.

Successful completion means that the task has been queued to run and not necessarily that it is actually running.

This directive requires either three or four nodes from the system node pool. These nodes are charged to the issuing task and are released when the requested task exits.

MACRO CALL

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

where:

- tsk - is the task name
- prt - is the partition name
- pri - is the priority
- ugc - is the UIC group code
- umc - is the UIC member code

LOCAL SYMBOL DEFINITIONS

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

- R.QSTN - (Length 4 bytes) Task name in Radix-50
- R.QSPN - (4) Partition name in Radix-50
- R.QSPR - (2) Priority
- R.QSGC - (1) UIC group
- R.QSPC - (1) UIC member

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.INS	-02	Task not installed
IE.PTS	-03	Partition too small for task
IE.HWR	-06	Handler task not resident to load task
IE.ACT	-07	Task is already active
IE.ITS	-08	Task is disabled
IE.PRI	-16	Directive privilege violation
IE.IUI	-91	Invalid UIC. Executive privilege is required to request a task to run under a changed UIC.
IE.PNS	-94	Partition not in system
IE.IPR	-95	Invalid priority specified (<0 or > 250)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size invalid

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (1st half)
  - — iop(2) - Radix-50 partition name (2nd half)
  - — iop(3) - Run priority
  - — iop(4) - UIC (User Identification Code)

  NOTE: The iop arguments (1), (2), (3) and (4) default to zero when none is specified.

- ids - is an integer variable to receive the Directive Status Word (DSW).

RREF$

The RECEIVE BY REFERENCE directive causes the Executive to dequeue the next reference in the receive-by-reference queue of the issuing (receiver) task. Optionally, the task will exit, suspend or stop if there are no references in the queue. The directive can also specify that the Executive proceed to map the region referred.

The detailed action, if there are no references to receive, is as follows:

- WS.RSX set - task will exit.
- WS.RSX clear, WS.RSU set - task will be suspended. When it is resumed, the status will be IS.SPD. No reference will have been received.
- WS.RSX, WS.RSU clear - task will be stopped. When it is resumed, the WS.RST set status will be IS.SPD. No reference will have been received.
- WS.RSX, WS.RSU, WS.RST all clear - task will receive an error status of IE.ITS.

If the directive is successful and the sending task specified an event flag to be set upon receipt of a reference, an event is declared.

If the Executive finds a reference, it writes the information provided to the corresponding words in the window definition block. This information provides sufficient information to map the reference, according to the sender task's specifications, with a previously created address window.

If the address of a 10-word receive buffer has been specified (W.NSRB in the window definition block), then the sender task name and the eight additional words (specified by the sender task) are placed in the specified buffer. If the sender task did not pass any additional information, the Executive writes in the sender task name and eight words of zero.

If the WS.MAP bit in the window status word has been set, the Executive attempts to map the reference as though a MAP ADDRESS WINDOW (MAP$) directive had been issued.

When a task that has unreceived references in its receive-by-reference queue exits or is removed, the Executive removes the references from the queue and deallocates them. Any related event flags are not set.

When a reference is received, the corresponding region is attached to the receiving task. Therefore, an Attachment Descriptor Block (ADB) is required in the task header, in the same way as for an ATTACH REGION (ATRG$) directive.

If the issuing task is multiuser, references will only be received if the TI specified in the corresponding SEND BY REFERENCE (SREF$) directive is equal to that for which it is running. If no TI was specified, the sending task must have been running with the same TI.

An additional 8-word buffer can be specified in the directive. The first word of this buffer will be filled with the TI of the sending task or the TI used in the SREF$ directive which sent this reference if this parameter was used. Such information can be used to return a reference or data to the sending task, by specifying it in a SREF$ or SEND DATA (VSDA$/SDAT$) directive. This information is only useful if the issuing task is not multiuser.

MACRO CALL

```
RREF$ wdb[,tibuf]
```

where:

- wdb - is the window definition block address
- tibuf - is the address of an 8-word buffer. The first word is filled with the terminal identification of the sending task. The remaining 7 words are reserved for future use.

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
iwdb(1) bits 0-7	W.NID	ID of an existing window if region is to be mapped
iwdb(7)	W.NSTS	Bit settings in the window status word:
		WS.MAP - 1 if received reference is to be mapped
		WS.RCX - 1 if task exit desired if no reference is found in the queue
		WS.RSU - 1 if task is to be suspended if no reference is found in the queue
		WS.RST - 1 if task is to be stopped if no reference is found in the queue.
iwdb(8)	W.NSRB	Optional address of a 10-word buffer, to contain the sender task name and additional information.

Output Parameters

Array Element	Offset	
iwdb(4)	W.NRID	Region ID (pointer to attachment description)
iwdb(5)	W.NOFF	Offset word specified by sender task
iwdb(6)	W.NLEN	Length word specified by sender task
iwdb(7)	W.NSTS	Bit settings in the window status word:
		WS.RED - 1 if attached with read access
		WS.WRT - 1 if attached with write access
		WS.EXT - 1 if attached with extend access
		WS.DEL - 1 if attached with delete access

Output Parameters		
Array Element	**Offset**	
		WS.RRF - 1 if receive was successful
		WS.UNM - 1 if a region has to be unmapped before the region referred to by the RREF$ directive can be mapped. The Executive clears the remaining bits.

LOCAL SYMBOL DEFINITIONS

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

- R.REBA - (Length 2 bytes) Window definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion.
IS.SPD	+02	No reference was received and task was suspended or stopped as indicated by WS.RSU or WS.RST.
IE.ITS	-08	No reference found in the receive-by-reference queue
IE.ALG	-84	Task specified an invalid region offset and length combination in the window definition block parameters; or WS.64B =0 and the value of W.NOFF is not a multiple of 8.
IE.WOV	-85	No attachment descriptors available in task header.
IE.NVW	-87	Invalid address window ID.
IE.ADP	-98	Invalid DPB, WDB, or receive buffer (W.NSRB) address.
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
RREF$     WDBADR
.BYTE     81.,3       ;RREF$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     WDBADR      ;WDB ADDRESS
.WORD     TIBUF
```

FORTRAN CALL

```
CALL RREF (iwdb,[isrb],[itibuf][,ids])
```

where:

- iwdb - is an 8-word integer array containing a window definition block
- isrb - is a 10-word integer array to be used as the receive buffer. If the call omits this parameter, the contents of iwdb(8) are unchanged.
- itibuf - is an 8-word integer array. The first word is filled with the terminal identification of the sending task. The remaining 7 words are reserved for future use.
- ids - is an integer variable to receive the Directive Status Word.

RSUM$

The **RESUME** directive instructs the system to resume the execution of a task that has been suspended as a result of issuing a SUSPEND (SPND$), RECEIVE DATA OR SUSPEND (VRCS$/RCVS$), or RECEIVE BY REFERENCE (RREF$) directive. If the task being resumed is a multiuser task, it is resumed only if its TI matches that of the task issuing the resume directive.

It is possible for a task to RESUME itself using the asynchronous trap feature. That is, the AST service routine can issue a RESUME directive specifying its own taskname.

MACRO CALL

```
RSUM$  tsk
```

where:

- tsk - is the task name

LOCAL SYMBOL DEFINITIONS

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

- R.SUTN - (Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.ACT	-07	Task not active
IE.ITS	-08	Task not suspended
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

RSUS$

The **RESUME OR UNSTOP** directive instructs the system to continue the execution of a task that has issued a SUSPEND (SPND$), STOP (STOP$), RECEIVE DATA OR SUSPEND (VRCS$/RCVS$), or RECEIVE DATA OR STOP (VRCT$/RCST$), or RECEIVE BY REFERENCE (RREF$) directive. If the task being resumed or unstopped is a multiuser task, it is continued only if its TI matches that of the task issuing the resume or unstop directive.

It is possible for a task to RESUME OR UNSTOP itself using the asynchronous trap feature. That is, the AST service routine can issue a RESUME OR UNSTOP directive specifying its own taskname.

MACRO CALL

```
RSUS$ tsk
```

where:

- tsk - is the task name

LOCAL SYMBOL DEFINITIONS

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

- R.SUTN - Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.ACT	-07	Task not active
IE.ITS	-08	Task not suspended
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space

Code	Value Returned	Explanation
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
RSUS$    ALPHA
.BYTE    171.,3          ;RSUS$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50   /ALPHA/         ;TASK 'ALPHA'
```

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6-character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

RUN$

The **RUN** directive causes a task to be requested at a specified future time, and optionally repeated periodically. The schedule time is specified in terms of time after issuance.

A real-time task will be loaded into a partition if it is of sufficiently high priority and there is sufficient memory available. If the task cannot be loaded it is queued in a list of tasks waiting for memory and will run when memory becomes free. When a task requests memory space, checkpointing can occur. If a checkpointable task of lower priority than the requesting task is currently resident, it will be checkpointed onto a checkpoint file and the requesting task will be loaded. Also, timesharing tasks of lower priority (as is normally the case) will be swapped out of memory. See the *IAS Executive Facilities Reference Manual* for details of checkpointing.

Successful completion means the task has been queued, not that it is actually running or that it will run when the specified time becomes due.

If you do not want optional rescheduling of the task, then you must omit the macro arguments rmg and rmt (see below).

This directive can specify a partition name to override the task's default partition, a priority to override the task's default execution priority and, if the issuing task is executive privileged, a UIC to override the task's default execution UIC. To specify a UIC other than that under which the requested task was installed, a task must have executive privilege.

The terminal identification (TI) of the requested task is always the same as the TI of the task issuing the REQUEST directive.

This directive requires two nodes from the system node pool. These nodes are charged to the issuing task and are returned when the request comes due, if a reschedule interval was not specified. Otherwise, the nodes are returned when the periodic rescheduling request is cancelled, by a CANCEL SCHEDULED REQUESTS (CSRQ$) directive or by the CANCEL command. In addition, either three or four nodes are required each time the task is requested.

MACRO CALL

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

where:

- tsk - is the task name
- prt - is the partition name
- pri - is the priority
- ugc - is the UIC group code
- umc - is the UIC member code
- smg - is the time interval magnitude before the task is scheduled, in the units specified as snt (below)

- snt - is the time unit used for smg (above)
  - — 1 = clock ticks
  - — 2 = seconds
  - — 3 = minutes
  - — 4 = hours
- rmg - is the reschedule interval magnitude
- rnt - is the reschedule interval unit

LOCAL SYMBOL DEFINITIONS

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

- R.UNTN - (Length 4 bytes) Task name in Radix-50
- R.UNPN - (4) Partition name in Radix-50
- R.UNPR - (2) Priority
- R.UNGC - (1) UIC group
- R.UNPC - (1) UIC member
- R.UNSM - (2) Schedule magnitude
- R.UNSU - (2) Schedule unit
- R.UNRM - (2) Reschedule magnitude
- R.UNRU - (2) Reschedule unit

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.INS	-02	Task not installed
IE.PTS	-03	Partition too small for task
IE.PRI	-16	Directive privilege violation
IE.IUI	-91	Invalid UIC. Executive privilege is required to run a task under a changed UIC.
IE.ITI	-93	Invalid time parameter specified

Code	Value Returned	Explanation
IE.PNS	-94	Partition not in system
IE.IPR	-95	Invalid priority specified (<0 or >250)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

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             ;DEFAULT PARTITION
.WORD   0               ;DEFAULT PRIORITY
.BYTE   10,20           ;UIC [20,10] TO RUN TASK UNDER
.WORD   20.             ;SCHEDULE MAGNITUDE=20
.WORD   3               ;SCH TIME UNIT=MINUTE(=3)
.WORD   10.             ;RESCH. INTERVAL MAGNITUDE=10
.WORD   3               ;RESCH.  INTERVAL UNIT=MINUTE (=3)
```

FORTRAN CALL

```
CALL RUN (tsk,[iop],[isd],isu,[iri],[iru][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (1st half)
  - — iop(2) - Radix-50 partition name (2nd half)
  - — iop(3) - run priority
  - — iop(4) - UIC (User Identification Code)

  NOTE: The iop arguments (1), (2), (3) and (4) default to zero when none is specified.
- isd - is a schedule delta magnitude.
- isu - is a schedule delta unit (1-4).
- iri - is a reschedule interval.
- iru - is a reschedule unit (1-4).

- ids - is an integer variable to receive the Directive Status Word.

