RSX-11M-PLUS and Micro/RSX Executive Reference Manual

Order No. AA-JS17A-TC

RSX-11M-PLUS Version 4.0
Micro/RSX Version 4.0

Digital Equipment Corporation Maynard, Massachusetts

First Printing, September 1987

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

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

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.

The following are trademarks of Digital Equipment Corporation:

DEC	EduSystem	UNIBUS
DEC/CMS	IAS	VAX
DEC/MMS	MASSBUS	VAXcluster
DECnet	MicroPDP-11	VMS
DECsystem-10	Micro/RSX	VT
DECSYSTEM-20	PDP	
DECUS	PDT	
DECwriter	RSTS	digital
DIBOL	RSX	

ZK3077

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*	CANADA	INTERNATIONAL
Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061	Digital Equipment of Canada Ltd. 100 Herzberg Road Kanata, Ontario K2K 2A6 Attn: Direct Order Desk	Digital Equipment Corporation PSG Business Manager c/o Digital's local subsidiary or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a trademark of the American Mathematical Society.

Contents

Chapter 2 Significant Events, System Traps, and Stop-Bit Synchronization

Chapter 3 Memory Management Directives

Chapter 4 Parent/Offspring Tasking

Chapter 5 Directive Descriptions

Figures

Tables

Preface

Manual Objectives

The *RSX–11M–PLUS and Micro/RSX Executive Reference Manual* describes the system directives that allow experienced programmers who are familiar with MACRO–11 or with high-level languages such as FORTRAN to use the Executive services to control the execution and interaction of tasks.

Intended Audience

This manual is intended for software developers who are experienced users of MACRO–11 or high-level languages for user-task generation. Information contained in this manual is intended for reference only; no attempt is made to describe the procedures involved in developing user tasks beyond the detailed reference information normally required for directive use. However, Chapters 1 through 4 do contain information that will promote a better understanding of how directives can be used effectively in the multitasking environment. Convenient quick-reference material is included in appendixes for use by the more advanced programmer.

Structure of This Document

A Summary of Technical Changes provides experienced users of the RSX–11M–PLUS and Micro/RSX operating systems with a quick summary of changes to the system software since the previous release of this manual.

Chapter 1 defines system directives and describes their use in both MACRO–11 and high-level language programs.

Chapter 2 defines significant events, event flags, system traps, and stop-bit synchronization, and describes their relationship to system directives.

Chapter 3 introduces the concept of extended logical address space within the framework of memory management directives.

Chapter 4 introduces the concept of parent/offspring tasking, including associated directives, generated data structures, and task communications.

Chapter 5 begins with a short summary of all the directives, arranged according to their functional categories. The summary is followed by detailed descriptions of each directive. The directives are arranged alphabetically according to macro call.

Appendix A contains summaries of the directives, arranged alphabetically according to macro call. These abbreviated descriptions include only the directive name, FORTRAN call, macro call, and parameters.

Appendix B lists the standard error codes returned by the Executive.

Appendix C lists Directive Identification Codes for all directives, using the same octal values that they have in the Directive Parameter Block. A description of how the values are obtained is included.

Appendix D lists all of the directives and the system generation option required (if applicable) to obtain that directive support.

Associated Documents

Manuals that are prerequisite sources of information for readers of this manual are the *Micro/RSX User's Guide* or the *RSX-11M-PLUS and Micro/RSX Task Builder Manual,* and the *PDP-11 MACRO-11 Language Reference Manual* or any other reference manual or user's guide for the appropriate high-level language.

Conventions Used in This Document

The following conventions are used in this manual:

Convention	Meaning
>	A right angle bracket is the default prompt for the Monitor Console Routine (MCR), which is one of the command interfaces used on RSX-11M-PLUS systems. All systems include MCR.
$	A dollar sign followed by a space is the default prompt of the DIGITAL Command Language (DCL), which is one of the command interfaces used on RSX-11M-PLUS and Micro/RSX systems. Many systems include DCL.
MCR>	This is the explicit prompt of the Monitor Console Routine (MCR).
DCL>	This is the explicit prompt of the DIGITAL Command Language (DCL).
xxx>	Three characters followed by a right angle bracket indicate the explicit prompt for a task, utility, or program on the system.
UPPERCASE	Uppercase letters in a command line indicate letters that must be entered as they are shown. For example, utility switches must always be entered as they are shown in format specifications.

Convention	Meaning
command abbreviations	Where short forms of commands are allowed, the shortest form acceptable is represented by uppercase letters. The following example shows the minimum abbreviation allowed for the DCL command DIRECTORY: `$ DIR`
lowercase	Any command in lowercase must be substituted for. Usually the lowercase word identifies the kind of substitution expected, such as a filespec, which indicates that you should fill in a file specification. For example: `filename.filetype;version` This command indicates the values that comprise a file specification; values are substituted for each of these variables as appropriate.
/keyword, /qualifier, or /switch	A command element preceded by a slash (/) is an MCR keyword; a DCL qualifier; or a task, utility, or program switch. Keywords, qualifiers, and switches alter the action of the command they follow.
parameter	Required command fields are generally called parameters. The most common parameters are file specifications.
[option]	Square brackets indicate optional entries in a command line or a file specification. If the brackets include syntactical elements, such as periods (.) or slashes (/), those elements are required for the field. If the field appears in lowercase, you are to substitute a valid command element if you include the field. Note that when an option is entered, the brackets are not included in the command line.
[,...]	Square brackets around a comma and an ellipsis mark indicate that you can use a series of optional elements separated by commas. For example, (argument[,...]) means that you can specify a series of optional arguments by enclosing the arguments in parentheses and by separating them with commas.
{ }	Braces indicate a choice of required options. You are to choose from one of the options listed.
:argument	Some parameters and qualifiers can be altered by the inclusion of arguments preceded by a colon. An argument can be either numerical (COPIES:3) or alphabetical (NAME:QIX). In DCL, the equal sign (=) can be substituted for the colon to introduce arguments. COPIES=3 and COPIES:3 are the same.

Convention	Meaning
( )	Parentheses are used to enclose more than one argument in a command line. For example: `SET PROT = (S:RWED,O:RWED)`
,	Commas are used as separators for command line parameters and to indicate positional entries on a command line. Positional entries are those elements that must be in a certain place in the command line. Although you might omit elements that come before the desired element, the commas that separate them must still be included.
[g,m] [directory]	The convention [g,m] signifies a User Identification Code (UIC). The g is a group number and the m is a member number. The UIC identifies a user and is used mainly for controlling access to files and privileged system functions. This may also signify a User File Directory (UFD), commonly called a directory. A directory is the location of files. Other notations for directories are: [ggg,mmm], [gggmmm], [ufd], [name], and [directory]. The convention [directory] signifies a directory. Most directories have 1- to 9-character names, but some are in the same [g,m] form as the UIC. Where a UIC, UFD, or directory is required, only one set of brackets is shown (for example, [g,m]). Where the UIC, UFD, or directory is optional, two sets of brackets are shown (for example, [[g,m]]).
filespec	A full file specification includes device, directory, file name, file type, and version number, as shown in the following example: `DL2:[46,63]INDIRECT.TXT;3` Full file specifications are rarely needed. If you do not provide a version number, the highest numbered version is used. If you do not provide a directory, the default directory is used. Some system functions default to particular file types. Many commands accept a wildcard character (*) in place of the file name, file type, or version number. Some commands accept a filespec with a DECnet node name.
.	A period in a file specification separates the file name and file type. When the file type is not specified, the period may be omitted from the file specification.
;	A semicolon in a file specification separates the file type from the file version. If the version is not specified, the semicolon may be omitted from the file specification.

Convention	Meaning
@	The at sign invokes an indirect command file. The at sign immediately precedes the file specification for the indirect command file, as follows: `@filename[.filetype;version]`
...	A horizontal ellipsis indicates the following: • Additional, optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis shows where elements of command input or statements in an example or figure have been omitted because they are irrelevant to the point being discussed.
KEYNAME	This typeface denotes one of the keys on the terminal keyboard; for example, the RETURN key.
"print" and "type"	The term "print" refers to any output sent to a terminal by the system. The term "type" refers to any user input from a terminal.
black ink	In examples, what the system prints or displays is printed in black.
red ink	In interactive examples, what the user types is printed in red. System responses appear in black.
blue ink	Text in blue ink indicates that the information pertains to RSX–11M–PLUS multiprocessor systems only.
xxx	A symbol with a 1- to 3-character abbreviation, such as x or RET, indicates that you press a key on the terminal. For example, RET indicates the RETURN key, LF indicates the LINE FEED key, and DEL indicates the DELETE key.
CTRL/a	The symbol CTRL/a means that you are to press the key marked CTRL while pressing another key. Thus, CTRL/Z indicates that you are to press the CTRL key and the Z key together in this fashion. CTRL/Z is echoed on some terminals as ^Z. However, not all control characters echo.

Summary of Technical Changes

The following section describes a feature that is new to the Executive. This new feature is documented in this revision of the *RSX-11M-PLUS and Micro/RSX Executive Reference Manual.*

General Information (GIN$) Directive

The General Information (GIN$) directive provides general information for user tasks. It instructs the system to perform the function found in the Directive Parameter Block (DPB). The functions either set parameters or get information. Each function includes a macro call, buffer format, macro expansion, and Directive Status Word (DSW) return codes.

Chapter 1
Using System Directives

This chapter describes the use of system directives and the ways in which they are processed. The discussion of the system directives assumes that all possible features are present in your system. See the appropriate system generation manual for a list of optional features.

1.1 Introduction

The process that occurs when a task requests the Executive to perform an indicated operation is called a system directive. You use the directives to control the execution and interaction of tasks. If you are a MACRO–11 programmer, you usually issue directives in the form of macros defined in the system macro library. If you are a FORTRAN or other high-level language programmer, you issue system directives in the form of calls to subroutines contained in the system object module library.

System directives enable tasks to perform the following functions:

- Obtain task and system information
- Measure time intervals
- Perform I/O functions
- Spawn other tasks
- Communicate and synchronize with other tasks
- Manipulate a task's logical and virtual address space
- Suspend and resume execution
- Exit

Directives are implemented by the EMT 377 instruction. EMT 0 through EMT 376 (or 375 for unmapped tasks and mapped privileged tasks) are considered to be non-RSX EMT synchronous system traps. These traps cause the Executive to abort the task unless the task has specified that it wants to receive control when such traps occur.

If you are a MACRO-11 programmer, use the system directive macros supplied in the system macro library for directive calls instead of coding individual calls. That way, you need only reassemble the program to incorporate any changes in the directive specifications.

Sections 1.2, 1.3, and 1.6 are intended for all users. Section 1.4 specifically describes the use of macros, while Section 1.5 describes the use of high-level language subroutine calls.

1.2 Directive Processing

Processing a system directive involves the following four steps:

1. The user task issues a directive with arguments that are used only by the Executive. The directive code and parameters that the task supplies to the system are known as the Directive Parameter Block (DPB). The DPB can be either on the user task's stack or in a user task's data section.
2. The Executive receives an EMT 377 generated by the directive macro (or a DIR$ macro) or high-level language subroutine.
3. The Executive processes the directive.
4. The Executive returns directive status information to the task's Directive Status Word (DSW).

Note that the Executive preserves all task registers when a task issues a directive.

The user task issues an EMT 377 (generated by the directive) together with the address of a DPB (or a DPB itself) on the top of the issuing task's stack. When the stack contains a DPB address, the Executive removes the address after processing the directive, and the DPB itself remains unchanged. When the stack contains the actual DPB, the Executive removes the DPB from the stack after processing the directive.

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 and 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 an odd-numbered value. This allows the Executive to determine whether the word on the top of the stack (before EMT 377 was issued) was the address of the DPB (even-numbered value) or the first word of the DPB (odd-numbered value).

The Executive normally returns control to the instruction following the EMT. Exceptions to this are directives that result in an exit from the task that issued them and an asynchronous system trap (AST) exit.

The Executive also clears or sets the Carry bit in the Processor Status Word (PSW) to indicate acceptance or rejection, respectively, of the directive. The DSW, addressed symbolically as $DSW, [1] is set to indicate a more specific cause for acceptance or rejection of the directive. The DSW usually has a value of +1 for acceptance and a range of negative values for rejection (exceptions are success return codes for the directives CLEF$, SETF$, and GPRT$, among others). The RSX-11M-PLUS and Micro/RSX operating systems associate DSW values with symbols, using mnemonics that report either successful completion or the cause of an error (see Section 1.3). (The Instrument Society of America (ISA) FORTRAN calls CALL WAIT and CALL

[1] The Task Builder resolves the address of $DSW. Users addressing the DSW with a physical address are not guaranteed compatibility with IAS, and may experience incompatibilities with future releases of the RSX-11M-PLUS and Micro/RSX operating systems.

START are exceptions because ISA requires positive numeric error codes. The specific return values are listed with the description of each directive.)

In the case of successful Exit directives, the Executive does not return control to the task. If an Exit directive fails, however, control is returned to the task with an error status in the DSW.

On Exit, the Executive frees task resources as follows:

- Detaches all attached devices
- Flushes the AST queue and despecifies all specified ASTs
- Flushes the receive and receive-by-reference queues
- Flushes the clock queue for outstanding Mark Time requests for the task
- Closes all open files (files open for write access are locked)
- Detaches all attached regions, except in the case of a fixed task (where no detaching occurs)
- Runs down the task's I/O
- Deaccesses the group global event flags for the task's group
- Disconnects from interrupts
- Flushes all outstanding CLI command buffers for the task
- Breaks the connection with any offspring tasks
- Marks for deallocation all virtual terminal units that the task has created
- Frees the task's memory if the task was not fixed

If the Executive rejects a directive, it usually does not clear or set any specified event flag. Thus, the task may wait indefinitely if it indiscriminately executes a Wait-for directive corresponding to a previously issued Mark Time directive that the Executive has rejected. You should always ensure that a directive has completed successfully.

1.3 Error Returns

As stated above, the RSX-11M-PLUS and Micro/RSX operating systems associate the error codes with mnemonics that report the cause of the error. In the text of this manual, the mnemonics are used exclusively. The macro DRERR$, which is expanded in Appendix B, provides a correspondence between each mnemonic and its numeric value.

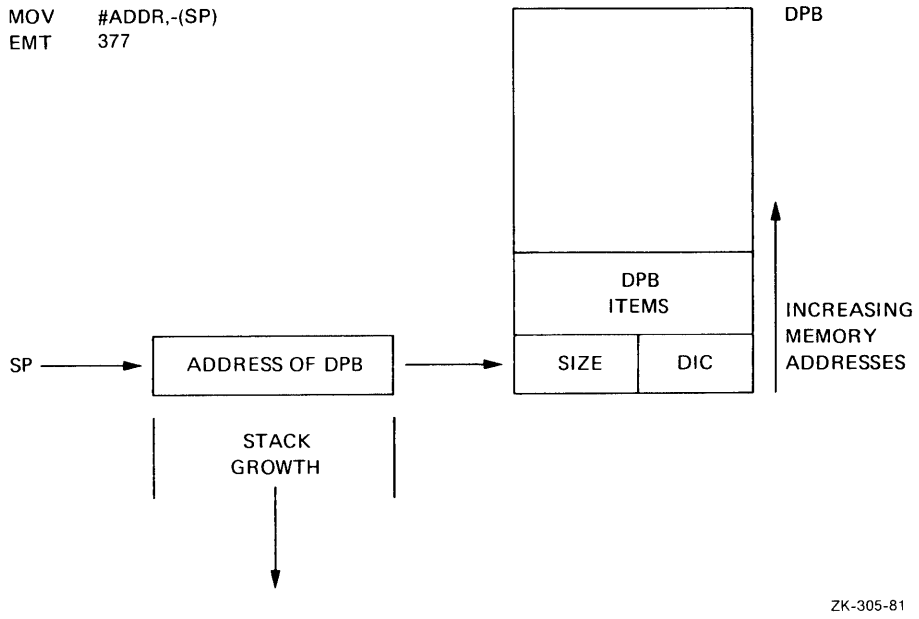
Appendix B also gives the meaning of each error code. In addition, each directive description in Chapter 5 contains specific, directive-related interpretations of the error codes.

1.4 Using the Directive Macros

If you are programming in MACRO–11, you must decide how to create the DPB before you issue a directive. The DPB may either be created on the stack at run time (see Section 1.4.1.3, which describes the $S form) or created in a data section at assembly time (see Sections 1.4.1.1 and 1.4.1.2, which describe the $ form and $C form, respectively). If parameters vary and the code must be reentrant, the DPB must be created on the stack.

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

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

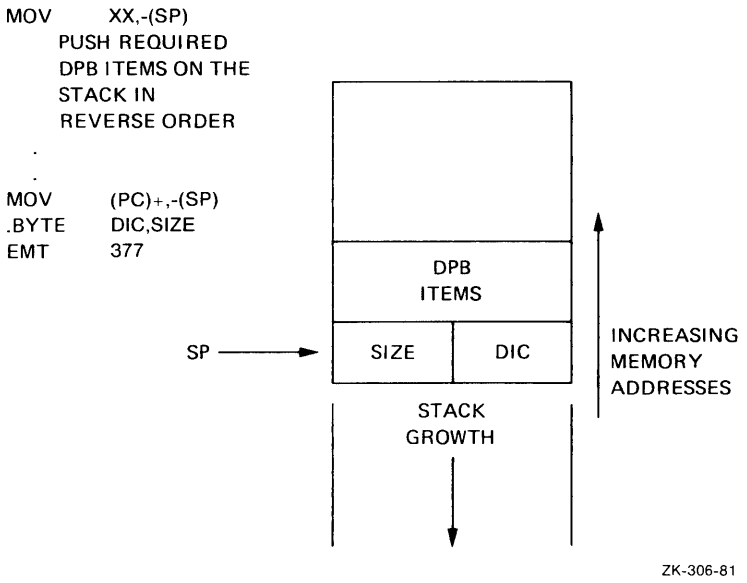


1.4.1 Macro Name Conventions

When you are programming in MACRO–11, you use system directives by including directive macro calls in your programs. The macros for the directives are contained in the System Macro Library (LB:[1,1]RSXMAC.SML). The .MCALL assembler directive makes these macros available to a program. The .MCALL arguments are the names of all the macros used in the program. For example:

```
;
; CALLING DIRECTIVES FROM THE SYSTEM MACRO LIBRARY
; AND ISSUING THEM.
;
.MCALL  MRKT$S,WTSE$S
    .
    .
    .
```

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




```
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, a C or an S. The optional letter or its absence specifies which of three possible macro expansions you want to use. The following sections explain these expansion forms.

1.4.1.1 $ Form

The $ form is useful for a directive operation that is to be issued several times from different locations in a non-reentrant program segment. The $ form is most useful when the directive is issued several times with varying parameters (one or more but not all parameters change) or in a reentrant program section when a directive is issued several times even though the DPB is not modified. 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 in the $ form is produced by a special macro, DIR$ (discussed in Section 1.4.2).

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

1. A $ form of a given directive needs to be issued only once (to produce its DPB).
2. A DIR$ macro associated with a given directive can be issued several times without incurring the cost of generating a DPB each time it is issued.

3. It is easy to access and change the directive's parameters by labeling the start of the DPB and using the offsets defined by the directive.

When a program issues the $ form of a macro call, the parameters required for DPB construction must be valid expressions for MACRO–11 data storage instructions (such as .BYTE, .WORD, and .RAD50). You can alter individual parameters in the DPB. You might do this if you want to use the directive many times with varying parameters.

1.4.1.2 $C Form

Use the $C form when a directive is to be issued only once. 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, in the $C form macro expansion of Section 1.4.5, that the new value of the assembler's location counter redefines the DPB address $$$ each time an additional $C directive is issued.)

The $C form generates a DPB in a separate program section[1] called $DPB$$. The DPB is first followed by a return to the user-specified program section, then by an instruction to push the DPB address onto the stack, and finally by an EMT 377. To ensure that the program reenters the correct program section, you must specify the program section name in the argument list immediately following the DPB parameters. If the argument is not specified, the program reenters the blank (unnamed) program section.

This form also accepts an optional final argument that specifies the address of a routine to be called (by a JSR instruction) if an error occurs during the execution of the directive (see Section 1.4.2).

When a program issues the $C form of a macro call, the parameters required for DPB construction must be valid expressions for MACRO–11 data storage instructions (such as .BYTE, .WORD, and .RAD50). (This is not true for the program-section argument and the error-routine argument, which are not a part of the DPB.)

1.4.1.3 $S Form

Program segments that need to be reentrant should use the $S form. Only the $S form produces the DPB at run time. The other two forms produce the DPB at assembly time.

In this form, 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 for MOV-type instructions. For a 2-word Radix–50 name parameter, the argument must be the address of a 2-word block of memory containing the name. Note that you should not use the stack pointer (or any reference to the stack pointer) to address directive parameters when the $S form is used.[2] (In the example in Section 1.4.1, the error-routine argument ERR is a target address for a JSR instruction; see Section 1.4.3.)

Note that in the $S form of the macro, the macro arguments are processed from right to left. Therefore, when using code of the following form, the result may be obscure:

```
MACRO$S,,(R4)+,(R4)+
```

---

[1] Refer to the *PDP–11 MACRO–11 Language Reference Manual* for a description of program sections.

[2] Subroutine or macro calls can use the stack for temporary storage, thereby destroying the positional relationship between SP and the parameters.

1.4.2 DIR$ Macro

The DIR$ macro allows you to execute a directive with a DPB predefined by the $ form of a directive macro. This macro pushes the DPB address onto the stack and issues an EMT 377 instruction.

The DIR$ macro generates an Executive trap using a predefined DPB:

Format (Macro Call)

DIR$ [adr][,err]

Parameters

adr

The address of the DPB (optional). If specified, the address must be a valid source address for a MOV instruction. If this address is not specified, the DPB or its address must be on the stack.

err

The address of the error return (optional; see Section 1.4.3). If this error return is not specified, an error simply sets the Carry bit 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. The DIR$ macro is not an Executive directive, and DIR$C and DIR$S are not valid macro calls.

1.4.3 Optional Error-Routine Address

The $C and $S forms of macro calls and the DIR$ macro can accept an optional final argument. The argument must be a valid assembler destination operand that specifies the address of a user error routine. For example, the DIR$ macro

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

generates the following code:

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

Since the $ form of a directive macro does not generate any executable code, it does not accept an error-address argument.

1.4.4 Symbolic Offsets

Most system directive macro calls generate local symbolic offsets describing the format of the DPB. The symbols are unique to each directive, and each is assigned an index value corresponding to the offset of a given DPB element.

Because the offsets are defined symbolically, you can refer to or modify DPB elements without knowing the offset values. Symbolic offsets also eliminate the need to rewrite programs if a future release of the RSX–11M–PLUS or Micro/RSX operating system changes a DPB specification.

All $ and $C forms of macros that generate DPBs longer than one word generate local offsets. All informational directives (see Section 5.1.3), including the $S form, also generate local symbolic offsets for the parameter block returned.

If the program uses either the $ or $C form and has defined the symbol $$$GLB (for example, $$$GLB=0), the macro generates the symbolic offsets as global symbols and does not generate the DPB itself. The purpose of this facility is to enable the use of a DPB defined in a different module. The symbol $$$GLB has no effect on the expansion of $S macros.

When using symbolic offsets, you should use the $ form of directives.

1.4.5 Examples of Macro Calls

The examples below show the expansions of the different macro call forms.

- The $ form generates only a DPB, in the current program section.

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

generates the following code:

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

- The $C form generates a DPB in program section DPB. and, in the specified section, the code to issue the directive.

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

generates the following code:

```
.PSECT  $DPB$.
$$$=.                   ; DEFINE TEMPORARY SYMBOL
.BYTE   23.,5           ; "MARK-TIME" DIC AND DPB SIZE
.WORD   1               ; EVENT FLAG NUMBER
.WORD   5               ; TIME INTERVAL MAGNITUDE
.WORD   2               ; TIME INTERVAL UNIT (SECONDS)
.WORD   MTRAP           ; AST ENTRY POINT ADDRESS
.PSECT  PROG1           ; RETURN TO THE ORIGINAL PSECT
MOV     #$$$,-(SP)      ; PUSH DPB ADDRESS ONTO STACK
EMT     377             ; TRAP TO THE EXECUTIVE
```

```
BCC     .+6             ; BRANCH ON DIRECTIVE ACCEPTANCE
JSR     PC,ERR          ; ELSE, CALL ERROR SERVICE ROUTINE
```

- The $S form generates code to push the DPB onto the stack and to issue the directive.

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

generates the following code:

```
MOV     R2,-(SP)        ; PUSH AST ENTRY POINT,
MOV     #2,-(SP)        ; TIME INTERVAL UNIT (SECONDS),
MOV     #5,-(SP)        ; TIME INTERVAL MAGNITUDE,

MOV     #1,-(SP)        ; EVENT FLAG NUMBER,

MOV     (PC)+,-(SP)     ; AND "MARK-TIME" DIC AND DPB SIZE

.BYTE   23.,5           ; ONTO THE STACK

EMT     377             ; TRAP TO THE EXECUTIVE

BCC     .+6             ; BRANCH ON DIRECTIVE ACCEPTANCE
JSR     PC,ERR          ; ELSE, CALL ERROR SERVICE ROUTINE
```

- The DIR$ macro issues a directive that has a predefined DPB.

```
DIR$    R1,(R3)         ; DPB ALREADY DEFINED; ADDRESS IN R1.
```

generates the following code:

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

1.5 Subroutines for FORTRAN and Other High-Level Languages

The RSX–11M–PLUS and Micro/RSX operating systems provide an extensive set of subroutines to perform system directive operations for FORTRAN and other high-level languages, such as BASIC-PLUS-2 and COBOL–81.

Caution

Some Executive routines could interfere with the operation of your high-level language programs, causing unexpected results.

The directive descriptions in Chapter 5 describe the high-level language subroutine calls as well as the macro calls.

The high-level language subroutines fall into three basic groups, as follows:

- Subroutines based on the Instrument Society of America (ISA) Standard ISA 62.1. These subroutines are CALL WAIT and CALL START, which are documented with the descriptions of the Mark Time and Run directives, respectively.
- Subroutines designed to use and control specific process control interface devices supplied by DIGITAL and supported by the RSX–11M–PLUS and Micro/RSX operating systems.

- Subroutines for performing RSX–11M–PLUS and Micro/RSX system directive operations. In general, one subroutine is available for each directive. (Exceptions are the Mark Time and Run directives. The description of Mark Time includes both CALL MARK and CALL WAIT. The description of Run includes both CALL RUN and CALL START.)

All the subroutines described in this manual can be called by FORTRAN programs compiled by either the FORTRAN IV or FORTRAN–77 compiler, and by PDP–11 BASIC-PLUS-2/RSX, PDP–11 Pascal/RSX, PDP–11 DIBOL-83/RSX, and PDP–11 COBOL–81/RSX programs. See Section 1.5.1 for more information.

These subroutines can also be called from programs written in the MACRO–11 assembly language by using PDP–11 FORTRAN calling sequence conventions. These conventions are described in the *VAX FORTRAN User's Guide* and in the *PDP–11 FORTRAN–77 User's Guide*.

Although the subroutines are supported for all the high-level languages listed above, FORTRAN is used in the examples in this chapter and in the descriptions of the directives in Chapter 5. FORTRAN is also the only high-level language discussed in detail in this section.

1.5.1 Supported High-Level Languages

The subroutines support several PDP–11 high-level languages. However, some of the supported languages have restrictions. This section lists the supported languages and describes any restrictions that may apply.

Language	Restrictions
FORTRAN IV	Complete support. No restrictions.
FORTRAN–77	Complete support. No restrictions.
PDP–11 BASIC-PLUS-2/RSX	Complete support. No restrictions.
PDP–11 Pascal/RSX	Complete support. No restrictions.
PDP–11 DIBOL-83/RSX	Complete support. No restrictions.
PDP–11 COBOL–81/RSX	Does not allow external arguments.

Any language using the R5 calling convention can call the routines. Using the R5 calling convention means that calls are made by means of a JSR PC,xxx instruction with R5 pointing to an argument list. The first word of the list is the number of arguments in the list. The remaining words are successive arguments in the list.

If the language does not support EXTERNAL GLOBAL parameters, AST routines cannot be used. If the language does not support null arguments, special care must be taken with omitted parameters. See Section 1.5.2.5.

1.5.2 Subroutine Usage

You call the high-level language subroutines by including the appropriate CALL statement in the program. When the program is linked to form a task, the Task Builder first checks to see whether each specified subroutine is user-defined. If a subroutine is not user-defined, the Task Builder automatically searches for it in the system object module library. If the subroutine is located, it is included in the linked task.

1.5.2.1 Optional Arguments

Many of the subroutines described in this manual have optional arguments. In the subroutine descriptions associated with the directives, optional arguments are designated as such by being enclosed in square brackets ([]). An argument of this kind can be omitted if the comma that immediately follows it is retained. If the argument (or string of optional arguments) comes last, it can simply be omitted and no comma need end the argument list. For example, the format of a call to SUB could be the following:

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

In this case, you may omit the arguments BB, CC, EE, and FF in one of the following ways:

- CALL SUB (AA,,,DD,,)
- CALL SUB (AA,,,DD)

In other cases, a subroutine will use a default value for an unspecified optional argument. Such default values are noted in each subroutine description in Chapter 5.

1.5.2.2 Task Names

In the subroutines, task names may be up to six characters long. Characters permitted in a task name are the letters A to Z, the numerals 0 to 9, and the special characters dollar sign ($) and period (.). Task names are stored as Radix-50 code, which permits up to three characters from the set above to be encoded in one PDP-11 word.

The subroutine calls require that a task name be defined as a 2-word variable or array that contains the task name as Radix-50 code. As an example, for FORTRAN this variable may be any of the following:

- REAL
- INTEGER*4
- An INTEGER*2 array of two elements

The variable may 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 name CCMF1 is to be used in a system directive call, the task name could be defined and used as follows:

```
DATA CCMF1/5RCCMF1/
    .
    .
    .
CALL REQUES (CCMF1)
```

A program may define task names during execution by using the IRAD50 subroutine or the RAD50 function as described in the *VAX FORTRAN User's Guide* or in the *PDP-11 FORTRAN-77 User's Guide*.

1.5.2.3 Integer Arguments for FORTRAN

All of the subroutines described in this manual assume that integer arguments are INTEGER*2-type arguments. Both the FORTRAN IV and FORTRAN–77 systems normally treat an integer variable as one PDP–11 storage word, provided that its value is within the range -32,768 to +32,767. However, if you specify the /I4 option switch when compiling a program, ensure that all integer array arguments used in these subroutines are explicitly specified as type INTEGER*2.

1.5.2.4 GETADR Subroutine

Some subroutine calls include an argument described as an integer array. The integer array contains some values that are the addresses of other variables or arrays. The FORTRAN language does not provide a means of assigning such an address as a value, so you must use the GETADR subroutine described below.

Format for Calling Sequence:

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

ipm

An array of dimension n.

arg1,...argn

Arguments whose addresses are to be inserted in ipm. Arguments are inserted in the order specified. If a null argument is specified, the corresponding entry in ipm is left unchanged. When the argument is an array name, the address of the first array element is inserted into ipm.

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 you to specify a buffer address in the CALL QIO directive.

1.5.2.5 ARGCHA Routine

Some high-level languages do not accept null parameters. To compensate for this, there is an alternate copy of the $ARGCK routine in the system library. The alternate routine is part of the ARGCHA module (SYSLIB/LB:ARGCHA). The routine treats any subroutine parameters specified as -1 as null arguments.

The entry point in the ARGCHA module is deleted from the entry-point table for the system library routines. To use the module, it must be explicitly extracted when the task that wants to use it is built.

Caution

Specified parameter variables that are returned by the Executive (for example, directive status) must be reinitialized if there is any possibility that their returned value may have been set to -1. For example, the standard technique for recovering from low pool (IE.UPN=-1) —executing a Wait for Significant Event directive and then reissuing the original directive—will not work if the Directive Status Word is not reinitialized.

The alternate routine in the ARGCHA module cannot be used for AST addresses. For calls omitting the AST parameter, use the "N" variant of the call, such as CALL SPAWNN for the Spawn directive. Every call with an AST parameter has an "N" variant that suppresses the parameter. For more information, see Section 1.5.5.

1.5.3 The Subroutine Calls

Table 1-1 is a list of the FORTRAN subroutine calls (and corresponding macro calls) associated with the system directives. See Chapter 5 for detailed descriptions.

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

Table 1-1: FORTRAN Subroutines and Corresponding Macro Calls

Directive	FORTRAN Subroutine	Macro Call
Abort Task	CALL ABORT	ABRT$
Assign Channel	CALL ACHN	ACHN$
Alter Priority	CALL ALTPRI	ALTP$
Assign LUN	CALL ASNLUN	ALUN$
Attach Region	CALL ATRG	ATRG$
Create Logical Name	CALL CRELOG CALL CRELON	CLOG$ CLON$
Cancel Scheduled Initiation Requests	CALL CANALL	CSRQ$

Table 1–1 (Cont.): FORTRAN Subroutines and Corresponding Macro Calls

Directive	FORTRAN Subroutine	Macro Call
Cancel Mark Time Requests	CALL CANMT	CMKT$
Checkpoint Common Region	CALL CPCR	CPCR$
Clear Event Flag	CALL CLREF	CLEF$
Connect	CALL CNCT CALL CNCTN	CNCT$
Create Address Window	CALL CRAW	CRAW$
Create Group Global Event Flags	CALL CRGF	CRGF$
Create Region	CALL CRRG	CRRG$
Create Virtual Terminal	CALL CRVT	CRVT$
Declare Significant Event	CALL DECLAR	DECL$S
Delete Logical Name	CALL DELLOG CALL DELLON	DLOG$ DLON$
Disable AST Recognition	CALL DSASTR	DSAR$S
Disable Checkpointing	CALL DISCKP	DSCP$S
Detach Region	CALL DTRG	DTRG$
Eliminate Address Window	CALL ELAW	ELAW$
Eliminate Group Global Event Flags	CALL ELGF	ELGF$
Eliminate Virtual Terminal	CALL ELVT	ELVT$
Emit Status	CALL EMST	EMST$
Enable AST Recognition	CALL ENASTR	ENAR$S
Enable Checkpointing	CALL ENACKP	ENCP$S
Exit If	CALL EXITIF	EXIF$
Exit with Status	CALL EXST	EXST$
Extend Task	CALL EXTTSK	EXTK$
Test for Specified System Feature	CALL FEAT	FEAT$
File Specification Scanner	CALL FSS	FSS$

Table 1–1 (Cont.): FORTRAN Subroutines and Corresponding Macro Calls

Directive	FORTRAN Subroutine	Macro Call
Get Command for Command Interpreter	CALL GTCMCI	GCCI$
Get Command Interpreter Information	CALL GETCII	GCII$
Get Default Directory	CALL GETDDS	GDIR$
Get LUN Information	CALL GETLUN	GLUN$
Get Mapping Context	CALL GMCX	GMCX$
Get MCR Command Line	CALL GETMCR	GMCR$
Get Partition Parameters	CALL GETPAR	GPRT$
Get Region Parameters	CALL GETREG	GREG$
Get Sense Switches	CALL READSW CALL SSWTCH	GSSW$S
Get Task Parameters	CALL GETTSK	GTSK$
Get Time Parameters	CALL GETTIM	GTIM$
Inhibit AST Recognition	CALL INASTR	IHAR$S
Map Address Window	CALL MAP	MAP$
Mark Time	CALL MARK CALL WAIT (ISA Standard call)	MRKT$
Parse FCS	CALL PRSFCS	PFCS$
Parse RMS	CALL PRSRMS	PRMS$
Queue I/O Request	CALL QIO	QIO$
Queue I/O Request and Wait	CALL WTQIO	QIOW$
Read All Event Flags	CALL READEF (only a single, local, common, or group global event flag can be read by a FORTRAN task)	RDXF$ RDAF$
Read Single Event Flag	CALL READEF	RDEF$
Recursive Translation of Logical Name	CALL RCTLON CALL RCTLOG	RLON$ RLOG$
Receive By Reference	CALL RREF	RREF$
Receive by Reference or Stop	CALL RRST	RRST$

Table 1–1 (Cont.): FORTRAN Subroutines and Corresponding Macro Calls

Directive	FORTRAN Subroutine	Macro Call
Receive Data	CALL RECEIV	RCVD$
Receive Data or Exit	CALL RECOEX	RCVX$
Receive Data or Stop	CALL RCST	RCST$
Remove Affinity (RSX–11M–PLUS multiprocessor systems only)	CALL RMAF	RMAF$S
Request and Pass Offspring Information	CALL RPOI	RPOI$
Request	CALL REQUES	RQST$
Resume	CALL RESUME	RSUM$
Run	CALL RUN CALL START (ISA Standard call)	RUN$
Send By Reference	CALL SREF	SREF$
Send Data	CALL SEND	SDAT$
Send Data Request and Pass OCB	CALL SDRP	SDRP$
Send Message	CALL SMSG	SMSG$
Send Next Command	CALL SNXC	SNXC$
Send, Request, and Connect	CALL SDRC CALL SDRCN	SDRC$
Set Affinity (RSX–11M–PLUS multiprocessor systems only)	CALL STAF	STAF$
Set Command Line Interpreter	CALL SCLI	SCLI$
Set Event Flag	CALL SETEF	SETF$
Set System Time	CALL SETTIM	STIM$
Set Default Directory	CALL SETDDS	SDIR$
Spawn	CALL SPAWN CALL SPAWNN	SPWN$
Specify Power Recovery AST	EXTERNAL SUBNAM CALL PWRUP (SUBNAM) (to establish an AST) CALL PWRUP (to remove an AST)	SPRA$

Table 1-1 (Cont.): FORTRAN Subroutines and Corresponding Macro Calls

Directive	FORTRAN Subroutine	Macro Call
Specify Requested Exit AST	CALL SREA CALL SREX	SREA$ SREX$
Stop	CALL STOP	STOP$S
Stop for Logical OR of Event Flags	CALL STLOR CALL STLORS	STLO$
Stop for Single Event Flag	CALL STOPFR	STSE$
Suspend	CALL SUSPND	SPND$S
Task Exit	CALL EXIT	EXIT$S
Test for Specified System Feature	CALL TFEA	TFEA$
Translate Logical Name	CALL TRALON CALL TRALOG	TLON$ TLOG$
Unlock Group Global Event Flags	CALL ULGF	ULGF$S
Unmap Address Window	CALL UNMAP	UMAP$
Unstop	CALL USTP	USTP$
Variable Receive Data	CALL VRCD	VRCD$
Variable Receive Data or Exit	CALL VRCX	VRCX$
Variable Receive Data or Stop	CALL VRCS	VRCS$
Variable Send Data	CALL VSDA	VSDA$
Variable Send, Request, and Connect	CALL VSRC CALL VSRCN	VSRC$
Wait for Single Event Flag	CALL WAITFR	WTSE$
Wait for Logical OR of Event Flags	CALL WFLOR CALL WFLORS	WTLO$
Wait for Significant Event	CALL WFSNE	WSIG$S

Note

The following directives are not available as FORTRAN subroutines:

Directive	Macro Call
AST Service Exit	ASTX$S
Connect to Interrupt Vector	CINT$
General Information	GIN$
Map Supervisor D-space (RSX–11M–PLUS systems only)	MSDS$
Move to/from Supervisor or User I- or D-space (RSX–11M–PLUS systems only)	MVTS$
Specify Command Arrival AST	SCAA$
Specify Floating Point Exception AST	SFPA$
Specify Parity Error AST	SPEA$
Specify Receive By Reference AST	SRRA$
Specify Receive Data AST	SRDA$
Specify SST Vector Table for Debugging Aid	SVDB$
Specify SST Vector Table for Tasks	SVTK$
Supervisor Call (RSX–11M–PLUS systems only)	SCAL$S
Switch State	SWST$

1.5.4 Error Conditions

Each subroutine call includes an optional argument that specifies the integer to receive the Directive Status Word (idsw). When you specify this argument, the subroutine returns a value that indicates whether the directive operation succeeded or failed. If the directive failed, the value indicates the reason for the failure. The possible values are the same as those returned to the Directive Status Word (DSW) in MACRO–11 programs (see Appendix B), except for the two ISA calls, CALL WAIT and CALL START. The ISA calls have positive numeric error codes.

In addition, two types of errors caused by incorrect use of the high-level language subroutines result in a task terminating by means of a breakpoint instruction (BPT). The instruction causes the task to abort with a message such as:

```
Task "tsknam" terminated
Executive interface parameter error
    .
    .
    .
```

```
(register dump)
    .
    .
    .
```

R0 contains the value that identifies the cause of the error. The value can be one of the following:

100000 Indicates that at least one necessary argument was missing from a call to a system directive routine.

000001 Indicates that an event flag number in a call to the STLOR (Stop for Logical OR of Event Flags) routine or to the WTFLOR (Wait for Logical OR of Event Flags) routine was not in the range of 1 through 96 or that not all of the event flags specified in the call were in the same group of 16 event flags.

1.5.5 AST Service Routines

The following routines, which are callable by high-level languages, provide support for ASTs in FORTRAN programs:

- CALL CNCT
- CALL CRVT
- CALL PWRUP
- CALL SDRC
- CALL SPAWN
- CALL SREA
- CALL SREX

Use great caution when coding an AST routine in FORTRAN. The following types of FORTRAN operations may not be performed at AST state (although this list is specific to FORTRAN, other high-level languages will have similar or additional restrictions; consult your language documentation for the level of support and any additional restrictions.):

- FORTRAN I/O of any kind (including ENCODE and DECODE statements and internal file I/O):

  FORTRAN I/O is not reentrant. Therefore, the information in the impure data area may be destroyed.

- Floating-point operations:

  The floating-point processor's context is not saved while in AST state. Since the scientific subroutines use floating-point operations, they may not be called at AST state.

- Traceback information in the generated code:

  Use of traceback corrupts the error recovery in the FORTRAN run-time library. Any FORTRAN modules that will be called at AST state must be compiled without traceback. See the *VAX FORTRAN User's Guide* or the *PDP-11 FORTRAN-77 User's Guide* for more information.

- Virtual array operations:

  Use of virtual arrays at AST state remaps the current array such that any operations at non-AST state will be executed incorrectly.

- Subprograms may not be shared between AST processing and normal task processing.

- EXIT or STOP statements with files open:

  FORTRAN flushes the task's buffers, which could be in an intermediate state. Therefore, data might be lost if any output files are open when the EXIT or STOP statement is executed.

  You can EXIT or STOP at AST state if no output files are open.

  Since the message put out by STOP uses a different mechanism from the normal FORTRAN I/O routines, the act of putting out this message does not corrupt impure data in the run-time system. Therefore, you can issue a STOP statement at AST state unless there are output files open.

Note also the following:

- Any execution-time error at AST state will corrupt the program.

- Use extreme care if the FORTRAN task is overlaid. Both the interface routine and the actual code of the FORTRAN AST routine must be located in the root segment. Any routines that are called at AST state must also be in the root segment.

If you do not want to use ASTs in your program, you can use alternative versions of some of the calls listed at the beginning of this section. The alternative calls use a module in the system library routines called SPNUL that suppresses AST handling. The alternative calls are:

```
CALL CNCTN
CALL CRVT
CALL SDRCN
CALL SPAWNN
CALL VSRCN
```

If you do not want to use ASTs with any of the routines listed at the beginning of this section, using the SPUNL routine is helpful because it saves space. To use the routine, include the following in the command line to the Task Builder:

```
LB:[1,1]SYSLIB/LB:SPNUL
```

1.6 Task States

Many system directives cause a task to change from one state to another. There are two basic task states in RSX-11M-PLUS and Micro/RSX systems: dormant and active. The active state has three substates: ready-to-run, blocked, and stopped.

The Executive recognizes the existence of a task only after it has been successfully installed and has an entry in the System Task Directory (STD). (Task installation is the process whereby a task is made known to the system; see the *RSX-11M-PLUS MCR Operations Manual*, the *RSX-11M-PLUS Command Language Manual*, or the *Micro/RSX User's Guide*.) Once a task has been installed, it is either dormant or active. These states are defined as follows:

Dormant Immediately following the processing of an MCR or DCL INSTALL command, a task is known to the system but dormant. A dormant task has an entry in the STD, but no request has been made to activate it.

Active A task is active from the time it is requested until the time it exits. Requesting a task means issuing the RQST$, RUN$, SPWN$, SDRC$, VSRC$, RPOI$, or SDRP$ macro, or an MCR or DCL RUN command. An active task is eligible for scheduling; a dormant task is not.

The three substates of an active task are as follows:

Ready-to-run A ready-to-run task competes with other tasks for CPU time on the basis of priority. The highest priority ready-to-run task obtains CPU time and thus becomes the current task.

Blocked A blocked task is unable to compete for CPU time for synchronization reasons or because a needed resource is not available. Task priority effectively remains unchanged, allowing the task to compete for memory space.

Stopped A stopped task is unable to compete for CPU time because of pending I/O completion, event flag or flags that are not set, or because the task stopped itself. When stopped, a task's priority effectively drops to zero and the task can be checkpointed by any other task, regardless of that task's priority. If an AST occurs for the stopped task, its normal task priority is restored only for the duration of the AST routine execution; once the AST is completed, task priority returns to zero.

1.6.1 Task State Transitions

Dormant to Active—The following commands or directives cause the Executive to activate a dormant task:

- A RUN$ directive
- A RQST$ directive
- A SPWN$ directive
- An SDRC$ directive
- A VSRC$ directive
- An RPOI$ directive
- An SDRP$ directive
- An MCR or DCL RUN command

Ready-to-Run to Blocked—The following events cause an active, ready-to-run task to become blocked:

- A SPND$ directive
- An unsatisfied Wait-for condition
- Checkpointing of a task out of memory by the Executive

Ready-to-Run to Stopped—The following events cause an active, ready-to-run task to become stopped:

- A STOP$S directive is executed, or an RCST$, SDRP$, GCCI$, or VRCS$ directive is issued when no data packet is available
- An unsatisfied Stop-for condition
- An unsatisfied Wait-for condition while the task has outstanding buffered I/O [1]

Blocked to Ready-to-Run—The following events return a blocked task to the ready-to-run state:

- A RSUM$ directive issued by another task
- An MCR RESUME command or a DCL CONTINUE command
- A Wait-for condition is satisfied
- The Executive reads a checkpointed task into memory

Stopped to Ready-to-Run—The following events return a stopped task to the ready-to-run state, depending upon how the task became stopped:

- A task stopped by the STOP$, RCST$, or VRCS$ directive becomes unstopped by USTP$ directive execution, or with an MCR UNSTOP or DCL START command.
- A Wait-for condition is satisfied for a task with outstanding buffered I/O.
- A task stopped for one or more event flags becomes unstopped when the specified event flag or flags become set.

Active to Dormant—The following events cause an active task to become dormant:

- An EXIT$S, EXIF$, RCVX$, or VRCX$ directive, or an RREF$ or GCCI$ directive that specifies the exit option
- An ABRT$ directive
- An MCR or DCL ABORT command
- A synchronous system trap (SST) for which a task has not specified a service routine

Blocked to Stopped—The following event causes a task that is blocked due to an unsatisfied Wait-for condition to become stopped:

- The task initiates buffered I/O at AST state and then exits from AST state.

Stopped to Blocked—The following event causes a task that is stopped due to an unsatisfied Wait-for condition and outstanding buffered I/O to return to a blocked state:

- Completion of all outstanding buffered I/O

[1] Only in systems that support the checkpointing of tasks during buffered I/O. An I/O request can be buffered only when the task is checkpointable and when the region that I/O is being done to or from is checkpointable.

1.6.2 Removing an Installed Task

You remove an installed task from the system by issuing the MCR or DCL command REMOVE from a privileged terminal. Refer to the *RSX-11M-PLUS MCR Operations Manual,* the *RSX-11M-PLUS Command Language Manual,* or the *Micro/RSX User's Guide.*

1.7 Directive Restrictions for Nonprivileged Tasks

Nonprivileged tasks cannot issue certain Executive directives, except as noted in the following list:

Directive	Macro Call	Comments
Abort Task	ABRT$	A nonprivileged task can abort only those tasks with the same TI: as the task issuing the directive.
Alter Priority	ALTP$	A nonprivileged task can alter its own priority only to those values less than or equal to the task's installed priority.
Cancel Scheduled Initiation Requests	CSRQ$	A nonprivileged task cannot issue this directive except for tasks with the same TI: as the issuing task.
Connect to Interrupt Vector	CINT$	A nonprivileged task cannot issue this directive.
Set Command Line Interpreter	SCLI$	A nonprivileged task cannot issue this directive under any circumstances.

Chapter 2

Significant Events, System Traps, and Stop-Bit Synchronization

This chapter introduces the concept of significant events and describes the ways in which your code can make use of event flags, synchronous and asynchronous system traps, and stop-bit synchronization.

2.1 Significant Events

A significant event is a change in system status that causes the Executive to reevaluate the eligibility of all active tasks to run. (For some significant events, specifically those in which the current task becomes ineligible to run, only those tasks of lower priority are examined.) A significant event is usually caused (either directly or indirectly) by a system directive issued from within a task. Significant events include the following:

- An I/O completion
- A task exit
- The execution of a Send Data directive
- The execution of a Send Data Request and Pass OCB directive
- The execution of a Send, Request, and Connect directive
- The execution of a Send By Reference, Receive By Reference, or Receive By Reference or Stop directive
- The execution of an Alter Priority directive
- The removal of an entry from the clock queue (for instance, resulting from the execution of a Mark Time directive or the issuance of a rescheduling request)
- The execution of a Declare Significant Event directive
- The execution of the round-robin scheduling algorithm at the end of a round-robin scheduling interval
- The execution of an Exit, an Exit with Status, or an Emit Status directive

2.2 Event Flags

Event flags are a means by which tasks recognize specific events. (Tasks also use asynchronous system traps (ASTs) to recognize specific events. See Section 2.3.3.) In requesting a system operation (such as an I/O transfer), a task may associate an event flag with the completion of the operation. When the event occurs, the Executive sets the specified flag. Several examples later in this section describe how tasks can use event flags to coordinate task execution.

Ninety-six event flags are available to enable tasks to distinguish one event from another. Each event flag has a corresponding unique event flag number (EFN). Numbers 1 through 32 form a group of flags that are unique to each task and are set or cleared as a result of that task's operation. Numbers 33 through 64 form a second group of flags that are common to all tasks; hence their name "common flags." Common flags may be set or cleared as a result of any task's operation. The last eight flags in each group, local flags (25-32) and common flags (57-64), are reserved for use by the system. Numbers 65 through 96 form the third group of flags, known as "group global event flags." You can use group global event flags in any application where common event flags are used except that, instead of applying to all tasks, group global event flags apply only to tasks running under UICs containing the group number specified when the flags were created. Four directives (Create Group Global Event Flags, Eliminate Group Global Event Flags, Unlock Group Global Event Flags, and Read Extended Event Flags) provide the Executive support needed for implementing group global event flags.

Tasks can use the common or group global event flags for intertask communication, or use their own local event flags internally. They can set, clear, and test event flags by using the Set Event Flag, Clear Event Flag, and Read All Event Flags directives. (The Read All Event Flags directive will not return the group global event flags. When these flags are in use, read all event flags using the Read Extended Event Flags directive.) Be careful to coordinate the use of group global event flags between multiple applications.

Examples 1 and 2 illustrate the use of common event flags (33-64) to synchronize task execution. Examples 3 and 4 illustrate the use of local flags (1-32).

Example 1

Task B clears common event flag 35 and then blocks itself by issuing a Wait-for directive that specifies common event flag 35.

Subsequently, another task, Task A, specifies event flag 35 in a Set Event Flag directive to inform Task B that it may proceed. Task A then issues a Declare Significant Event directive to ensure that the Executive will schedule Task B.

Example 2

In order to synchronize the transmission of data between Tasks A and B, Task A specifies Task B and common event flag 42 in a Send Data directive.

Task B has specified flag 42 in a Wait-for directive. When Task A's Send Data directive has caused the Executive to set flag 42 and to cause a significant event, Task B proceeds and issues a Receive Data directive because its Wait-for condition has been satisfied.

Example 3

A task contains a Queue I/O Request directive and an associated Wait-for directive; both directives specify the same local event flag. When the task queues its I/O request, the Executive clears the local flag. If the requested I/O is incomplete when the task issues the Wait-for directive, the Executive blocks the task.

When the requested I/O has been completed, the Executive sets the local flag and causes a significant event. The task then resumes its execution at the instruction that follows the Wait-for directive. Using the local event flag in this manner ensures that the task does not manipulate incoming data until the transfer is complete.

Example 4

A task specifies the same local event flag in a Mark Time and an associated Wait-for directive. When the Mark Time directive is issued, the Executive first clears the local flag and subsequently sets it when the indicated time interval has elapsed.

If the task issues the Wait-for directive before the local flag has been set, the Executive blocks the task. The task resumes when the flag is set at the end of the proper time interval. If the flag has been set first, the directive is a no-op and the task is not blocked.

Specifying an event flag does not mean that a Wait-for directive must be issued. Event-flag testing can be performed at any time. The purpose of a Wait-for directive is to stop task execution until an indicated event occurs. Hence, it is not necessary to issue a Wait-for directive immediately following a Queue I/O Request directive or a Mark Time directive.

If a task issues a Wait-for directive that specifies an event flag that is already set, the blocking condition is immediately satisfied and the Executive returns control to the task.

Tasks can issue Stop-for directives as well as Wait-for directives. When this is done, an event-flag condition that is not satisfied results in the task's being stopped (instead of being blocked) until the event flag or flags are set. A task that is blocked still competes for memory resources at its running priority. A task that is stopped competes for memory resources at priority 0.

The simplest way to test a single event flag is to issue the Clear Event Flag or Set Event Flag directive. Both of these directives can cause the following return codes:

```
IS.CLR - Flag was previously clear

IS.SET - Flag was previously set
```

For example, if a set common event flag indicates the completion of an operation, a task can issue the Clear Event Flag directive both to read the event flag and, simultaneously, to reset it for the next operation. If the event flag was previously clear (the current operation was incomplete), the flag remains clear.

2.2.1 Creating, Deleting, and Displaying Group Global Event Flags

The DCL SET GROUPFLAGS command creates and deletes group global event flags. Privileged users can create and delete group global event flags for any group. Nonprivileged users can create and delete global event flags only for the group of which they are members.

The SET GROUPFLAGS command line has the following formats:

```
SET GROUPFLAGS[/qualifier]
Flag? g

SET GROUPFLAGS:g[/qualifier]
```

The qualifiers for the SET GROUPFLAGS command are /CREATE and /DELETE.

The /CREATE qualifier indicates that you want to create a set of group global event flags. This is the default qualifier; it does not need to be specified. Nonprivileged users can create and delete group global event flags for their own login group. Privileged users can create and delete group global event flags for any group.

The /DELETE qualifier indicates that you want to delete a set of group global event flags.

For both qualifiers, g is the group number with which the flags are associated.

The DCL SHOW GROUPFLAGS command displays the group global event flags currently in the system. The command line has the following format:

```
SHOW GROUPFLAGS
```

In the display, the first column is the group number with which the flags are associated. The second column is the access count, which is the number of tasks using the event flags.

The group global event flags are represented in the display by two octal words. The first word represents event flags 65 through 80 (from right to left) and the second word represents event flags 81 through 96 (from right to left).

The final column in the display is reserved for the delete flag DEL, which means the group global event flags are marked for deletion and are not available.

Example