The ISA standard call for initiating a task (CALL START) is also provided.

The CALL START call causes the execution of a designated task after a specified time delay. The actual time delay available is subject to the resolution of the real-time clock. Execution of the designated program will start at its first executable statement.

```
CALL START (tsk,isd,isu,ids)
```

where:

- tsk - is a 2-word integer array or a Real variable that specifies the program to be executed (in Radix-50 representation).
- isd - specifies the length of the time delay, in units as specified by isu, before program execution. If the value is zero or negative, the requested program will run as soon as permissible.
- isu - specifies units of time as follows:
  - — 0 - Basic counts of the system's clock
  - — 1 - Milliseconds (converted to basic counts of the system clock).
  - — 2 - Seconds
  - — 3 - Minutes
- ids - is an integer set on return to the calling program to indicate the disposition of the request as follows:
  - — 1 - Request accepted
  - — 2 or greater - Request not accepted

The START call is implemented by using the RUN Directive. An error indication means that the directive was rejected. Subtract one from the status variable and negate to obtain the standard IAS error code.

SCHD$

The SCHEDULE directive causes a task to be requested at a specified future time, and optionally, repeated periodically. You specify the schedule time in terms of absolute time-of-day. The SCHEDULE, RUN, and SYNC directives are similar in effect, differing only in the form in which the schedule data is presented. Refer to the RUN (RUN$) directive for further information.

You specify the time at which the task is to be run in the form hours, minutes, seconds, ticks.

MACRO CALL

```
SCHD$  tsk,[prt],[pri],[ugc,umc],hrs,min,sec,tck,[mag][,rnt]
```

where:

- tsk - is the task name
- prt - is the partition name
- pri - is the priority
- ugc - is a UIC group code
- umc - is a UIC member code
- hrs - is the schedule hours
- min - is the schedule minutes
- sec - is the schedule seconds
- tck - is the schedule clock ticks
- mag - is the reschedule interval magnitude (how many of the units defined by rnt)
- rnt - is the reschedule interval unit (1 = clock ticks; 2 = seconds; 3 = minutes; 4 = hours)

LOCAL SYMBOL DEFINITIONS

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

- S.CHTN - (Length 4 bytes) Task name in Radix-50
- S.CHPN - (4) Partition name in Radix-50
- S.CHPR - (2) Priority
- S.CHGC - (1) UIC group
- S.CHPC - (1) UIC member

- S.CHHR - (2) Hours
- S.CHMI - (2) Minutes
- S.CHSC - (2) Seconds
- S.CHCT - (2) Clock ticks
- S.CHRM - (2) Reschedule magnitude
- S.CHRU - (2) Reschedule unit

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.INS	-02	Task not installed
IE.PTS	-03	Partition too small for task
IE.PRI	-16	Directive privilege violation
IE.IUI	-91	Invalid UIC
IE.ITI	-93	Invalid time parameter specified
IE.PNS	-94	Partition not in system
IE.IPR	-95	Invalid priority specified (<0 or >250)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
SCHD$   POOL,CORE,200,300,40,10.,30.,00,00,5,3
.BYTE   15.,13. ;SCHD$ MACRO DIC, DPB SIZE=13. WORDS
.RAD50  /POOL/  ;TASK 'POOL'
.RAD50  /CORE/  ;PARTITION 'CORE'
.WORD   200     ;PRIORITY = 200
.BYTE   40,300  ;UIC [300,40] TO RUN TASK UNDER
.WORD   10.     ;SCHEDULE TIME HOURS=10
.WORD   30.     ;SCHEDULE TIME MINUTES=30
.WORD   00      ;SCHEDULE TIME SECONDS=0
.WORD   00      ;SCHEDULE TIME TICKS=0
.WORD   5       ;RESCHEDULE INTERVAL MAGNITUDE=5
.WORD   3       ;RESCHEDULE INTERVAL UNIT=MINUTE(=3)
                ;SCHEDULED TO RUN AT 10:30:00:00
                ;THEN EVERY 5 MINUTES
```

FORTRAN CALL

```
CALL SCHED (tsk,[iop],ih,im,is,it,[iri],[iru][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (1st half)
  - — iop(2) - Radix-50 partition name (2nd half)
  - — iop(3) - Run priority
  - — iop(4) - UIC (User Identification Code)

    NOTE: The iop arguments (1), (2), (3) and (4) default to zero when none is specified.
- ih - schedule hours
- im - schedule minutes
- is - schedule seconds
- it - schedule ticks
- iri - rescheduling interval
- iru - reschedule unit (1-4)
- ids - integer variable to receive the Directive Status Word

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

The TRNON call causes a designated task to be executed at a specific time of day. Execution of the designated task will start at its first executable statement. The form of the call is:

```
CALL TRNON (tsk,tim,ids)
```

where:

- tsk - is a 2-word integer array or a real variable that specifies the program to be executed (in Radix-50)
- tim - designates a 1-dimensional integer array of length three; the array contains the absolute time at which the program is to be executed. The elements of the array are as follows:
  - — tim (1) - Hours (using a 24-hour clock)
  - — tim (2) - Minutes
  - — tim (3) - Seconds

- ids - is an integer set on return to the calling program to indicate the disposition of the request as follows:
  - — 1 = request accepted
  - — 2 or greater = request not accepted.

The subroutine issues a SCHEDULE directive by building the appropriate Directive Parameter Block (DPB) on the stack. An error indication means that the directive was rejected. Subtract one from the status variable and negate to obtain the standard IAS error code.

SETF$

The SET EVENT FLAG directive sets an indicated event flag and reports the flag's polarity, in the DSW, as it was before setting. Setting an event flag does not cause a significant event to occur, it merely sets the specified flag. To set global event flags a task must have real-time directive privilege. Event flags 25 to 32 and 57 to 64 are conventionally reserved for use by IAS and should not be explicitly manipulated by the user.

MACRO CALL

```
SETF$  efn
```

where:

- efn - is an event flag number

LOCAL SYMBOL DEFINITIONS

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

- S.ETEF - (Length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.CLR	+0	Flag was clear
IS.SET	+2	Flag was already set
IE.PRI	-16	Privileged function (global flag cannot be set by a task that does not have real-time directive privilege).
IE.IEF	-97	Invalid event flag number (event flag number <1 or >64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
SETF$   1
.BYTE   33.,2       ; SETF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   1           ; EVENT FLAG NUMBER 1
```

FORTRAN CALL

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

where:

- iefn - is an integer containing an event flag number.
- ids - is an integer variable to receive the Directive Status Word.

SFPA$

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

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

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

When the directive is issued with an AST service routine entry point specified, the issuing task will be notified of future floating point exceptions via an AST at the specified location. See the IAS Executive Facilities Reference Manual for further details of ASTs.

When the directive is issued without an AST service routine entry point, the issuing task will not be notified of any future floating point exceptions until the directive is issued again with an AST service routine entry point specified.

Floating point exception ASTs are queued when a floating point exception trap occurs. No further floating point exception ASTs will be queued for the task until the first one queued has actually been effected.

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.

This directive cannot be issued when AST recognition is inhibited or from an AST service routine.

The directive requires nodes from the system node pool, which are charged to the issuing task. If the directive is used to specify a floating point exception AST where none currently exists, two nodes are required. These are returned when the task cancels its floating point exception AST by using a SPECIFY FLOATING POINT EXCEPTION AST (SFPA$) directive with a parameter of zero. Should the directive be used to change an existing floating point exception AST entry point, no additional nodes are needed.

MACRO CALL

```
SFPA$  [ast]
```

where:

- ast - is an AST service routine entry address, or zero if notification of floating point exception ASTs is no longer desired.

LOCAL SYMBOL DEFINITIONS

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

- S.FPAE - (Length 2 bytes) AST service routine entry address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Unavailable pool node
IE.ITS	-08	AST entry is already null
IE.AST	-80	Directive issued during AST service or while AST recognition is inhibited
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO CALL

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

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.

SPND$

The SUSPEND directive suspends the execution of the issuing task. A task can suspend only itself and not another task. A suspended task:

1 Remains on the Active Task List (ATL) but with a task status of "suspended".

2 Retains control of the system resources allocated to that task. No attempt is made to free the resources.

3 Can receive notification of events via the AST mechanism. In particular it can resume execution following a RESUME (RSUM$) directive issued from an AST service routine.

4 May resume execution following the PDS CONTINUE/REALTIME or MCR RESUME command. See the appropriate User's Guide for further details.

A suspended real-time task can be checkpointed if it was built checkpointable and has not disabled checkpointing. However, it is no more eligible for checkpointing than any other task of the same priority. In particular, it will never be checkpointed to make space for a task of lower priority. The description of the STOP (STOP$) directive explains how a task can suspend execution and temporarily release its memory.

If a timesharing task issues a SUSPEND (SPND$) directive, its owner will be informed. For example, a task running under the control of PDS in a timesharing IAS system will result in a "TASK SUSPENDED" message and PDS will prompt for command input.

If the SUSPEND directive succeeds, it returns a directive status of IS.SPD, not IS.SUC.

MACRO CALL

```
SPND$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SPD	+2	Task successfully suspended
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
SPND$
.BYTE  45.,1    ;SPND$ MACRO DIC, DPB SIZE=1 WORD
```

FORTRAN CALL

```
CALL SUSPND
```

SPRA$

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

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

2 That power recovery ASTs for the issuing task are no longer desired.

When the directive is issued with an AST entry point specified, the issuing task will be notified of any future power recovery by means of an AST to the specified location. See the *IAS Executive Facilities Reference Manual* for a detailed description of ASTs.

When the directive is issued without an AST service routine entry point, the issuing task will not be notified of any future power recovery until the directive is issued again with an AST service routine entry point specified.

Power recovery ASTs are queued when the power-up interrupt occurs following a power failure. Subsequent occurrences of powerfail will be ignored (no AST will occur) until the first AST has been effected.

The power recovery 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 (ASTX$) directive must be executed with the stack in the same state as when the AST was effected.

If a power recovery AST entry point is specified and the power fails while the task is not resident in memory, the AST is not effected or queued. A checkpointable task should disable checkpointing over critical regions where power recovery ASTs are essential.

This directive requires nodes from the system node pool which are charged to the issuing task. If the directive is used to specify a power recovery AST where none currently exists, two nodes are needed. These are returned when the task cancels its power recovery AST by using a SPECIFY POWER RECOVERY AST (SPRA$) directive with a parameter of zero. Should the directive be used to change an existing power recovery AST entry point, no additional nodes are needed.

An error is returned if the directive is issued when AST recognition is inhibited or from an AST service routine.

MACRO CALL

```
SPRA$  [ast]
```

where:

- ast - is an AST service routine entry address or zero if no longer desired.

LOCAL SYMBOL DEFINITIONS

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

- S.PRAE - (Length 2 bytes) AST service routine entry address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Unavailable pool node
IE.ITS	-08	AST entry is already null
IE.AST	-80	Directive issued during AST service or while AST recognition is inhibited
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

Task-resident subroutine PWRUP to call another subroutine upon recovery from a power failure.