```
$ SHOW GROUPFLAGS [RET]
7       0       000000 000000
200     1       000000 000000
$ SET GROUPFLAGS:303 [RET]
$ SHOW GROUPFLAGS [RET]
7       0       000000 000000
200     1       000000 000000
303     1       000010 000000
```

In this example, the first SHOW GROUPFLAGS command displays the group global event flags currently being used in the system. The display shows that one task is using the event flags for group 200.

The SET GROUPFLAGS command creates group global event flags for group 303. The second SHOW GROUPFLAGS command displays the event flags from before and the newly created flags for group 303. The display also shows that event flag 68 (000010) has been set for group 303.

2.3 System Traps

System traps (also called software interrupts) are a means of transferring control to tasks to allow them to monitor and react to events. The Executive initiates system traps when certain events occur. The trap transfers control to the task associated with the event and gives the task the opportunity to service the event by entering a user-written routine.

There are two kinds of system traps:

- Synchronous system traps (SSTs)—SSTs detect events directly associated with the execution of program instructions. They are synchronous because they always recur at the same point in the program when trap-causing instructions occur. For example, an illegal instruction causes an SST.
- Asynchronous system traps (ASTs)—ASTs detect events that occur asynchronously to the task's execution. That is, the task has no direct control over the precise time that the event—and therefore the trap—may occur. For example, the completion of an I/O transfer may cause an AST to occur.

A task that uses the system-trap facility issues system directives that establish entry points for user-written service routines. Entry points for SSTs are specified in a single table. AST entry points are set by individual directives for each kind of AST. When a trap condition occurs, the task automatically enters the appropriate routine (if its entry point has been specified).

2.3.1 Synchronous System Traps (SSTs)

SSTs can detect the execution of:

- Illegal instructions
- Instructions with invalid addresses
- Trap instructions (TRAP, EMT, IOT, BPT)
- FIS floating-point exceptions (PDP–11/40 processors only)

You can set up an SST vector table that contains one entry per SST type. Each entry is the address of an SST routine that services a particular type of SST (a routine that services illegal instructions, for example). When an SST occurs, the Executive transfers control to the routine for that type of SST. If a corresponding routine is not specified in the table, the task is aborted. The SST routine enables you to process the failure and then return to the interrupted code. Note that if a debugging aid and a user's task both have an SST vector enabled for a given condition, the debugging-aid vector is referenced first to determine the service-routine address.

SST routines must always be reentrant if there is a possibility that an SST can occur within the SST routine itself. Although the Executive initiates SSTs, the execution of the related service routines is indistinguishable from the task's normal execution. Therefore, an AST or another SST can interrupt an SST routine.

2.3.2 SST Service Routines

The Executive initiates SST service routines by pushing the task's processor status (PS), program counter (PC), and trap-specific parameters onto the task's stack. After removing the trap-specific parameters, the service routine returns control to the task by issuing an RTI or RTT instruction. Note that the task's general-purpose registers R0-R5 and SP are not saved. If the SST routine makes use of them, it must save and restore them itself.

To the Executive, SST-routine execution is indistinguishable from normal task execution, so all directive services are available to an SST routine. An SST routine can remove the interrupted PS and PC from the stack and transfer control anywhere in the task; the routine does not have to return control to the point of interruption. Note that any operations performed by the routine (such as the modification of registers or the DSW, or the setting or clearing of event flags) remain in effect when the routine eventually returns control to the task.

A trap vector table within the task contains all the service-routine entry points. You can specify the SST vector table by means of the Specify SST Vector Table for Task directive or the Specify SST Vector for Debugging Aid directive. The trap vector table has the following format:

Word	Offset	Associated Vector	Trap
0	S.COAD	4	Odd or nonexistent memory error (on some PDP-11 processors—for example, the PDP 11/45—an illegal instruction traps here rather than through word 04)
1	S.CSGF	250	Memory protect violation
2	S.CBPT	14	T-bit trap or execution of a BPT instruction
3	S.CIOT	20	Execution of an IOT instruction
4	S.CILI	10	Execution of a reserved instruction
5	S.CEMT	30	Execution of a non-RSX EMT instruction
6	S.CTRP	34	Execution of a TRAP instruction
7	S.CFLT	244	Synchronous floating-point exception (PDP-11/40 processors only)

A zero appearing in the table means that no entry point is specified.

On Micro/RSX systems, an odd address in the table causes another SST to occur when an SST tries to use that particular address as an entry point. If an SST occurs and an associated entry point is not specified in the table, the Executive aborts the task.

On RSX-11M-PLUS systems, an even vector entry causes the SST routine to be executed in the same mode (either user or supervisor) that the processor was in when the SST vector was specified. An odd vector entry causes the SST routine to be executed in the other mode. For example, if the processor was in supervisor mode and the vector entry was odd, the SST routine is executed in user mode.

Depending on the reason for the SST, the task's stack may also contain additional information, as follows:

- Memory protect violation (complete stack)

  SP+10 PS

  SP+06 PC

  SP+04 Memory protect status register (SR0)[1]

  SP+02 Virtual PC of the faulting instruction (SR2)[1] [2]

  SP+00 Instruction backup register (SR1)[1]

[1] For details of SR0, SR1, and SR2, see the section on the memory management unit in the appropriate PDP-11 processor handbook.

[2] On systems with a processor based on the DCJ11 microprocessor chip, the value of SR2 may be random.

- TRAP instruction or EMT other than 377 (and 376 in the case of unmapped tasks and mapped privileged tasks) (complete stack)

  SP+04 PS

  SP+02 PC

  SP+00 Instruction operand (low-order byte) multiplied by 2, non-sign-extended

All items except the PS and PC must be removed from the stack before the SST service routine exits.

2.3.3 Asynchronous System Traps (ASTs)

The primary purpose of an AST is to inform the task that a certain event has occurred—for example, the completion of an I/O operation. As soon as the task has serviced the event, it can return to the interrupted code.

Some directives can specify both an event flag and an AST; with these directives, ASTs can be used as an alternative to event flags or the two can be used together. Therefore, you can specify the same AST routine for several directives, each with a different event flag. Thus, when the Executive passes control to the AST routine, the event flag can determine the action required.

AST service routines must save and restore all registers used. If the registers are not restored after an AST has occurred, the task's subsequent execution may be unpredictable.

Although not able to distinguish execution of an SST routine from task execution, the Executive is aware that a task is executing an AST routine. An AST routine can be interrupted by an SST routine, but not by another AST routine.

The following notes describe general characteristics and uses of ASTs:

- If an AST occurs while the related task is executing, the task is interrupted so that the AST service routine can be executed.
- If an AST occurs while another AST is being processed, the Executive queues the latest AST (first-in/first-out, or FIFO). The task then processes the next AST in the queue when the current AST routine is complete (unless AST recognition was disabled by the AST service routine).

- If a task is suspended or stopped when an associated AST occurs, the task remains suspended or stopped after the AST routine has been executed unless it is explicitly resumed or unstopped either by the AST service routine itself or by another task (the MCR RESUME or DCL CONTINUE command, for example).
- If an AST occurs while the related task is waiting or stopped for an event flag to be set (a Wait-for or Stop-for directive), the task continues to wait after execution of the AST service routine unless the event flag is to be set when the AST exits.
- If an AST occurs for a checkpointed task, the Executive queues the AST (FIFO), brings the task into memory, and then activates the AST when the task returns to memory.
- The Executive allocates the necessary dynamic memory when an AST is specified. Thus, no AST condition lacks dynamic memory for data storage when it actually occurs. The AST reuses the storage allocated for I/O and Mark Time directives. Therefore, no additional dynamic storage is required.
- Two directives, Disable AST Recognition and Enable AST Recognition, allow a program to queue ASTs for subsequent execution during critical sections of code. (A critical section might be one that accesses databases also accessed by AST service routines, for example.) If ASTs occur while AST recognition is disabled, they are queued (FIFO) and then processed when AST recognition is enabled.

2.3.4 AST Service Routines

When an AST occurs, the Executive pushes the task's Wait-for mask word, the DSW, the PS, and the PC onto the task's stack. This information saves the state of the task so that the AST service routine has access to all the available Executive services. The preserved Wait-for mask word allows the AST routines to establish the conditions necessary to unblock the waiting task. Depending on the reason for the AST, the stack may also contain additional parameters. Note that the task's general-purpose registers R0-R5 and SP are not saved. If the routine makes use of them, it must save and restore them itself.

On all RSX–11M–PLUS and Micro/RSX systems, the Wait-for mask word comes from the offset T.EFLM in the task's Task Control Block (TCB). On systems that do not support those features, the Wait-for mask word comes from the offset H.EFLM in the task's header. Its value and the event-flag range to which it corresponds depend on the last Wait-for or Stop-for directive issued by the task. For example, if the last such directive issued was Wait for Single Event Flag 42, the mask word has a value of 1000_8 and the event flag range is from 33 to 48. Bit 0 of the mask word represents flag 33, bit 1 represents flag 34, and so on.

The Wait-for mask word is meaningless if the task has not issued any type of Wait-for or Stop-for directive.

Your code should not attempt to modify the Wait-for mask while in the AST routine. For example, putting a zero in the Wait-for mask results in an unclearable Wait-for state.

After processing an AST, the task must remove the trap-dependent parameters from its stack. That is, everything from the top of the stack down to, but not including, the task's Directive Status Word must be removed. It must then issue an AST Service Exit directive with the stack set as indicated in the description of that directive (see Section 5.3). When the AST service routine exits, it returns control to one of two places: another AST or the original task.

There are 14 variations on the format of the task's stack, as follows:

1. If a task needs to be notified when a Floating Point Processor exception trap occurs, it issues a Specify Floating Point Processor Exception AST directive. If the task specifies this directive, an AST will occur when a Floating Point Processor exception trap occurs. The stack will contain the following values:

   SP+12 Event-flag mask word

   SP+10 PS of task prior to AST

   SP+06 PC of task prior to AST

   SP+04 Task's Directive Status Word

   SP+02 Floating exception code

   SP+00 Floating exception address

   Note

   Refer to the appropriate processor handbook for a description of the FPU exception-code values.

2. If the task needs to be notified of power-failure recoveries, it issues a Specify Power Recovery AST directive. An AST will occur when the power is restored if the task is not checkpointed. The stack will contain the following values:

   SP+06 Event-flag mask word

   SP+04 PS of task prior to AST

   SP+02 PC of task prior to AST

   SP+00 Task's Directive Status Word

3. If a task needs to be notified when it receives either a message or a reference to a common area, it issues either a Specify Receive Data AST or a Specify Receive By Reference AST directive. An AST will occur when the message or common reference is sent to the task. The stack will contain the following values:

   SP+06 Event-flag mask word

   SP+04 PS of task prior to AST

   SP+02 PC of task prior to AST

   SP+00 Task's Directive Status Word

4. When a task queues an I/O request and specifies an appropriate AST service entry point, an AST will occur upon completion of the I/O request. The task's stack will contain the following values:

   SP+10 Event-flag mask word

   SP+06 PS of task prior to AST

   SP+04 PC of task prior to AST

SP+02 Task's Directive Status Word

SP+00 Address of I/O status block for I/O request (or zero if none was specified)

5. When a task issues a Mark Time directive and specifies an appropriate AST service entry point, an AST will occur when the indicated time interval has elapsed. The task's stack will contain the following values:

   SP+10 Event-flag mask word

   SP+06 PS of task prior to AST

   SP+04 PC of task prior to AST

   SP+02 Task's Directive Status Word

   SP+00 Event flag number (or zero if none was specified)

6. An offspring task, connected by a Spawn, Connect, or Send, Request, and Connect directive, returns status to the connected (parent) task or tasks upon exiting by the Exit AST. The parent task's stack contains the following values:

   SP+10 Event-flag mask word

   SP+06 PS of task prior to AST

   SP+04 PC of task prior to AST

   SP+02 Task's Directive Status Word

   SP+00 Address of exit status block

7. If a command arrives for a CLI, the Command Arrival AST routine is entered. The stack contains:

   SP+10 Event-flag mask word

   SP+06 PS of task prior to AST

   SP+04 PC of task prior to AST

   SP+02 Task's Directive Status Word

   SP+00 Command-buffer address

8. If a parent task issues a Create Virtual Terminal directive, the input and output AST routines are entered. The task's stack contains the following values:

   SP+14 Event-flag mask word

   SP+12 PS of task prior to AST

   SP+10 PC of task prior to AST

   SP+06 Task's Directive Status Word

   SP+04 Third parameter word (vertical format control - VFC) of the offspring request

   SP+02 Byte count of offspring request

SP+00 Virtual terminal unit number (low byte); I/O subfunction code of offspring request (high byte)

9. If the Attach/Detach AST routine is entered for attaching to a virtual terminal, the task's stack contains the following values:

   SP+14 Event-flag mask word

   SP+12 PS of task prior to AST

   SP+10 PC of task prior to AST

   SP+06 Task's Directive Status Word

   SP+04 Second word of offspring task name

   SP+02 First word of offspring task name

   SP+00 Virtual terminal unit number (low byte); I/O subfunction code of offspring request (high byte)

10. If the Attach/Detach AST routine is entered for detaching from a virtual terminal, the task's stack contains the following values:

    SP+14 Event-flag mask word

    SP+12 PS of task prior to AST

    SP+10 PC of task prior to AST

    SP+06 Task's Directive Status Word

    SP+04 Second word of offspring task name = 0

    SP+02 First word of offspring task name = 0

    SP+00 Virtual terminal unit number (low byte); I/O subfunction code of offspring request (high byte)

11. If a task issues a Specify Parity Error AST directive, the parity-error AST service routine is entered. The task's stack contains the following values:

    SP+62 Event-flag mask word

    SP+60 PS of task prior to AST

    SP+56 PC of task prior to AST

    SP+54 Task's Directive Status Word

    SP+52

    SP+50

    SP+46

    SP+44

    SP+42

SP+40

SP+36

SP+34

SP+32 Contents of memory-parity CSRs

SP+30 (hardware-dependent information)

SP+26

SP+24

SP+22

SP+20

SP+16

SP+14

SP+12 Contents of cache control register

SP+10 Contents of memory system-error register

SP+06 Contents of high-error-address register

SP+04 Contents of low-error-address register

SP+02 Processor identification (single-processor system=0)

SP+00 Number of bytes to add to SP to clean the stack (52)

12. If a task is aborted by a directive or a DCL or MCR command when the Specify Requested Exit AST is in effect, the abort AST is entered. The task's stack contains the following values:

    SP+06 Event-flag mask word

    SP+04 PS of task prior to AST

    SP+02 PC of task prior to AST

    SP+00 Task's Directive Status Word

13. If a task is aborted by a directive or a DCL or MCR command when the Extended Specify Requested Exit AST is in effect, the abort AST is entered. The task's stack contains the following values:

    SP+12 Event-flag mask word

    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 Trap-dependent parameter

SP+00 Number of bytes to add to SP to clean the stack

14. If a task issues a QIO IO.ATA function to the full-duplex terminal driver, unsolicited terminal input will cause the AST service routine to be entered. Upon entry into the routine, the task's stack contains the following values:

    SP+10 Event-flag mask word

    SP+06 PS of task prior to AST

    SP+04 PC of task prior to AST

    SP+02 Task's Directive Status Word

    SP+00 Unsolicited character in low byte; parameter 2 in high byte

2.4 Stop-Bit Synchronization

Stop-bit synchronization allows tasks to be checkpointed during terminal (buffered) I/O or while waiting for an event to occur (for example, an event flag to become set or an Unstop directive to become issued). You can control synchronization between tasks by the setting of the task's Task Control Block (TCB) stop bit.

When the task's stop bit is set, the task is blocked from further execution, its priority for memory allocation effectively drops to zero, and it may be checkpointed by any other task in the system regardless of priority. If checkpointed, the task remains out of memory until its stop bit is cleared, at which time the task becomes unstopped, its normal priority for memory allocation becomes restored, and it is considered for memory allocation based on the restored priority.

If the stopped task receives an AST, it becomes unstopped until it exits from the AST routine. Memory allocation for the task during the AST routine is based on the task's priority prior to the stopped state. Note that a task cannot be stopped when an AST is in progress, but the AST routine can issue either an Unstop or Set Event Flag directive to reference the task. This causes it to remain unstopped after it issues the AST Service Exit directive.

There are three ways in which a nonprivileged task can become stopped and three corresponding ways for it to become unstopped. Only one method for stopping a task can be applied at one time. The methods are as follows:

- A task is stopped whenever it is in a Wait-for state and has outstanding buffered I/O. A task is unstopped when the buffered I/O is completed or when the Wait-for condition is satisfied.
- You can stop a task for event flag or flags by issuing the Stop for Single Event Flag directive or the Stop for Logical OR of Event Flags directive. In this case, the task can be unstopped only by setting the specified event flag or flags.
- You can stop a task by issuing the Stop directive, the Receive Data or Stop directive, or the Get Command for Command Interpreter directive. In this case, the task can be unstopped only by issuing the Unstop directive or the MCR UNSTOP or DCL START command.

You cannot stop a task when an AST is in progress (AST state). Any directives that cause a task to become stopped are invalid at the AST state.

When a task is stopped for any reason at the task state, it can still receive ASTs. If the task has been checkpointed, it becomes eligible for entry back into memory when an AST is queued for it. The task retains its normal priority in memory while it is at the AST state or has ASTs queued. Once it has exited the AST routine with no other ASTs queued, the task is again stopped and effectively has zero priority for memory allocation.

You can use the following directives for stop-bit synchronization:

Directive	Function
Stop	This directive stops the issuing task and cannot be issued at the AST state.
Receive Data or Stop and Variable Receive Data or Stop	These directives attempt to dequeue send-data packets from the specified task (or any task if none is specified). If there is no such packet to be dequeued, the issuing task is stopped. These directives cannot be issued at the AST state.
Receive by Reference or Stop	This directive requests the Executive to dequeue the next packet in the receive-by-reference queue of the issuing (receiver) task. The task will stop if there are no packets in the queue.
Stop for Logical OR of Event Flags	This directive stops the issuing task until the specified flags in the specified group of local event flags become set. If any of the specified event flags are already set, the task does not become stopped. This directive cannot be issued at the AST state.
Stop for Single Event Flag	This directive stops the issuing task until the indicated local event flag becomes set. If the specified event flag is already set, the task does not become stopped. This directive cannot be issued at the AST state.
Unstop	This directive unstops a task that has become stopped by the Stop or the Receive Data or Stop directive.
Get Command for Command Interpreter	This directive stops a CLI task when there is no command queued for it. The GC.CST option must be specified to force the task to stop. This directive cannot be issued at the AST state.

Chapter 3
Memory Management Directives

Within the framework of memory management directives, this chapter discusses extended logical address space, regions, virtual address windows, and fast mapping.

3.1 Addressing Capabilities of a Task

Without the overlaying of tasks, a task cannot explicitly refer to a location with an address greater than 177777 (32K words). The 16-bit word size of the PDP–11 imposes this restriction on a task's addressing capability. Overlaying a task means that it must first be divided into segments: a single root segment, which is always in memory, and any number of overlay segments, which can be loaded into memory as required. Unless a 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 a total of 32K words, a task requiring large amounts of data must access data that resides on disk. Data is disk-based not only because of limited memory space, but also because transmission of large amounts of data between tasks is only practical by means of disk. An overlaid task, 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 resident in memory at run time. A group of memory management directives provide a task with this capability. The directives overcome the 32K-word addressing restriction by allowing the task to dynamically change the physical locations that are referred to by a given 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 physical memory requirements.

On RSX–11M–PLUS operating systems, you can effectively triple the memory available for tasks on PDP–11 systems that are capable of operating in supervisor mode through the use of supervisor-mode library routines and separate user-mode instruction space (I-space) and data space (D-space). Supervisor-mode library routines are instruction-only routines that are mapped into supervisor-mode I-space (32K words maximum). User task parameters, stack, and any locations that may be written are mapped into supervisor-mode D-space (32K words maximum).

User tasks that use I- and D-space may consist of up to 32K words of instructions and 32K words of data.

3.1.1 Address Mapping

In a mapped system, you do not need to know where a task resides in physical memory. Mapping, the process of associating task addresses with available physical memory, is transparent and is accomplished by the KT11 memory management hardware. (See the appropriate PDP–11 processor handbook for a description of the KT11.) When a task references a location (virtual address), the KT11 determines the physical address in memory. The memory management directives use the KT11 to perform address mapping at a level that is visible to and controlled by you.

3.1.2 Address Space

The following concepts—logical address space and virtual address space—provide a basis for understanding the functions performed by the memory management directives:

- Logical address space—A task's logical address space is the total amount of physical memory to which the task has access rights. This includes various areas called regions (see Section 3.3). Each region occupies a contiguous block of memory.
- Virtual address space—A task's virtual address space corresponds to the 32K-word address range imposed by the PDP–11's 16-bit word length. The task can divide its virtual address space into segments called virtual address windows (see Section 3.2).

If the capabilities supplied by the 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 point 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.

3.1.3 Supervisor-Mode Addressing

RSX–11M–PLUS systems support PDP–11 processors that are capable of operating in supervisor mode. Supervisor mode is one of three possible modes (user, kernel, and supervisor) in which these systems can operate. In user mode, eight active page registers (APRs) are available for address mapping of user tasks. Note that only I-space APRs are employed in user mode for both instructions and data.

Supervisor-mode support doubles the instruction space available to tasks because 16 APRs (8 user-mode I-space and 8 supervisor-mode I-space) are available for address mapping. The contents of user-mode D-space APRs (I-space APRs on systems that do not support user data space) are copied into supervisor-mode D-space APRs to allow supervisor-mode routines to access user-mode data. (Refer to the appropriate PDP–11 processor handbook for a complete description of address mapping, memory management, and the various APR registers).

3.1.4 Mapping Structure of I- and D-Space Tasks

RSX–11M–PLUS systems support user-mode I- and D-space. Tasks that do not use D-space execute with I- and D-space overmapped. However, these tasks may create D-space windows. This allows tasks to increase the total virtual size without a full implementation of I- and D-space.

Tasks in which the Task Builder has separated the I-space and D-space structures are mapped separately (I- and D-space are not overmapped). The overall mapping structure for these tasks is as follows:

Window 0 Root I-space

Window 1 Task header, stack, and root D-space

Window 2 I-space of the read-only section if a multiuser task; memory-resident overlays if not a multiuser task

Window 3 D-space of the read-only section if a multiuser task; memory-resident overlays if not a multiuser task.

Window 4 Memory-resident overlays

. .

. .

. .

The multiuser section of a multiuser task is also separated into I- and D-space areas. Memory-resident libraries are not separated and are normally mapped by both I- and D-space. Common regions are also normally mapped through D-space only. The memory management directives can be used to attach to and map a data common with an explicit D-space window.

3.2 Virtual Address Windows

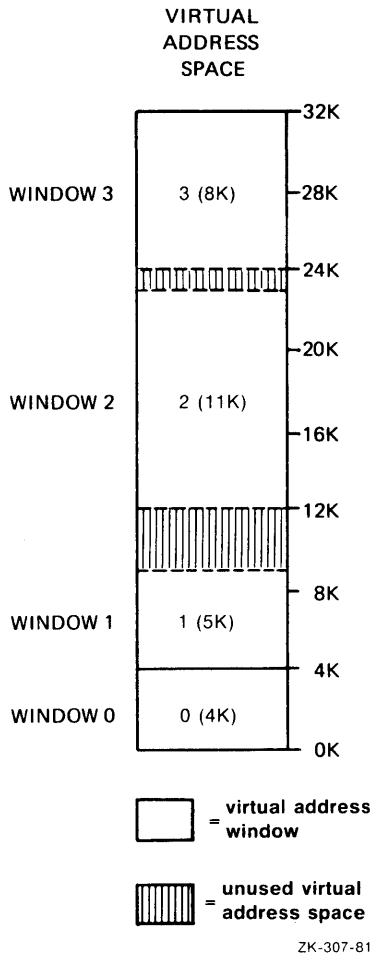
In order to manipulate the mapping of virtual addresses to various logical areas, you must first divide a task's 32K words of virtual address space into segments. These segments are called virtual address windows. Each window encompasses a contiguous range of virtual addresses, which must begin on a 4K-word boundary (that is, the first address must be a multiple of 4K). The number of windows defined by a task can vary from 1 to 23. For all tasks, window 0 is not available to you. For tasks on RSX–11M–PLUS using I- and D-space, windows 0 and 1 are not available to you. The size of each window can range from a minimum of 32 words to a maximum of 32K words.

A task that includes directives to manipulate address windows dynamically must have window blocks set up in its task header. The Executive uses window blocks to identify and describe each currently existing window. You specify the required number of additional window blocks (the number used for windows created by the memory management directives) to be set up by the Task Builder when linking the task (see the *RSX–11M–PLUS and Micro/RSX Task Builder Manual*). The number of blocks that you specify should equal the maximum number of windows that will exist at any one time when the task is running.

A window's identification is a number from 0 to 15_{10} for either user or, on RSX-11M-PLUS, supervisor windows on systems that support supervisor-mode libraries (0 to 23_{10} for systems with user and supervisor I- and D-space). The number is an index to the window's corresponding window block. The address window identified by 0 is the window that maps the task's header and root segment. For tasks on RSX-11M-PLUS using I- and D-space, window 0 maps the task's root instruction segment; window 1 maps the task's header, stack, and root data segment. On all systems, the Task Builder automatically creates window 0, which is mapped by the Executive and cannot be specified in any directive.

Figure 3-1 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.

Figure 3-1: Virtual Address Windows



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, reference to that address will cause an illegal address trap to occur. 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 (see Section 3.3).

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

- Map a window to all or part of a region
- Unmap a window from one region in order to map it to another region
- Unmap a window from one part of a region in order to map it to another part of the same region

3.3 Regions

A region is a portion of physical memory to which a task has (or potentially may have) access. The current window-to-region mapping context determines that 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 regions, as follows:

- Task region—A contiguous block of memory in which the task runs.
- Static common region—An area, such as a global common area, defined by an operator at run time or at system-generation time. Static common regions are dynamically loaded whenever needed.
- Dynamic region—A region created dynamically at run time by issuing the memory management directives.
- Shareable region—A read-only portion of multiuser tasks that are in shareable regions.

Tasks refer to a region by means of a region ID returned to the task by the Executive. A region ID from 0 to 23 refers to a task's static attachment. Region ID 0 always refers to a task's task region. On RSX–11M–PLUS and Micro/RSX systems, region ID 1 always refers to the read-only (pure code) portion of multiuser tasks. All other region IDs are actually addresses of the attachment descriptor maintained by the Executive in the system dynamic storage region (pool).

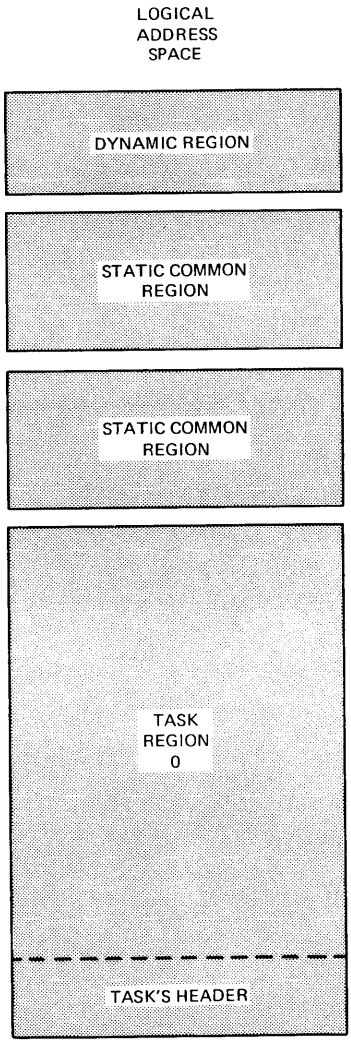
Figure 3-2 shows a sample collection of regions that could make up a task's logical address space at some given time. The header and root segment are always part of the task region. Since a region occupies a contiguous area of memory, each region is shown as a separate block.

Figure 3-3 illustrates a possible mapping relationship between the windows and regions shown in Figures 3-1 and 3-2.

3.3.1 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 time. One result is an increased potential for task interaction by means of shared regions. For example, a task can create a dynamic region to accommodate 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. Thus, tasks can map to required routines at run time rather than linking to them at task-build time.

Figure 3–2: Region Definition Block

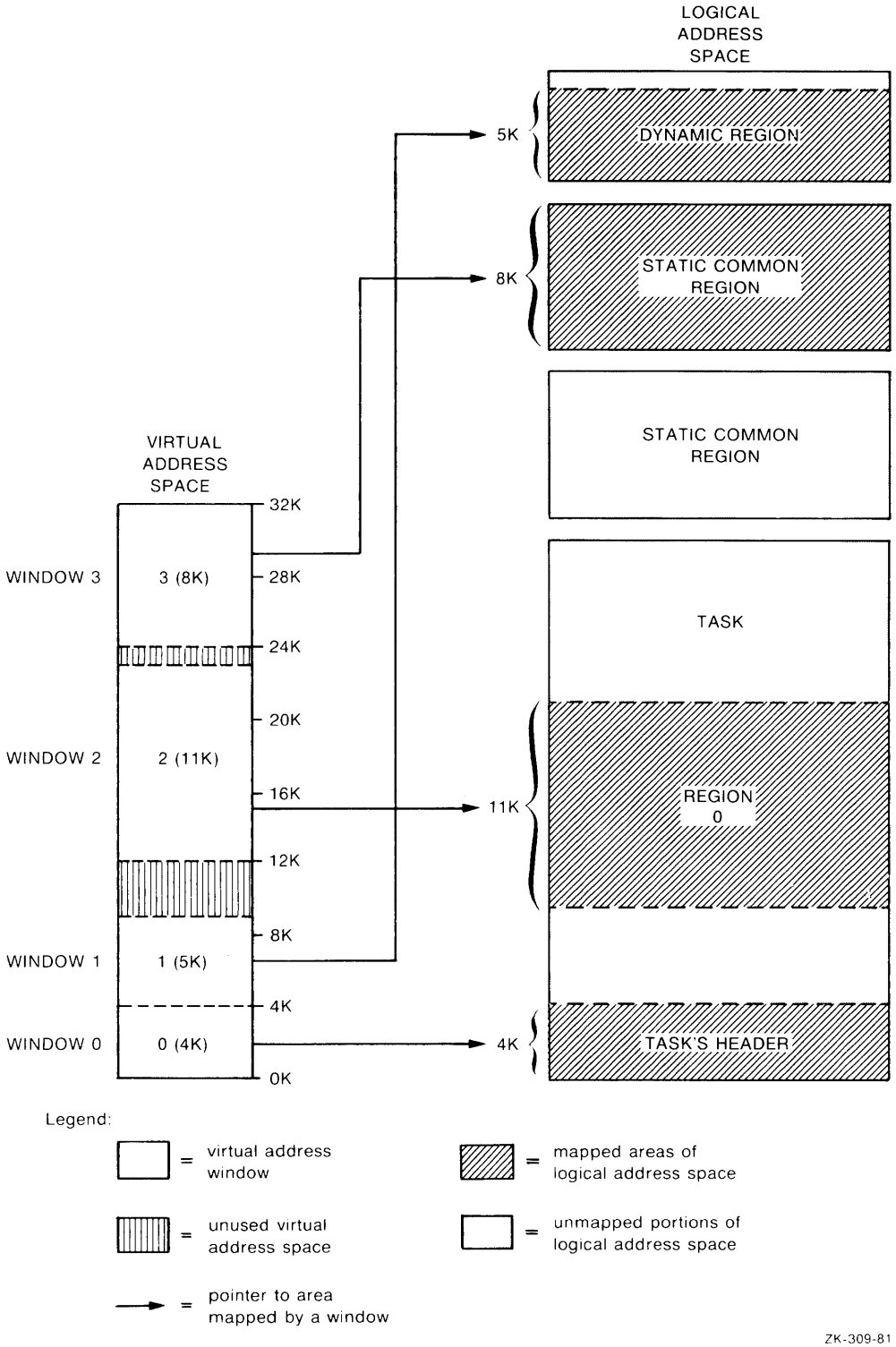


3.3.2 Attaching to Regions

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

- All tasks are automatically attached to regions that are linked to them at task-build time.
- A task can issue a directive to attach itself to a named static common region or a named dynamic region.
- A task can request the Executive to attach another specified task to any region within the logical address space of the requesting task.

Figure 3–3: Mapping Windows to Regions



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 from it. (It should be noted that fixed tasks do not automatically become detached from regions upon exiting.)

Note

Each Send By Reference directive issued by a sending task creates a new attachment descriptor for the receiving task. However, multiple Send By Reference directives referencing the same region require only one attachment descriptor. After the receiving task issues a series of Receive By Reference directives and all pending data requests have been received, the task should detach from the region in order to return the attachment descriptors to pool.

It is possible to avoid multiple attachment descriptors when sending and receiving data by reference. Setting the WS.NAT bit in the Window Descriptor Block (see Section 3.5.2) causes the Executive to create a new attachment descriptor for that region only if necessary (that is, if the task is currently not attached to the region).

3.3.3 Region Protection

A task cannot indiscriminately attach to any region. 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.

To determine when tasks may add to their logical address space by attaching regions, the following points must be considered (note that all considerations presume there is no protection violation):

- Any task can attach to a named dynamic region, provided it knows the name. In the case of an unnamed dynamic region, a task can attach to the region only after receiving a Send By Reference directive from the task that created the region.
- Any task can issue a Send By Reference directive to attach another task to any region. The reference sent includes the access rights with which the receiving task attaches to the region. The sending task can grant only those access rights that it has itself.
- Any task can map to a named static common region.

3.4 Directive Summary

This section briefly describes the function of each memory management directive.

3.4.1 Create Region Directive (CRRG$)

The Create Region directive creates a dynamic region in a designated system-controlled partition and optionally attaches the issuing task to it.

3.4.2 Attach Region Directive (ATRG$)

The Attach Region directive attaches the issuing task to a static common region or to a named dynamic region.

3.4.3 Detach Region Directive (DTRG$)

The Detach Region directive detaches the issuing task from a specified region. Any of the task's address windows that are mapped to the region are automatically unmapped.

3.4.4 Create Address Window Directive (CRAW$)

The Create Address Window directive creates an address window, establishes its virtual address base and size, and optionally maps the window. Any other windows that overlap with the range of addresses of the new window are first unmapped and then eliminated.

3.4.5 Eliminate Address Window Directive (ELAW$)

The Eliminate Address Window directive eliminates an existing address window, unmapping it first if necessary.

3.4.6 Map Address Window Directive (MAP$)

The Map Address Window directive maps an existing window to an attached region. The mapping begins at a specified offset from the start of the region and goes to a specified length. If the window is already mapped elsewhere, the Executive unmaps it before carrying out the map assignment described in the directive.

3.4.7 Unmap Address Window Directive (UMAP$)

The Unmap Address Window directive unmaps a specified window. After the window has been unmapped, its virtual address range cannot be referenced until the task issues another mapping directive.

3.4.8 Send by Reference Directive (SREF$)

The Send By Reference directive 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 referred to.

3.4.9 Receive by Reference Directive (RREF$)

The Receive By Reference directive requests the Executive to select the next packet from the receive-by-reference queue of the issuing task and make the information in the packet available to the task. Optionally, the directive can map a window to the referenced region or cause the task to exit if the queue does not contain a receive-by-reference packet.

3.4.10 Receive by Reference or Stop Directive (RRST$)

The Receive By Reference or Stop directive requests the Executive to select the next packet from the receive-by-reference queue of the issuing task and make the information in the packet available to the task. The directive can map a window to the referenced region or cause the task to stop if the queue does not contain a receive-by-reference packet.

3.4.11 Get Mapping Context Directive (GMCX$)

The Get Mapping Context directive 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 through a series of Create Address Window directives.

3.4.12 Get Region Parameters Directive (GREG$)

The Get Region Parameters directive causes the Executive to supply the issuing task with information about either its task region (if no region ID is given) or an explicitly specified region.

3.5 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, and 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:

- Allow the task to specify which directive options it wants the Executive to perform
- 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:

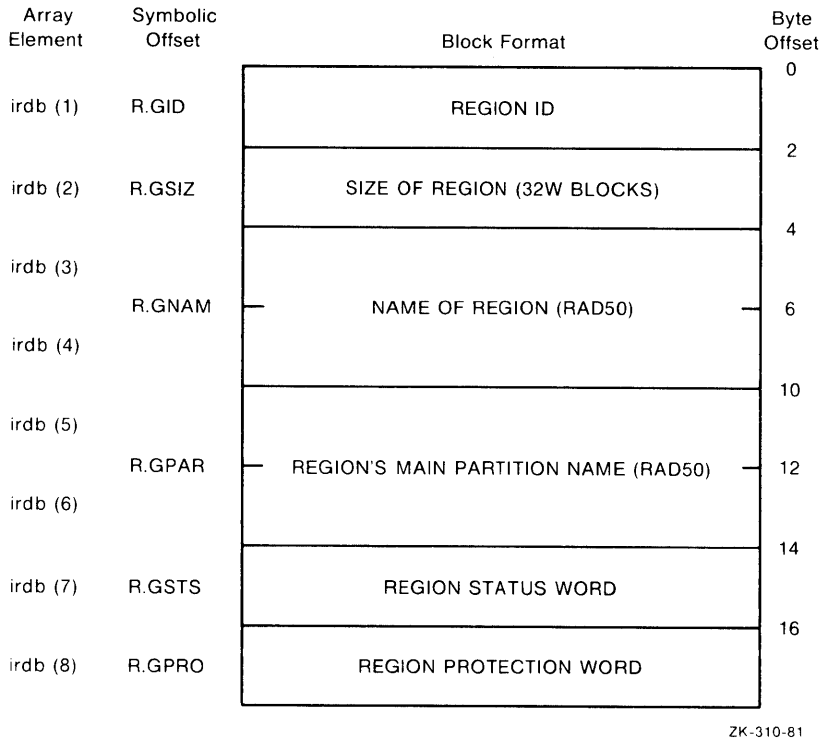
- The Region Definition Block (RDB)
- The Window Definition Block (WDB)

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. Symbolic address offset values are assigned by the task, pointing to locations within an RDB or a WDB. The task can change the contents of these locations to 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.

3.5.1 Region Definition Block

Figure 3-4 illustrates the format of a Region Definition Block (RDB). In addition to the symbolic offsets defined in the diagram, the region status word R.GSTS contains defined bits that may be set or cleared by the Executive or the task. (Undefined bits are reserved for future expansion.) The bits and their definitions follow.

Figure 3–4: Region Definition Block



Bit	Definition
RS.CRR=100000	Region was created successfully.
RS.UNM=40000	At least one window was unmapped on a detach.
RS.MDL=200	Mark region for deletion on last detach. When a region is created by means of a CRRG$ directive, it is normally marked for deletion on the last detach. However, if RS.NDL is set when the CRRG$ directive is executed, the region is not marked for deletion. Subsequent execution of a DTRG$ directive with RS.MDL set marks the region for deletion.
RS.NDL=100	Created region is not to be marked for deletion on last detach.
RS.ATT=40	Attach to created region.
RS.NEX=20	Created region is not extendable.
RS.DEL=10	Delete access desired on attach.
RS.EXT=4	Extend access desired on attach.

Bit	Definition
RS.WRT=2	Write access desired on attach.
RS.RED=1	Read access desired on attach.

These symbols are defined by the RDBDF$ macro, as described in Section 3.5.1.1.

The following memory management directives require a pointer to an RDB:

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 or RS.UNM bit to indicate to the task what actions were taken. The Executive never modifies the other bits.

3.5.1.1 Using Macros to Generate an RDB

RSX–11M–PLUS and Micro/RSX systems provide 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 as follows:

```
RDBDF$
```

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

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

Parameters

siz

The region size in 32-word blocks.

nam

The region name (Radix–50).

par

The name (Radix–50) of the partition in which to create the region.

sts

The bit definitions of the region status word.

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

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

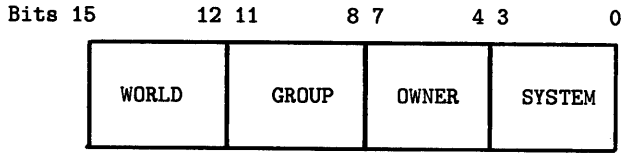
bit

A defined bit to be set. See Section 3.5.1.

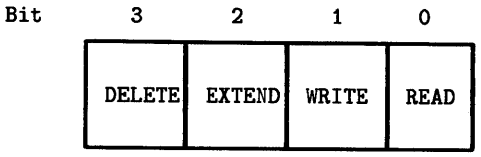
pro

The region's default protection word.

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



Each of these four categories has four bits, with each bit representing a type of access:



A bit value of 0 indicates that the specified type of access is to be allowed. A bit value of 1 indicates that the specified 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 pointed to the RDB defined by this expanded macro call, the Executive would create a region 102_{10} 32-word blocks in length, named ALPHA, in a partition named GEN. The defined bits specified in the sts argument tell the Executive:

- Not to mark the region for deletion on the last detach
- To attach region ALPHA to the task issuing the directive macro call
- To grant read and write access to the attached task

The protection word specified as 167000_8 assigns a default protection mask to the region. The octal number, which has a binary equivalent of 1110 1110 0000 0000, grants access as follows:

World	(1110)	Read access only
Group	(1110)	Read access only
Owner	(0000)	All access
System	(0000)	All access

If the Create Region directive is successful, the Executive returns to the issuing task a region-ID value in the location accessed by symbolic offset R.GID and sets the defined bit RS.CRR in the status word R.GSTS.

3.5.1.2 Using FORTRAN to Generate an RDB

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

```
CALL ATRG (Attach Region directive)
CALL CRRG (Create Region directive)
CALL DTRG (Detach Region directive)
```

See the *PDP-11 FORTRAN IV Language Reference Manual* or the *PDP-11 FORTRAN-77 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 the paragraph following this list)
irdb(8)	Region protection code

You can modify the region status word irdb(7) by setting or clearing the appropriate bits. See the list in Section 3.5.1 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 the FORTRAN library routine IRAD50. (See the appropriate FORTRAN User's Guide.)

3.5.2 Window Definition Block

Figure 3-5 illustrates the format of a Window Definition Block (WDB). The block consists of a number of symbolic address offsets to specific WDB locations. One of the locations is the window status word W.NSTS, which contains defined bits that can be set or cleared by the Executive or the task. (All undefined bits are reserved for future expansion.) The bits and their definitions follow.

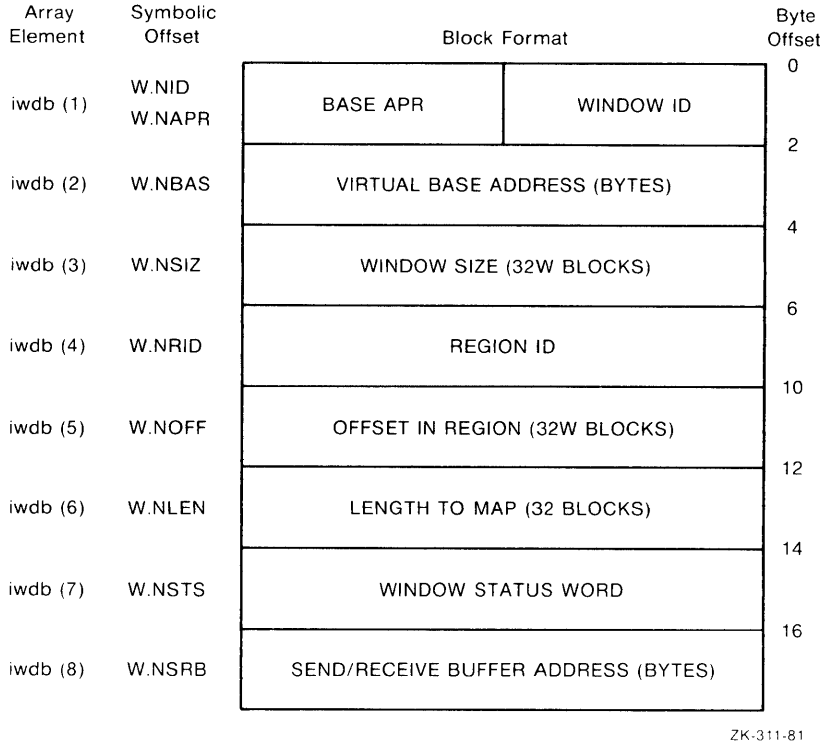
Bit	Definition
WS.CRW=100000	Address window was created successfully.
WS.UNM=40000	At least one window was unmapped by a Create Address Window, Map Address Window, or Unmap Address Window directive.
WS.ELW=20000	At least one window was eliminated by a Create Address Window or Eliminate Address Window directive.
WS.RRF=10000	Reference was received successfully.
WS.NBP=4000	Do not bypass cache for CRAW$ directives.
WS.BPS=4000	Always bypass cache for MAP$ directives.
WS.RES=2000	Map only if resident.
WS.NAT=1000	Create attachment descriptor only if necessary for Send By Reference directives.
WS.64B=400	Define 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 by a Create Address Window, Receive By Reference, or Receive By Reference or Stop directive.
WS.RCX=100	Exit if no references to receive.
WS.SIS=40	Create window in supervisor I-space (RSX–11M–PLUS systems only).
WS.UDS=20	Create window in user-mode D-space (RSX–11M–PLUS systems only).
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.

These symbols are defined by the WDBDF$ macro, as described in Section 3.5.2.1.

The following directives require a pointer to a WDB:

Create Address Window (CRAW$)
Eliminate Address Window (ELAW$)
Map Address Window (MAP$)
Unmap Address Window (UMAP$)

Figure 3–5: Window Definition Block



Send By Reference (SREF$)
Receive By Reference (RREF$)
Receive By Reference or Stop (RRST$)

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. After completing the directive operation, the Executive can then set any of these bits to tell the task what actions were taken. The Executive never modifies the other bits.

3.5.2.1 Using Macros to Generate a WDB

RSX-11M-PLUS and Micro/RSX systems provide 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 as follows:

WDBDF$

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

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

Parameters

apr

A number from 0 to 7 that specifies the window's base Active Page Register (APR). The APR determines the 4K 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.

siz

The size of the window in 32-word blocks.

rid

A region ID.

off

The offset within the region to be mapped, in 32-word blocks.

len

The length within the region to be mapped, in 32-word blocks (defaults to the value of siz).

sts

The bit definitions of the window status word.

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

<bit1[!...!bitn]>

bit

A defined bit to be set. See Section 3.5.2.

srb

A send/receive buffer virtual address.

The macro call:

```
WDBBK$   5,76.,0,50.,,<WS.64B!WS.MAP!WS.WRT>
```

expands to:

```
.BYTE   0,5      (Window ID returned in low-order byte)
.WORD   0        (Base virtual address returned here)
.WORD   76.
.WORD   0
.WORD   50.
.WORD   0
.WORD   WS.64B!WS.MAP!WS.WRT
.WORD   0
```

If a Create Address Window directive pointed to the WDB defined by the macro call expanded above, the Executive would perform the following actions:

- Create a window 76_{10} blocks long beginning at APR 5 (virtual address 20K or 120000_8) and align the window on a 64-byte boundary (WS.64B)
- Map 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)

- Start the map 50_{10} blocks from the base of the region, and map an area either equal to the length of the window (76 decimal blocks) or to the length remaining in the region, whichever is smaller (because the macro call defaulted the len argument)
- Return values to the symbolic W.NID (the window's ID) and W.NBAS (the window's virtual base address)

3.5.2.2 Using FORTRAN to Generate a WDB

When programming in FORTRAN, you must create an 8-word, single-precision integer array as the WDB to be supplied in the following 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)
CALL RRST (Receive By Reference or Stop directive)

See the *PDP-11 FORTRAN IV Language Reference Manual* or the *PDP-11 FORTRAN-77 Language Reference Manual* for information on the creation of arrays.

A WDB array has the following format:

Word	Contents
iwdb(1)	Bits 0 through 7 contain the window ID; bits 8 through 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 the paragraph following this list)
iwdb(8)	Address of send/receive buffer

You can modify the window status word iwdb(7) by setting or clearing the appropriate bits. See the list in Section 3.5.2 that describes the defined bits. The bit values are listed alongside the symbolic offsets.

Please note the following:

- For any directive other than Create Address Window, the contents of bits 8 through 15 of iwdb(1) must normally be set without destroying the value in bits 0 through 7.
- A call to GETADR (see Section 1.5.1.4) can be used to set up the address of the send/receive buffer. For example:

```
CALL GETADR(IWDB,,,,,,,,IRCVB)
```

This call places the address of buffer IRCVB in array element 8. The remaining elements are unchanged. The subroutines SREF, RREF, and RRST also set up this value. If you use these routines, you do not need to use GETADR.

3.5.3 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 required as input can have any value when the directive is issued. Chapter 5 describes which offsets and settings are relevant for each memory management directive. The values assigned by the task are called input parameters, whereas those assigned by the Executive are called output parameters.

3.6 Privileged Tasks

When a privileged task maps to the Executive and the I/O page, the system normally dedicates five or six APRs to this mapping. A privileged task can issue memory management directives to remap any number of these APRs to regions. Take great care when using the directives in this way because such remapping can cause obscure bugs to occur. When a directive unmaps a window that formerly mapped the Executive or the I/O page, the Executive restores the former mapping.

Note

Tasks should not remap APR 0. If APR 0 is remapped, information such as the DSW, overlay structures, or language run-time systems will become inaccessible.

3.7 Fast Mapping

The RSX–11M–PLUS and Micro/RSX operating systems provide a special addition to the memory management facilities called fast mapping. Fast mapping provides a mechanism for executing a subset of the Map directive at a greatly increased speed. For tasks that use this subset, fast mapping can be as much as ten to thirty times faster than the Map directive.

However, the fast-mapping facility has the following restrictions:

1. Only the offset to the map field (W.NOFF) and, optionally, the length to the map field (W.NLEN) may be modified by the fast-mapping facility.
2. The interface to the fast-mapping facility is designed for speed, not for ease of programming. Debugging a task using fast mapping may be more difficult than using the Map directive. Specifically, protecting the operating system and its data structures is the only validation of parameters that is done. For example, specifying a random value for the window ID may cause a random address window to be modified.
3. The interface uses the IOT instruction. Tasks use IOT instructions for internal communications and other functions, but tasks that use fast mapping cannot use the IOT instruction for any purpose other than fast mapping.
4. The interface uses registers for passing arguments rather than using a DPB (saving 200-300 instructions over the Map directive). This means that the MACRO–11 programmer must be careful about register usage when using fast mapping.

5. Fast mapping increases the size of the task header, which means that fast mapping can be used only with tasks with external headers. (Most tasks on RSX–11M–PLUS systems have external headers.)

These restrictions (particularly the first one in number 2) should not deter the use of fast mapping in high-performance applications. However, it is recommended that you first get the application running with the Map directive, varying only the W.NOFF and W.NLEN fields, and then replace the directive with fast mapping.

3.7.1 Using Fast Mapping

To use fast mapping, the task must first have an extended header to include the fast-mapping extension area. This is achieved by using the Task Builder fast map switch (see the *RSX–11M–PLUS and Micro/RSX Task Builder Manual*, the *RSX–11M–PLUS Command Language Manual*, or the *Micro/RSX User's Guide*) or by installing the task with the fast-mapping option (see the *RSX–11M–PLUS MCR Operations Manual*, the *RSX–11M–PLUS Command Language Manual*, or the *Micro/RSX User's Guide*).

Before issuing a fast-mapping call, the task must create and map the window by using the Create Address Window and Map directives or the CRAW and MAP high-level language calls.

Three parameters are required for the fast-mapping call. The first parameter is a window identifier, which is a function of the first APR mapped by the window. (It is 10 octal times the W.NAPR field in the WDB, plus 100_8 if the window is in user D-space; see the following table.) The second parameter is the offset field to map and the third parameter is the length of the window to map. The ID and offset fields are required; the length is optional. If the length is to be specified, the high bit of the ID field must be set. Thus, the following values are used for window IDs (all values are octal):

Starting APR number	ID if length not set	ID if length set
User I-space 0	000000	100000
User I-space 1	000010	100010
User I-space 2	000020	100020
User I-space 3	000030	100030
User I-space 4	000040	100040
User I-space 5	000050	100050
User I-space 6	000060	100060
User I-space 7	000070	100070

Starting APR number	ID if length not set	ID if length set
User D-space 0	000100	100100
User D-space 1	000110	100110
User D-space 2	000120	100120
User D-space 3	000130	100130
User D-space 4	000140	100140
User D-space 5	000150	100150
User D-space 6	000160	100160
User D-space 7	000170	100170

The offset field is specified in 32-word blocks, the same as it would be for the W NOFF value in the Map directive. If the length-to-map field is not specified, it is assumed to be the same as W.NSIZ. If it is specified (high bit of window ID set), then that length is mapped unless the value is specified as zero. If it is zero, then either the size (W.NSIZ) or the size of the region minus the offset field, whichever is smaller, is used. This handling is identical to that for W.NLEN in the Map directive.

Note that the speed of fast mapping is affected by the parameter values. Not specifying the length-to-map field is the fastest form, requiring about 25 instructions for a single APR window, plus a minimum of two additional instructions for each APR. Specifying a fixed length is slower, and forcing the length calculation is slower still. The fastest form is about thirty times the speed of the Map directive, the slowest form about ten times that speed.

3.7.2 MACRO-11 Calling Sequence

MACRO-11 programs call the fast-mapping facility by placing the window ID in register 0, the offset in register 1, and the length in register 2, and then issuing an IOT instruction. R0 is returned as the status (IS.SUC or IE.ALG) and R2 is returned as the length if it was defaulted. The contents of register 3 are destroyed by the call.

Examples

Changing only the window offset field:

```
MOV     #40,R0          ; Window starts in user-I APR 4
MOV     #200,R1         ; Offset = 4K words (200 32-word blocks)
IOT                     ; Issue fast map
TST     R0              ; Success?
BPL     GOOD            ; If PL yes
```

Changing the window offset field, fixed length specified:

```
MOV     #100150,R0      ; Window starts in user-D APR 5
                        ; High bit set to indicate length specified
MOV     #100,R1         ; Offset = 2K words (100 32-word blocks)
MOV     #100,R2         ; Set length to map to 2K words
IOT                     ; Issue fast map
TST     R0              ; Success?
BPL     GOOD            ; If PL yes
```

Changing the window offset field, defaulted length specified:

```
MOV     #100150,R0      ; Window starts in user-D APR 5
                        ; High bit set to indicate length specified
MOV     #100,R1         ; Offset = 2K words (100 32-word blocks)
CLR     R2              ; Force calculation to W.NSIZ or remaining size
                        ; of region
IOT                     ; Issue fast map
TST     R0              ; Success?
BPL     GOOD            ; If PL yes
```

3.7.3 High-Level Language Interface

High-level languages (FORTRAN–77 is used in the following examples) call either the FMAP or FMAPL interface routines, specifying the three parameters as previously described. Two of the variables are updated to reflect the directive status and the length (if it was defaulted). All parameters should be specified as 16-bit integer values.

Unlike other high-level language routines, FMAP and FMAPL do not validate parameters. Omitting a parameter or specifying a bad value will probably cause a task SST to occur.

Examples

Changing only the window offset field:

```
INTEGER*2 WINDID , WINDOF               ! Force 16-bit integer values
WINDID = '40'0                          ! Set fast map window ID for user-I APR 4
WINDOF = '200'0                         ! Set offset to 4K words (200 32-word blocks)
CALL FMAP ( WINDID , WINDOF )           ! Do fast map
IF ( WINDID .GT. 0 ) ...                ! If successful...
```

Changing the window offset field, fixed length specified:

```
INTEGER*2 WINDID , WINDOF                ! Force 16-bit integer values
INTEGER*2 WINDLN
WINDID = '100150'0                       ! Set fast map window ID for user-D
                                         ! APR 5
WINDOF = '100'0                          ! Set offset to 2K words
                                         ! (100 32-word blocks)
WINDLN = '100'0                          ! Set length to map to 2K words
CALL FMAP ( WINDID , WINDOF , WINDLN )   ! Do fast map
IF ( WINDID .GT. 0 ) ...                 ! If successful ...
```

Changing the window offset field, defaulted length specified:

```
INTEGER*2 WINDID , WINDOF                ! Force 16-bit integer values
INTEGER*2 WINDLN
WINDID = '100150'0                       ! Set fast map window ID for
                                         ! user-D APR 5
WINDOF = '100'0                          ! Set offset to 2K words
                                         ! (100 32-word blocks)
WINDLN = '100'0                          ! Set length to map to 2K words
CALL FMAPL ( WINDID , WINDOF , WINDLN )  ! Do fast map
IF ( WINDID .GT. 0 ) ...                 ! If successful...
```

3.7.4 Status Returns

There are two possible status returns from the fast-mapping call:

IS.SUC Operation successful.

IE.ALG The specified mapping parameters are illegal for the region to which the target window is mapped. This means that the sum of the offset and length fields is greater than the accessible part of the window. This may also imply that the specified window ID was not valid.

There is no specific error code for an invalid window ID because the Executive code that checks for invalid window-offset parameters also traps invalid ID errors. The Executive clears bits 14 through 7 and 2 through 0 of the window ID before it is interpreted. Specifying random values in the window ID may cause legitimate mapping changes.

Chapter 4
Parent/Offspring Tasking

4.1 Overview of Parent/Offspring Tasking Support

Parent/offspring tasking has many real-time applications in establishing and controlling complex interrelationships between tasks. A parent task is one that starts or connects to another task, called an offspring task. A major application for the parent-offspring task relationship is batch processing: when running tasks, you can set up task relationships and parameters on line to control the processing of a batch job (or jobs) that run off line.

Starting (or activating) offspring tasks is called "spawning." Spawning also includes the ability to establish task communications; a parent task can be notified when an offspring task exits and can receive status information from the offspring task. Status returned from an offspring task to a parent task indicates successful completion of the offspring task or identifies specific error conditions.

4.2 Directive Summary

This section summarizes the directives for parent/offspring tasking and intertask communication.

4.2.1 Parent/Offspring Tasking Directives

There are two classes of parent/offspring tasking directives:

- Spawning—directives that create a connection between tasks
- Chaining—directives that transfer a connection

The following directives can connect a parent task to an offspring task:

- Spawn—This directive requests activation of, and connects to, a specific offspring task.

  An offspring task spawned by a parent task has the following three task functions that are not provided by the Request or Run directives:

  - A spawned offspring task can be a command line interpreter (CLI).
  - A spawned offspring task on an RSX-11M-PLUS or Micro/RSX system can have a virtual terminal as its terminal input device (TI:).

- A spawned offspring task can return current status information or exit status information to a connected parent task or tasks.

The Spawn directive includes the following options:

- Queuing a command line for the offspring task (which may be a command line interpreter)
- Establishing the offspring task's TI: as a physical terminal or as a previously created virtual terminal unit
- For privileged or CLI tasks, designating any terminal as the offspring TI:

- Connect—This directive establishes task communications for synchronizing with the exit status or emit status issued by a task that is already active.
- Send, Request, and Connect—This directive sends data to the specified task, requests activation of the task if it is not already active, and connects to the task.

The following directives allow one task to chain to another task:

- Request and Pass Offspring Information—This directive allows an offspring task to pass its parent connection to another task, thus making the new task the offspring of the original parent task. The RPOI$ directive offers all the options of the Spawn directive.
- Send Data Request and Pass Offspring Control Block—This directive sends a data packet for a specified task, passes its parent connection to that task, and requests activation of the task if it is not already active.

A parent task can connect to more than one offspring task using the Spawn and Connect directives, as appropriate. In addition, the parent task can use the directives in any combination to make multiple connections to offspring tasks.

An offspring task can be connected to multiple parent tasks. An Offspring Control Block is produced (in addition to those already present) each time a parent task connects to the offspring task.

4.2.2 Task Communication Directives

The following directives in an offspring task return status to connected parent tasks:

- Exit with Status—This directive in an offspring task causes the offspring task to exit, passing status words to all connected parent tasks (one or more) that have been previously connected by a Spawn, Connect, or Send, Request, and Connect directive.
- Emit Status—This directive causes the offspring task to pass status words to either the specified connected task or to all connected parent tasks if no task is explicitly specified.

When status is passed to tasks in this manner, the parent task or tasks no longer remain connected.

The following standard offspring-task status values can be returned to parent tasks:

Status	Value	Action
EX$WAR	0	Warning - task succeeded, but irregularities are possible
EX$SUC	1	Success - results should be as expected
EX$ERR	2	Error - results are unlikely to be as expected
EX$SEV	4	Severe error - one or more fatal errors detected, or task aborted

These symbols are defined in the file DIRSYM.MAC. They become defined locally when the EXST$ macro is invoked. However, the exit status may be any 16-bit value.

4.3 Connecting and Passing Status

Offspring-task exit status can be returned to a connected (parent) task or tasks by issuing the Exit with Status directive. Offspring tasks can return status to one or more connected parent tasks at any time by issuing the Emit Status directive. Note that only connected parent-offspring tasks can pass status.

The means by which a task connects to another task are indistinguishable once the connecting process is complete. For example, Task A can become connected to Task B in one of four ways:

- Task A spawned Task B when Task B was inactive.
- Task A connected to Task B when Task B was active.
- Task A issued a Send, Request, and Connect directive to Task B when Task B was either active or inactive.
- Task A either spawned or connected to Task C, which then chained to Task B by means of either an RPOI$ directive or an SDRP$ directive.

Regardless of the way in which Task A became connected to Task B, Task B can pass status information back to Task A, set the event flag specified by Task A, or cause the AST specified by Task A to occur in any of the following ways (note that once offspring-task status is returned to one or more parent tasks, the parent tasks become disconnected):

- Task B issues a normal (successful) exit directive. Task A receives a status of EX$SUC.
- Task B is aborted. Task A receives a severe error status of EX$SEV.
- Task B issues an Exit with Status directive, returning status to Task A upon completion of Task B.
- Task B issues an Emit Status directive specifying Task A. If Task A is multiply connected to Task B, the OCBs that contain information about these multiple connections are stored in a FIFO queue. The first OCB is used to determine which event flag, AST address, and exit status block to use.
- Task B issues an Emit Status directive to all connected tasks (no task name specified).

When a task has previously specified another task in a Spawn, Connect, or Send, Request, and Connect directive and then exits, and if status has not yet been returned, the OCB representing this connection remains queued. However, the OCB is marked to indicate that the parent task has exited. When this OCB is subsequently dequeued due to an Emit Status directive, or any type of exit, no action is taken because the parent task has exited. This procedure is followed to help a multiply connected task to remain synchronized when parent tasks exit unexpectedly.

The following examples show directives being used for intertask synchronization (the macro calls for the directives are given). Task A is the parent task and Task B is the offspring task.

Task A	Task B	Action
SPWN$	EXST$	Task A spawns Task B. Upon Task B's completion, Task B returns status to Task A.
CNCT$	EXST$	Task A connects to active Task B. Upon Task B's completion, Task B returns status to Task A.
SDRC$	RCVX$, EMST$	Task A sends data to Task B, requests Task B if it is presently not active, and connects to Task B. Task B receives the data, does some processing based on the data, returns status to Task A (possibly setting an event flag or declaring an AST), and becomes disconnected from Task A.
SDRC$, USTP$	RCST$, EMST$	Task A sends data to Task B, requests Task B if it is presently not active, connects to Task B, and unstops Task B. Task B becomes unstopped (if Task B previously could not dequeue the data packet), receives the data, does some processing based on the data, and returns status to Task A (possibly setting an event flag or declaring an AST).
SDAT$, USTP$	RCST$	Task A queues a data packet for Task B and unstops Task B. Task B receives the data.
SPWN$	RPOI$ SDRP$	Task A spawns Task B. Task B chains to Task C by issuing an RPOI$ or an SDRP$ directive. Task A is now Task C's parent. Task A is no longer connected to Task B.

4.4 Spawning System Tasks

One special use of the Spawn directive is to pass a command line to a system task. You may use the Spawn directive to pass a command line to a command line interpreter or to an installed utility.

4.4.1 Spawning a Command Line Interpreter

Command line interpreters can be broken into three classes: MCR, the CLI that is active from TI: (for example, DCL), and all others.

- To pass a command line to MCR, use the MCR... task name.
- To pass a command line to the CLI that is currently active from TI:, use the CLI... task name. You can determine which CLI is active from your TI: by issuing the GCII$ directive.
- To pass a command to a specific CLI other than MCR or the CLI active from TI:, simply use that CLI's task name in your Spawn directive. The task name of DCL is ...DCL. Check with your system manager for the task names of any user-written CLIs.

4.4.2 Spawning a Utility

Utilities are generally installed under task names of the form ...tsk. You can pass commands to a utility in one of two ways. You can spawn the utility directly, using the task name ...tsk, or you can spawn MCR and pass it a command line that begins with the 3-character task name.

Whenever you spawn a task using a name of the form ...tsk, the Executive activates the task as tskTnn. (A task with its name in the form ...tsk is considered to be a prototype task. Prototype tasks cannot be run on the RSX–11M–PLUS and Micro/RSX operating systems.)

4.4.2.1 Passing Command Lines to Utilities

Even when you spawn a utility directly, pass a command line to it that is exactly as you would type it at the terminal or pass to MCR: include the 3-character task name followed by a space. This method maintains compatibility with the format used by MCR to pass commands to utilities. For more information, see the description of the GMCR$ directive in Chapter 5.

Chapter 5
Directive Descriptions

The directive descriptions consist of an explanation of the directive's function and use, the names of the corresponding macro and FORTRAN calls, the associated parameters, and the possible return values of the Directive Status Word (DSW). The descriptions generally show the $ form of the macro call (for instance, QIO$), although the $C and $S forms are often also available. Where the $S form of a macro requires less space and performs as fast as a DIR$ macro (because of a small DPB), it is recommended. For these macros, the expansion for the $S form is shown rather than that for the $ form.

In addition to the directive macros themselves, you can use the DIR$ macro to execute a directive if the directive has a predefined DPB. See Sections 1.4.1.1 and 1.4.2 for further details.

5.1 Directive Categories

For ease of reference, the directive descriptions are presented alphabetically in Section 5.3 according to the directive macro calls. This section, however, groups the directives by function. The directives are grouped into the following categories:

- Task execution control directives
- Task status control directives
- Informational directives
- Event-associated directives
- Trap-associated directives
- I/O- and intertask communications-related directives
- Memory management directives
- Parent/offspring tasking directives
- System directives
- Command line interpreter (CLI) support directives

5.1.1 Task Execution Control Directives

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

Macro	Directive Name
ABRT$	Abort Task
CSRQ$	Cancel Scheduled Initiation Requests
EXIT$S	Task Exit ($S form recommended)
EXTK$	Extend Task
RQST$	Request Task
RSUM$	Resume Task
RUN$	Run Task
SPND$S	Suspend ($S form recommended)
SWST$	Switch State

5.1.2 Task Status Control Directives

Two task status control directives alter the checkpointable attribute of a task. A third directive changes the running priority of an active task. These directives are:

Macro	Directive Name
ALTP$	Alter Priority
DSCP$S	Disable Checkpointing ($S form recommended)
ENCP$S	Enable Checkpointing ($S form recommended)

5.1.3 Informational Directives

Several directives provide the issuing task with system information and parameters, such as the time of day, the task parameters, the console switch settings, and partition or region parameters. These directives are:

Macro	Directive Name
FEAT$	Test for Specified System Feature
GDIR$	Get Default Directory
GIN$	General Information
GPRT$	Get Partition Parameters

Macro	Directive Name
GREG$	Get Region Parameters
GSSW$S	Get Sense Switches ($S form recommended)
GTIM$	Get Time Parameters
GTSK$	Get Task Parameters
TFEA$	Test for Specified Task Feature

5.1.4 Event-Associated Directives

The event and event-flag directives provide inter- and intratask synchronization and signaling and the means to set the system time. You must use these directives carefully because software faults resulting from erroneous signaling and synchronization are often obscure and difficult to isolate. The directives are:

Macro	Directive Name
CLEF$	Clear Event Flag
CMKT$	Cancel Mark Time Requests
CRGF$	Create Group Global Event Flags
DECL$S	Declare Significant Event ($S form recommended)
ELGF$	Eliminate Group Global Event Flags
EXIF$	Exit If
MRKT$	Mark Time
RDAF$	Read All Event Flags
RDXF$	Read Extended Event Flags
SETF$	Set Event Flag
STIM$	Set System Time
STLO$	Stop for Logical OR of Event Flags
STOP$S	Stop ($S form recommended)
STSE$	Stop for Single Event Flag
ULGF$S	Unlock Group Global Event Flags ($S form recommended)
USTP$	Unstop
WSIG$S	Wait for Significant Event ($S form recommended)
WTLO$	Wait for Logical OR of Event Flags
WTSE$	Wait for Single Event Flag

5.1.5 Trap-Associated Directives

The trap-associated directives provide trap facilities that allow transfer of control (software interrupts) to the executing tasks. These directives are:

Macro	Directive Name
ASTX$S	AST Service Exit ($S form recommended)
DSAR$S	Disable AST Recognition ($S form recommended)
ENAR$S	Enable AST Recognition ($S form recommended)
IHAR$S	Inhibit AST Recognition ($S form recommended)
SCAA$	Specify Command Arrival AST
SFPA$	Specify Floating Point Processor Exception AST
SPRA$	Specify Power Recovery AST
SRDA$	Specify Receive Data AST
SREA$	Specify Requested Exit AST
SREX$	Specify Requested Exit AST (extended)
SRRA$	Specify Receive-By-Reference AST
SVDB$	Specify SST Vector Table for Debugging Aid
SVTK$	Specify SST Vector Table for Task

5.1.6 I/O- and Intertask Communications-Related Directives

The I/O- and intertask communications-related directives allow tasks to access I/O devices at the driver interface level or interrupt level, to communicate with other tasks in the system, and to retrieve the MCR command line used to start the task. These directives are:

Macro	Directive Name
ALUN$	Assign LUN
CINT$	Connect to Interrupt Vector
GLUN$	Get LUN Information
GMCR$	Get MCR Command Line
QIO$	Queue I/O Request
QIOW$	Queue I/O Request and Wait
RCST$	Receive Data or Stop
RCVD$	Receive Data
RCVX$	Receive Data or Exit

Macro	Directive Name
SDAT$	Send Data
SMSG$	Send Message

5.1.7 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. These directives are:

Macro	Directive Name
ATRG$	Attach Region
CRAW$	Create Address Window
CRRG$	Create Region
DTRG$	Detach Region
ELAW$	Eliminate Address Window
GMCX$	Get Mapping Context
MAP$	Map Address Window
RREF$	Receive By Reference
RRST$	Receive By Reference or Stop
SREF$	Send By Reference
UMAP$	Unmap Address Window

5.1.8 Parent/Offspring Tasking Directives

Parent/offspring tasking directives permit tasks to start other tasks and to connect to other tasks in order to receive status information. These directives are:

Macro	Directive Name
CNCT$	Connect
EMST$	Emit Status
EXST$	Exit with Status
RPOI$	Request and Pass Offspring Information
SDRC$	Send, Request, and Connect

Macro	Directive Name
SDRP$	Send Data Request and Pass OCB
SPWN$	Spawn

5.1.9 System Directives

In addition to the directives just listed, RSX-11M-PLUS and/or Micro/RSX systems include directives that support virtual terminals, named directories, logical names, CPU/UNIBUS affinity, supervisor-mode library routines, variable-length send/receive data buffers, and parity error AST routine support. These directives are:

Macro	Directive Name
ACHN$	Assign Channel
CLON$	Create Logical Name
CPCR$	Checkpoint Common Region
CRVT$	Create Virtual Terminal
DLON$	Delete Logical Name
ELVT$	Eliminate Virtual Terminal
FSS$	File Specification Scanner
MSDS$	Map Supervisor D-Space
MVTS$	Move to/from I/D-Space
PFCS$	Parse FCS (File Control Services)
PRMS$	Parse RMS (Record Management Services)
RDEF$	Read Single Event Flag
RLON$	Recursive (iterative) Translation of Logical Name
RMAF$S	Remove Affinity ($S form only)
SCAL$S	Supervisor Call ($S form only)
SDIR$	Set Default Directory
SPEA$	Specify Parity Error AST
SNXC$	Send Next Command
STAF$	Set Affinity
TLON$	Translate Logical Name
VRCD$	Variable Receive Data
VRCS$	Variable Receive Data or Stop

Macro	Directive Name
VRCX$	Variable Receive Data or Exit
VSRC$	Variable Send, Request, and Connect
VSDA$	Variable Send Data

These functions provide for the dispatching of multiuser tasks and can enhance the interface to slave tasks.

The dispatching algorithm used by the Executive is identical to the algorithm used by MCR. Thus, the ability to dispatch copies of multiuser tasks is available at both the MCR command and Executive directive level. A consistent scheme for communication and synchronization between multiuser tasks is made available at the Executive level.

Executive-level dispatching uses the same naming scheme as is used in the RSX-11M-PLUS MCR dispatching algorithm. A single copy of the multiuser task must be installed with a name of the form ...mmm. When a task issues a directive specifying a task name of the form ...mmm, the Executive first forms the task name mmmtnn, where t is the first character of the device name of the TI: of the issuing task and nn is the unit number. The Executive then attempts to perform the directive as if the task name mmmtnn has been specified. If the directive is one that could activate the task (Request; Spawn; or Send, Request, and Connect), a TCB may be dynamically created and filled in from the ...mmm TCB. If the directive is a send user-type directive and the TCB mmmtnn does not exist, the send packet is queued to the ...mmm TCB until mmmtnn is activated. At that time, any send packets for mmmtnn that are queued to the ...mmm TCB are moved to the mmmtnn TCB.

This naming scheme allows for the specification of a specific copy of a multiuser task in a directive whose TI: is different from that of the issuing task. If the TI: of the target task is known, the task's name can be calculated and explicitly specified in a directive.

5.1.10 CLI Support Directives

The CLI support directives allow CLI tasks to get command lines, request and pass offspring information, get command line interpreter information, and set a specified CLI for a terminal. These directives are:

Macro	Directive Name
GCCI$	Get Command for Command Interpreter
GCII$	Get Command Interpreter Information
SCLI$	Set Command Line Interpreter

5.2 Directive Conventions

When using system directives, observe the following conventions:

- In MACRO-11 programs, unless a number is followed by a decimal point (.), the system assumes the number is octal.

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

- In MACRO–11 programs, task and partition names can be from one to six characters in length, and should be 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.

- Device names are two characters long and are represented by one word of ASCII code.
- Some directive descriptions state that a certain parameter must be provided even though the system ignores it. Such parameters are included to maintain compatibility between the RSX–11M, RSX–11M–PLUS, Micro/RSX, IAS, and RSX–11D operating systems.
- 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.
- Logical unit numbers (LUNs) can range from 1 through 255_{10}.
- Event flag numbers range from 1 through 96_{10}. Numbers from 1 to 32_{10} denote local flags. Numbers from 33 to 64 denote common flags. Numbers 65 to 96 denote group global event flags.

Note that the Executive preserves all task registers when a task issues a directive.

5.3 System Directive Descriptions

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

Name

This describes the function of the directive.

FORTRAN Call

This shows the FORTRAN subroutine call and defines each parameter.

Macro Call

This shows the macro call, defines each parameter, and gives the defaults for optional parameters in parentheses following the definition of the parameter. Since zero is supplied for most defaulted parameters, only nonzero default values are shown. Parameters ignored by RSX–11M, RSX–11M–PLUS, and Micro/RSX systems are required for compatibility with IAS and RSX–11D systems.

Macro Expansion

Most of the directive descriptions expand the $ form of the macro. Where the $S form is recommended for a directive, the expansion for that form is shown instead. Section 1.4.5 illustrates expansions for all three forms and for the DIR$ macro.

Definition Block Parameters

Only the memory management directive descriptions include these parameters. This section describes all the relevant input and output parameters in the Region or Window Definition Block (see Section 3.5).

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 corresponding DPB element. This section lists these symbols. The length in bytes of the element pointed to by the symbol appears in parentheses following the symbol's description. Thus,

A.BTTN — Task name (4)

defines A.BTTN as pointing to a task name in the Abort Task DPB. The task name has a length of four bytes.

DSW Return Code

This section lists valid return codes for the directive. For more information, see Appendix B, which lists the standard directive error codes.

Notes

The notes presented with some directive descriptions expand on the function, use, and/or consequences of using the directives. Always read the notes carefully.

ABRT$

5.4 Abort Task

The Abort Task directive instructs the system to terminate the execution of the indicated task. ABRT$ is intended for use as an emergency or fault exit. ABRT$ displays a termination notification based on the described condition, at one of the following terminals:

- The terminal from which the aborted task was requested
- The originating terminal of the task that requested the aborted task
- The operator's console (CO:) if the task was started internally from another task by a Run directive, or by an MCR or DCL RUN command that specified one or more time parameters

On systems without multiuser protection, a task may abort any task, including itself. When a task is aborted, its state changes from active to dormant. Therefore, to reactivate an aborted task, a task or an operator must request it.

On systems that support multiuser protection, a task must be privileged to issue the Abort Task directive (unless it is aborting a task with the same TI:).

FORTRAN Call

CALL ABORT (tsk[,ids])

Parameters

tsk

Name (Radix–50) of the task to be aborted

ids

Directive status

Macro Call

ABRT$ tsk

Parameter

tsk

Name (Radix–50) of the task to be aborted

Macro Expansion

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

Local Symbol Definition

A.BTTN Task name (4)

DSW Return Codes

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

Notes

1. When a task is aborted, the Executive frees all the task's resources. In particular, the Executive:
   - Detaches all attached devices
   - Flushes the AST queue and despecifies all specified ASTs
   - Flushes the receive and receive-by-reference queue
   - Flushes the clock queue for outstanding Mark Time requests for the task
   - Closes all open files (files open for write access are locked)
   - Detaches all attached regions, except in the case of a fixed task
   - Runs down the task's I/O
   - Deaccesses the group global event flags for the task's group
   - Disconnects from interrupts
   - Flushes all outstanding CLI command buffers for the task
   - Breaks the connection with any offspring tasks
   - Returns a severe error status (EX$SEV) to the parent task when a connected task is aborted
   - Marks virtual terminals created by the aborted task for deallocation; the virtual terminals actually become deallocated when all tasks using the virtual terminal or terminals are aborted or exit; nonprivileged tasks using virtual terminal units that are marked for deallocation as TI: are also aborted
   - Frees the task's memory if the aborted task was not fixed
2. If the aborted task had a requested exit AST specified, the task will receive that AST instead of being aborted. No indication that this has occurred is returned to the task that issued the abort request.
3. When the aborted task actually exits, the Executive declares a significant event.

ACHN$

5.5 Assign Channel

The Assign Channel directive performs all of the processing of the file specification required to find the actual device name and then assigns the LUN to that device. This processing involves expanding the file specification and using the final device specification to assign the LUN.

FORTRAN Call

CALL ACHN ([mod],[itbmsk],lun,fsbuf,fssz[,idsw])

Parameters

mod

Optional modifier to be matched against the logical name within a table. Ordinarily, no value will be specified to allow any logical name in table to be found.

itbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

fsbuf

Array containing the file specification buffer

fssz

Size (in bytes) of the file specification buffer

idsw

Integer to receive the Directive Status Word

Macro Call

ACHN$ [mod],[tbmsk],lun,fsbuf,fssz

Parameters

mod

Optional modifier to be matched against the logical name within a table. Ordinarily, no value will be specified to allow any logical name in table to be found.

tbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

fsbuf

Address of file specification buffer

fssz

Size (in bytes) of the file specification buffer

Macro Expansion