```
EXTERNAL subnam
CALL PWRUP (subnam)
```

where:

- subnam - is the name of a subroutine to be executed upon power recovery. The PWRUP subroutine will effect a CALL SUBNAM (no arguments). SUBNAM is called as a result of a power recovery AST and therefore may be controlled at critical points by using INHIBIT and ENABLE AST recognition Directives.

Subroutine call of PWRUP to remove a power fail AST:

```
CALL PWRUP
```

SPWN$

The SPAWN directive requests the specified task for execution, optionally establishing exit events and queueing a command line. The effect of this directive is similar to the REQUEST (RQST$) directive, but has the following exceptions:

1 The issuing task can request notification of the termination of the spawned task (via an event flag or an AST).

2 The issuing task may determine the exit status of the spawned task via an exit status block.

3 A command line may be passed to the spawned task for subsequent retrieval by means of the GET MCR COMMAND LINE (GMCR$) directive, or the FCS Macro GCML$.

If an event flag is specified it will be cleared when the directive is issued and set when the spawned task exits or terminates for any reason.

If an AST address is specified an AST will be queued for the specified service routine when the spawned task exits. One additional word is placed on the stack: the address of the exit status block. The state of the stack on entry to the AST service routine is 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 DSW of task prior to AST
- SP+00 Address of exit status block (zero if none specified)

Thus, one word must be removed from the stack before the AST SERVICE EXIT (ASTX$) directive is issued.

If an Exit Status Block (ESB) is specified, its first word will be cleared when the directive is issued and set to the exit status of the spawned task when the latter exits. The exit status is determined as follows:

1 If the task exits by means of the EXIT WITH STATUS INFORMATION (EXST$) directive, the exit status is the argument supplied to this directive.

2 If the task exits using the EXIT (EXIT$) or EXITIF (EXIF$) directive, a success status (EX$SUC) is returned.

3 If the task is aborted or terminates because of any fault condition (for example, segment fault, memory parity error), a severe error status (EX$SEV) is returned.

If you specify a command line, it must not exceed 79 characters in length (it may have a length of zero characters). The address of the command line specified in the directive must be an even address (that is, the command line must start on a word boundary). The command line must contain only ASCII characters in the range 40 to 176 (octal). The line will be passed as specified to the spawned task (when the latter issues a GET MCR COMMAND LINE directive), except that a termination carriage return (octal code 015) will be appended to the line. The character count returned to the GET MCR

COMMAND LINE directive will be equal to that specified in the SPAWN directive and will not include a carriage return.

For standard system tasks (for example, MACRO, TKB) the command line should be prefixed by the task name and a space, in the same format as an MCR command line.

If the exit status block is invalid (that is, its address is odd or cannot be written to), the SPAWN directive will fail with a status of IE.ADP. However, a task can issue a SPAWN directive with a valid exit status block address and then use the memory management directives to make that address invalid. In this case, when the spawned task exits, the requesting task will be aborted by the Executive, with an error message "INVALID STATUS BLOCK".

The issuing task can exit before the spawned task exits. In this case, the spawned task will continue to run but no notification will occur when it exits. In particular, if a global event flag was specified in the SPAWN directive, the flag will not be set.

The SPAWN directive allows the specification of:

1 A partition in which the task is to run, other than that in which it was installed

2 A priority at which the task is to run, other than that at which it was installed

3 A UIC under which the task is to run, other than that under which it was installed. To do this, the issuing task must have executive privilege.

If the issuing task is non-privileged, this directive can specify a UIC under which the task is to run, provided that the UIC is:

1 The UIC of the spawning task, or

2 The logged-in UIC for the terminal.

A real-time task will be loaded into a partition if it is of sufficiently high priority and there is sufficient memory available. If the task cannot be loaded it is queued in a list of tasks waiting for memory and will run when memory becomes free. When a task requests memory space, checkpointing can occur. If a checkpointable task of lower priority than the requesting task is currently resident, it will be checkpointed onto a checkpoint file and the requesting task will be loaded. Additionally, timesharing tasks of lower priority (the normal case) will be swapped out of memory. See the *IAS Executive Facilities Reference Manual* for further details of checkpointing.

You can issue a SPAWN directive without specifying an event flag, AST, exit status block, or command line. In this case it is equivalent to a REQUEST (RQST$) directive. If a spawned task issues a SPAWN directive of this form, its requestor will not be notified when it exits. Instead, its requestor will be notified when the task spawned by this directive exits. Hence, this provides a chaining mechanism for spawned tasks such that the original requestor is notified only when the last task of the chain exits.

Before using the SPAWN directive in this way, a task must issue a GET MCR COMMAND LINE (GMCR$) directive to clear its command line.

The SPAWN directive requires up to 10 nodes from the system node pool, depending on the arguments specified in the macro call and upon the task being successfully spawned. These nodes are charged to the issuing task.

MACRO CALL

```
SPWN$ tsk,[prt],[pri],[ugc,umc],[efn],
[east],[esb],[cmdlin],[cmdlen][,vtun]
```

where:

- tsk - is the name of task to be spawned
- prt - is the partition name
- pri - is the priority
- ugc - is the group code number for the UIC of the spawned task.
- umc - is the member code number for the UIC of the spawned task.
- efn - is the event flag number to be cleared on issuance and set upon the termination of the spawned task.
- east - is the address of an AST routine to be entered when the spawned task exits
- esb - is the address of an eight-word exit status block to be written upon spawned task exit:
  - WD.00 - offspring task exit status
  - WD.01 to WD.07 - reserved
- cmdlin - is the address of a command line to be queued for the spawned task. The address must be even (that is, aligned on a word boundary).
- cmdlen - is the length of the command line. The maximum length is 79 characters.
- vtun - reserved.

LOCAL SYMBOL DEFINITIONS

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

- S.PWTN - (Length 4 bytes) Task name in Radix-50
- S.PWPN - (4) Partition name
- S.PWPR - (2) Priority
- S.PWPC - (1) UIC member code
- S.PWGC - (1) UIC group code
- S.PWEF - (2) Event flag number
- S.PWEA - (2) Address of AST routine
- S.PWES - (2) Address of exit status block

- S.PWCA - (2) Address of command line
- S.PWCL - (2) Length of command line
- S.PWVT - (2) Reserved

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	01	Unavailable pool node.
IE.INS	-02	Task not installed.
IE.PTS	-03	Partition too small for task.
IE.HWR	-06	Handler task not resident to load task.
IE.ACT	-07	The specified task was already active
IE.ITS	-08	Task is disabled.
IE.PRI	-16	Directive privilege violation.
IE.IUI	-91	Invalid UIC. Executive privilege is required to spawn a task to run under a changed UIC.
IE.PNS	-94	Partition not in system
IE.IPR	-95	Invalid priority specified (<0 or >250).
IE.IEF	-97	An invalid event flag number was specified.
IE.ADP	-98	Either: 1 Part of the DPB is out of the issuing task's address space. 2 The Exit Status Block has an odd address or cannot be written to. 3 The command line buffer has an odd address or cannot be accessed.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
SPWN$ TSKNAM,PRTNAM,PRI,UGC,UMC,EFN,EAST,ESB,CMDLIN,CMDLEN,VTUN
.BYTE  11.,13.      ;SPWN$ MACRO DIC,DPB SIZE=13 WORDS
.RAD50 /TSKNAM/     ;TASK 'TSKNAM'
.RAD50 /PRTNAM/     ;PARTITION 'PRTNAM'
.WORD  PRI          ;PRIORITY
.BYTE  UMC,UGC      ;UIC CODES
.WORD  EFN          ;EVENT FLAG NUMBER
.WORD  EAST         ;ADDRESS OF AST ON SPAWN EXIT
.WORD  ESB          ;EXIT STATUS BLOCK ADDRESS
.WORD  CMDLIN       ;COMMAND LINE ADDRESS
.WORD  CMDLEN       ;COMMAND LINE LENGTH
.WORD  VTUN         ;RESERVED
```

FORTRAN CALL

```
CALL SPAWN (tsk,[iop],[iefn],[iesb],[icmd],[icmdl],[ivtun][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (first half)
  - — iop(2) - Radix-50 partition name (second half)
  - — iop(3) - run priority
  - — iop(4) - UIC (User Identification Code)

    NOTE: The iop arguments (1), (2), (3) and (4) default to zero when none is specified.
- iefn - is an integer array event flag.
- iesb - is an 8-element integer for the exit status block.
- icmd - is a byte or integer array for the command line.
- icmdl - is an integer length for the command line.
- ivtun - reserved
- ids - is an integer variable to receive the Directive Status Word.

SRDA$

The **SPECIFY RECEIVE DATA AST** directive allows the specification of an AST service routine which will be executed when any data is sent to the issuing task by another task.

Subsequently, a receive AST occurs at the specified address when any task issues a SEND DATA (VSDA$/SDAT$) or SEND BY REFERENCE (SREF$) directive which results in data being sent to the receiving task. Only one receive AST is ever queued at a time; thus, if a delay occurs before the AST is effected, further sends to the task can occur without an AST being queued. This means that a task which waits for data being sent and then processes the data when a receive AST occurs must issue RECEIVE DATA directives until the directive indicates that no more data is present. Such a task must not assume that it will receive a separate AST for each packet of data sent.

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

You cannot issue this directive when AST recognition is inhibited or from an AST service routine.

If data is sent to a task while it is checkpointed, the receive AST will occur the next time the task is run, after it has been reloaded.

This directive uses nodes from the system node pool, which are charged to the issuing task. If you use the directive to specify a receive AST when none currently exists, two nodes are needed. These nodes are returned to the pool when the task cancels its receive AST by issuing a SPECIFY RECEIVE AST directive with a parameter of zero. If you use the directive to change the existing receive AST entry point, no additional nodes are needed.

MACRO CALL

```
SRDA$  [ast]
```

where:

- ast - is an AST service entry point address. Receives occuring for the issuing task cause control to be transferred to the AST entry point. Alternatively, if this parameter is zero, receive ASTs will no longer occur for the task.

LOCAL SYMBOL DEFINITIONS

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

- S.RDAE - (Length 2 bytes) AST entry address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.ITS	-08	AST entry already unspecified
IE.AST	-80	Directive issued during AST service or while AST is inhibited
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

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.

SREF$

The SEND BY REFERENCE directive inserts a reference to a region into the receive-by-reference queue of a specified (receiver) task. The Executive effectively attaches the region referred (the region identified in W.NRID of the window definition block) to the receiver task. (In practice, no Attachment Descriptor Block is required until the receiving task issues a RECEIVE BY REFERENCE [RREF$] directive.) The successful execution of this directive causes a significant event to occur.

The reference contains:

1 The identity of the region being sent

2 The offset and length words specified in W.NOFF and W.NLEN of the window definition block (which the Executive passes without checking)

3 The receiver task's permitted access to the region, specified in the window status word W.NSTS

4 The sender task name

5 Optionally, the address of an 8-word buffer that contains additional information. If the packet does not include a buffer address, the Executive sends 8 words of 0.

The receiver task automatically has access to the entire region as specified in W.NSTS. The sender task must be attached to the region with at least the same types of access. By setting all the bits in W.NSTS to 0, the permitted access can be defaulted to that of the sender task.

If the directive specifies an event flag, the Executive clears the flag when the directive is issued. The flag is set for the sender task when the receiver task acknowledges the reference by issuing the RECEIVE BY REFERENCE directive. When the sender task exits, the system searches for any unreceived references that specify event flags and prevents any invalid attempts to set the flags. The references themselves remain in the receiver task's receive-by-reference queues.

If the specified event flag is global, it will be set when the reference is received, even if the sender task has exited.

The TI parameter can be used to send a reference to a particular invocation of a multi-user task. The value to use is that placed in the TI buffer when data or a reference was received from the task.

This directive needs three nodes from the system node pool. These nodes are charged to the issuing task and are returned when the reference is received.

The ordering of the SREF$ macro arguments does not directly correspond to the format of the DPB. The arguments have been arranged so that the optional arguments efn and ti are at the end of the macro call. This arrangement is also compatible with the SDAT$ macro.

MACRO CALL

```
SREF$   tsk,wdb,[efn][,ti]
```

where:

- tsk - is the name of the receiver task

- wdb - is the window definition block address
- efn - is the event flag number
- ti - is the terminal identification (TI) of the task to which a reference is to be sent.

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
iwdb(4)	W.NRID	ID of the region to be sent by reference
iwdb(5)	W.NOFF	Offset word, passed without checking
iwdb(6)	W.NLEN	Length word, passed without checking
iwdb(7)	W.NSTS	Bit settings in window status word (the receiver task's permitted access):
		WS.RED - 1 if read access is permitted
		WS.WRT - 1 if write access is permitted
		WS.EXT - 1 if extend access is permitted
		WS.DEL - 1 if delete access is permitted
iwdb(8)	W.NSRB	Optional address of an 8-word buffer containing additional information

Output Parameters

NONE

LOCAL SYMBOL DEFINITIONS

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

- S.RETN - (Length 4 bytes) Receiver task name
- S.REBA - Window definition block base address (2)
- S.REEF - Event flag number (2)
- S.RETI - TI for SEND (2)

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.UPN	-01	A node could not be picked for a reference
IE.INS	-02	The sender task attempted to send a reference to a task which was not installed
IE.PRI	-16	Specified access not allowed to sender task itself
IE.NVR	-86	Invalid region ID
IE.IEF	-97	Invalid event flag number
IE.ADP	-98	The DPB, WDB or send buffer is wholly or partially outside the task's address space, or the WDB cannot be written to.
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
SREF$     ALPHA,WDBADR,48.
.BYTE     69.,5       ;SREF$ MACRO DIC,DPB SIZE=5 WORDS
.RAD50    /ALPHA/     ;RECEIVER TASK NAME
.WORD     48.         ;EVENT FLAG NUMBER
.WORD     WDBADR      ;WDB ADDRESS
```

FORTRAN CALL

```
CALL SREF (tsk,[efn],iwdb,[isrb],[iti][,ids])
```

where:

- tsk - is a single precision, floating point variable containing the name of the receiving task in Radix-50 format.
- efn - is an event flag number
- iwdb - is an 8-word integer array containing a window definition block
- isrb - is an 8-word integer array containing additional information. If specified, the address of isrb is placed in iwdb(8). If isrb is omitted, the contents of iwdb(8) remain unchanged.
- iti - is the terminal identification (TI) of the task to which a reference is to be sent.
- ids - is an integer variable to receive the Directive Status Word

SRFR$

The SEND BY REFERENCE AND REQUEST OR RESUME directive inserts a reference to a region into the receive-by-reference queue of a specified (receiver) task; then requests, resumes, or unstops the execution of the receiver task. You cannot successfully issue this directive unless:

1 You have real-time directive privilege, or

2 You have built the task you are trying to request or resume using the /SR switch (MCR TKB) or /REQUEST qualifier (PDS LINK).

This directive is equivalent to the SEND BY REFERENCE (SREF$) directive when followed by one of:

1 The REQUEST (RQST$) directive, if the receiver task is inactive.

2 The RESUME (RSUM$) directive, if the receiver task is active and suspended.

3 The UNSTOP (USTP$) directive, if the receiver task is active and stopped.

Real-time directive privilege is needed to specify a global event flag.

For the significance of the Window Definition Block (wdb), event flag number (efn) and terminal identification (ti) parameters, refer to the SEND BY REFERENCE (SREF$) directive.

For the significance of the partition (prt), priority (pri) and UIC group codes (ugc and umc), refer to the REQUEST (RQST$) directive.

If the receiver task is multi-user, the ti parameter can be used to specify which invocation of the task is to receive the data. If the receiver task is not active with the specified TI, it will be requested with that TI irrespective of other invocations which can be active with different TI assignments. If the task is active with the specified TI, this is the invocation which will be resumed or unstopped.

If the receiver task is already active, the prt, pri, ugc and umc parameters are ignored.

MACRO CALL

```
SRFR$   tsk,[prt],[pri],[ugc,umc],wdb,[efn][,ti]
```

where:

- tsk - is the name of the receiver task
- prt - is the partition name
- pri - is the priority
- ugc - is the UIC group code
- umc - is the UIC member code
- wdb - is the window definition block address
- efn - is the event flag number
- ti - is the terminal identification

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameter		
Array Element	**Offset**	
iwdb(4)	W.NRID	ID of the region to be sent by reference
iwdb(5)	W.NOFF	Offset word, passed without checking
iwdb(6)	W.NLEN	Length word, passed without checking
iwdb(7)	W.NSTS	Bit settings in window status word (the receiver task's permitted access):
		WS.RED - 1 if read access is permitted
		WS.WRT - 1 if write access is permitted
		WS.EXT - 1 if extend access is permitted
		WS.DEL - 1 if delete access is permitted
iwdb(8)	W.NSRB	Optional address of an 8-word buffer containing additional information

Output Parameters
NONE

LOCAL SYMBOL DEFINITIONS

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

- S.RRTN - (Length 4 bytes) Receiver task name
- S.RRPN - Partition name (4)
- S.RRPR - Priority (2)
- S.RRUG - UIC Group code (1)
- S.RRUP - UIC Member code (1)
- S.RRBA - Window definition block base address (2)
- S.RREF - Event flag number (2)
- S.RRTI - Terminal Identification (2)

DSW RETURN CODES

Code	Value Returned	Explanation
	+03	Reference sent and task already active and not suspended
	+02	Reference sent and task resumed
IS.SUC	+01	Reference sent and task requested
IE.UPN	-01	Unavailable pool node to run receiver task
IE.INS	-02	The sender task tried to send a reference to a task which was not installed
IE.PTS	-03	Partition to small for receiver task
IE.UNS	-04	Unavailable pool node for send packet
IE.HWR	-06	Handler task not resident to load receiver task
IE.ITS	-08	Receiver task is disabled
IE.FIX	-09	Receiver task is irrevocably exiting
IE.PRI	-16	Sender task not allowed specified access
IE.NVR	-86	Invalid region ID
IE.ITP	-88	Invalid TI parameter
IE.IUI	-91	Invalid UIC
IE.PNS	-94	Partition not in system
IE.IEF	-97	Invalid event flag number
IE.ADP	-98	The address check of the DPB, the WDB, or the send buffer failed
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
SRFR$   ALPHA,PART,40,200,200,WDBADR,48.,TIADR
.BYTE   153.,$$          ;SRFR$ MACRO DIC, DPB SIZE=$$ WORDS
.RAD50  /ALPHA/          ;RECEIVER TASK NAME
.WORD   /PART/           ;PARTITION NAME
.WORD   40               ;PRIORITY
.BYTE   200,200          ;UIC
.WORD   48.              ;EVENT FLAG NUMBER
.WORD   WDRADR           ;WDB ADDRESS
.WORD   TIADR            ;TI ADDRESS
```

FORTRAN CALL

```
CALL SRRF (tsk,[iop],[efn],iwdb,[isrb],[iti][,ids])
```

where:

- tsk - is a single precision, floating point variable containing the name of the receiving task in Radix-50 format.
- iop - is a 4-word integer array containing optional parameters where:
  - — iop(1) - Partition name (1st half) (Radix-50)
  - — iop(2) - Partition name (2nd half) (Radix-50)
  - — iop(3) - Run priority
  - — iop(4) - UIC User identification code
- efn - is an event flag number
- iwdb - is a window descriptor block address
- isrb - is an 8-word integer array containing additional information. If specified, the address of the isrb is placed in iwdb(8). If isrb is omitted, the contents of iwdb(8) remain unchanged.
- iti - is the terminal identification (TI) of the task to which a reference is to be sent.
- ids - is an integer variable to receive the Directive Status Word.

SRRA$

The SPECIFY RECEIVE-BY-REFERENCE AST directive instructs the system to record either of the following:

1 That receive-by-reference ASTs for the issuing task are needed, and the address to which the Executive transfers control when a receive-by-reference AST occurs.

2 That receive-by-reference ASTs for the issuing task are no longer needed.

When the directive specifies an AST service routine entry point, receive-by-reference ASTs for the task will occur; the Executive will transfer control to the specified address whenever a reference is sent to the task.

Only a single receive-by-reference AST can be queued at one time. If further references are sent to the task before an AST has been effected, only a single AST occurs. This means that a task which waits for ASTs, and then receives the reference and performs some action, must be prepared to receive more than one reference for each AST.

When the directive omits an entry point address, the Executive stops the occurrence of receive-by-reference ASTs for the issuing task. Receive-by-reference ASTs will not occur until the task issues another SPECIFY RECEIVE-BY-REFERENCE AST directive that specifies an entry point address.

The task enters the receive-by-reference AST service routine 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-by-reference AST; therefore, the AST SERVICE EXIT (ASTX$) directive must be executed with the stack in the same state as when the AST was effected.

This directive cannot be issued from an AST service routine or when AST recognition is inhibited.

If a task is checkpointed when a reference is sent, the AST occurs when the task is next run, after being reloaded into memory. If the task is stopped, it will be temporarily unstopped for the duration of the AST and therefore will be reloaded if it is checkpointed as a result of being stopped.

This directive requires nodes from the system node pool, which are charged to the issuing task. If you use the directive to specify a receive-by-reference AST when none currently exists, two nodes are required. These nodes are returned when the task cancels its receive-by-reference AST by using a SPECIFY RECEIVE BY REFERENCE AST (SRRA$) directive with a parameter of zero. If you use the directive to change an existing receive-by-reference AST entry point, no additional nodes are required.

MACRO CALL

```
SRRA$    [ast]
```

where:

- ast - is the AST service routine point address entry (0)

LOCAL SYMBOL DEFINITIONS

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

- S.RRAE - (Length 2 bytes) AST entry address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.UPN	-01	Unavailable pool node
IE.ITS	-08	AST entry is already null
IE.AST	-80	Directive issued during AST service or while AST recognition is inhibited
IE.ADP	-98	Part of the DPB is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid receive-by-reference ASTs.

MACRO EXPANSION

```
SRRA$     RECAST
.BYTE     21.,2  ;SRRA$ MACRO DIC, DPB SIZE=2 WORDS
.WORD     RECAST ;ADDRESS OF RECEIVE AST
```

FORTRAN CALL

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

STLO$

The STOP FOR LOGICAL OR OF EVENT FLAGS directive stops the execution of the issuing task until an indicated event flag of one of the following groups is set:

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

If one of the indicated flags is set when the directive is issued, task execution is not affected.

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

A task which is stopped has indicated that it does not require to be in memory. A stopped task will be checkpointed or swapped in favour of any other task which requires memory, irrespective of priority. When the task is no longer stopped (in this case, because one or more of the specified event flags has been set), it will be reloaded into memory according to its priority. A stopped task can also be reloaded into memory to allow an AST routine to be serviced. When the AST routine exits, the task becomes stopped again and is removed from memory to make room for any other tasks, unless it was unstopped while the AST was serviced.

The order in which stopped tasks are removed from memory to allow another task to be loaded is not defined. In particular, it is not related to task priority.

A task will normally use the stop facility, rather than the WAITFOR or SUSPEND, when it does not expect to be able to run for a relatively long period, and time is not critical when it is able to run. For example, the use of the stop facility would normally be appropriate while a spawned task was executing, or for a task which remained active while waiting for data to be sent, but not during a fast I/O transfer (for example, to disk).

MACRO CALL

```
STLO$  grp,msk
```

where:

- grp - is the desired group of event flags
- mask - If 'grp' is 0,1,2 or 3, 'mask' is a 16 bit (16-flag) mask word.

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

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

LOCAL SYMBOL DEFINITIONS

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

- S.TLGR - (Length 2 bytes) Event flag group
- S.TLMS - (2) 16-bit mask word

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.IEF	-97	No event flag specified in mask word(s) or; flag set indicator other than 0, 1, 2, 3, or 4.
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
STLO$  1,MSK
.BYTE  137.,3     ;STLO$ MACRO DIC, DPB SIZE=3 WORDS
.WORD  1          ;EVENT FLAG GROUP 1
.WORD  MSK        ;MASK WORD
```

FORTRAN CALL

```
CALL STOPOR (ief1,ief2,...iefn)
```

where:

- ief1,...iefn - is a list of event flag numbers to be taken as the set of event flags to be specified in the directive.

NOTE: The argument list specified in the FORTRAN call must contain only event flag numbers that lie with one event flag group. If event flag numbers are specified that lie in more than one group (that is, 0-3) or an invalid event flag number is specified, a fatal FORTRAN error is generated.

STOP$

The STOP directive instructs the system to stop the execution of the issuing task. The task can subsequently be unstopped by the UNSTOP (USTP$) directive, issued by one of the following:

1 Itself, at AST level, or by another task.

2 By means of a SEND DATA AND RESUME OR REQUEST RECEIVER (VSDR$/SDRQ$) directive.

3 By means of a SEND BY REFERENCE AND REQUEST OR RESUME (SRFR$) directive.

A task which is stopped has indicated that it does not require to be in memory. A stopped task will be checkpointed or swapped in favour of any other task which requires memory, irrespective of priority. When the task is no longer stopped it will be reloaded into memory according to its priority. A stopped task can also be reloaded into memory to allow an AST routine to be serviced. When the AST routine exits, the task becomes stopped again and will be removed from memory to make room for any other tasks, unless it was unstopped while the AST was serviced.

The order in which stopped tasks are removed from memory to allow another task to be loaded is not defined. In particular, it is not related to task priority.

A task will normally use the stop facility, rather than WAITFOR or SUSPEND, when it does not expect to be able to run for a relatively long period, and time is not critical when it is able to run. For example, the use of the stop facility would normally be appropriate while a spawned task was executing, or for a task which remained active whilst waiting for data to be sent, but not during a fast I/O transfer (for example, to disk).