```
ACHN$ MOD,TBMSK,LUN,FSBUF,FSSZ
.BYTE     207.,5          ;ACHN$ MACRO DIC, DPB SIZE = 5 WORDS
.BYTE     6               ;ACHN$ SUBFUNCTION
.BYTE     MOD             ;MODIFIER
.BYTE     LUN             ;LUN TO BE ASSIGNED
.BYTE     TBMSK           ;TABLE MASK
.WORD     FSBUF           ;ADDRESS OF FILE SPECIFICATION BUFFER
.WORD     FSSZ            ;LENGTH OF FILE SPECIFICATION
```

Local Symbol Definitions

A.LFUN	Subfunction value (1)
A.LMOD	Logical name modifier (1)
A.LLUN	LUN number (1)
A.LTBL	Table inhibit mask (1)
A.LSBF	Address of file specification buffer (2)
A.LSSZ	Size (in bytes) of the file specification buffer (2)

DSW Return Codes

IS.SUC	Successful completion.
IE.IDU	Invalid device or unit.
IE.ILU	Invalid LUN.
IE.LNF	Logical translation failed.
IE.LNL	LUN in use.

ACHN$

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.

IE.SDP DIC or DPB size is invalid.

Notes

1. A return code of IE.LNL indicates that the specified LUN cannot be assigned as directed. Either the LUN is already assigned to a device with a file open for that LUN or the LUN is currently assigned to a device attached to the task, and the directive attempted to change the LUN assignment. If a task has a LUN assigned to a device and the task has attached the device, the LUN can be reassigned, provided that the task has another LUN assigned to the same device.
2. Physical I/O (output) operations cannot be executed with spooled devices. Output should be performed using the File Control Services (FCS).

5.6 Alter Priority

The Alter Priority directive instructs the system to change the running priority of a specified active task to either a new priority indicated in the directive call or to the task's default (installed) priority if the call does not specify a new priority.

The specified task must be installed and active. The Executive resets the task's priority to its installed priority when the task exits.

If the directive call omits a task name, the Executive defaults to the issuing task.

The Executive reorders any outstanding I/O requests for the task in the I/O queue and reallocates the task's partition. The partition reallocation may cause the task to be checkpointed.

On systems that support multiuser protection, a nonprivileged task can issue ALTP$ only for itself, and only for a priority equal to or lower than its installed priority. A privileged task can change the priority of any task to any value less than 250_{10}.

FORTRAN Call

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

Parameters

tsk

Active task name

ipri

A 1-word integer value equal to the new priority, a number from 1 to 250_{10}

ids

Directive status

Macro Call

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

Parameters

tsk

Active task name

pri

New priority, a number from 1 to 250_{10}

ALTP$

Macro Expansion

```
ALTP$    ALPHA,75.
.BYTE    9.,4              ;ALTP$ MACRO DIC, DPB SIZE = 4 WORDS
.RAD50   /ALPHA/           ;TASK ALPHA
.WORD    75.               ;NEW PRIORITY
```

Local Symbol Definitions

A.LTTN Task name (4)

A.LTPR Priority (2)

DSW Return Codes

IS.SUC Successful completion.

IE.INS Task not installed.

IE.ACT Task not active.

IE.PRI Issuing task is not privileged (multiuser protection systems only).

IE.IPR Invalid priority.

IE.RSU Resource (the task's header) unavailable because task is checkpointed with outstanding I/O.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.7 Assign LUN

The Assign LUN directive instructs the system to assign a physical device unit to a logical unit number (LUN). It does not indicate that the task has attached itself to the device.

The actual physical device assigned to the logical unit is dependent on the logical assignment table (see the description of the ASSIGN command in the *RSX-11M-PLUS MCR Operations Manual*, the *RSX-11M-PLUS Command Language Manual*, or the *Micro/RSX User's Guide*). The Executive first searches the logical assignment table for a device name match. If it finds a match, the Executive assigns the physical device unit associated with the matching entry to the logical unit. Otherwise, the Executive searches the physical device tables and assigns the actual physical device unit named to the logical unit. In systems that support multiuser protection, the Executive does not search the logical assignment table if the task has been installed with the slave option.

When a task reassigns a LUN from one device to another, the Executive cancels all I/O requests for the issuing task in the previous device queue.

FORTRAN Call

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

Parameters

lun
Logical unit number

dev
Device name (format: 1A2)

iunt
Device unit number

ids
Directive status

Macro Call

ALUN$ lun,dev,unt

Parameters

lun
Logical unit number

dev
Device name (two uppercase characters)

unt
Device unit number

ALUN$

Macro Expansion

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

Local Symbol Definitions

A.LULU Logical unit number (2)

A.LUNA Physical device name (2)

A.LUNU Physical device unit number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.LNL LUN use is interlocked (see Note 1).

IE.IDU Invalid device and/or unit.

IE.ILU Invalid logical unit number.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. A return code of IE.LNL indicates that the specified LUN cannot be assigned as directed. Either the LUN is already assigned to a device with a file open for that LUN or the LUN is currently assigned to a device attached to the task, and the directive attempted to change the LUN assignment. If a task has a LUN assigned to a device and the task has attached the device, the LUN can be reassigned, provided that the task has another LUN assigned to the same device.
2. Physical I/O (output) operations cannot be executed with spooled devices. Output should be performed using the File Control Services (FCS).

5.8 AST Service Exit ($S Form Recommended)

The AST Service Exit directive instructs the system to terminate execution of an AST service routine.

If another AST is queued and ASTs are not disabled, then the Executive immediately effects the next AST. Otherwise, the Executive restores the task's pre-AST state. See the Notes.

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.

Macro Call

ASTX$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

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

Notes

1. A return to the AST service routine occurs only if the directive is rejected. Therefore, no Branch on Carry Clear instruction is generated if an error-routine address is given. (The return occurs only when the Carry bit is set.)

2. When an AST occurs, the Executive pushes, at minimum, the following information onto the task's stack:

   SP+06 Event flag mask word

   SP+04 PS of task prior to AST

   SP+02 PC of task prior to AST

   SP+00 DSW of task prior to AST

   The task stack must be in this state when the AST Service Exit directive is executed.

   In addition to the data parameters, the Executive pushes supplemental information onto the task stack for certain ASTs. For I/O completion, the stack contains the address of the I/O status block; for Mark Time, the stack contains the Event Flag Number; for a floating-point-processor exception, the stack contains the exception code and address.

   These AST parameters must be removed from the task's stack prior to issuing an AST exit directive. The following example shows how to remove AST parameters when a task uses an AST routine on I/O completion.

Example

```
;
; EXAMPLE PROGRAM
;
; LOCAL DATA
;
IOSB:    .BLKW   2               ;I/O STATUS DOUBLEWORD
BUFFER:  .BLKW   30.             ;I/O BUFFER

;
; START OF MAIN PROGRAM
;
START:   .                       ;PROCESS DATA
         .
         .
         QIOW$C   IO.WVB,2,1,,IOSB,ASTSER,<BUFFER,60.,40>
         .
         .                       ;PROCESS AND WAIT
         .
         EXIT$S                  ;EXIT TO EXECUTIVE

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

3. The task can alter its return address by manipulating the information on its stack prior to executing an AST exit directive. For example, to return to task state at an address other than the pre-AST address indicated on the stack, the task can simply replace the PC word on the stack. This procedure may be useful in those cases in which error conditions are discovered in the AST routine, but you should use extreme caution when doing this alteration since AST service routine problems are difficult to isolate.
4. Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as the DIR$ macro.

ATRG$

5.9 Attach Region

The Attach Region directive attaches the issuing task to a static common region or to a named dynamic region. (No other type of region can be attached 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, the Executive grants the desired access. 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 the 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 want 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 time.

FORTRAN Call

CALL ATRG (irdb[,ids])

Parameters

irdb

An 8-word integer array containing a Region Definition Block (see Section 3.5.1.2)

ids

Directive status

Macro Call

ATRG$ rdb

Parameter

rdb

Region Definition Block address

Macro Expansion

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

Region Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
irdb(3)(4)	R.GNAM	Name of the region to be attached
irdb(7)	R.GSTS	Bit settings[1] in the region status word (specifying desired access to the region):

Bit	Definition
RS.RED	1 if read access is desired
RS.WRT	1 if write access is desired
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.

[1] If you are a FORTRAN programmer, refer to Section 3.5.1 to determine the bit values represented by the symbolic names described.

Output parameters:

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

Local Symbol Definition

A.TRBA Region definition block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN An attachment descriptor cannot be allocated.

IE.PRI Privilege violation.

IE.NVR Invalid region ID.

IE.PNS Specified region name does not exist. (The specified region is a main partition.)

IE.HWR Region had parity error or load failure.

IE.ADP Part of the DPB or RDB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

CINT$

5.10 Connect to Interrupt Vector

The Connect to Interrupt Vector directive enables a task to process hardware interrupts through a specified vector. The Interrupt Service Routine (ISR) is included in the task's own space. In a mapped system, the issuing task must be privileged.

The overhead entails the execution of about 10 instructions before entry into the ISR and 10 instructions after exit from the ISR. The Executive provides a mechanism for transfer of control from the ISR to task-level code, using either an AST or a local event flag.

After a task has connected to an interrupt vector, it can process interrupts on three different levels: interrupt, fork, and task. The task level may be subdivided into AST level and non-AST level. The task levels are as follows:

- Interrupt Level

  When an interrupt occurs, control is transferred, with the Interrupt Transfer Block (ITB) that has been allocated by the CINT$ directive, to the Executive subroutine $INTSC. From there, control goes to the ISR specified in the directive.

  The ISR processes the interrupt and either dismisses the interrupt directly or enters fork level through a call to the Executive routine $FORK2.

- Fork Level

  The fork-level routine executes at priority 0, the lowest processor priority, allowing interrupts and more time-dependent tasks to be serviced promptly. If required, the fork routine sets a local event flag for the task and/or queues an AST to an AST routine specified in the directive.

- Task Level

  At task level, entered as the result of a local event flag or an AST, the task does final interrupt processing and has access to Executive directives.

Typically, the ISR does the minimal processing required for an interrupt and stores information for the fork routine or task-level routine in a ring buffer. The fork routine is entered after a number of interrupts have occurred as deemed necessary by the ISR and further condenses the information. Finally, the fork routine wakes up the task-level code for ultimate processing that requires access to Executive directives. The fork level may, however, be a transient stage from ISR to task-level code without doing any processing.

In a mapped system, a task must be built privileged to use the CINT$ directive. However, it is legal to use the /PR:0 switch to the Task Builder to have "unprivileged mapping," that is, up to 32K words of virtual address space available. This precludes use of the Executive subroutines from task-level code; however, the ISR and fork-level routines are always mapped to the Executive when they are executed. In any case, the Executive symbol table file (RSX11M.STB) should be included as input to the Task Builder.

However, be aware that including the symbol definition (table) file can cause references to system subroutines to be resolved from that file instead of from the system library. To avoid this problem, explicitly include the required library modules before specifying the RSX11M.STB file. Specifying the /SS switch with the file causes the Task Builder to resolve any symbols that are still undefined. (Specifying the /SS switch is necessary because it prevents the Task Builder from trying to use multiply defined symbols.)

As will be described later, in a mapped system special considerations apply to the mapping of the ISR, fork routine, and enable/disable routine as well as all task data buffers accessed by these routines.

FORTRAN Call

Not supported

Macro Call

CINT$ vec,base,isr,edir,pri,ast

Parameters

vec

Interrupt vector address; must be in the range 60_8 through highest vector specified during system generation, and must be a multiple of 4

base

Virtual base address for kernel APR 5 mapping of the ISR and enable/disable interrupt routines. This address is automatically truncated to a 32_{10}-word boundary. The "base" argument is ignored in an unmapped system.

isr

Virtual address of the ISR or 0 to disconnect from the interrupt vector

edir

Virtual address of the enable/disable interrupt routine

pri

Initial priority at which the ISR is to execute. This is normally equal to the hard-wired interrupt priority and is expressed in the form n*40, where n is a number in the range 0-7. This form puts the value in bits 5-7 of pri. It is recommended that you make use of the symbols PR4, PR5, PR6, and PR7 for this purpose. These are implemented by means of the macro HWDDF$ found in the file [1,1]EXEMC.MLB. Also, you should take care to specify the correct value for this parameter. An incorrect initial priority (for example, specifying PR4 for a device that interrupts at PR5) may result in a system crash.

ast

Virtual address of an AST routine to be entered after the fork-level routine queues an AST

To disconnect from interrupts on a vector, the argument isr is set to 0 and the arguments base, edir, psw, and ast are ignored.

CINT$

Macro Expansion

```
CINT$   420,BADR,TADR,EDADR,PR5,ASTADR
.BYTE   129.,7          ;CINT$ MACRO DIC, DPB SIZE = 7 WORDS
.WORD   420             ;INTERRUPT VECTOR ADDRESS = 420
.WORD   BADR            ;VIRTUAL BASE ADDRESS FOR KERNEL APR
.WORD   IADR            ;VIRTUAL ADDRESS OF THE INTERRUPT SERVICE ROUTINE
.WORD   EDADR           ;VIRTUAL ADDRESS OF THE INTERRUPT ENABLE/DISABLE ROUTINE
.BYTE   PR5,0           ;INITIAL INTERRUPT SERVICE ROUTINE PRIORITY (LOW BYTE).
                        ;(HIGH BYTE = 0.)
.WORD   ASTADR          ;VIRTUAL ADDRESS OF AST ROUTINE
```

Local Symbol Definitions

C.INVE Vector address (2)

C.INBA Base address (2)

C.INIS ISR address (2)

C.INDI Enable/disable interrupt routine address (2)

C.INPS Priority (1)

C.INAS AST address (2)

DSW Return Codes

IE.UPN An ITB could not be allocated (no pool space).

IE.ITS The function requested is "disconnect" and the task is not the owner of the vector.

IE.PRI Issuing task is not privileged (not applicable in unmapped system).

IE.RSU The specified vector is already in use.

IE.ILV The specified vector is illegal (lower than 60 or higher than highest vector specified during system generation, or not a multiple of 4).

IE.MAP ISR or enable/disable interrupt routine is not within 4K words from the value (base address and 177700).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. Checkpointable Tasks

   The following points should be noted only for checkpointable tasks:

   - When a task connects to an interrupt vector, checkpointing of the task is automatically disabled.
   - When a task disconnects from a vector and is not connected to any other vector, checkpointing of the task is automatically enabled, regardless of its state before the first connect or any change in state while the task was connected.

2. Mapping Considerations

   In an unmapped system, the argument "base" is ignored and the arguments "isr," "edir," and "ast" are physical addresses.

   In a mapped system, the argument "base," after being truncated to a 32_{10}-word boundary, is the start of a 4K-word area mapped in kernel APR 5. All code and data in the task that are used by the routines must fall within that area or a fatal error will occur, probably resulting in a system crash.

   Furthermore, the code and data must be either position independent (refer to the *PDP-11 MACRO-11 Language Reference Manual* for more information on position-independent code) or coded in such a way that the code can execute in APR 5 mapping. When the routines execute, the processor is in kernel mode and the virtual address space includes all of the Executive, the pool, and the I/O page.

   References within the task image must be PC-relative or use a special offset defined below. References outside the task image must be absolute.

   The following solutions are possible:

   - Write the ISR, enable/disable interrupt routines, and data in position-independent code.
   - Include the code and data in a common partition, task build it with absolute addresses in APR 5 (PAR=ISR:120000:20000), and link the task to the common partition.
   - Build the task privileged with APR 5 mapping and use the constant 120000 as argument "base" in the CINT$ directive.
   - When accessing locations within the task image in immediate or absolute addressing mode, use the following offset:

     <120000- <base and 177700> >

     (In immediate mode, only relocatable addresses need to use this offset.)

3. ISR

   When the ISR is entered, R5 points to the fork block in the Interrupt Transfer Block (ITB), and R4 is saved and free to be used. Registers R0 through R3 must be saved and restored if used. If one ISR services multiple vectors, the interrupting vector can be identified by the vector address, which is stored at offset X.VEC in the ITB. The following example loads the vector address into R4:

```
MOV X.VEC-X.FORK(R5),R4
```

   The ISR either dismisses the interrupt directly by an RTS PC instruction or calls $FORK2 if the fork routine is to be entered. When calling $FORK2, R5 must point to the fork block in the ITB and the stack must be in the same state as it was upon entry to the ISR. Note that the call must use absolute addressing: CALL @#$FORK2.

Note

Do not put the ISR in a common. Commons can be checkpointed or shuffled independently from the task and the Executive disables checkpointing and shuffling for the task region only.

4. Fork-Level Routine

   The fork-level routine starts immediately after the call to $FORK2. On entry, R4 and R5 are the same as when $FORK2 was called. All registers are free to be used. The first instruction of the fork routine must be CLR @R3, which declares the fork block free.

   The fork-level routine should be entered if servicing the interrupt takes more than 500 microseconds. It must be entered if an AST is to be queued or an event flag is to be set. (Fork level is discussed in greater detail in the *RSX-11M-PLUS and Micro/RSX Guide to Writing an I/O Driver* manuals.)

   An AST is queued by calling the subroutine $QASTC.

   Input: R5 Pointer to fork block in the ITB

   Output: If AST successfully queued, Carry bit = 0
   If AST was not specified by CINT$, Carry bit = 1

   Registers altered: R0, R1, R2, and R3

   An event flag is set by calling the subroutine $SETF.

   Input: R0 Event flag number

   R5 Task Control Block (TCB) address of task for which flag is to be set. This is usually, but not necessarily, the task that has connected to the vector. This task's TCB address is found at offset X.TCB in the ITB.

   Output: Specified event flag set

   Registers altered: R1 and R2

   Note that absolute addressing must be used when calling these routines (and any other Executive subroutines) from fork level, as follows:

```
CALL @#$QASTC
CALL @#$SETF
```

5. Enable/Disable Interrupt Routine

   The purpose of the enable/disable interrupt routine, whose address is included in the directive call, is to allow you to have a routine automatically called in the following cases:

   a. When the directive is successfully executed to connect to an interrupt vector (argument isr $\neq$ 0). The routine is called immediately before return to the task.

   b. When the directive is successfully executed to disconnect from an interrupt vector (argument isr=0).

   c. When the task is aborted or exits with interrupt vectors still connected.

   In case a, the routine is called with the Carry bit cleared; in cases b and c, with the Carry bit set. In all three cases, R1 is a pointer to the Interrupt Transfer Block (ITB). Registers R0, R2, and R3 are free to be used; other registers must be returned unmodified. Return is accomplished by means of an RTS PC instruction.

Typically, the routine dispatches to one of two routines, depending on whether the Carry bit is cleared or set. One routine sets interrupt enable and performs any other necessary initialization; the other clears interrupt enable and cleans up.

Note that the ITB contains the vector address, in the event that common code is used for multiple vectors.

6. AST Routine

   The fork routine may queue an AST for the task through a call to the Executive routine $QASTC as described above. When the AST routine is entered (at task level), the top word of the stack contains the vector address and must be popped off the stack before AST exit (ASTX$S).

7. ITB Structure

   The following offsets are defined relative to the start of the ITB:

X.LNK	Link word
X.JSR	Subroutine call to $INTSC
X.PSW	PSW for ISR (low-order byte)
X.ISR	ISR address (relocated)
X.FORK	Start of fork block
X.REL	APR 5 relocation (mapped systems only)
X.DSI	Address of enable/disable interrupt routine (relocated)
X.TCB	TCB address of owning task
X.AST	Start of AST block
X.VEC	Vector address
X.VPC	Saved PC from vector
X.LEN	Length in bytes of ITB

   The symbols X.LNK through X.TCB are defined locally by the macro ITBDF$, which is included in the file [1,1]EXEMC.MLB. All global symbols are defined globally by the file [1,54]RSX11M.STB.