MACRO CALL

```
STOP$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SPD	+02	Successful completion.
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
STOP$
.BYTE     131.,1
```

FORTRAN CALL

```
CALL STOPTK
```

STSE$

The STOP FOR SINGLE EVENT FLAG directive instructs the system to stop the execution of the issuing task until the indicated event flag is set.

A task which is stopped has indicated that it does not require to be in memory. A stopped task will be checkpointed or swapped in favour of any other task which requires memory, irrespective of priority. When the task is no longer stopped (in this case, by the setting of the specified event flag), it will be reloaded into memory according to its priority. A stopped task can also be reloaded into memory to allow an AST routine to be serviced. When the AST routine exits, the task becomes stopped again and will be removed from memory to make room for any other tasks.

The order in which stopped tasks are removed from memory to allow another task to be loaded is not defined. In particular, it is not necessarily related to task priority.

A task normally uses the stop facility, rather than WAITFOR or SUSPEND, when it does not expect to be able to run for a relatively long period, and time is not critical when it is able to run. For example, the use of the stop facility would normally be appropriate while a spawned task was executing, or for a task which remained active whilst waiting for data to be sent, but not during a fast I/O transfer (for example, to disk).

MACRO CALL

```
STSE$   efn
```

where:

- efn - is the event flag number

LOCAL SYMBOL DEFINITIONS

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

- S.TSEF - (Length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.IEF	-97	An event flag number other than a local event flag was specified (not in the range 1-32).
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
STSE$  EFN
.BYTE  135.,2     ;STSE$ MACRO DIC, DPB SIZE=2 WORDS
.WORD  EFN        ;EVENT FLAG NUMBER
```

FORTRAN CALL

```
CALL STOPFR (iefn[,ids])
```

where:

- iefn - is an integer containing an event flag number.
- ids - is an integer variable to receive the Directive Status Word.

SVDB$

The SPECIFY SST VECTOR TABLE FOR DEBUGGING AID directive specifies the address within the task's virtual address space of a table of entry points for synchronous system trap service routines for use by an intra-task debugging aid (for example, ODT uses SVDB$).

When the issuing task contains both a task SST vector (from SVTK$) and a debugging SST vector (from SVDB$) and both contain an entry for a particular trap, then the debugging vector takes precedence.

The table can contain up to nine entry points. Each entry point or trap-type instruction corresponds to a type of error or trap-type instruction that could occur. The table need only be long enough to include the entries of interest to the program. The table is of the following format:

- WD. 00 - Odd address error or other trap through 4
- WD. 01 - Segment fault
- WD. 02 - T-bit trap or execution of a BPT instruction
- WD. 03 - Execution of an IOT instruction
- WD. 04 - Execution of a reserved instruction
- WD. 05 - Execution of a non-IAS EMT
- WD. 06 - Execution of a TRAP instruction
- WD. 07 - PDP-11/40 floating point exception
- WD. 10 - Memory parity error

The way in which a particular trap is serviced is determined by the following sequence of checks, made in order of decreasing priority:

1 If the debugging SST vector exists (SVDB$) and the corresponding entry is non-zero, the trap is serviced by the debugging aid.

2 If the task SST vector exists (SVTK$) and the corresponding entry is non-zero, the trap is serviced by the task.

3 The task is aborted.

The vector exists if the directive was issued successfully. The vector can be eliminated by re-issuing the directive with a length parameter of zero.

MACRO CALL

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

where:

- adr - is the address of the SST vector table

- len - is the number of entries in table

LOCAL SYMBOL DEFINITIONS

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

- S.VDTA - (Length 2 bytes) SST vector table address
- S.VDTL - (2) Table length

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ADP	-98	Part of DPB or table is out of task's address space, or table address not specified (zero address)
IE.SDP	-99	DIC or DPB size is invalid

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
```

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.

SVTK$

The SPECIFY SST VECTOR TABLE FOR TASK directive specifies the address within the task's virtual address space of a table of entry points for synchronous system trap service routines for use by the issuing task. SSTs are described in the IAS Executive Facilities Reference Manual.

When the issuing task contains both a task SST vector (from SVTK$) and a debugging SST vector (from SVDB$) and both contain an entry for a particular trap, then the debugging vector takes precedence.

The table can contain up to nine entry points. Each entry point corresponds to a type of error or trap-type instruction that could occur. The table need only be long enough to include the entries of interest to the program. The table is of the following format:

- WD.00 - Odd address error or other trap through 4
- WD.01 - Segment fault
- WD.02 - T-Bit trap or execution of a BPT instruction
- WD.03 - Execution of an IOT instruction
- WD.04 - Execution of a reserved instruction
- WD.05 - Execution of a non-IAS EMT
- WD.06 - Execution of a TRAP instruction
- WD.07 - PDP-11/40 floating point exception
- WD.10 - Memory parity error

The way in which a particular trap is serviced is determined by the following sequence of checks, made in order of decreasing priority:

1 If the debugging SST vector exists (SVDB$) and the corresponding entry is non-zero, the trap is serviced by the debugging aid.

2 If the task SST vector exists (SVTK$) and the corresponding entry is non-zero, the trap is serviced by the task.

3 The task is aborted.

The vector exists if the directive was issued successfully. The vector can be eliminated by re-issuing the directive with a length parameter of zero.

MACRO CALL

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

where:

- adr - is the address of SST vector table

- len - is the length (number of entries) in the table

LOCAL SYMBOL DEFINITIONS

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

- S.VTTA - (Length 2 bytes) SST vector table address
- S.VTTL - (2) Table length

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ADP	-98	Part of DPB or table is out of task's address space, or table address is not specified (zero address)
IE.SDP	-99	DIC or DPB size is invalid

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
```

FORTRAN CALL

Neither the FORTRAN IV language nor the ISA standard permits linking to system trapping mechanisms. Therefore, this directive is not available to FORTRAN tasks. The FORTRAN OTS has its own SST vector table that traps these errors.

SYNC

The SYNCHRONIZE directive requests a task at a specified future time, and optionally, repeats it periodically. The directive allows the execution of a task to be synchronized with respect to a given clock unit. For example, a task can cause another task to run at 30 seconds past the next minute without needing to know about the current time.

The directive allows the specification of the time unit to be synchronized with and a delay after the synchronization, which can be specified in hours, minutes, seconds and ticks. In addition, the task can optionally be rescheduled to run periodically at a specified interval.

This directive requires two nodes from the system node pool. These nodes are charged to the issuing task and are released when the request comes due, if a reschedule interval was not specified, or otherwise when the periodic scheduling request is cancelled (by either the CSRQ$ directive or by the CANCEL command). In addition, either three or four nodes are required each time the task is requested (see the RQST$ directive).

MACRO CALL

```
SYNC$  tsk,[prt],[pri],[ugc,umc],[smg],[snt],sync,[rmg][,rnt]
```

where:

- tsk - is the task name
- prt - is the partition name
- pri - is the priority
- ugc - is the group code
- umc - is the member code
- smg - is the time interval after synchronization, in the units specified as snt (below).
- snt - is the time unit used for smg (above)
  - — 1 = clock ticks
  - — 2 = seconds
  - — 3 = minutes
  - — 4 = hours
- sync - is a synchronization unit. The task will be requested at the specified interval after the indicated synchronization unit. For example, if sync is 4, snt is 3 and smg is 12, the task is requested 12 minutes after the next hour.
- rmg - is the reschedule interval magnitude
- rnt - is a reschedule interval unit

LOCAL SYMBOL DEFINITIONS

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

- S.YNTN - (Length 4 bytes) Task name in Radix-50
- S.YNPN - (4) Partition name in Radix-50
- S.YNPR - (2) Priority
- S.YNGC - (1) UIC group
- S.YNPC - (1) UIC member
- S.YNSM - (2) Schedule magnitude
- S.YNSU - (2) Schedule unit
- S.YNSY - (2) Synchronization
- S.YNRM - (2) Reschedule magnitude
- S.YNRU - (2) Reschedule unit

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.UPN	-01	Insufficient pool nodes available to requester
IE.INS	-02	Task not installed
IE.PTS	-03	Partition too small for task
IE.PRI	-16	Directive privilege violation
IE.IUI	-91	Invalid UIC. Executive privilege is required to "run synchronized" a task under a changed UIC.
IE.ITI	-93	Invalid time parameter specified
IE.PNS	-94	Partition not in system
IE.IPR	-95	Invalid priority specified (<0 or >250)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
SYNC$   THIS,MARTO,55,300,40,6,2,3,10.,2
.BYTE   19.,12. ;SYNC$ MACRO DIC, DPB SIZE=12 WORDS
.RAD50  /THIS/  ;TASK 'THIS'
.RAD50  /MARTO/ ;PARTITION 'MARTO'
.WORD   55      ;PRIORITY = 55
.BYTE   40,300  ;UIC TO RUN TASK UNDER
.WORD   6       ;SCHEDULE MAGNITUDE = 6
.WORD   2       ;SCHEDULE UNIT = SECONDS (=2)
.WORD   3       ;SYNCHRONISATION UNIT = MINUTE (=3)
.WORD   10.     ;RESCHEDULE MAGNITUDE = 10
.WORD   2       ;RESCHEDULE UNIT = SECONDS (=2)
```

FORTRAN CALL

```
CALL SYNC (tsk,[iop],iom,iou,isyu,[iri],[iru][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (1st half)
  - — iop(2) - Radix-50 partition name (2nd half)
  - — iop(3) - Run priority
  - — iop(4) - User Identification Code

    NOTE: The iop arguments (1), (2), (3) and (4) default to zero when none is specified.
- iom - is a synchronization offset magnitude
- iou - is a synchronization offset unit (1-4)
- isyu - is a synchronization unit (1-4)
- iri - is a reschedule interval
- iru - is a reschedule unit (1-4)
- ids - is an integer variable to receive the Directive Status Word

UFIX$

The UNFIX directive nullifies the effect of a FIX directive, allowing the memory occupied by a fixed task to be freed for other users. The previously fixed task can now be removed from memory.

The UNFIX directive will only work for a task which was fixed with the same TI as the issuing task.

MACRO CALL

```
UFIX$  tsk
```

where:

- tsk - is the task name

LOCAL SYMBOL DEFINITIONS

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

- U.FXTN - (Length 4 bytes) Task name in Radix-50

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Task not installed
IE.FIX	-09	Task is not fixed
IE.PRI	-16	Directive privilege violation
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
UFIX$   POOL
.BYTE   87.,3    ;UFIX$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50  /POOL/   ;TASK 'POOL'
```

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

UMAP$

The UNMAP ADDRESS WINDOW directive unmaps a specified window. After the window has been unmapped, references to the corresponding virtual addresses are invalid and cause a processor trap to occur. This directive will fail if the directive is issued when the task has I/O in progress.

MACRO CALL

```
UMAP$   wdb
```

where:

- wdb - is the window definition block address

WINDOW DEFINITION BLOCK PARAMETERS

Input Parameters

Array Element	Offset	
iwdb(1) bits 0-7	W.NID	ID of the window to be unmapped

Output Parameters

Array Element	Offset	
iwdb(7)	W.NSTS	Bit settings in the window status word: WS.UNM - 1 if the window was successfully unmapped

LOCAL SYMBOL DEFINITIONS

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

- U.MABA - (Length 2 bytes) Window definition block address

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion
IE.ITS	-08	The specified address window is not mapped
IE.NVW	-87	Invalid address window ID
IE.IOP	-83	Task has I/O in progress
IE.ADP	-98	DPB or WDB out of range
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
UMAP$     WDBADR
.BYTE     123.,2      ;UMAP MACRO DIC, DPB SIZE=2 WORDS
.WORD     WDBADR      ;WDB ADDRESS
```

FORTRAN CALL

```
CALL UNMAP (iwdb[,ids])
```

where:

- iwdb - is an 8-word integer array containing a window definition block
- ids - is an integer variable to receive the Directive Status Word

USTP$

The **UNSTOP** directive instructs the system to unstop a task which has been stopped by either a STOP (STOP$), RECEIVE DATA OR STOP (VRCT$/RCST$) or RECEIVE BY REFERENCE (RREF$) directive.

The task will be returned to the runnable state and, if it has been checkpointed or swapped, it will be reloaded according to its priority.

The directive cannot be used to resume the execution of a task which is stopped and is currently waiting for event flags, having issued a STLO$ or STSE$ directive.

If the specified task is not stopped because it is executing an AST, but is stopped at non-AST level, it will be unstopped at the non-AST level. In particular, a task can issue the UNSTOP directive at AST level, specifying itself, so as to cause the main task to continue execution.

MACRO CALL

```
USTP$ tsk
```

where:

- tsk - is the name of the task to be unstopped.

LOCAL SYMBOL DEFINITIONS

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

- U.STTN - (Length 4 bytes) Taskname

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	The specified task is not installed in the system.
IE.ACT	-07	The specified task is not active.
IE.ITS	-08	The specified task is not stopped, or it is stopped for event flag(s)
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION

```
USTP$  TSKNAM
.BYTE  133., 3     ;USTP$ MACRO DIC, DPB SIZE=3 WORDS
.RAD50 /TSKNAM/    ;TASK 'TSKNAM'
```

FORTRAN CALL

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

where:

- tsk - is a 2-word, 1- to 6- character task name in Radix-50 form.
- ids - is an integer variable to receive the Directive Status Word.

VRCD$/RCVD$

The RECEIVE DATA (VRCD$) directive receives a variable-length data block that has been queued for it in priority order. An alternative macro call is RCVD$ which receives a 13-word data block (see below). The SEND DATA (VSDA$/SDAT$) or SEND DATA AND RESUME OR REQUEST RECEIVER (VSDR$/SDRQ$) directives queue data for a receiver.

The directive allows a sender task name to be specified. In this case, only data sent by the indicated task is received. When a sender task is not specified, data sent by any task is received.

If the buffer size is not specified, a default size of 13 words is used (see RCVD$, below). A 2-word sending task name and the data block are placed in the indicated buffer. The task name is in the first two words. The receive buffer must be long enough to include these two words. The buffer length specified in the directive should not include these words.

If the location to store the TI is specified, a TI indicator is placed in this location. This can then be used to identify the sender if the receiving task wishes to communicate back to the sender, by specifying it as the "ti" parameter in a SEND DATA directive.

If the receiver task is multi-user, only data with the same TI assignment is received. The TI assignment of sent data is equal to the TI assignment of the sending task, unless the "ti" parameter is used when sending the data.

Data is transferred from the sending task to the receiving task by means of nodes picked from the system node pool which are charged to the sender. When the data is received, the nodes are returned and subtracted from the sender's node usage count. The number of nodes required depends on the length of the data block. The first three words require a single node; subsequently, one node is required for each eight words.

The condition code "V" (CC-V) is set upon successful completion of this directive, if the sender task had executive privilege.

MACRO CALL FOR VRCD$

```
VRCD$ [tsk],adr,len[,ti]
```

- tis - the sender task name
- adr - is the buffer address
- len - is the data length in words
- ti - is the address to store sender's TI

LOCAL SYMBOL DEFINITIONS

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

- R.VDTN - (Length 4 bytes) Task name in Radix-50
- R.VDBA - (2) Buffer address
- R.VDBL - (2) Data length
- R.VDTI - (2) Address in which to store TI

CONDITION CODES

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

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Sender task not installed
IE.ITS	-08	No data queued (sent)
IE.RBS	-15	Receiver's buffer too small, data truncated
IE.IBS	-89	Invalid buffer size (>255)
IE.ADP	-98	Part of DPB or buffer is out of task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION OF VRCD$

```
VRCD$    OTHER,DATAIN,25.,TIADDR
.BYTE    75.,$$$T9      ;VRCD$ MACRO DIC, DPB SIZE=VARIABLE (4-6)
.RAD50   /OTHER/        ;SENDER TASK 'OTHER'
.WORD    DATAIN         ;RECEIVE BUFFER
.IIF GE  $$$T9-5,.WORD 25. ;DATA LENGTH
.IIF EQ  $$$T9-6,.WORD TIADDR ;ADDRESS TO STORE TI
```

FORTRAN CALL FOR VRCD$

```
CALL VRECEV ([tsk],idata,[bufsz],[ti],[iprv][,ids])
```

where:

- tsk - is a two-word, 1- to 6- character sender task name in Radix-50 form
- idata - is an integer array for data received (1-255 words)
- bufsz - is the size in words, to receive
- ti - is the variable in which to receive the TI of the sender
- iprv - is a full-word logical variable set as follows:
  - — Directive failed: iprv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer to receive the directive status word.

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

MACRO CALL FOR RCVD$

```
RCVD$   [tsk],bufadr
```

FORTRAN CALL FOR RCVD$

```
CALL RECEIV ([tsk],idata,[iprv][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character sender task name in Radix-50 form.
- idata - is a 15-word integer array for data received.
- iprv - is a full-word logical variable set as follows:
  - — Directive failed: iprv unchanged
  - — Sender task not executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer variable to receive the Directive Status Word.

VRCS$/RCVS$

The **RECEIVE DATA OR SUSPEND** (VRCS$) directive receives a variable-length data block that has been queued according to priority for it or suspends if no data blocks can be received. An alternative macro call is RCVS$ which receives a 13-word data block (see below). Use the SEND DATA and the SEND DATA AND RESUME OR REQUEST RECEIVER directives to queue data for a receiver.

The RECEIVE DATA OR SUSPEND directive is useful in avoiding a possible "race condition" that can occur between two tasks communicating by means of 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 SUSPEND, the other task sends it a message. Since the first task has already decided to suspend, it will not receive the message until another message causes it to be resumed, which may be much later. This condition can be avoided by using the combined RECEIVE DATA OR SUSPEND directive rather than specifying separate RECEIVE and SUSPEND directives.

The directive allows a sender task name to be specified. In this case, only data sent by the indicated task is received. When a sender is not specified, data sent by any task is received.

If the buffer size is not specified, a default size of 13 words is used (see RCVS$, below). A 2-word sending task name and the data block are returned in the indicated buffer. The task name is in the first two words. The receive buffer must be long enough to include these two words. The buffer length specified in the directive should not include these words.

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

If the receiving task is multi-user, only data with the same TI assignment is received. The TI assignment of sent data is equal to the TI assignment of the sending task, unless the 'ti' parameter is used when sending the data.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool which are charged to the sender. When the data is received, the nodes are returned and subtracted from the sender's node usage count. The number of nodes required depends on the length of the data block. The first three words require a single node; subsequently, one node is required for each eight words.

The condition code "V" (CC-V) is set upon successful completion of this directive, if the sender task had executive privilege.

If no data is received, the task is suspended. When the task is subsequently resumed, a RECEIVE DATA directive is issued to receive any data. The task can check whether it was suspended (and hence that no data was received) by examining its Directive Status Word (DSW). If this word has a value of IS.SUC, data was received. Otherwise, a value of IS.SPD indicates that the task was suspended and no data was received.

MACRO CALL FOR VRCS$

```
VRCS$   [tsk],bufadr,buflen[,ti]
```

where:

- tsk - is the sender task name
- bufadr - is the buffer address
- buflen - is the data length in words
- ti - is the address in which to store the sender's TI indicator

LOCAL SYMBOL DEFINITIONS

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

- R.VSTN - (Length 4 bytes) Task name in Radix-50
- R.VSBA - (2) Buffer address
- R.VSBL - (2) Data length
- R.VSTI - (2) Address in which to store the TI indicator

CONDITION CODES

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

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion of the receive
IS.SPD	+02	Successful suspension of the task. No data has been received. The task has since been resumed. A receive directive is now needed to receive any data sent meanwhile.
IE.INS	-02	Sender task not installed
IE.RBS	-15	Receiver's buffer too small, data truncated
IE.IBS	-89	Invalid buffer size (>255)
IE.ADP	-98	Part of DPB or buffer is out of task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION FOR VRCS$

```
VRCS$    TASK2,DATAIN,10.,TIADDR
.BYTE    79.,$$$T9              ;VRCS$ MACRO DIC, DPB SIZE=VARIABLE
                                ;(4-6)
.RAD50   /TASK2/                ;SENDER TASK 'TASK2'
.WORD    DATAIN                 ;RECEIVE BUFFER
.IIF GE  $$$T9-5,.WORD 10.      ;DATA LENGTH
.IIF EQ  $$$T9-6,.WORD TIADDR ;ADDRESS TO STORE TI
```

FORTRAN CALL FOR VRCS$

```
CALL VRECSP ([tsk],idata,[bufsz],[ti],[iprv][,ids])
```

where:

- tsk - is a two-word, 1- to 6- character sender task name in Radix-50 format
- idata - is an integer array for data received (1-255 words)
- bufsz - is the size, in words, of the data to be received
- ti - is the variable into which the TI of the sender is received
- iprv - is a full-word logical variable set as follows:
  - — Directive failed: iprv unchanged
  - — Receiving task suspended: iprv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer to receive the Directive Status Word

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

MACRO CALL FOR RCVS$

```
RCVS$   [tsk],bufadr
```

FORTRAN CALL FOR RCVS$

```
CALL RECOSP ([tsk],idata,[iprv][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character sender task name in Radix-50 form.
- idata - is a 15-word integer array for data received.
- iprv - is a full-word logical variable, set as follows:
  - — Directive failed: iprv unchanged
  - — Receiving task suspended: iprv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer variable to receive the Directive Status Word.

VRCT$/RCST$

The RECEIVE DATA OR STOP (VRCT$) directive receives a variable-length data block that has been queued according to the specified priority or stops if no data can be received. An alternative macro call is RCST$ which receives a 13-word data block. Use the SEND DATA and the SEND DATA AND RESUME OR REQUEST RECEIVER directives to queue data for a receiver.

The function of this directive is similar to that of the RECEIVE DATA OR SUSPEND (VRCS$) directive, except that if no data is present, the issuing task is stopped, rather than suspended. This allows the task to be removed from memory irrespective of priority. Use this directive in preference to VRCS$ when it is likely that there will be a long wait (that is, several seconds or more) before further data is available and it is not essential that the data be received in the fastest possible time.

MACRO CALL FOR VRCT$

```
VRCT$ [tsk],bufadr,buflen[,ti]
```

- tsk - is the name of the task from which data is to be received. If the task name is not specified, data may be received from any task.
- bufadr - is the address of a 15-word buffer to receive the sender task name and data
- buflen - is the data length in words
- ti - is the address in which to store the sender's TI indicator

LOCAL SYMBOL DEFINITIONS

The following symbols are defined with the assigned values equal to the byte offset from the start of the DPB to the DPB element:

- R.CSTN - (Length 4 bytes) Task name
- R.CSBF - (2) Buffer address
- R.CSBL - (2) Buffer length
- R.CSTI - (2) Terminal identification

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+01	Successful completion.
IS.SPD	+02	No data was received and task was stopped. (Naturally the task must be unstopped before it will see this status.)
IE.INS	-02	Sender task not installed
IE.RBS	-15	Receiver's buffer too small, data truncated
IE.IBS	-89	Invalid buffer size (>255)
IE.ADP	-98	Part of the DPB is out of the issuing task's address space.
IE.SDP	-99	DIC or DPB size is invalid.

MACRO EXPANSION FOR VRCT$

```
VRCT$    TSKNAM,BUFADR,BUFLEN,TI
.BYTE    139.,$$$T9  ;VRCT$ MACRO DIC,DPB SIZE=VARIABLE (4-6)
.RAD50   /TSKNAM/    ;TASK 'TSKNAM'
.WORD    BUFADR      ;BUFFER ADDRESS
.IIF GE  $$$T9.5,.WORD   ;BUFFER LENGTH
.IIF EQ  $$$T9.6,.WORD   ;ADDRESS TO STORE TI
```

FORTRAN CALL FOR VRCT$

```
CALL VRECST ([tsk],idata,[bufsz],[ti],[iprv][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character sender task name in Radix-50 format
- idata - is an integer array for data received (1-255 words)
- bufsz - is the size, in words, of the data to be received
- ti - is the variable into which the TI of the sender is received
- iprv - is a full-word logical variable set as follows:
  - — Directive failed: iprv unchanged
  - — Receiving task stopped: iprv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)

- ids - is an integer to receive the Directive Status Word

An alternative macro call is RCST$ which receives a 13-word data block.