Example

The following programming example illustrates the use of the CINT$ directive:

```
  .TITLE PUNTSK PUNCH ASCII TEXT ON PAPER TAPE PUNCH
 ;++
 ; THIS TASK WILL PUNCH AN ASCII STRING TO THE PAPER TAPE PUNCH
 ; USING THE CINT$ DIRECTIVE.
```

CINT$

```
;
; IT MUST BE BUILT USING THE /PR:0 TASK BUILDER SWITCH.
; NOTE THAT THIS METHOD ALLOWS A TASK TO BE A FULL 32K
; WORDS LONG.  IF IT IS NECESSARY TO ACCESS THE I/O PAGE
; IN OTHER THAN THE ENABLE/DISABLE ROUTINE OR THE ISR
; THE TASK MUST BE LINKED TO A COMMON BLOCK COVERING
; THE CORRECT PART OF THE I/O PAGE.
;

;
; TASK BUILD COMMAND FILE:
;
; PUNTSK/MM/PR:0/-FP,PUNTSK/-SP/MA=PUNTSK
; [1,54]RSX11M.STB/SS
; /
; GBLDEF=$VECTR:74
; GBLDEF=$DVCSR:177554
; UNITS=1
; ASG=TI:1
; PAR=GEN:0:40000
;

;
; IT IS POSSIBLE TO HAVE THIS TASK TYPE ON THE CONSOLE TERMINAL
; IF THERE IS NO PAPER TAPE PUNCH AVAILABLE.  TO DO THIS THE
; VECTOR FOR THE CONSOLE OUTPUT MUST APPEAR TO BE UNUSED.  THIS
; MAY BE DONE BY (ON A TERMINAL OTHER THAN THE CONSOLE!) OPENING
; THE VECTOR LOCATION (64) AND REPLACING ITS CONTENTS WITH
; THE VALUE OF '$NSO' AS OBTAINED FROM A MAP OF THE SYSTEM. BE
; SURE TO REMEMBER THE OLD VALUE OR YOUR CONSOLE WILL BE DEAD
; UNTIL YOU REBOOT THE SYSTEM.  NOW TASK BUILD USING THE FOLLOWING
; COMMAND FILE:
;
; PUNTTY/MM/PR:0,/-FP,PUNTTY/-SP/MA=PUNTSK
; [1,54]RSX11M.STB/SS
; /
; GBLDEF=$VECTR:64
; GBLDEF=$DVCSR:177564
; UNITS=1
; ASG=TI:1
; PAR=GEN:0:40000
;

;
; NOTE THAT IN THE ABOVE TWO TKB COMMAND FILES THE FOLLOWING
; CHANGES MUST BE MADE IN ORDER TO RUN ON AN UNMAPPED SYSTEM:
;
; 1) /MM SHOULD BE CHANGED TO /-MM
; 2) 'PAR=GEN:0:40000' SHOULD BE CHANGED TO
;         'PAR=GEN:40000:40000'
;
; IN ADDITION, PLACE A SEMICOLON IN FRONT OF THE SOURCE LINE
; BELOW THAT DEFINES THE SYMBOL 'M$$MGE'.
;--
 .MCALL CINT$, QIOW$, CLEF$S, WTSE$S, EXIT$S, DIR$
;
; LOCAL SYMBOLS
;
```

```
LUN.TT      =       1       ;LUN FOR TERMINAL I/O
EFN.TT      =       1       ;EFN FOR TERMINAL I/O
EFN.WF      =       2       ;EFN TO WAIT FOR PUNCHING TO COMPLETE
M$$MGE      =       0       ;DEFINE THIS SYMBOL TO RUN ON MAPPED SYSTEM

;++
;       MACRO TO GENERATE AN ASCII STRING AND A QIO TO OUTPUT
;       THE STRING TO THE TERMINAL.
;
;       MESSG   NAM,STRING
;
;       WHERE:
;
;       NAM     IS THE NAME OF THE GENERATED QIO DPB
;       STRING  IS THE ASCII STRING TO OUTPUT
;
;--

        .MACRO  MESSG   NAM,STRING,?LBL
        $CHR=0
        .IRPC   X,<STRING>
        $CHR=$CHR+1
        .ENDR
        .ENABL  LSB
LBL:    .ASCII  /(STRING/
        .EVEN
NAM:    QIOW$   IO.WVB,LUN.TT,EFN.TT,,,,<LBL,$CHR,40>
        .DSABL  LSB
        .ENDM

        MESSG   HELLO,<CONNECT TO INTERRUPT TEST>
        MESSG   CINWRK,<CONNECT TO INTERRUPT WORKS--CHECK THE PAPER TAPE PUNCH>

CINT:   CINT$   $VECTR,$BASE,$PNISR,$PNEDI,PR4
                                ;CONNECT TO INTERRUPT
                                ;       VECTOR=$VECTR
                                ;       BASE.FOR.MAPPING=$BASE
                                ;       ISR=$PNISR
                                ;       ENB.DSABL.RTN=$PNEDI
                                ;       PRIO=PR4
DISCON: CINT$   $VECTR,0,0      ;DISCONNECT FROM INTERRUPT
                                ;       VECTOR=74

;++
;    ENTRY POINT TO THE PUNCH TASK.  THE TASK WILL ANNOUNCE
;    ITSELF ON THE INITIATING TERMINAL, CONNECT TO THE
;    SPECIFIED VECTOR, OUTPUT THE ASCII STRING, AND THEN
;    OUTPUT A MESSAGE THAT IT WAS SUCCESSFUL.  IF THE TASK
;    TERMINATES WITH AN I/O TRAP THE CONNECT-TO-INTERRUPT
;    DIRECTIVE FAILED, AND R1 WILL CONTAIN THE DSW RETURNED
;    IN ORDER TO DIAGNOSE THE ERROR.
;--
```

CINT$

```
$PUNTK::DIR$     #HELLO     ;ANNOUNCE THAT WE ARE HERE
         DIR$     #CINT      ;CONNECT TO THE PUNCH
                             ;       THIS CAN BE EITHER THE TERMINAL
                             ;       OR THE PAPER TAPE PUNCH.
         BCS      ERR1       ;IF CS THEN DIRECTIVE ERROR
         WTSE$S   #EFN.WF    ;WAIT FOR PUNCH TO FINISH
         DIR$     #DISCON    ;DISCONNECT FROM INTERRUPTS
         DIR$     #CINWRK    ;TELL USER THAT CINT WORKS
         EXIT$S

ERR1:    MOV      #1,R0      ;ERROR # 1
         MOV      $DSW,R1    ;GET THE DSW TO SHOW THE CINT ERROR RETURN
         IOT                 ;DUMP REGISTERS

BASE:                        ;THIS IS THE BASE OF THE MAPPING USED
                             ;BY THE EXECUTIVE WHEN MAPPING TO THE
                             ;'DRIVER.' THIS MAPPING IS REQUIRED
                             ;ONLY ON MAPPED SYSTEMS; UNMAPPED
                             ;SYSTEMS DO NOT HAVE THIS PROBLEM.

;++
;       FOLLOWING IS THE ASCII STRING PUNCHED BY THIS TASK.
;--

         .NLIST  BEX
PUNMSG:  .ASCIZ  /ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^&*()_+-=/<15><12>
         .LIST   BEX
         .EVEN

PUNPTR:  .WORD   0           ;POINTER INTO PUNMSG FOR ISR
TSKTCB:  .WORD   0           ;TCB ADDRESS OF TASK
PUNCSR:  .WORD   $DVCSR      ;PAPER TAPE PUNCH CSR ADDRESS
PUNBUF:  .WORD   $DVCSR+2    ;PAPER TAPE PUNCH BUFFER ADDRESS

;++
;    ENABLE/DISABLE ROUTINE.
;
;    THIS ROUTINE IS CALLED BY THE EXEC ON EITHER A CONNECT OR DISCONNECT
;    FROM INTERRUPT VECTOR REQUEST, OR WHEN THE TASK EXITS WITH INTERRUPT
;    VECTORS STILL CONNECTED.
;
;    ENTRY CONDITIONS:
;
;    C-CLEAR        THIS IS A SUCCESSFUL CONNECT.
;    C-SET          THIS IS A DISCONNECT.
;
;    $TKTCB         THE TCB ADDRESS OF THE CURRENTLY EXECUTING TASK (ME).
;
;    ACTION:
;
;    IF THE C-BIT IS SET WE MERELY DISABLE THE PUNCH AND RETURN.  IF
;    THE C-BIT IS CLEAR WE WILL ENABLE THE PUNCH TO INTERRUPT.  THIS
;    WILL IMMEDIATELY CAUSE AN INTERRUPT AND THE INTERRUPT SERVICE
;    ROUTINE WILL OUTPUT CHARACTERS TO THE PUNCH (ONE PER
```

```
;    INTERRUPT) UNTIL A ZERO BYTE IS OUTPUT.  THE ISR WILL THEN FORK
;    AND SET THE LOCAL EVENT FLAG 'EFN.WF.' THIS WILL THEN CAUSE THE
;    TASK PORTION OF THIS TASK TO CONTINUE EXECUTING AND EVENTUALLY
;    EXIT.
;
;--

$PNEDI::BCS     20$         ;IF CS THEN DISCONNECT
       MOV      @#$TKTCB,TSKTCB ;COPY TASK TCB ADDRESS FOR LATER
                            ;SO WE CAN SET EFN.

       .IFDF    M$$MGE      ;MAPPED SYSTEM?

       MOV      #PUNMSG+120000-<$BASE&177700>,PUNPTR ;RELOCATE ADDRESS
                            ;TO APR 5 MAPPING, AND SET UP
                            ;BUFFER POINTER

       .IFF     M$$MGE      ;UNMAPPED SYSTEM?

       MOV      #PUNMSG,PUNPTR ;SET UP BUFFER POINTER

       .ENDC

       BIS      #100,@PUNCSR ;ALLOW INTERRUPTS
       RETURN
                            ;WHEN WE ARE DONE PUNCHING
20$:   BIC      #100,@PUNCSR ;DISABLE INTERRUPTS
       RETURN

;++
;       INTERRUPT SERVICE ROUTINE
;
;       THIS IS THE 'BARE-BONES' INTERRUPT SERVICE ROUTINE.  THERE IS NO
;       ERROR CHECKING.  THIS ROUTINE MERELY OUTPUTS THE NEXT CHARACTER
;       IN THE STRING.  WHEN IT ENCOUNTERS THE ZERO BYTE AT THE END, IT
;       WILL CALL $FORK2.  THIS CREATES A SYSTEM PROCESS AND WE THEN
;       SET THE LOCAL EVENT FLAG 'EFN.WF' TO WAKE UP THE TASK PART OF
;       THIS TASK.
;
;       INPUTS:
;
;       R5        POINTS TO FORK BLOCK IN THE INTERRUPT TRANSFER BLOCK.
;       R4        IS FREE TO USE.
;
;--

$PNISR::MOVB    @PUNPTR,R4              ;GET THE NEXT CHARACTER IN THE BUFFER
       BEQ      20$                     ;IF EQ THEN END OF STRING
       MOVB     R4,@PUNBUF              ;PUNCH THE CHARACTER
       INC      PUNPTR                  ;MOVE THE POINTER
       RETURN                           ;RETURN TO INTERRUPT EXIT CODE

;
; WE HAVE FINISHED PUNCHING THE STRING.  DISABLE INTERRUPTS, FORK, AND
; SET THE LOCAL EVENT FLAG.
;
```

```
20$:    BIC     #100,@PUNCSR            ;DISABLE FURTHER INTERRUPTS
        CALL    @#$FORK2                ;CREATE SYSTEM PROCESS
        CLR     (R3)                    ;DECLARE THE FORK BLOCK FREE

;
; IF IT IS DESIRABLE TO QUEUE AN AST FOR THE TASK, THERE ARE TWO
; THINGS THAT MUST BE DONE:
;
;       1) AN AST ADDRESS MUST HAVE BEEN SPECIFIED IN THE CINT$
;          DIRECTIVE (THERE WAS NONE IN THIS CASE).
;
;       2) THE FOLLOWING CODE MUST BE EXECUTED:
;
;            NOTE - R5 POINTS TO THE FORK BLOCK WITHIN THE
;                   INTERRUPT TRANSFER BLOCK (THIS IS SET
;                   UP UPON RECEIPT OF THE INTERRUPT)

;       CALL    @#$QASTC                ;QUEUE AN AST FOR THE TASK
;
;
;       IT IS POSSIBLE TO QUEUE AN AST AND SET AN EVENT FLAG.
;       HOWEVER, THIS TASK IS ONLY USING EVENT FLAGS, SO NOW
;       WE WILL SET THE EVENT FLAG.
;
        MOV     #EFN.WF,R0              ;GET EFN NUMBER TO SET
        MOV     TSKTCB,R5               ;GET TASK TCB ADDRESS FOR $SETF
        CALL    @#$SETF                 ;SET THE LOCAL EVENT FLAG TO AWAKE TASK
        RETURN                          ;EXIT

        .END    $PUNTK
```

5.11 Clear Event Flag

The Clear Event Flag directive instructs the system to report an indicated event flag's polarity and then clear the flag.

FORTRAN Call

CALL CLREF (efn[,ids])

Parameters

efn

Event flag number

ids

Directive status

Macro Call

CLEF$ efn

Parameter

efn

Event flag number

Macro Expansion

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

Local Symbol Definition

C.LEEF Event flag number (2)

DSW Return Codes

IS.CLR Successful completion; flag was already clear.

IS.SET Successful completion; flag was set.

IE.IEF Invalid event flag number (EFN <1, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

CLON$, CLOG$

5.12 Create Logical Name

The Create Logical Name directive establishes the relationship between a logical name string and an equivalence name string. The maximum length for each string is 255_{10} characters. If you create a logical name string with the same name, the new definition supersedes the old one.

The CRELON and CLON$ calls are the preferred calls to use on RSX-11M-PLUS and Micro/RSX operating systems. The CRELOG and CLOG$ calls are provided for compatibility with the P/OS operating system. See the Notes.

FORTRAN Calls

CALL CRELON ([mod],itbnum,lns,lnssz,iens,ienssz[,idsw])
CALL CRELOG ([mod],itbnum,lns,lnssz,iens,ienssz[,idsw])

Parameters

mod

Modifier of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB; if specified, nonzero values must correspond to the valid symbolic references used by the system (see the Notes)

itbnum

Logical name table number in the lower byte and the status byte in the upper byte, as follows:

Table number:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

Status:

LS.TRM	1	Terminal status. Iterative translations will not proceed beyond this logical name.
LS.PRV	2	Privileged status. Only privileged tasks may delete this logical name.

lns

Character array containing the logical name string

lnssz

Size (in bytes) of the logical name string

iens

Character array containing the equivalence string to be created

ienssz

Size (in bytes) of the data area for the equivalence string

idsw
Integer to receive the Directive Status Word

Macro Calls

CLON$ [mod], <prmlst> ,lns,lnssz,ens,enssz
CLOG$ [mod], <prmlst> ,lns,lnssz,ens,enssz

Parameters

mod
Modifier of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB. See the Notes.

<prmlst>
<[tbnum][,status]>

(Angle brackets not required if only tbnum is specified.)

tbnum
Logical name table number. The following are the symbolic offsets for the table:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

status
Logical status definition value. The following are the valid bits for the value:

LS.TRM	1	Terminal status. Iterative translations will not proceed beyond this logical name.
LS.PRV	2	Privileged status. Only privileged tasks may delete this logical name.

lns
Logical name string

lnssz
Size (in bytes) of the logical name string

ens
Equivalence name string

enssz
Size (in bytes) of the equivalence name string

CLON$, CLOG$

Macro Expansion

```
     .MACRO CLON$ MOD,PRMLST,LNS,LNSSZ,ENS,ENSSZ

     .BYTE     207.,7            ;CLON$ MACRO DIC, DPB SIZE = 7 WORDS
     .BYTE     11.               ;SUBFUNCTION (CLOG$ = 0)
     .BYTE     MOD               ;MODIFIER OF LOGICAL NAME

$$$ARG = 0
     .IRP      SYM     <PRMLST> ;TABLE NUMBER AND STATUS
     .BYTE     SYM

$$$ARG=$$$ARG+1
     .ENDM
     .IFF      LT 2-$$$ARG,.ERROR
     .IF       GT 2-$$$ARG
     .REPT     <2-$$$ARG>

     .BYTE     0
     .ENDR
     .ENDC

     .WORD     LNS               ;ADDRESS OF LOGICAL NAME STRING
     .WORD     LNSSZ             ;SIZE IN BYTES OF LOGICAL NAME STRING

     .WORD     ENS               ;ADDRESS OF EQUIVALENCE NAME STRING
     .WORD     ENSSZ             ;SIZE IN BYTES OF EQUIVALENCE NAME STRING
```

Local Symbol Definitions

C.LENS Address of equivalence name buffer (2)

C.LESZ Byte count of equivalence name buffer (2)

C.LFUN Subfunction value (1)

C.LLNS Address of logical name string (2)

C.LLSZ Byte count of logical name string (2)

C.LMOD Logical name modifier (1)

C.LSTS Address of status block for LNB (1)

C.LTBL Logical table number (1)

DSW Return Codes

IS.SUC Successful completion.

IS.SUP Previous value of logical name was superseded.

IE.ITN Invalid table number specified.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.

IE.SDP DIC or DPB size is invalid.

Notes

1. You may specify any value up to 255_{10} for the logical name modifier. The logical names will be translated in ascending order. However, the RSX–11M–PLUS and Micro/RSX operating systems create and display the values of 1 (LB.LOC) and 2 (LB.LOG) only.
2. The CRELON and CLON$ calls are the preferred calls to use on RSX–11M–PLUS and Micro/RSX operating systems. The CRELOG and CLOG$ calls are provided for compatibility with the P/OS operating system. When you use CRELOG or CLOG$, the system performs the following actions:
   - If a device name or node name ends with one or more colons, strips off one to two of the terminating colons.
   - If a physical device name string is in the form ddnnn:, compresses any leading zeros. For example, DR005: becomes DR5.

CMKT$

5.13 Cancel Mark Time Requests

The Cancel Mark Time Requests directive instructs the system to cancel a specific Mark Time Request or all Mark Time requests that have been made by the issuing task.

FORTRAN Call

CALL CANMT ([efn][,ids])

Parameters

efn

Event flag number

ids

Directive status

Macro Call

CMKT$ [[efn],[ast],[err]]

Parameters

efn

Event flag number

ast

Mark time AST address

err

Error-routine address

Macro Expansion

```
CMKT$   52.,MRKAST,ERR  ;NOTE: THERE ARE TWO IGNORED ARGUMENTS
.BYTE   27.,3           ;CMKT$ MACRO DIC, DPB SIZE = 3 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52
.WORD   MRKAST          ;ADDRESS OF MARK TIME REQUEST AST ROUTINE
```

Note

The above example will cancel only the Mark Time requests that were specified with efn 52 or the AST address MRKAST. If no ast or efn parameters are specified, all Mark Time requests issued by the task are canceled and the DPB size equals 1.

Local Symbol Definitions

C.MKEF Event flag number (2)

C.MKAE Mark Time Request AST routine address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. If neither the efn nor ast parameters are specified, all Mark Time Requests issued by the task are canceled. In addition, the DPB size is one word. (When either the efn and/or ast parameters are specified, the DPB size is three words.)
2. If both efn and ast parameters are specified (and nonzero), only Mark Time Requests issued by the task specifying either that event flag or AST address are canceled.
3. If only one efn or ast parameter is specified (and nonzero), only Mark Time Requests issued by the task specifying the event flag or AST address are canceled.
4. If the specified event flag is a group global, then the use count for the event flag's group is run down when a Mark Time request is canceled.

CNCT$

5.14 Connect

The Connect directive synchronizes the task issuing the directive with the exit or emit status of another task (offspring) that is already active. Execution of this directive queues an Offspring Control Block (OCB) to the offspring task and increments the issuing task's rundown count (contained in the issuing task's Task Control Block). The rundown count is maintained to indicate the combined total number of tasks presently connected as offspring tasks and the total number of virtual terminals the task has created. The 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 cannot be issued to connect to command line interpreter (CLI) tasks because it is illegal to connect to a CLI task.

FORTRAN Call

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

Parameters

rtname

Name (Radix–50) of the offspring task to be connected

iefn

Event flag to be set when the offspring task exits or emits status

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL CNCTN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the iefn parameter above.

iparm

Name of a word to receive the status block address when an AST occurs

ids

Integer to receive the Directive Status Word

Macro Call

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

Parameters

tname

Name (Radix-50) of the offspring task to be connected

efn

The event flag to be cleared on issuance and set when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the efn parameter above.

Macro Expansion

```
CNCT$   ALPHA,1,CONAST,STBUF
.BYTE   143.,6          ;CNCT$ MACRO DIC, DPB SIZE = 6 WORDS
.RAD50  ALPHA           ;OFFSPRING TASK NAME
.BYTE   1               ;EVENT FLAG NUMBER = 1
.BYTE   16.             ;EXIT STATUS BLOCK CONSTANT
.WORD   CONAST          ;AST ROUTINE ADDRESS
.WORD   STBUF           ;EXIT STATUS BLOCK ADDRESS
```

Local Symbol Definitions

C.NCTN Task name (4)

C.NCEF Event flag (2)

C.NCEA AST routine address (2)

C.NCES Exit status block address (2)

CNCT$

DSW Return Codes

IS.SUC	Successful completion.
IE.UPN	Insufficient dynamic memory to allocate an Offspring Control Block.
IE.INS	The specified task was a command line interpreter.
IE.ACT	The specified task was not active.
IE.IEF	Invalid event flag number (EFN <0, or EFN$>$ 96 if group global event flags exist for the task's group or EFN$>$ 64 if not).
IE.ADP	Part of the DPB or exit status block is not in the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

Notes

1. If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - The connected task returns status.
   - The issuing task exits before status is returned.
2. Do not change the virtual mapping of the exit status block while the connection is in effect. Doing so may cause obscure errors because the exit status block is always returned to the virtual address specified regardless of the physical address to which it is mapped.

5.15 Checkpoint Common Region

The Checkpoint Common Region directive instructs the system to force the specified read/write common region to be checkpointed. This directive stops all the tasks that are mapped to the common region, writes the common region out to the disk, and then unstops the tasks.

Before the common region can be checkpointed with this directive, it must be installed with the VMR or MCR INSTALL /WB=YES command or the DCL INSTALL /WRITE_BACK command.

The issuing task must be privileged (PR:0) and must be attached to the specified common region.

If the issuing task is mapped to the specified common region, it is blocked. Any task (including the issuing task) is also blocked if it maps to the common region while the checkpoint is in progress. If the task was built with the /COMMON= qualifier, the task will be blocked when it issues this directive. If the task becomes attached by means of the Attach Region directive, it is not blocked unless it issues a Map directive.

You can use this directive to preserve changes made to a memory-resident common region. When a region installed with the /WB=YES switch or /WRITE_BACK qualifier is checkpointed, it is copied to its own image on the disk and not to the checkpoint file. Therefore, any update to the memory-resident copy of the common region becomes permanent.

FORTRAN Call

CALL CPCR (name[,ids])

Parameters

name

Name (Radix–50) of the common region to be checkpointed

ids

Directive Status

Macro Call

CPCR$ name

Parameter

name

Name of the common region to be checkpointed

Macro Expansion

```
CPCR$   NAME
.BYTE   205.,3          ;CPCR$ MACRO DIC, DPB SIZE = 3 WORDS
.RAD50  /NAME/
```

Local Symbol Definition

C.PCNM Name of common region

CPCR$

DSW Return Codes

IE.SUC	Successful completion.
IE.PRI	Privilege violation.
IE.NSP	The specified common region does not exist.
IE.ITS	I/O is in progress to the specified region or, if the region is a memory management (PLAS) common, the common was not installed with the /WB=YES switch or /WRITE_BACK qualifier.
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

5.16 Create Address Window

The Create Address Window directive creates a new virtual address window by allocating a window block from the header of the issuing task and establishing its virtual address base and size. (Space for the window block has to be reserved at task-build time by means of the WNDWS keyword. See the *RSX-11M-PLUS and Micro/RSX Task Builder Manual.*) Execution of this directive unmaps and then eliminates any existing windows that overlap the specified range of virtual addresses. If the window is created successfully, 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 through 23_{10}, which is an index to the window block in the task's header. The window block describes the created address window.

On RSX-11M-PLUS systems, if WS.SIS in the window status word (W.NSTS) is set, the Executive creates the window in supervisor-mode I-space. Program control can subsequently be transferred to supervisor-mode I-space upon issuing a Supervisor Call directive. If WS.UDS in the window status word is set, the Executive creates the window in user-mode D-space.

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 the window size specified when the window was created, and 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. (Because 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, as follows:

- 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.
- If WS.64B = 1, the task can align on 32-word boundaries; you 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 RSX emulators. To avoid future incompatibility, you should write applications adaptable to either alignment requirement. The bit setting of WS.64B could be a parameter chosen at assembly time (by means of a prefix file), at task-build time (as input to the GBLDEF option), or at run time (by means of command input or by means of the G.TSSY field returned from the GTSK$ directive).

FORTRAN Call

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

CRAW$

Parameters

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids

Directive status

Macro Call

CRAW$ wdb

Parameter

wdb

Window Definition Block address

Macro Expansion

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

Window Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
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[1] in the window status word:

Bit	Definition
WS.MAP	1 if the new window is to be mapped.

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Array Element	Offset	Meaning	
		Bit	**Definition**
		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.
		WS.NBP	Do not bypass the cache (for RSX–11M–PLUS multiprocessor systems and systems that have the conditional assembly parameter C$$CBP defined in RSXMC.MAC; C$$CBP is also included in the RL02 ID and MICROD pregenerated systems).
		WS.RES	Map only if resident.
		WS.NAT	Create attachment descriptor only if necessary (for Send By Reference directives).
		WS.MAP	Window is to be mapped.
		WS.RCX	Exit if no references to receive.
		WS.SIS	Create window in supervisor I-space.
		WS.UDS	Create window in user D-space (RSX–11M–PLUS systems only).
		WS.DEL	Send with delete access.
		WS.EXT	Send with extend access.
		WS.WRT	Send with write access or map with write access.
		WS.RED	Send with read access.

Output parameters:

Array Element	Offset	Meaning
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[1] in the window status word:

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

CRAW$

Array Element	Offset	Meaning	
		Bit	**Definition (if bit=1)**
		WS.CRW	Address window was created successfully.
		WS.UNM	At least one window was unmapped.
		WS.ELW	At least one window was eliminated.
		WS.RRF	Reference was received successfully.

Local Symbol Definition

C.RABA Window Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.HWR Directive failed in mapping storage because region has incurred a parity error.

IE.PRI Requested access denied at mapping stage.

IE.NVR Invalid region ID.

IE.ALG Task specified either an invalid base APR and window size combination or an invalid region offset and length combination in the mapping assignment, or WS.64B = 0 and the value of W.NOFF is not a multiple of 8.

IE.WOV No window blocks available in task's header.

IE.ADP Part of the DPB or WDB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.17 Create Group Global Event Flags

The Create Group Global Event Flags directive creates a Group Global Event Flag Control Block (GFB) and links it into the GFB list. If a GFB for the specified group is not present when the directive is issued, the Executive creates the GFB data structure with all event flags initialized to zero. If a GFB is present when the directive is issued, the Executive uses the present GFB and the event flags are not initialized. However, if the GFB is marked for deletion (by a previously issued Eliminate Group Global Event Flags directive), the Executive clears the GS.DEL bit.

If the specified group code matches the group code of the issuing task's protection UIC (H.CUIC+1), this directive increments the access count for the event flags. This locks the event flags so they cannot be eliminated by another task that is sharing them. The issuing task can explicitly unlock the event flags with an Unlock Group Global Event Flags directive or an Eliminate Group Global Event Flags directive. The Executive automatically unlocks the event flags when the task exits if necessary. Note that a task may not lock the event flags more than once in succession. Any attempt to lock event flags that are already locked will return the IE.RSU error code.

FORTRAN Call

CALL CRGF ([group][,idsw])

Parameters

group

Group number for the flags to be created. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

idsw

Integer to receive the Directive Status Word

Macro Call

CRGF$ [group]

Parameter

group

Group number for the flags to be created. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

CRGF$

Macro Expansion

```
CRGF$   4
.BYTE   157.,2              ;CRGF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   4                   ;GROUP 4 GLOBAL EVENT FLAGS
```

Local Symbol Definition

C.RGRP Group number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic storage.

IE.PRI Privilege violation.

IE.IUI Invalid group.

IE.RSU Event flags already exist or are already locked.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

A privileged task may specify group numbers other than the group UIC of the issuing task. However, the task can lock the event flags created for its own group only. This directive does not return an error if it does not lock the event flags.

5.18 Create Region

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. If no region name has been specified, your program 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, set RS.NDL in the region status word as an input parameter. Be careful if you consider overriding the delete-on-last-detach option. An error within a program can cause the system to lock by leaving no free space in a system-controlled partition.

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

Named regions are put in the Common Block Directory (CBD). However, memory is not allocated until the Executive maps a task to the region.

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

FORTRAN Call

CALL CRRG (irdb[,ids])

Parameters

irdb

An 8-word integer array containing a Region Definition Block (see Section 3.5.1.2)

ids

Directive status

Macro Call

CRRG$ rdb

Parameter

rdb

Region Definition Block address

Macro Expansion

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

Region Definition Block Parameters

Input parameters:

CRRG$

Array Element	Offset	Meaning
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[1] in the region status word:

Bit	Definition (if bit=1)
RS.CRR	Region was created successfully.
RS.UNM	At least one window was unmapped on a detach.
RS.MDL	Mark region for deletion on last detach.
RS.NDL	The region should not be deleted on last detach.
RS.ATT	Created region should be attached.
RS.NEX	Created region is not extendable.
RS.RED	Read access is desired on attach.
RS.WRT	Write access is desired on attach.
RS.EXT	Extend access is desired on attach.
RS.DEL	Delete access is desired on attach.

irdb(8)	R.GPRO	Protection word for the region (DEWR,DEWR,DEWR,DEWR)

[1] If you are a FORTRAN programmer, refer to Section 3.5.1 to define the bit values represented by the symbolic names described.

Output parameters:

Array Element	Offset	Meaning
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)

Array Element	Offset	Meaning
irdb(7)	R.GSTS	Bit settings[1] in the region status word:

Bit	Definition
RS.CRR	1 if the region was created successfully

[1] If you are a FORTRAN programmer, refer to Section 3.5.1 to define the bit values represented by the symbolic names described.

Local Symbol Definition

C.RRBA Region Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN A Partition Control Block (PCB) or an attachment descriptor could not be allocated, or the partition was not large enough to accommodate the region, or there is currently not enough continuous space in the partition to accommodate the region.

IE.HWR The directive failed in the attachment stage because a region parity error was detected.

IE.PRI Attach failed because desired access was not allowed.

IE.PNS Specified partition in which the region was to be allocated does not exist, or no region name was specified and RS.ATT = 0. The specified region is a main partition.

IE.ADP Part of the DPB or RDB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. The Executive does not return an error if the named region already exists. In this case, the Executive clears the RS.CRR bit in the status word R.GSTS. If RS.ATT has been set, the Executive attempts to attach the already existing named region to the issuing task.
2. The protection word (see R.GPRO above) has the same format as that of the file system protection word. There are four categories and the access for each category is coded into four bits. From low order to high order, the categories follow this order: system, owner, group, world. The access code bits within each category are arranged (from low order to high order) as follows: read, write, extend, delete. A bit that is set indicates that the corresponding access is denied.

   The issuing task's UIC is the created region's owner UIC.

   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.

CRVT$

5.19 Create Virtual Terminal

The Create Virtual Terminal directive creates a virtual terminal for use by a parent task in communicating with its offspring tasks. When the offspring task issues a read or write to its TI: terminal, the request is sent to the parent task through the virtual terminal. For example, when the batch processor invokes a task, it communicates with that task through a virtual terminal rather than a physical terminal.

This directive creates a Device Control Block (DCB) and a Unit Control Block (UCB) for each virtual terminal unit, and links the unit to the device list. Each newly created virtual terminal unit is assigned the lowest available virtual terminal unit number.

Only a single copy of the Status Control Block (SCB) is required. The data structure for Virtual Terminal Unit 0 (VT0:) is used as a template for these dynamically created data structures. Therefore, VT0: is never assigned as a virtual terminal unit number.

On successful completion of this directive, the assigned VT: unit number is returned in the DSW with the Carry bit clear. The task must save this number if this virtual terminal is to be referenced in another directive.

A rundown count is maintained in the issuing task's TCB to indicate the total (current) number of virtual terminals the task has created and the number of connected offspring tasks. This count is reduced when an Eliminate Virtual Terminal directive is issued specifying this VT: unit.

The input and output AST routines for the virtual terminal unit are entered with the following three words on the stack:

SP+04 Third parameter word (VFC) of offspring request

SP+02 Byte count of offspring request

SP+00 Virtual terminal unit number (low byte); I/O subfunction code of offspring request (high byte)

The attach and detach AST routines are entered with the following three words on the stack:

SP+04 Second word of offspring task name (0 if detach AST)

SP+02 First word of offspring task name (0 if detach AST)

SP+00 Virtual terminal unit number (low byte); I/O subfunction code of offspring request (high byte)

Note that the detach AST routine is entered with 0 in both task name words on the stack. The AST routine must remove the three words from the stack before it issues an AST Service Exit directive.

Parent tasks can service each offspring input or output request with a corresponding output or input request to the correct virtual device unit. For example, where MACRO-11 has been activated as an offspring task of the batch processor with a TI: of VT3:, the following actions occur:

1. MACRO-11 issues an IO.RVB or IO.RLB to TI: for its first input line. The virtual terminal driver queues the read request internally and effects an AST in the batch processor at the virtual address "iast" with the unit number 3 and the byte count from MACRO-11's I/O request on the stack.

2. In its AST routine, the batch processor retrieves an input line for MACRO–11 from the batch stream and specifies this line in a QIO directive to a LUN assigned to VT3: with an IO.WVB or IO.WLB function, a byte count of the line, and the status to be returned (such as IS.CR).
3. The virtual terminal driver reads the line from the batch processor's buffer, writes the line to MACRO–11's buffer, and then signals I/O completion for both I/O requests.
4. Similarly, if MACRO–11 needs to print an error message, it does so with an IO.WVB or IO.WLB to TI:. The virtual terminal driver queues the write request internally and effects an AST in the batch processor at the virtual address "oast" with the unit number 3, the byte count, and the VFC from MACRO–11's I/O request on the stack.
5. In its output AST routine, the batch processor issues an IO.RVB or IO.RLB to retrieve the line by means of the virtual terminal driver. The batch processor may then output this line to its log file. The third word on the AST stack in the batch output AST routine is the vertical format character, telling batch what type of carriage control is expected for the output line. This word would be ignored in the input AST routine.