MACRO CALL FOR RCST$

```
RCST$ [tsk],bufadr
```

FORTRAN CALL FOR RCST$

```
CALL RECOST ([tsk],idata,[iprv][,ids])
```

- tsk - is a 2-word, 1- to 6- character sender taskname in Radix-50 form.
- idata - is a 15-word integer array for data received.
- iprv - is a full-word logical variable set as follows:
  - — Directive failed: iprv unchanged
  - — Receiving task stopped: iprv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer to receive the Directive Status Word

VRCX$/RCVX$

The **RECEIVE DATA OR EXIT** (VRCX$) directive receives a variable-length data block that has been queued for it according to priority or exits if no data block can be received. An alternative macro call is RCVX$ which receives a 13-word data block (see below).

The RECEIVE DATA OR EXIT directive is useful in avoiding a possible "race condition" that can occur between two tasks communicating by means of the SEND and RECEIVE directives. The race condition can occur when one task executes a RECEIVE directive and finds its receive queue empty. It then exits, but before the task can EXIT, the other task sends it a message. Although the first task has already decided to exit, it is still active and, therefore, a request directive issued by the second task will fail. Thus, the data will not be received unless the first task is activated for some other reason. If the task is built with the "flush receive queues" option, the send data will be lost altogether when the receive queues are flushed as the task exits. This condition can be avoided by using the combined RECEIVE DATA OR EXIT directive rather than specifying separate RECEIVE and EXIT directives.

The SEND DATA and the SEND DATA AND RESUME OR REQUEST directives queue data for a receiver.

The directive allows a sender task name to be specified. In this case, only data sent by the indicated task is received. When a sender task is not specified, data sent by any task is received.

If the buffer size is not specified, a default size of 13 words is used (see RCVX$, below). A 2-word sending task name and the data block are returned in the indicated buffer. The task name is in the first two words. The receive buffer must be long enough to include these two words, the buffer length specified in the directive should not include these words.

If the location to store TI is specified, the TI indicator is transferred from the SEND/RECEIVE node to this specified location. This can then be used to identify the sender if the receiver task wishes to communicate bask to the sender, by specifying it as the "ti" parameter in a SEND DATA directive.

If the receiving task is multi-user, only data with the same TI assignment is received.

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool which are charged to the sender. When the data is received, the nodes are returned and subtracted from the sender's node usage count. The number of nodes required depends on the length of the data block. The first three words require a single node; subsequently, one node is required for each eight words.

The condition code "V" (CC-V) is set upon successful completion of this directive, if the sender task had executive privilege. If the task exits, the Executive will declare an event.

MACRO EXPANSION FOR VRCX$

```
VRCX$  [tsk],bufadr,buflen[,ti]
```

- tsk - is the sender task name
- bufadr - is the buffer address
- buflen - is the data length in words
- ti - is the address at which to store the TI indicator

LOCAL SYMBOL DEFINITIONS

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

- R.VXTN - (Length 4 bytes) Task name
- R.VXBA - (2) Buffer address
- R.VXBL - (2) Data length
- R.VXTI - (2) Address in which to store the TI indicator

CONDITION CODES

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

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Sender task not installed
IE.RBS	-15	Receiver's buffer too small, data truncated
IE.IBS	-89	Invalid buffer size (>255)
IE.ADP	-98	Part of DPB or buffer is out of task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION FOR VRCX$

```
VRCX$    TASK9,BUFIN,8.,TIADDR
.BYTE    77.,$$$T9              ;VRCX$ MACRO DIC, DPB SIZE =
                                ;VARIABLE (4-6)
.RAD50   /TASK9/                ;SENDER TASK 'TASK9'
.WORD    BUFIN                  ;RECEIVE BUFFER
.IIF GE  $$$T9-5,.WORD 8.       ;DATA LENGTH
.IIF EQ  $$$T9-6,.WORD TIADDR ;ADDRESS TO STORE TI
```

FORTRAN CALL FOR VRCX$

```
CALL VRECEX ([tsk],idate,[bufsz],[ti],[iprv][,ids])
```

- tsk - is a 2-word, 1- to 6- character sender task name in Radix-50 form
- idata - is an integer array for data received (1-255 words)
- bufsz - is the size, in words, of the data to be received
- ti - is the variable into which the TI of the sender is received
- iprv - is a full-word logical variable set as follows:
  - — Directive failed: iprv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer to receive the Directive Status Word

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

MACRO CALL FOR RCVX$

```
RCVX$ [tsk],bufadr
```

FORTRAN CALL FOR RCVX$

```
CALL RECOEX ([tsk],idata,[iprv][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character sender task name in Radix-50 form.
- idata - is a 15-word integer array for data received.
- iprv - is a full-word logical set as follows:
  - — Directive failed: ipvrv unchanged
  - — Sender task executive privileged: iprv set to .TRUE. (-1)
  - — Sender task not executive privileged: iprv set to .FALSE. (0)
- ids - is an integer variable to receive the Directive Status Word.

VSDA$/SDAT$

The **SEND DATA** (VSDA$) directive queues a variable-length data block for a task to receive. An alternative macro call is SDAT$, which sends a 13-word data block (see below). A maximum of 255 (decimal) words may be sent in one block.

If the buffer size is not specified, a default of 13 words is always used (see SDAT$ below).

A significant event is declared if the directive is successful. The indicated event flag, if any, is set. Normally, the event flag is used to trigger the receiver into some action. To be effective, the task must set a global event flag (33 through 64), because the local flags are visible only to the sending task and will have no effect on the receiving task. Real-time directive privilege is required to specify a global event flag.

All data sent to a task is queued according to the priority specified in the send request. Thus it will be received in priority order, not first-in, first-out (FIFO). If no priority is specified, the priority of the task is used. If a number of tasks, running at different priorities send related data to the same task, obscure problems can result if the send priority facility is not taken into account.

If a TI is specified, this is used to determine which invocation of the receiving task (if it is multi-user) is to receive the data. If no TI is specified, the TI of the sending task is used.

If the receiving task has specified a receive AST service routine, an AST will be queued (unless the task already has a receive AST queued).

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked is then charged to the sender. When the data is received, the nodes are returned and subtracted from the sender's node usage count. The number of nodes required depends on the amount of data sent. The first three words require a single node. Subsequently, each eight words require one further node.

MACRO CALL FOR VSDA$

```
VSDA$ [tsk],bufadr,[buflen],[efn],[sndpri][,ti]
```

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

LOCAL SYMBOL DEFINITIONS

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

- S.DATN - (Length 4 bytes) Task name in Radix-50
- S.DABA - (2) buffer address
- S.DAEF - (2) event flag number
- S.DABL - (2) buffer length
- S.DASP - (2) send priority
- S.DATI - (2) TI indicator

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.INS	-02	Receiver task not installed
IE.UNS	-04	Insufficient pool nodes for SEND
IE.PRI	-16	Task does not have the privilege to use global event flags
IE.ITP	-88	Invalid TI indicator
IE.IBS	-89	Invalid buffer size (size >255)
IE.IEF	-97	Invalid event flag number (event flag number < 0 or >64)
IE.ADP	-98	Part of DPB or data block is out of task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION FOR VSDA$

```
VSDA$    TASK4,DATA,24,36,200,TIADDR
.BYTE    71.,$$$T9          ;VSDA$ MACRO DIC, DPB SIZE=VARIABLE
                            (5-8);
.RAD50   /TASK4/            ;RECEIVER TASK NAME
.WORD    DATA               ;BUFFER ADDRESS OF DATA TO BE SENT
.WORD    36                 ;EVENT FLAG TO BE SET ON SUCCESSFUL SEND
.IIF GE  $$$T9-6,.WORD 24    ;LENGTH OF DATA TO BE SENT
.IIF GE  $$$T9-7,.WORD 200   ;PRIORITY AT WHICH TO SEND
                             ;DATA
.IIF EQ  $$$T9-8.,.WORD TIADDR   ;TI OF RECEIVER TASK
```

FORTRAN CALL FOR VSDA$

```
CALL VSEND (tsk,idata,[iefn],[bufsz],[pri],[ti][,ids])
```

where:

- tsk - is a 2 word 1- to 6- character receiver task name in Radix-50 form
- idata - is an integer array of data to be sent (1 to 255 words)
- iefn - is the number of an event flag to be set
- bufsz - is the number of words to send
- pri - is the priority of the send
- ti - is the TI of the task to which data is to be sent
- ids - is an integer to receive the Directive Status Word.

An alternative macro call is SDAT$, which sends a 13-word data ck.

MACRO CALL FOR SDAT$

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

FORTRAN CALL FOR SDAT$

```
CALL SEND (tsk,idata,[iefn][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character receiver task name in Radix-50 form.
- idata - is a 13-word integer array of data to be sent.
- iefn - is the number of an event flag to be set.
- ids - is an integer variable to receive the Directive Status Word.

VSDR$/SDRQ$

The SEND DATA AND REQUEST OR RESUME RECEIVER (VSDR$) queues a variable-length data block for a task to receive and to request, resume or unstop the receiver. You cannot successfully issue these directives unless:

1 You have real-time directive privilege, or

2 You have built the task you are trying to request or resume using the /SR switch (MCR TKB) or /REQUEST qualifier (PDS LINK).

An alternative macro call is SDRQ$, which sends a 13-word data block. A maximum of 255 (decimal) words may be sent in one block.

If the buffer size is not specified, a default buffer size of 13 words is used (see SDRQ$, below).

A significant event is declared if the directive is successful. The indicated event flag, if any, is set. The event flag is used commonly to trigger the receiver into some action. To be effective, the task must set a global event flag (33 through 64), because the local flags are visible only to the sending task and will have no effect on the receiving task. You need real-time directive privilege to specify a global event flag.

All data sent to a task is queued according to the priority specified in the send request. Thus it will be received in priority order, not first-in, first-out (FIFO). If no priority is specified, the priority of the task is used. If a number of tasks, running at different priorities, send related data to the same task, obscure problems can result if the send priority facility is not taken into account.

If a TI is specified, this is used to determine which invocation of the receiving task (if it is multi-user) is to receive the data. If no TI is specified, the TI of the sending task is used.

If the receiving task has specified a receive AST service routine, an AST will be queued (unless the task already has a receive AST queued).

Data is transferred from the sending task to the receiving task by means of nodes picked from the pool. The number of nodes picked from the pool is charged to the sender. When the data is received, the nodes are returned and subtracted from the sender's usage count. The number of nodes required depends on the amount of data sent. The first three words require a single node. Subsequently, each eight words require one further node.

As such, the REQUEST part of this directive is equivalent to the SEND DATA (VSDA$/SDAT$) directive, followed by one of the following operations:

1 The REQUEST (RQST$) directive, if the receiver task is inactive.

2 The RESUME (RSUM$) directive, if the receiver task is active and suspended.

3 The UNSTOP (USTP$) directive, if the receiver task is active and stopped.

For the significance of the event flag number (efn) and terminal identification (ti) parameters, refer to the SEND DATA (VSDA$/SDAT$) directive.

For the significance of the partition (prt), priority (pri) and UIC group codes (ugc and umc), refer to the REQUEST (RQST$) directive.

If the receiver task is multi-user, the ti parameter can be used to specify which invocation of the task is to receive the data. If the receiver task is not active with the specified TI, it will be requested with that TI irrespective of other invocations which can be active with different TI assignments. If the task is active with the specified TI, this is the invocation which will be resumed or unstopped.

If the receiver task is already active, the prt, pri, ugc and umc parameters are ignored.