The virtual terminal driver does not interpret or modify transferred bytes, I/O subfunction codes, or vertical format characters. However, this driver does automatically truncate offspring I/O requests to the maximum byte count specified in the "mlen" parameter, notifying neither the parent nor offspring task. The actual number of bytes transferred on each request is equal to the smaller of the byte counts specified in the offspring and parent I/O requests. The total number of bytes transferred is returned in the corresponding I/O status blocks. Note that offspring tasks can receive "mlen" in the fourth characteristics word when a Get LUN Information directive is issued.

Intermediate buffering in secondary pool, when enabled by the parent task, is performed on offspring input and output requests when the offspring task is checkpointable. Offspring tasks, therefore, may be stopped and checkpointed. If the parent task is stopped and checkpointed when the offspring task issues an I/O request, the resulting AST brings the parent task to an unstopped state from which it may return to memory to service the I/O request. Upon exit from the AST routine, the parent task is again stopped. This mode of operation allows the parent and offspring tasks to share the same physical memory, even while the parent task services the terminal I/O requests for the offspring task. Whenever, for any reason, the virtual terminal driver determines that it should not use intermediate buffering, offspring tasks are locked in memory when I/O requests are issued, and transfers occur directly between parent and offspring buffers.

The intermediate buffering of offspring I/O requests can normally be enabled and disabled by the parent task with the IO.STC function, as described below. An exception to this exists for virtual terminals created with an "mlen" parameter greater than a systemwide maximum specified at system-generation time. (The system generation procedure does not allow this maximum to be greater than 512.) If a Create Virtual Terminal directive is specified with an "mlen" parameter greater than the systemwide maximum, the parameter is accepted, but intermediate buffering for the created virtual terminal unit is automatically disabled. Furthermore, intermediate buffering for that unit cannot be enabled by the parent task with the IO.STC function.

Parent tasks specify the first word of the I/O completion status for the offspring request in the third word of the QIO DPB. For example, consider an offspring input request for 10 characters or more that is honored with a write logical of 10 characters and IS.CR in the third parameter word. The second word of the I/O status would be set to 10 and 10 characters would be transferred. Another example is when a parent task issues a read request to satisfy a write request that was issued by the offspring task. To notify the offspring task that its write request was satisfied, the parent task would specify IS.SUC in the third parameter word.

A special I/O function, IO.STC, returns status to an offspring task without a data transfer. The parameter word format for the IO.STC function is as follows:

- Word 0 with bit 0 set indicates that status is being returned.
- Word 0 with bit 1 clear, if the virtual terminal is in full-duplex mode, indicates that status is being returned for an offspring read request.
- Word 0 with bit 1 set, if the virtual terminal is in full-duplex mode, indicates that status is being returned for an offspring write request.

Note

If the virtual terminal is in half-duplex mode, bit 1 is ignored.

- Word 1 is the second word of I/O return status.
- Word 2 is the first word of I/O return status.

The status words are reversed in order to be similar to the format in which status must be passed back in a parent read or write function to an offspring task. The IO.STC function must be used to return status when no transfer is desired because a byte count of 0 is not allowed in an IO.RLB or IO.WLB (read logical block and write logical block operations, respectively). For example, IE.EOF (write end-of-file tape mark) would normally be returned with IO.STC.

Note that it is important to specify an I/O completion status for all parent read and write requests that satisfy corresponding requests from the offspring task. If a return status is not specified, it defaults to zero. A zero indicates that the I/O is still pending (IS.PND). This causes the offspring task to hang if it examines the I/O status block to determine whether the I/O is completed.

In addition to returning status, the IO.STC function has an additional purpose. It can also enable or disable intermediate buffering of I/O requests. (Note that a task cannot perform both IO.STC functions in the same I/O request.) If bit 0 of the first parameter word in IO.STC is clear, bit 1 in this word is interpreted as a disable buffering flag, as follows:

- If bit 0 is clear and bit 1 is set, intermediate buffering of offspring I/O is disabled.
- If bit 0 is clear and bit 1 is clear, buffering is enabled.

Buffering cannot be enabled on a virtual terminal unit that has been created with an "mlen" parameter greater than the systemwide maximum specified at system-generation time. An attempt to do both results in an error return of IE.IFC.

The only tasks that can assign LUNs to a virtual terminal unit are:

- The task that created the virtual terminal unit
- That task's offspring task or tasks, whose TI: is the virtual terminal unit

Attachment of a virtual terminal unit by an offspring task prevents the dequeuing of I/O requests to that unit from other offspring tasks. Parent I/O requests are always serviced.

Both parent and offspring tasks can specify the I/O functions IO.GTS, SF.GMC, and SF.SMC. However, SF.GMC and SF.SMC support only a limited number of terminal characteristics for virtual terminals. Refer to the *RSX-11M-PLUS and Micro/RSX I/O Drivers Reference Manual* for a list of valid characteristics.

Note that the parent task is not notified when the offspring issues any of the above directives.

When an offspring task issues a read-with-prompt request (IO.RPR), the virtual terminal driver separates the request into an IO.WLB request and an IO.RLB request. The parent task cannot issue an IO.RPR.

When a virtual terminal is in half-duplex mode, the virtual terminal driver handles only one offspring request at a time. For example, if the offspring task issues a read request and then issues a write request without waiting for the read to be completed, the driver queues the write request to be processed when the read is completed.

The parent task may issue an SF.SMC function to set the virtual terminal to full-duplex mode. In full-duplex mode, the write request in the previous example would be processed even if the previous read was not yet completed. If the parent task is at AST state, it will not receive notification of the I/O request.

Both parent and offspring tasks can issue an SF.GMC request to determine the mode of the virtual terminal. However, only the parent task can change the mode (using SF.SMC).

FORTRAN Call

CALL CRVT ([iiast],[ioast],[iaast],[imlen],iparm[,idsw])

Parameters

iiast

AST address at which input requests from offspring tasks are serviced

ioast

AST address at which output requests from offspring tasks are serviced

iaast

AST address at which the parent task may be notified of the completion of successful offspring attach and detach requests to the virtual terminal unit

Note

At least one of the above optional parameters should be specified. Otherwise, the virtual terminal created is treated as the null device.

imlen

Maximum buffer length allowed for offspring I/O requests

iparm

Address of 3-word buffer to receive information from the stack when an AST occurs

CRVT$

idsw

Integer to receive the Directive Status Word containing the virtual terminal number

Macro Call

CRVT$ [iast],[oast],[aast],[mlen]

Parameters

iast

AST address at which input requests from offspring tasks are serviced. If iast=0, offspring input requests are rejected with IE.IFC returned.

oast

AST address at which output requests from offspring tasks are serviced. If oast=0, offspring output requests are rejected with IE.IFC returned.

aast

AST address at which the parent task may be notified of the completion of successful offspring attach and detach requests to the virtual terminal unit. If aast=0, no notification of offspring attach/detach is returned to the parent task.

Note

At least one of the above optional parameters should be specified. Otherwise, the virtual terminal created is treated as the null device.

mlen

Maximum buffer length (in bytes) allowed for offspring I/O requests (default and maximum values for this parameter are system generation options)

Macro Expansion

```
CRVT$   IASTRU,OASTRU,PAST,20.
.BYTE   149.,5          ;CRVT$ MACRO DIC, DPB SIZE = 5 WORDS
.WORD   IASTRU          ;INPUT REQUEST AST ROUTINE ADDRESS
.WORD   OASTRU          ;OUTPUT REQUEST AST ROUTINE ADDRESS
.WORD   PAST            ;SUCCESSFUL VT ATTACH NOTIFICATION AST ROUTINE ADDRESS
.WORD   20.             ;MAXIMUM BUFFER LENGTH = 20(10) BYTES
```

Local Symbol Definitions

C.RVIA Input request AST routine address (2)

C.RVOA Output request AST routine address (2)

C.RVAA VT attach notification AST routine address (2)

C.RVML Maximum buffer length (2)

DSW Return Codes

unit	Successful completion results in the return of the unit number of the created virtual terminal unit with the Carry bit clear.
IE.UPN	Insufficient dynamic memory to allocate the virtual terminal device unit data structures.
IE.HWR	Virtual terminal device driver not resident.
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

CSRQ$

5.20 Cancel Scheduled Initiation Requests

The Cancel Scheduled Initiation Requests directive instructs the system to cancel all time-synchronized initiation requests for a specified task, regardless of the source of each request. These requests result from a Run directive or from any of the time-synchronized variations of the MCR or DCL RUN commands.

In a multiuser protection system, a nonprivileged task can cancel scheduled initiation requests only for a task with the same TI:.

FORTRAN Call

CALL CANALL (tsk[,ids])

Parameters

tsk

Task name

ids

Directive status

Macro Call

CSRQ$ tsk

Parameter

tsk

Scheduled (target) task name

Macro Expansion

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

Local Symbol Definition

C.SRTN Target task name (4)

DSW Return Codes

IS.SUC Successful completion.

IE.INS Task is not installed.

IE.PRI The issuing task is not privileged and is attempting to cancel requests made by another task.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

If you specify an error-routine address when using the $C or $S macro form, you must include a null argument for compatibility with RSX–11D systems. For example:

```
CSRQ$S  #TNAME,,ERR       ;CANCEL REQUESTS FOR "ALPHA"
    .
    .
    .
TNAME:  .RAD50  /ALPHA/
```

DECL$S

5.21 Declare Significant Event ($S Form Recommended)

The Declare Significant Event directive instructs the system to declare a significant event.

Declaration of a significant event causes the Executive to scan the Active Task List from the beginning, searching for the highest-priority task that is ready to run. Use this directive with discretion to avoid excessive scanning overhead.

FORTRAN Call

CALL DECLAR ([,ids])

Parameter

ids

Directive status

Macro Call

DECL$S [,err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

IS.SUC Successful completion.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as the DIR$ macro.

5.22 Delete Logical Name

The Delete Logical Name directive deletes a logical name from a logical name table and returns the resources used by that logical name to the system. You should delete logical names when they are no longer needed. If you do not specify a logical name string, DLOG$/DLON$ deletes all of the logical names within the specified logical name table.

The DELLON and DLON$ calls are the preferred calls to use on RSX–11M–PLUS and Micro/RSX operating systems. The DELLOG and DLOG$ calls are provided for compatibility with the P/OS operating system. See the Notes.

FORTRAN Calls

```
CALL DELLON   ([mod],itbnum,[lns],[lnssz][,idsw])
CALL DELLOG   ([mod],itbnum,[lns],[lnssz][,idsw])
```

Parameters

mod

Modifier of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB; if specified, any nonzero value must correspond to the valid symbolic references used by the system (see the Notes)

itbnum

Logical name table number. The tables and their corresponding numbers are:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

lns

Character array name containing the logical name string

lnssz

Size (in bytes) of the logical name string to delete

idsw

Integer to receive the Directive Status Word

Macro Calls

```
DLON$   [mod],tbnum[,lns,lnssz]
DLOG$   [mod],tbnum[,lns,lnssz]
```

DLON$, DLOG$

Parameters

mod

Modifier value of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB; if specified, any nonzero value must correspond to the valid symbolic references used by the system (see the Notes)

tbnum

Logical name table number. The tables and their corresponding numbers are:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

lns

Address of logical name string to be deleted

lnssz

Size (in bytes) of the logical name string

Macro Expansion

```
.MACRO DLON$ MOD,TBNUM,LNS,LNSSZ
.BYTE     207.,5          ;DLON$ MACRO DIC, DPB SIZE = 5 WORDS
.BYTE     12.             ;SUBFUNCTION (DLOG$ = 2)
.BYTE     MOD             ;MODIFIER OF LOGICAL NAME
.BYTE     TBNUM           ;TABLE NUMBER
.BYTE     0               ;RESERVED FOR FUTURE USE
.BYTE     LNS             ;ADDRESS OF LOGICAL NAME STRING
.BYTE     LNNSSZ          ;BYTE COUNT OF LOGICAL NAME STRING
```

Local Symbol Definitions

D.LFUN	Subfunction value (1)
D.LLNS	Address of logical name string (2)
D.LLSZ	Byte count of logical name string (2)
D.LMOD	Logical name modifier (1)
D.LTBL	Logical table number (1)

DSW Return Codes

IS.SUC	Successful completion.
IE.ITN	Invalid table number specified.
IE.LNF	The specified logical name string was not found.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. You may specify any value up to 255_{10} for the logical name modifier. The only logical names that will be deleted are those that match this value exactly. However, the RSX–11M–PLUS and Micro/RSX operating systems create and display the values of 1 (LB.LOC) and 2 (LB.LOG) only.
2. The DELLON and DLON$ calls are the preferred calls to use on RSX–11M–PLUS and Micro/RSX operating systems. The DELLOG and DLOG$ calls are provided for compatibility with the P/OS operating system. When you use DELLOG or DLOG$, the system performs the following actions:
   - If a device name or node name ends with one or more colons, strips off one to two of the terminating colons.
   - If a physical device name string is in the form ddnnn:, compresses any leading zeros. For example, DR005: becomes DR5.

DSAR$S, IHAR$S

5.23 Disable (or Inhibit) AST Recognition ($S Form Recommended)

The Disable (or Inhibit) AST Recognition directive instructs the system to disable recognition of ASTs for the issuing task. The ASTs are queued as they occur and are effected when the task reenables AST recognition. There is an implied disable AST recognition directive whenever an AST service routine is executing. When a task's execution is started, AST recognition is enabled. See the Notes.

FORTRAN Call

CALL DSASTR [(ids)]
CALL INASTR [(ids)]

Parameter

ids
Directive status

Macro Call

DSAR$S [err]
IHAR$S [err]

Paramdeflist

err
Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

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

Notes

1. This directive disables only the recognition of ASTs; the Executive still queues the ASTs. They are queued FIFO and will occur in that order when the task reenables AST recognition.
2. The FORTRAN calls, DSASTR (or INASTR) and ENASTR exist solely to control the possible jump to the PWRUP (power-up) routine. FORTRAN is not designed to link to a system's trapping mechanism. The PWRUP routine is strictly controlled by the system, which both accepts the trap and subsequently dismisses it. The FORTRAN program is notified by a jump to PWRUP, but must use DSASTR (or INASTR) and ENASTR to ensure the integrity of FORTRAN data structures (most importantly, the stack) during power-up processing.
3. Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

DSCP$S

5.24 Disable Checkpointing ($S Form Recommended)

The Disable Checkpointing directive instructs the system to disable the checkpointability of a task that has been installed as a checkpointable task. Only the affected task can issue this directive. A task cannot disable the ability of another task to be checkpointed.

FORTRAN Call

CALL DISCKP [(ids)]

Parameter

ids

Directive status

Macro Call

DSCP$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

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

Notes

1. When a checkpointable task's execution is started, checkpointing is enabled (that is, the task can be checkpointed).
2. Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

DTRG$

5.25 Detach Region

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 the directive is issued, 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.

FORTRAN Call

CALL DTRG (irdb[,ids])

Parameters

irdb

An 8-word integer array containing a Region Definition Block (see Section 3.5.1.2)

ids

Directive status

Macro Call

DTRG$ rdb

Parameter

rdb

Region Definition Block address

Macro Expansion

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

Region Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
irdb(1)	R.GID	ID of the region to be detached

Array Element	Offset	Meaning
irdb(7)	R.GSTS	Bit settings[1] in the region status word:

Bit	Definition
RS.MDL	1 if the region should be marked for deletion when the last task detaches from it

[1] If you are a FORTRAN programmer, refer to Section 3.5.1 to determine the bit values represented by the symbolic names described.

Output parameters:

Array Element	Offset	Meaning
irdb(7)	R.GSTS	Bit settings[1] in the region status word:

Bit	Definition
RS.UNM	1 if any windows were unmapped

[1] If you are a FORTRAN programmer, refer to Section 3.5.1 to determine the bit values represented by the symbolic names described.

Local Symbol Definition

D.TRBA Region Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.PRI The task, which is not attached with delete access, has attempted to mark the region for deletion on the last detach or the task has outstanding I/O.

IE.NVR The task specified an invalid region ID or attempted to detach region 0 (its own task region).

IE.ADP Part of the DPD or RDB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

ELAW$

5.26 Eliminate Address Window

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.

FORTRAN Call

CALL ELAW (iwdb[,ids])

Parameters

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids

Directive status

Macro Call

ELAW$ wdb

Parameter

wdb

Window Definition Block address

Macro Expansion

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

Window Definition Block Parameters

Input parameters:

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

Output parameters:

Array Element	Offset	Meaning
iwdb(7)	W.NSTS	Bit settings[1] in the window status word:

Bit	Definition
WS.ELW	1 if the address window was eliminated successfully
WS.UNM	1 if the address window was unmapped

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Local Symbol Definition

E.LABA Window Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.NVW Invalid address window ID.

IE.ADP Part of the DPB or WDB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

ELGF$

5.27 Eliminate Group Global Event Flags

The Eliminate Group Global Event Flags directive marks group global event flags for deletion. If no tasks in this group are using the group global event flags (the use count for this group maintained by the Executive in G.CNT is 0), the Group Global Event Flags Control Block (GFB) is immediately unlinked and deallocated. If tasks are using flags in this group, the Executive marks the flags for deletion (GS.DEL is set to 1) and the GFB is eliminated when no remaining tasks are using the flags in this group. However, if a Create Group Global Event Flags directive is issued before the flags are eliminated, the Executive clears GS.DEL.

If the specified group code matches the group code of the issuing task's protection UIC and the event flags are locked by this task (by a previous Create Group Global Event Flags directive), this directive unlocks the event flags by decrementing the access count. Note that a task may not unlock the event flags more than once in succession. Any attempt to unlock event flags that are already unlocked will return the IE.RSU error code.

FORTRAN Call

CALL ELGF ([group][,idsw])

Parameters

group

Group number of flags to be eliminated. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

idsw

Integer to receive the Directive Status Word

Macro Call

ELGF$ [group]

Parameter

group

Group number of flags to be eliminated. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

Macro Expansion

```
ELGF$   303
.BYTE   159.,2              ;ELGF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   303                 ;GROUP NUMBER 303 FLAGS
```

Local Symbol Definition

E.LGRP Group number (2)

DSW Return Codes

IS.SUC	Successful completion.
IE.PRI	Privilege violation.
IE.IUI	Invalid group (group> 377_8).
IE.IEF	Group is not found.
IE.RSU	Event flags are already marked for deletion.
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

ELVT$

5.28 Eliminate Virtual Terminal

The Eliminate Virtual Terminal directive causes the specified virtual terminal unit data structures to be marked for deallocation and eventually to be unlinked from the device list and deallocated. This directive can be issued only by the task that created the virtual terminal device unit. Any active nonprivileged tasks are aborted whose TI: device units are the virtual terminal being deallocated. TKTN messages reporting the abortion of these tasks in this instance are directed to CO:. Any LUNs assigned by the issuing task, or by any offspring task being aborted, are deassigned.

A rundown count is maintained in the TCB of each parent task. This count reflects the total number of outstanding virtual terminal units the task has created, plus the number of connected (offspring) tasks. A series of ELVT$ directives are issued when a parent task, which has not eliminated virtual terminals it has created, exits. The virtual terminal data structures continue to exist until the last task exits whose TI: is the virtual terminal unit and until all CLI commands for that unit have been processed.

FORTRAN Call

CALL ELVT (iunum[,idsw])

Parameters

iunum

Virtual terminal unit number

idsw

Integer to receive the Directive Status Word

Macro Call

ELVT$ unum

Parameter

unum

Unit number of the virtual terminal to be eliminated. The task must provide this parameter after the virtual terminal is created. (See Note.)

Macro Expansion

```
ELVT$   0
.BYTE   151.,2          ;ELVT$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   0               ;VIRTUAL TERMINAL UNIT NUMBER
```

Local Symbol Definition

E.LVNM VT unit number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.IDU The specified virtual terminal unit does not exist or it was not created by the issuing task.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

The actual virtual terminal unit number is not known until after the virtual terminal is actually created (that is, after successfully completing a Create Virtual Terminal directive). The Create Virtual Terminal directive DSW contains the actual virtual terminal unit number for use in the Eliminate Virtual Terminal directive. Thus, the task must save DSWs for all virtual terminals it creates and later eliminate them using the Eliminate Virtual Terminal directive.

EMST$

5.29 Emit Status

The Emit Status directive returns the specified 16-bit quantity to the specified connected task. It possibly sets an event flag or declares an AST if previously specified by the connected task in a Send, Request, and Connect, a Spawn, or a Connect directive. If the specified task is multiply connected to the task issuing this directive, the first (oldest) Offspring Control Block (OCB) in the queue is used to return status. If no task name is specified, this action is taken for all tasks that are connected to the issuing task at that time. 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.

FORTRAN Call

CALL EMST ([rtname],status[,idsw])

Parameters

rtname
Name of a task connected to the issuing task to which the status is to be emitted

status
A 16-bit quantity to be returned to the connected task

idsw
Integer to receive the Directive Status Word

Macro Call

EMST$ [tname],status

Parameters

tname
Name of a task connected to the issuing task to which the status is to be emitted

status
A 16-bit quantity to be returned to the connected task

Macro Expansion

```
EMST$   ALPHA,STWD
.BYTE   147.,4              ;EMST$ MACRO DIC, DPB SIZE = 4 WORDS
.RAD50  ALPHA               ;NAME OF CONNECTED TASK TO RECEIVE STATUS
.WORD   STWD                ;VALUE OF STATUS TO BE RETURNED
```

Local Symbol Definitions

E.MSTN Task name (4)

E.MSST Status to be returned (2)

DSW Return Codes

IS.SUC	Successful completion.
IE.ITS	The specified task is not connected to the issuing task.
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

ENAR$S

5.30 Enable AST Recognition ($S Form Recommended)

The Enable AST Recognition directive instructs the system to recognize ASTs for the issuing task; that is, the directive nullifies a Disable AST Recognition directive. ASTs that were queued while recognition was disabled are effected at issuance. When a task's execution is started, AST recognition is enabled.

FORTRAN Call

CALL ENASTR [(ids)]

Parameter

ids

Directive status

Macro Call

ENAR$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

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

Notes

1. Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

2. The FORTRAN calls DSASTR (or INASTR) and ENASTR exist solely to control the jump to the PWRUP (power-up) routine. FORTRAN is not designed to link to a system's trapping mechanism. The PWRUP routine is strictly controlled by the system. It is the system that both accepts the trap and subsequently dismisses it. The FORTRAN program is notified by a jump to PWRUP, but must use DSASTR (or INASTR) and ENASTR to ensure the integrity of FORTRAN data structures (most importantly, the stack) during power-up processing.

ENCP$S

5.31 Enable Checkpointing ($S Form Recommended)

The Enable Checkpointing directive instructs the system to make the issuing task checkpointable after its checkpointability has been disabled; that is, the directive nullifies a DSCP$S directive. This directive cannot be used to enable checkpointing of a task that was built noncheckpointable.

FORTRAN Call

CALL ENACKP [(ids)]

Parameter

ids

Directive status

Macro Call

ENCP$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

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

Note

Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

5.32 Exit If

The Exit If directive instructs the system to terminate the execution of the issuing task only if an indicated event flag is not set. The Executive returns control to the issuing task if the specified event flag is set. See the Notes.

FORTRAN Call

CALL EXITIF (efn[,ids])

Parameters

efn

Event flag number

ids

Directive status

Macro Call

EXIF$ efn

Parameter

efn

Event flag number

Macro Expansion

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

Local Symbol Definition

E.XFEF Event flag number (2)

DSW Return Codes

IS.SET Indicated EFN set; task did not exit.

IE.IEF Invalid event flag number (EFN <1, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

EXIF$

Notes

1. The Exit If 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 exit, the other task sends it a message. The message is lost because the Executive flushed the receiver task's receive queue when it decided to exit. This condition can be avoided if the sending task specifies a common event flag in the Send directive and the receiving task executes an Exit If directive specifying the same common event flag. If the event flag is set, the Exit If directive will return control to the issuing task, signaling that something has been sent.
2. A FORTRAN program that issues the Exit If call must first close all files by issuing CLOSE calls. See the *VAX FORTRAN User's Guide* or the *PDP-11 FORTRAN-77 User's Guide* for instructions on how to ensure that such files are closed properly if the task exits. To avoid the time overhead involved in the closing and reopening of files, the task should first issue the appropriate test or clear event flag directive. If the Directive Status Word indicates that the flag was not set, then the task can close all files and issue the call to Exit If.
3. On exit, the Executive frees task resources. In particular, the Executive:
   - Detaches all attached devices
   - Flushes the AST queue and despecifies all specified ASTs
   - Flushes the receive and receive-by-reference queues
   - Flushes the clock queue for any outstanding Mark Time requests for the task
   - Closes all open files (files open for write access are locked)
   - Detaches all attached regions, except in the case of a fixed task
   - Runs down the task's I/O
   - Deaccesses the group global event flags for the task's group
   - Disconnects from interrupts
   - Flushes all outstanding CLI command buffers for the task
   - Breaks the connection with any offspring tasks
   - Returns a success status (EX$SUC) to any parent tasks
   - Marks for deallocation all virtual terminal units the task has created (see the description of the CRVT$ directive)
   - Frees the task's memory if the exiting task was not fixed
4. If the task exits, the Executive declares a significant event.

5.33 Task Exit ($S Form Recommended)

The Task Exit directive instructs the system to terminate the execution of the issuing task.

FORTRAN Call

CALL EXIT (istat)

Parameter

istat

A 16-bit quantity to be returned to the parent task

See Note 5.

Macro Call

EXIT$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. A return to the task occurs only if the directive is rejected. Therefore, no Branch on Carry Clear instruction is generated if an error-routine address is given because the return occurs only with Carry set.
2. Exit causes a significant event to be declared.
3. On exit, the Executive frees task resources. In particular, the Executive:
   - Detaches all attached devices
   - Flushes the AST queue and despecifies all specified ASTs
   - Flushes the receive and receive-by-reference queues

- Flushes the clock queue for any outstanding Mark Time requests for the task
- Closes all open files (files open for write access are locked)
- Detaches all attached regions, except in the case of a fixed task
- Runs down the task's I/O
- Deaccesses the group global event flags for the task's group
- Disconnects from interrupts
- Flushes all outstanding CLI command buffers for the task
- Breaks the connection with any offspring tasks
- Returns a success status (EX$SUC) to any parent tasks
- Marks for deallocation all virtual terminal units the task has created (see the description of the CRVT$ directive)
- Frees the task's memory if the exiting task was not fixed

4. Because this directive requires only a 1-word DPB, the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

5. You can terminate FORTRAN tasks with the STOP statement or with CALL EXIT. CALL EXIT is a FORTRAN OTS routine that closes open files and performs other cleanup before it issues an EXIT$S directive (or a CALL EXST (istat) call in FORTRAN–77). FORTRAN tasks that terminate with the STOP statement result in a message displayed on the task's TI:. This message includes the task name (as it appears in the Active Task List), the statement causing the task to stop, and an optional character string specified in the STOP statement. Tasks that terminate with CALL EXIT do not display a termination message. For example, a FORTRAN task containing the following statement:

```
20   STOP 'THIS FORTRAN TASK'
```

exits with the following message displayed on the task's TI: (TT37 in this example):

```
TT37 -- STOP THIS FORTRAN TASK
```

5.34 Exit with Status

The Exit with Status directive causes the issuing task to exit, passing a 16-bit status back to all connected tasks (by the Spawn, Connect, or Send, Request, and Connect directive). If the issuing task has no connected tasks, then the directive simply performs a Task Exit. No format of the status word is enforced by the Executive; format conventions are a function of the cooperation between parent and offspring tasks. However, if an offspring task aborts for any reason, a status of EX$SEV is returned to the parent task. This value is interpreted as a "severe error" by batch processors. Furthermore, if a task performs a normal exit with other tasks connected to it, a status of EX$SUC (successful completion) is returned to all connected tasks.

FORTRAN Call

CALL EXST (istat)

Parameter

istat

A 16-bit quantity to be returned to the parent task

Macro Call

EXST$ status

Parameter

status

A 16-bit quantity to be returned to the parent task

Macro Expansion

```
EXST$   STWD
.BYTE   29.,2           ;EXST$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   STWD            ;VALUE OF STATUS TO BE RETURNED
```

Local Symbol Definition

E.XSTS Value of status to be returned (2)

DSW Return Codes

No status is returned if the directive is successfully completed because the directive causes the issuing task to exit.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

EXST$

Notes

1. On exit, the Executive frees a task's resources. In particular, the Executive:
   - Detaches all attached devices
   - Flushes the AST queue and despecifies all specified ASTs
   - Flushes the receive and receive-by-reference queues
   - Flushes the clock queue for any outstanding Mark Time requests for the task
   - Closes all open files (files open for write access are locked)
   - Detaches all attached regions, except in the case of a fixed task
   - Runs down the task's I/O
   - Deaccesses the group global event flags for the task's group
   - Disconnects from interrupts
   - Flushes all outstanding CLI command buffers for the task
   - Breaks the connection with any offspring tasks
   - Returns the specified exit status to any parent tasks
   - Marks for deallocation all the virtual terminal units that the task has created (see the description of the CRVT$ directive)
   - Frees the task's memory if the exiting task was not fixed
2. If the task exits, the Executive declares a significant event.

5.35 Extend Task

The Extend Task directive instructs the system to modify the size of the issuing task by a positive or negative increment of 32-word blocks. If the directive does not specify an increment value or specifies an increment value of zero, the Executive makes the issuing task's size equal to its installed size. The issuing task must be running in a system-controlled partition and cannot have any outstanding I/O when it issues the directive. The task must also be checkpointable to increase its size; if necessary, the Executive checkpoints the task and then returns the task to memory with its size modified as directed.

In a system that supports the memory management directives, the Executive does not change any current mapping assignments if the task has memory-resident overlays. However, if the task does not have memory-resident overlays, the Executive attempts to modify, by the specified number of 32-word blocks, the mapping of the task to its task region.

If the issuing task is checkpointable but has no preallocated checkpoint space available, a positive increment may require dynamic memory and extra space in a checkpoint file sufficient to contain the task.

There are several constraints on the size to which a task can extend itself using the Extend Task directive. These constraints are as follows:

- No task can extend itself beyond the maximum size set by the MCR SET /MAXEXT or the DCL SET EXTENSION_LIMIT command or the size of the partition in which it is running. (See the *RSX-11M-PLUS MCR Operations Manual*, the *RSX-11M-PLUS Command Language Manual*, or the *Micro/RSX User's Guide*.)
- A task that does not have memory-resident overlays cannot extend itself beyond 32K minus 32 words.
- A task that has preallocated checkpoint space in its task image file cannot extend itself beyond its installed size.
- A task that has memory-resident overlays cannot reduce its size below the highest window in the task partition.

FORTRAN Call

CALL EXTTSK ([inc][,ids])

Parameters

inc

A positive or negative number equal to the number of 32-word blocks by which the task size is to be extended or reduced

ids

Directive status

Macro Call

EXTK$ [inc]

EXTK$

Parameter

inc

A positive or negative number equal to the number of 32-word blocks by which the task size is to be extended or reduced

Macro Expansion

```
EXTK$   40
.BYTE   89.,3           ;EXTK$ MACRO DIC, DPB SIZE = 3 WORDS
.WORD   40              ;EXTEND INCREMENT, 40(8) BLOCKS (1K WORDS)
.WORD   0               ;RESERVED WORD
```

Local Symbol Definition

E.XTIN Extend increment (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory or insufficient space in a checkpoint file.

IE.ITS The issuing task is not running in a system-controlled partition.