MACRO CALL FOR VSDR$

```
VSDR$ tsk,[prt],[pri],[ugc,umc],bufadr,[buflen],[efn],[sndpri][,ti]
```

where:

- tsk - is the receiver task name
- prt - is the partition
- pri - is the priority
- ugc - is the UIC group code
- umc - is the UIC member code
- bufadr - is an address of data block
- buflen - is the length of data block in words
- efn - is an event flag number (0 implies no event flag)
- sndpri - is the priority of send (1 through 250)
- ti - is a TI indicator

LOCAL SYMBOL DEFINITIONS

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

- S.DRTN - (length 4 bytes) Task name in Radix-50
- S.DRPN - (4) Partition name in Radix-50
- S.DRPR - (2) Request priority
- S.DRGC - (1) UIC group
- S.DRPC - (1) UIC member
- S.DRBA - (2) Buffer address
- S.DREF - (2) Event flag
- S.DRBL - (2) Buffer length
- S.DRSP - (2) Send priority

- S.DRTI - (2) TI indicator

DSW RETURN CODES

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

Code	Value Returned	Explanation
IS.SUC	+1	Data sent and task requested
IS.SET	+02	Data sent and task resumed
IS.ACT	+03	Data sent to a non-suspended task
IE.UPN	-01	[R] Insufficient pool nodes available for REQUEST
IE.INS	-02	[B] Receiver task not installed
IE.PTS	-03	[R] Partition too small for receiver task
IE.UNS	-04	[B] Insufficient pool nodes for SEND
IE.HWR	-06	[R] Handler task not resident to load task
IE.ITS	-08	[R] Receiver task is disabled
IE.FIX	-09	[R] Receiver task is in process of exiting
IE.PRI	-16	[B] Directive privilege violation
IE.ITP	-88	[B] Invalid TI indicator
IE.IBS	-89	[B] Invalid buffer size (>255)
IE.IUI	-91	[R] Invalid UIC
IE.PNS	-94	[R] Partition not in system
IE.IPR	-95	[R] Invalid priority specified (<0 or > 250)
IE.IEF	-97	[B] Invalid event flag number (event flag number <0 or >64)
IE.ADP	-98	[B] Part of DPB or data block is out of the task's address space
IE.SDP	-99	[B] DIC or DPB size is invalid

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

MACRO EXPANSION FOR VSDR$

```
VSDR$    YOU,PART,40,200,200,MYDATA,30,60,30,TIADDR
.BYTE    73.,$$$T9     ;VSDR$ MACRO DIC, DPB SIZE=VARIABLE (9-12)
.RAD50   /YOU/         ;RECEIVER TASK NAME
.IIF LT  $$$TI-4,.WORD 0   ;FILLER FOR TASK NAME LESS THAN
                       ;4 CHARS
.RAD50   /PART/        ;PARTITION IN WHICH TO REQUEST THE RECEIVER
.WORD    40            ;PRIORITY AT WHICH TO REQUEST THE RECEIVER
.BYTE    200,200       ;UIC AT WHICH TO REQUEST THE RECEIVER
.WORD    MYDATA        ;BUFFER ADDRESS OF DATA TO BE SENT
.WORD    60            ;EVENT FLAG TO BE SET ON SUCCESSFUL SEND
.IIF     GE      $$$T9-10.,.WORD 30   ;LENGTH OF DATA TO BE SENT
.IIF     GE      $$$T9-11.,.WORD 30   ;PRIORITY AT WHICH TO SEND
                       ;DATA
.IIF     EQ      $$$T9-12.,.WORD TIADDR   ;TI OF RECEIVER TASK
```

FORTRAN CALL FOR VSDR$

```
CALL VSNDRR (tsk,[iop],idata,[iefn],[bufsz],[pri],[ti][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character receiver task name in Radix-50 form
- iop - is a four-word integer array containing optional parameters
  - — where:
  - — iop(1) - Partition Name (1st Half) (Radix-50)
  - — iop(2) - Partition Name (2nd Half) (Radix-50)
  - — iop(3) - Run Priority
  - — iop(4) - UIC (User Identification Code)-Zero when none specified
- idata - is an integer array of data to be sent (1 to 255 words)
- iefn - is the number of an event flag to be set
- bufsz - is the size of the send in words
- pri - is the priority of the send
- ti - is the TI of the task to which data is to be sent
- ids - is an integer to receive the Directive Status Word

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

MACRO CALL FOR SDRQ$

```
SDRQ$  tsk,[prt],[pri],[ugc,umc],bufadr,[efn]
```

FORTRAN CALL FOR SDRQ$

```
CALL SNDROR (tsk,[iop],idata,[iefn][,ids])
```

where:

- tsk - is a 2-word, 1- to 6- character receiver task name in Radix-50 form
- iop - is a 6-word integer array containing optional parameters
  - — where:
  - — iop(1) - Radix-50 partition name (1st half)
  - — iop(2) - Radix-50 partition name (2nd half)
  - — iop(3) - run priority
  - — iop(4) - UIC (User Identification Code)

    NOTE: The iop arguments (1), (2), (3), and (4) default to zero when none is specified.
- idata - is a 13-word integer array of data to be sent
- iefn - is the number of an event flag to be set
- ids - is an integer variable to receive the Directive Status Word

WSIG$

The WAIT FOR SIGNIFICANT EVENT directive suspends execution of the issuing task until the next significant event occurs in the system. The directive provides a means of suspending execution for a short time without using a MARK TIME directive, which requires nodes from the system node pool. For example, it may be used to suspend a task which cannot continue because of lack of pool nodes.

Use this directive with discretion in a system which contains the IAS Scheduler because the scheduler causes events to occur many times every second.

MACRO CALL

```
WSIG$
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.ADP	-98	Part of the DPB is out of the issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

```
WSIG$
.BYTE  49.,1    ;WSIG$ MACRO DIC, DPB SIZE=1 WORD
```

FORTRAN CALL

```
CALL WFSNE
```

WTLO$

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

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

if the indicated condition is met when the directive is issued, task execution is not affected.

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

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.

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.

MACRO CALL

```
WTLO$ grp,mask
```

where:

- grp - is the desired group of event flags
- mask - if "grp" is 0,1,2 or 3, "mask" is a 16 bit (16-flag) mask word.

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

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

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

```
WTLO$      1,100034
```

Example: This macro can be used to wait for flag 1, 19, 20, 21, 32, or 64.

```
WTLO$      4,<00001,100034,0,100000>
```

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.IEF	-97	No event flag specified in mask word(s), or flag group indicator other than 0,1,2,3, or 4
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

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.
```

FORTRAN CALL

```
CALL WFLOR (ief1,ief2,...iefn)
```

where:

- ief1,...iefn - is a list of Event Flag Numbers to be taken as the set of Event Flags to be specified in the Directive.

NOTE: The FORTRAN call always generates the group 4 form of the directive with four mask words. Therefore, you can specify any combination of event flags between 1 and 64.

WRSE$

The WAIT FOR SINGLE EVENT FLAG directive suspends the execution of the issuing task until an indicated event flag is set. If the flag is set when the directive is issued, task execution is not affected.

MACRO CALL

```
WTSE$  efn
```

where:

- efn - is an event flag number

LOCAL SYMBOL DEFINITIONS

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

- W.TSEF - (Length 2 bytes) Event flag number

DSW RETURN CODES

Code	Value Returned	Explanation
IS.SUC	+1	Successful completion
IE.IEF	-97	Invalid event flag number (event flag number <1 or >64)
IE.ADP	-98	Part of DPB is out of issuing task's address space
IE.SDP	-99	DIC or DPB size is invalid

MACRO EXPANSION

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

FORTRAN CALL

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

where:

- iefn - is an integer containing an Event Flag Number.
- ids - is an integer variable to receive the Directive Status Word.

A Directive Status Error Returns

The symbols listed below are associated with the directive status codes returned by the Executive. To include these in a MACRO-11 program, the programmer should use the following two lines of code:

```
        .MCALL DRERR$
        DRERR$

;
; STANDARD ERROR CODES RETURNED BY DIRECTIVES IN THE DIRECTIVE STATUS
; WORD
;
        IE.UPN   -01.   INSUFFICIENT DYNAMIC STORAGE
        IE.INS   -02.   SPECIFIED TASK NOT INSTALLED
        IE.PTS   -03.   PARTITION TOO SMALL FOR TASK
        IE.UNS   -04.   INSUFFICIENT DYNAMIC STORAGE FOR SEND
        IE.ULN   -05.   UNASSIGNED LUN
        IE.HWR   -06.   DEVICE HANDLER NOT RESIDENT
        IE.ACT   -07.   TASK NOT ACTIVE
        IE.ITS   -08.   DIRECTIVE INCONSISTENT WITH TASK STATE
        IE.FIX   -09.   TASK ALREADY FIXED/UNFIXED
        IE.CKP   -10.   ISSUING TASK NOT CHECKPOINTABLE
        IE.TCH   -11.   TASK IS CHECKPOINTABLE
        IE.RBS   -15.   RECEIVE BUFFER IS TOO SMALL
        IE.PRI   -16.   PRIVILEGE VIOLATION
        IE.RSU   -17.   RESOURCE IN USE
        IE.NSW   -18.   NO SWAP SPACE AVAILABLE
        IE.ILV   -19.   ILLEGAL VECTOR SPECIFIED
;
;
        IE.AST   -80.   DIRECTIVE ISSUED/NOT ISSUED FROM AST OR
                        TASK NOT DIRECTIVE PRIVILEGED
        IE.MAP   -81.   ILLEGAL MAPPING SPECIFIED
        IE.IOP   -83.   WINDOW HAS I/O IN PROGRESS
        IE.ALG   -84.   ALIGNMENT ERROR
        IE.WOV   -85.   ADDRESS WINDOW ALLOCATION OVERFLOW
        IE.NVR   -86.   INVALID REGION ID
        IE.NVW   -87.   INVALID ADDRESS WINDOW ID
        IE.ITP   -88.   INVALID TI PARAMETER
        IE.IBS   -89.   INVALID SEND BUFFER SIZE
        IE.LNL   -90.   LUN LOCKED IN USE
        IE.IUI   -91.   INVALID UIC
        IE.IDU   -92.   INVALID DEVICE OR UNIT
        IE.ITI   -93.   INVALID TIME PARAMETERS
        IE.PNS   -94.   PARTITION/REGION NOT IN SYSTEM
        IE.IPR   -95.   INVALID PRIORITY
        IE.ILU   -96.   INVALID LUN
        IE.IEF   -97.   INVALID EVENT FLAG
        IE.ADP   -98.   PART OF DPB OUT OF USER'S SPACE
        IE.SDP   -99.   DIC OR DPB SIZE INVALID
```

B Directive Status Error Returns

The symbols listed below are assoicated with the directive status codes returned by the Executive. To include these in a MACRO-11 program, use the following two lines of code:

```
.MCALL DRERR$
DRERR$

;
; Standard Error Codes Returned By Directives in the Directive
; Status Word
;
      IE.UPN     -01.   Insufficient dynamic storage
      IE.INS     -02.   Specified task not installed
      IE.PTS     -03.   Partition too small for task
      IE.UNS     -04.   Insufficient dynamic storage for send
      IE.ULN     -05.   Unassigned LUN
      IE.HWR     -06.   Device handler not resident
      IE.ACT     -07.   Task not active
      IE.ITS     -08.   Directive inconsistent with task state
      IE.FIX     -09.   Task already fixed/unfixed
      IE.CKP     -10.   Issuing task not checkpointable
      IE.TCH     -11.   Task is checkpointable
      IE.RBS     -15.   Receive buffer is too small
      IE.PRI     -16.   Privilege violation
      IE.RSU     -17.   Resource in use
      IE.NSW     -18.   No swap space available
      IE.ILV     -19.   Illegal vector specified
;
;
;
      IE.AST     -80.   Directive issued/not issued from AST
                        or task not directive privileged
      IE.MAP     -81.   Illegal mapping specified
      IE.IOP     -83.   Window has I/O in progress
      IE.ALG     -84.   Alignment error
      IE.WOV     -85.   Address window allocation overflow
      IE.NVR     -86.   Invalid region ID
      IE.NVW     -87.   Invalid address window ID
      IE.ITP     -88.   Invalid TI parameter
      IE.IBS     -89.   Invalid send buffer size
      IE.LNL     -90.   LUN locked in use
      IE.IUI     -91.   Invalid UIC
      IE.IDU     -92.   Invalid device or unit
      IE.ITI     -93.   Invalid time parameters
      IE.PNS     -94.   Partition/region not in system
      IE.IPR     -95.   Invalid priority
      IE.ILU     -96.   Invalid LUN
      IE.IEF     -97.   Invalid event flag
      IE.ADP     -98.   Part of DPB out of user's space
      IE.SDP     -99.   DIC or DPB size invalid
```

Index

A

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W

Reader's Comments

This form is for document comments only. Digital will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

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

Did you find errors in this manual? If so, specify the error and the page number.

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
- ☐ Other (please specify)_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____
or Country

Do Not Tear - Fold Here and Tape

digital™

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 33 MAYNARD MASS

POSTAGE WILL BE PAID BY ADDRESSEE

IAS Engineering/Documentation
Digital Equipment Corporation
5 Wentworth Drive GSF/L20
Hudson, NH 03051-4929

Do Not Tear - Fold Here