IE.ALG The issuing task attempted to reduce its size to less than the size of its task header, or the task tried to increase its size beyond 32K words or beyond the maximum set by the MCR SET /MAXEXT or DCL SET EXTENSION_LIMIT command, or the task tried to increase its size to the extent that one virtual address window would overlap another, or the task has memory-resident overlays and it attempted to reduce its size below the highest window mapped to the task partition.

IE.RSU Other tasks are attached to this task partition.

IE.IOP I/O is in progress for this task partition.

IE.CKP The issuing task is not checkpointable and specified a positive integer.

IE.NSW The task attempted to extend itself to larger than the installed size (when checkpoint space is allocated in the task).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.36 Test for Specified System Feature

The Test for Specified System Feature directive tests for the presence of a specific system software or hardware option, such as floating-point support or the presence of the Commercial Instruction Set.

FORTRAN Call

CALL FEAT (isym,idsw)

Parameters

isym
Symbol for the specified system feature

idsw
Integer to receive the Directive Status Word

Macro Call

FEAT$ sym

Parameter

sym
Symbol for the specified system feature (see Table 5-1)

Table 5–1: System Feature Symbols

Symbol	Value	Meaning
FE$EXT	1	22-bit extended memory support (bit 1)
FE$MUP	2	Multiuser protection support
FE$EXV	3	Executive is supported to 20K words
FE$DRV	4	Loadable driver support
FE$PLA	5	PLAS support
FE$CAL	6	Dynamic checkpoint space allocation
FE$PKT	7	Preallocation of I/O packets
FE$EXP	8.	Extend Task directive support
FE$LSI	9.	Processor is an LSI-11
FE$OFF	10.	Parent/offspring tasking support
FE$FDT	11.	Full-duplex terminal driver support
FE$X25	12.	X.25 CEX is loaded
FE$DYM	13.	Dynamic memory allocation supported

Table 5–1 (Cont.): System Feature Symbols

Symbol	Value	Meaning
FE$CEX	14.	Comm Exec is loaded
FE$MXT	15.	MCR exit after each command mode
FE$NLG	16.	Logins disabled
FE$DAS	17.	Kernel data space supported (bit 17.)
FE$LIB	18.	Supervisor-mode libraries support
FE$MP	19.	System supports multiprocessing
FE$EVT	20.	System supports event trace feature
FE$ACN	21.	System supports CPU accounting
FE$SDW	22.	System supports shadow recording
FE$POL	23.	System supports secondary pools
FE$WND	24.	System supports secondary pool file windows
FE$DPR	25.	System has a separate directive partition
FE$IRR	26.	Install, run, and remove support
FE$GGF	27.	Group global event flag support
FE$RAS	28.	Receive/send data packet support
FE$AHR	29.	Alternate header refresh area support
FE$RBN	30.	Round-robin scheduling support
FE$SWP	31.	Executive level disk swapping support
FE$STP	32.	Event flag mask is in the TCB (1=YES)
FE$CRA	33.	System spontaneously crashed (1=YES) (bit 33.)
FE$XCR	34.	System crashed from XDT (1=YES)
FE$EIS	35.	System requires extended instruction set
FE$STM	36.	System has Set System Time directive
FE$UDS	37.	System supports user data space
FE$PRO	38.	System supports secondary pool prototype TCBs
FE$XHR	39.	System supports external task headers
FE$AST	40.	System has AST support
FE$11S	41.	RSX–11S system
FE$CLI	42.	System supports multiple CLIs

Table 5–1 (Cont.): System Feature Symbols

Symbol	Value	Meaning
FE$TCM	43.	System has separate terminal driver pool
FE$PMN	44.	System supports pool monitoring
FE$WAT	45.	System has watchdog timer support
FE$RLK	46.	System supports RMS record locking
FE$SHF	47.	System supports shuffler task
FE$CXD	49.	Comm Exec is deallocated (non-I/D only) (bit 49.)
FE$XT	50.	System is a P/OS system (1=YES)
FE$ERL	51.	System supports error logging
FE$PTY	52.	System supports parity memory
FE$DVN	53.	System supports decimal version numbers
FE$LCD	54.	System supports loadable crash drivers
FE$NIM	55.	System supports deleted fixed task images
FE$CHE	56.	System supports disk data caching
FE$LOG	57.	System supports extended logical names
FE$NAM	58.	System supports named directories
FE$FMP	59.	System supports Fast Map directive
FE$DCL	60.	DCL is default CLI
FE$DDS	61.	Named directory mode is default
FE$ACD	62.	System supports ACDs
HF$UBM	-1.	Processor has UNIBUS map (1=YES) (bit 1)
HF$EIS	-2.	Processor has extended instruction set
HF$QB	-3.	Processor has a Q-bus backplane
HF$DSP	-4.	Processor supports separate I/D space
HF$CIS	-8.	Processor supports commercial instruction set
HF$FPP	-16.	Processor has no floating-point unit (1=YES)
HF$NVR	-17.	PRO-300 nonvolatile RAM present (1=YES) (bit 17.)
HF$INV	-18.	Nonvolatile RAM present (1=YES)
HF$CLK	-19.	PRO-300 clock is present
HF$ITF	-20.	Invalid time format in nonvolatile RAM

FEAT$

Table 5–1 (Cont.): System Feature Symbols

Symbol	Value	Meaning
HF$PRO	-21.	Hardware system is a PRO-3xx
HF$BRG	-32.	PRO-300 bridge module present

Macro Expansion

```
FEAT$   FE$DVN
.BYTE   177.,2          ;FEAT$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   FE$DVN          ;FEATURE IDENTIFIER
```

Local Symbol Definition

F.EAF Feature identifier (2)

DSW Return Codes

IS.CLR Successful completion; feature not present.

IS.SET Successful completion; feature present.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.37 File Specification Scanner

The File Specification Scanner directive takes a string as input and returns a filled-in parse block.

FORTRAN Call

CALL FSS (fsbuf,fssz,prsblk,prssz,[reserv][,idsw])

Parameters

fsbuf
Array containing the file specification buffer

fssz
Size (in bytes) of the file specification buffer

prsblk
Array containing the parse block

prssz
Size (in bytes) of the parse block

reserv
Reserved parameter (must not be specified)

idsw
Integer to receive the Directive Status Word

Macro Call

FSS$ fsbuf,fssz,prsblk,prssz[,reserv]

Parameters

fsbuf
Address of the file specification buffer

fssz
Size (in bytes) of the file specification buffer

prsblk
Address of the parse block

prssz
Size (in bytes) of the parse block

reserv
Reserved parameter (must be blank)

FSS$

Macro Expansion

```
FSS$ FSBUF,FSSZ,PRSBLK,PRSSZ,RESERV
.BYTE   207.,7              ;FSS$ MACRO DIC, DPB SIZE = 7 WORDS
.BYTE   5                   ;FSS$ SUBFUNCTION
.BYTE   0                   ;RESERVED
.WORD   0                   ;RESERVED
.WORD   FSBUF               ;FILE SPECIFICATION BUFFER
.WORD   FSSZ                ;FILE SPECIFICATION SIZE
.WORD   PRSBLK              ;PARSE BLOCK ADDRESS
.WORD   PRSSZ               ;PARSE BLOCK SIZE
```

Local Symbol Definitions

F.LFUN Subfunction value (1)

F.RSV1 Reserved (1)

F.RSV2 Reserved (2)

F.LSBF Address of file specification buffer (2)

F.LSSZ Size (in bytes) of the file specification buffer (2)

F.LPBK Address of the parse block (2)

F.LPBZ Size (in bytes) of the parse block (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.

IE.SDP DIC or DPB size is invalid.

Notes

The parse block has the following format:

1. O$STAT (status word). Indicates the status of the operation. This field is always set to the value of 1. Any part of the string that could not be successfully parsed is returned in the trailing string descriptor.

   SU$SUC Success

2. O$FLAG (flag word). The following flags indicate what was found in the file specification:

   FS$NOD Node present

   FS$DEV Device present

   FS$DIR Directory

   FS$QUO Quoted file name present

   FS$NAM File name present

FS$TYP File type present

FS$VER File version present

FS$WCH Wildcard character present

FS$WDI Wild directory

FS$WNA Wild file name

FS$WTY Wild file type

FS$WVE Wild file version

3. O$NODS: Length of the node specification.
4. O$NODA: Address of the node specification.
5. O$DEVS: Length of the device specification.
6. O$DEVA: Address of the device specification.
7. O$DIRS: Length of the directory specification.
8. O$DIRA: Address of the directory specification.
9. O$NAMS: Length of the file name specification.
10. O$NAMA: Address of the file name specification.
11. O$TYPS: Length of the file type specification.
12. O$TYPA: Address of the file type specification.
13. O$VERS: Length of the file version specification.
14. O$VERA: Address of the file version specification.
15. O$TRLS: Length of the trailing string.
16. O$TRLA: Address of the trailing string.
17. O$ACCS: Length of the access control specification.
18. O$ACCA: Address of the access control specification.
19. O$LTYP (logical type byte). The first element that could be a logical name. This field can contain the following words:

    P.LNON No logical name present

    P.LNAM File name may be a logical name

    P.LDEV Device name may be a logical name

    P.LNOD Node specification may be a logical name

20. O$PLEN: Length of the parse block.

The above offsets are defined by the macro LNBDF$, not by FSS$.

Although the entire parse block is 20 words long, the size of the parse block specified in the call (prssz) determines how much of the block is returned.

GCCI$

5.38 Get Command for Command Interpreter

The Get Command for Command Interpreter directive instructs the system to retrieve a command buffer for a Command Line Interpreter (CLI) task and copy it to a buffer in the task's address space. Information about the issuing terminal can also be returned to the CLI task.

The directive can also return a message from the system to the CLI instead of a command if the CLI has been initialized with this capability. The offsets G.CCDV and G.CCUN indicate whether a system message has been returned. See the *RSX-11M-PLUS and Micro/RSX System Management Guide* for more information.

Only CLI tasks can issue this directive.

FORTRAN Call

CALL GTCMCI (icbf,icbfl,[iibuf],[iibfl],[iaddr],[incp][,idsw])

Parameters

icbf

Name of a byte array to receive the command

icbfl

Integer containing the size of the icbf array in bytes

iibuf

Name of an integer array to receive the optional information buffer

iibfl

Name of an integer containing the length of the optional information buffer. If you specify a length shorter than the information buffer, as much information as will fit in the specified length is returned.

iaddr

Name of an integer that contains the address in pool of the command desired. (This address was obtained by a previous call to GTCMCI with GC.CND specified.)

incp

Name of an integer containing a bit mask indicating the action to take if there is no command queued, as follows:

Bit	Octal Value	Definition
GC.CCS	000	Return with Carry set (default)
GC.CEX	001	Force CLI to exit instead of returning
GC.CST	002	Force CLI to stop instead of returning
GC.CND	200	Copy command into buffer, but do not dequeue it from the list

You must specify these as decimal values in your FORTRAN program.

idsw
Integer to receive the Directive Status Word

Macro Call

GCCI$ cbuf,cbfl,[ibuf],[ibfl],[addr],[ncp]

Parameters

cbuf
Address of buffer to receive command string

cbfl
Length of buffer; maximum buffer size is 266_{10}

ibuf
Address of buffer to receive information on the issuing terminal

ibfl
Length of buffer to receive information

addr
Address of command.

This address is returned in G.CCCA of the information buffer if GC.CND is specified in the ncp argument. If this argument is nonzero, then only the command with the address specified by this argument is copied and/or dequeued. Note that this address is filled in only if the command is not dequeued.

ncp
Action to take if no command buffer present, as follows:

Bit	Octal Value	Definition
GC.CCS	000	Return with Carry set (default)
GC.CEX	001	Force CLI to exit instead of returning
GC.CST	002	Force CLI to stop instead of returning
GC.CND	200	Copy command into buffer, but do not dequeue it from the list

Note

GC.CND can be supplied with one of the other options; for example, GC.CND!GC.CEX.

GCCI$

Command Buffer Format

G.CCDV If set, the ASCII device name of the issuing terminal; if cleared, a message from the system has been returned (2)

G.CCUN Octal unit number of the issuing terminal or the code identifying the system message (1)

G.CCCT Number of characters (1)

G.CCCL Number of characters in command line (2)

G.CCTC Terminator (1)

G.CCFL Flags (1)

The values returned in the flag byte G.CCFL are:

	Flag	Value	Definition
	GC.CNL	1	Null command line
	GC.CTE	2	Prompt from a task exit
	GC.CTC	100	Control-C notification packet

G.CCBF Command text in ASCII (256 bytes)

Information Buffer Format

The format of the information buffer in the CLI virtual address space is as follows:

G.CCW2 U.CW2 of issuing terminal (2)

G.CCPT Name of parent task (if any) (4)

G.CCOA Address of Offspring Control Block from parent (2)

G.CCPU Protection UIC of issuing task (if possible) (2); otherwise, protection UIC of issuing terminal

G.CCCU Default UFD of issuing task (if possible) (2); otherwise, default UFD of issuing terminal

G.CCCA Address of command, if not dequeued (2)

Macro Expansion

```
GCCI$   CBUF,CBFL,IBUF,IBFL,ADDR,NCP
.BYTE   127.,7.          ;GCCI$ MACRO DIC, DPB SIZE = 7 WORDS
.BYTE   NCP              ;ACTION TO TAKE IF NO COMMAND QUEUED
.BYTE   0
.WORD   ADDR             ;ADDRESS OF COMMAND
.WORD   CBUF             ;COMMAND BUFFER ADDRESS
.WORD   CBFL             ;COMMAND BUFFER LENGTH
.WORD   IBUF             ;INFORMATION BUFFER ADDRESS
.WORD   IBFL             ;INFORMATION BUFFER LENGTH
```

Local Symbol Definitions

G.CCNC Action if no command queued (2)

G.CCAD Address of command to be returned (2)

G.CCBA Address of command buffer (2)

G.CCBL Length of task's command buffer (2)

G.CCIA Address of optional information buffer (2)

G.CCIL Length of optional information buffer (2)

DSW Return Codes

IE.AST The stop-on-no-command option was set and the directive was issued from AST state.

IE.PRI Task is not a CLI.

IE.RSU The issuing task has a group global context active and the next command to be received would have caused the task's protection group to change.

IE.ITS No command was queued for the CLI and the directive was issued with the return-with-Carry-set option.

IS.CLR Returned with Carry clear when the CLI was unstopped due to command arrival, after having been stopped by a GCII$ directive with the stop-on-no-command-option set.

IE.ADP DPB, send buffer, or information buffer was outside the task's address space, or the information buffer was shorter than nine bytes.

IE.SDP DIC and DPB size is invalid.

Notes

1. The number of characters returned (G.CCCT) could be less than the number of characters in the command (G.CCCL) if the length of the command buffer in the task, as specified by the cbfl argument, is smaller than the actual command line. If there is sufficient room, a carriage return is placed at the end of the command line returned at G.CCBF in the command buffer inside the task to ease parsing.

2. If a command is returned successfully, the protection and default UICs for the issuing task are changed by this directive to match those of the originating task (if possible) or terminal. These values are returned in words G.CCPU and G.CCCU of the optional information buffer. If named directories are supported, the task context block pointer is changed to match the task context block pointer of the originating task (if possible) or to match the terminal context block pointer of the originating terminal. Note that the context block contains the default directory string.

5.39 Get Command Interpreter Information

The Get Command Interpreter Information directive instructs the system to fill a buffer with information about a specified CLI or the CLI associated with a given terminal. A task must be privileged in order to issue this directive for any terminal other than its own TI: or for a CLI to which its TI: is not set.

FORTRAN Call

CALL GETCII (ibuf,ibfl,[icli],[idev],[iunit][,ids])

Parameters

ibuf
Name of an integer array to receive the CLI information

ibfl
Length in bytes of the integer array to receive the CLI information

icli
Name of a 2-word array element containing the Radix-50 name of the CLI

idev
Name of an integer containing the ASCII name of the terminal (must be the name of a physical device; default = TI:)

iunit
Name of an integer containing the octal unit number of the terminal

ids
Directive status

Macro Call

GCII$ buf,bufl,[cli],[dev],[unit]

Parameters

buf
Address of buffer to receive information

bufl
Length of information buffer

cli
Name in Radix-50 of the CLI on which information is requested

dev
ASCII name of terminal whose CLI should be used (must be the name of a physical device; default = TI:)

GCII$

unit
Octal unit number of the terminal

Information Buffer Format

G.CICL Name of CLI (4)

G.CICS Bit settings in the CLI status word (2):

Bit	Value	Definition
CP.NUL	1	Pass empty command lines to CLI.
CP.MSG	2	CLI wants system messages.
CP.LGO	4	CLI wants commands from logged-out terminals.
CP.DSB	10	CLI is disabled (note that MCR does not check this bit).
CP.PRV	20	You must be privileged to set terminal to this CLI.
CP.SGL	40	Do not handle continuations.
CP.NIO	100	MCR..., HEL, BYE do no I/O to terminal. HEL and BYE also do not set CLI, and so forth.
CP.RST	200	Restricted access; only this CLI task can set a terminal to this CLI.
CP.EXT	400	Pass task exit prompt requests to CLI.
CP.CTC	2000	Pass Control-C notification packets.

G.CITK Name of task serving as CLI (4)

G.CIW2 Terminal's U.CW2 (2)

G.CIPU Terminal's protection UIC (2)

G.CICU Terminal's current UIC (2)

G.CIDP CLI default prompt string (16-word block; first byte is length of string)

Macro Expansion

```
GCII$ buf,bufl,cli,dev,unit
.BYTE   173.,7          ;DIC =173(10), DPB SIZE = 7 WORDS
.WORD   buf             ;ADDRESS OF BUFFER
.WORD   bufl            ;LENGTH OF BUFFER
.RAD50  /cli/           ;RADIX-50 NAME OF CLI
.ASCII  /dev/           ;ASCII NAME OF TERMINAL
.WORD   unit            ;TERMINAL UNIT NUMBER
```

Local Symbol Definitions

G.CIBF Address of buffer (2)

G.CIBL Length of buffer (2)

G.CICN Radix–50 name of CLI (4)

G.CIDV ASCII name of terminal (2)

G.CIUN Unit number of terminal (1)

DSW Return Codes

IE.MAP Both a terminal and a CLI were specified.

IE.INS Specified CLI does not exist.

IE.IDU Specified device was not a terminal or does not exist.

IE.PRI Nonprivileged task attempted to get information on a CLI other than its own.

IE.ADP Part of the DPB or buffer was out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. If the buffer is not long enough to contain all the information, the data that does not fit will not be supplied. No indication of this is returned to the issuing task. The buffer is filled from left to right.
2. You may not specify both a CLI and a terminal. If the cli argument is present, the dev and unit arguments must be zero.

GDIR$

5.40 Get Default Directory

The Get Default Directory directive retrieves the default directory string, and returns it and the string length to a user-specified buffer.

FORTRAN Call

CALL GETDDS (mod,iens,ienssz,[irsize][,idsw])

Parameters

mod

Modifier for the GDIR$ directive; specify one of the following values:

0	Get task default
GD.LOG	Get terminal default

iens

Character array containing the default directory string

ienssz

Size (in bytes) of the default directory string

irsize

Buffer address of the returned default directory string size

idsw

Integer to receive the Directive Status Word

Macro Call

GDIR$ [mod],ens,enssz[,rsize]

Parameters

mod

Modifier for the GDIR$ directive; specify one of the following values:

0	Get task default
GD.LOG	Get terminal default

ens

Buffer address of the default directory string

enssz

Size (in bytes) of the default directory string buffer

rsize

Buffer address to which the size of the default directory string is returned

Macro Expansion

```
GDIR$ MOD,ENS,ENSSZ,RSIZE
.BYTE   207.,6          ;GDIR$ MACRO DIC, DPB SIZE = 6 WORDS
.BYTE   4               ;SUBFUNCTION CODE FOR GET DEFAULT DIRECTORY
.BYTE   MOD             ;MODIFIER
.WORD   0               ;RESERVED
.WORD   ENS             ;BUFFER ADDRESS OF DEFAULT DIRECTORY STRING
.WORD   ENSSZ           ;BYTE COUNT OF DEFAULT DIRECTORY STRING
.WORD   RSIZE           ;BUFFER ADDRESS FOR RETURNED DEFAULT DIRECTORY STRING
```

Local Symbol Definitions

G.DENS Address of default directory string buffer (2)

G.DESZ Byte count of the default directory string (2)

G.DFUN Subfunction code (1)

G.DMOD Modifier (1)

G.DRSZ Buffer address for the returned default directory string size (2)

DSW Return Codes

IS.SUC Successful completion of service.

IE.RBS The resulting default directory string is too large for the buffer to receive it.

IE.LNF Default directory string does not exist.

IE.IBS The length of the default directory string is invalid. The string length must be greater than 0 but less than 256_{10}.

IE.ITN Illegal table number. The reserved word in the DPB was not a zero.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have proper access to that region.

IE.SDP DIC or DPB size is invalid, or an illegal subfunction code was specified.

Notes

In addition to the terminal default directory associated with each logged-in terminal, a default directory string is associated with each active task. The default directory string (DDS) is stored in a context block (CTX).

The following rules apply to default directory strings and their context blocks:

- Each logged-in terminal has a default directory string stored in a context block, referred to as the terminal_CTX. The context block is created by HELLO/LOGIN when you log in and is deleted by BYE when you log out. You can change the terminal_CTX by using either the MCR SET /DEF or DCL SET DEFAULT command. The context block is pointed to from the terminal's Unit Control Block (UCB).

- Each active task has associated with it a default directory string referred to as the task_CTX. Exceptions to this rule are system tasks running from the console terminal (CO:), such as LDR, F11ACP, SHF, and so on. The task_CTX is pointed to from the Task Control Block (TCB).
- When a task is activated from a terminal, the terminal_CTX is propagated to the task_CTX.
- When a task issues the GDIR$ directive, the DDS from the task_CTX is returned. If GD.LOG is specified as a modifier, the DDS is taken from the terminal_CTX.
- When a task spawns an offspring task, the parent's task_CTX is propagated.
- When an entry is inserted into the clock queue for time-based schedule requests from a task, the issuing task's task_CTX is propagated to the clq_CTX (the context block for the clock queue). When an entry is inserted into the clock queue for time-based schedule requests from a terminal CLI command, the issuing terminal's terminal_CTX is propagated to the clq_CTX. When the time expires and the task is activated, the task_CTX is propagated from the clq_CTX.
- When a task sends a packet to a slave task, the sender's task_CTX is propagated to the packet_CTX (the context block for the packet). When the slave task issues a Receive Data (RCVD$) directive to get the packet, the receiver's task_CTX is propagated from the packet_CTX.

5.41 General Information

The General Information directive provides general information for user tasks. It instructs the system to perform the function found in the Directive Parameter Block (DPB). The functions either set parameters or get information. Each function includes a macro call, buffer format, macro expansion, and Directive Status Word (DSW) return codes.

The following sections describe the functions.

Notes

1. For each of the following functions, you must include a variable for every element in the macro definition.
2. FORTRAN calls are not supported for the GIN$ functions.

5.41.1 GI.GAS - Get Assigned Device Name

The Get Assigned Device Name (GI.GAS) function searches the assignment list for logical assignments of the specified terminal. When the specified assignment is found, the name of the device to which the assignment applies is returned to the task.

Macro Call

GIN$ GI.GAS, buf, siz, dev, unt, udev, unum

Parameters

GI.GAS
GIN$ function code (0)

buf
Address of 6-word buffer to receive the LUN information

siz
Buffer size in words

dev
Device name

unt
Device unit number

udev
Device name for which this assignment holds (if blank, get global assignment)

unum
Unit number of terminal for which this assignment holds (if high bit set, get login assignment)

GIN$

Buffer Format

Word 0	Name of assigned device
Word 1	Unit number of assigned device and flags byte
Word 2	First device characteristics word
Word 3	Second device characteristics word
Word 4	Third device characteristics word
Word 5	Fourth device characteristics word

Macro Expansion

```
GIN$    GI.GAS, GABUF, GASIZ, "XQ, 0, "TT, 0
.BYTE   169.,10
.WORD   GI.GAS
.WORD   GABUF
.WORD   GASIZ
.WORD   "XQ
.WORD   0
.WORD   "TT
.WORD   0
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.IDU	The specified device is not a terminal.
IE.PRI	The issuing task is not privileged.
IE.SDP	Invalid function code or the DPB size is invalid.
IE.ULN	No assignment exists for the logical name.

Note

A task must be privileged to issue this function.

5.41.2 GI.UIC - Get System UIC Information

The Get System User Identification Code Information (GI.UIC) function returns the system UIC, the library UIC, the task's current and protection UICs, and the issuing terminal (TI:) login UIC. If more space is available, the current terminal UIC, terminal command line interpreter (CLI), system name, network UIC, and system size in 32-word blocks are also returned.

Macro Call

```
GIN$  GI.UIC, buf, siz
```

Parameters

GI.UIC
GIN$ function code (1)

buf
Address of 5- or 32-word buffer to receive the information

siz
Buffer size in words

Buffer Format

Word 0	System UIC
Word 1	Library UIC
Word 2	H.DUIC for the requesting task
Word 3	H.CUIC for the requesting task
Word 4	Login UIC
Word 5	(Optional) Current terminal UIC
Word 6	(Optional) Radix–50 CLI name, first half
Word 7	(Optional) Radix–50 CLI name, second half
Word 8	(Optional) ASCII system name, first third
Word 9	(Optional) ASCII system name, middle third
Word 10	(Optional) ASCII system name, last third
Word 11	(Optional) Network UIC
Word 12	(Optional) System size in 32-word blocks

Macro Expansion

```
GIN$    GI.UIC, GABUF, GASIZ
.BYTE   169.,4
.WORD   GI.UIC
.WORD   GABUF
.WORD   GASIZ
```

DSW Return Code

IS.SUC	Successful completion.

Note

The buffer size must be a minimum of 32 words for the optional information to be received.

5.41.3 GI.DEF - Set Task Default UIC

The Set Task Default UIC (GI.DEF) function sets the default UIC for the requesting task. If the task is not privileged, only the default UIC is changed. If the task is privileged, both the default and protection UICs are modified.

Macro Call

GIN$ GI.DEF, uic

Parameters

GI.DEF

GIN$ function code (2)

uic

User Identification Code

Macro Expansion

```
GIN$    GI.DEF, <<<1*400>+54>>
.BYTE   169.,3
.WORD   GI.DEF
.WORD   <1*400>+54
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.IUI	The specified UIC is invalid.
IE.RSU	Group global event flags are active for task.
IE.SDP	Invalid function code or the DPB size is invalid.

Note

If an immediate expression is used for the UIC, it must be enclosed in double angle brackets (< < > >).

5.41.4 GI.SPR - Set Task Privilege

The Set Task Privilege (GI.SPR) function requests the setting or clearing of the task privilege bit (T3.PRV) in the issuing task's Task Control Block (TCB). The previous state of the bit is saved in T4.PRV.

Macro Call

GIN$ GI.SPR, flg

Parameters

GI.SPR
GIN$ function code (7)

flg
New privilege bit in bit 0

Macro Expansion

```
GIN$    GI.SPR, 0
.BYTE   169.,3
.WORD   GI.SPR
.WORD   0
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.PRI	The task was not previously privileged.
IE.SDP	Invalid function code or the DPB size is invalid.

Note

The privilege bit may be set only if it was originally set, then cleared.

5.41.5 GI.REN - Rename Task

The Rename Task (GI.REN) function renames the issuing task to the supplied task name. The new name is checked for uniqueness and, if unique, the issuing task is renamed.

Macro Call

GIN$ GI.REN, nam1, nam2

Parameters

GI.REN
GIN$ function code (8)

nam1
Radix–50 task name, first half

nam2
Radix–50 task name, second half

Macro Expansion

```
GIN$    GI.REN, <<^RNEW>>, <<^RTSK>>
.BYTE   169.,4
.WORD   GI.REN,
.WORD   ^RNEW
.WORD   ^RTSK
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.RSU	The specified task name is already in use.
IE.SDP	Invalid function code or the DPB size is invalid.

Notes

1. If an immediate Radix-50 expression is used for the task name, it must be enclosed in double angle brackets (< < > >).
2. Tasks may rename to normally invalid task names such as all blanks. This should be avoided, however, because the CLI directive cannot abort such tasks.
3. Tasks that receive DECnet connections or send data packets should not use this directive.

5.41.6 GI.FMK - Get Feature Mask Words

The Get Feature Mask Words (GI.FMK) function returns the system Executive feature mask, hardware feature mask, system base level, system type, and system version words to the requesting task.

Macro Call

GIN$ GI.FMK, buf, siz

Parameters

GI.FMK
GIN$ function code (3)

buf
Address of 9-word buffer to receive the information

siz
Buffer size in words

Buffer Format

Word 0	First Executive feature mask word
Word 1	Second Executive feature mask word
Word 2	Third Executive feature mask word
Word 3	Fourth Executive feature mask word
Word 4	Hardware feature mask word
Word 5	ASCII system base level, first half
Word 6	ASCII system base level, second half
Word 7	ASCII system version, first half
Word 8	ASCII system version, second half
Words 9-14	ASCII system type

Macro Expansion

```
GIN$    GI.FMK, FMBUF, FMSIZ
.BYTE   169.,4
.WORD   GI.FMK
.WORD   FMBUF
.WORD   FMSIZ
```

DSW Return Codes

IS.SUC	Successful completion.

Note

The system type is returned if the buffer is 15_{10} words or longer.

5.41.7 GI.QMC - Queue MCR Command Line

The Queue MCR Command Line (GI.QMC) function queues a command line to the MCR command line interpreter on the task's host terminal.

Macro Call

GIN$ GI.QMC, buf, siz

Parameters

GI.QMC
GIN$ function code (4)

buf
Address of buffer containing the MCR command line

siz
Buffer size in words

Buffer Format

Words 00-x Command line characters

Macro Expansion

```
GIN$    GI.QMC, MQBUF, MQSIZ
.BYTE   169.,4
.WORD   GI.QMC
.WORD   MQBUF
.WORD   MQSIZ
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.IDU	The host terminal is a virtual terminal marked for elimination.
IE.SDP	Invalid function code or the DPB size is invalid.
IE.UPN	Insufficient pool available to queue command buffer.

Note

The command buffer to be queued should be terminated by a carriage return or escape character. If the command buffer is not terminated by a carriage return or escape character, the buffer is copied up to the length of an MCR command buffer. This may cause unpredictable results.

5.41.8 GI.UAB - Get User Account Block

The Get User Account Block (GI.UAB) function moves the contents of a User Account Block (UAB) to a user buffer.

Macro Call

GIN$ GI.UAB, buf, siz, dev, unt

Parameters

GI.UAB
GIN$ function code (5)

buf
Address of buffer to receive the UAB information

siz
Buffer size in words

dev
Device name (if blank, use task's TI:)

unt
Device unit number

Buffer Format

The format of this buffer is subject to change. Consult the *RSX-11M-PLUS and Micro/RSX Crash Dump Analyzer Reference Manual* or the *RSX-11M-PLUS and Micro/RSX Guide to Writing an I/O Driver* for the ACNDF$ system macro, which defines the format of a UAB.

Macro Expansion

```
GIN$    GI.UAB, UABUF, UASIZ, "TT, 1
.BYTE   169.,6
.WORD   GI.UAB
.WORD   UABUF
.WORD   UASIZ
.WORD   "TT
.WORD   1
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.IDU	The specified device is not a terminal, or no UAB exists.
IE.PRI	Nonprivileged user specified a terminal.
IE.SDP	Invalid function code or the DPB size is invalid.

Notes

1. The buffer size must be a minimum of <B.ULEN/2> words.
2. The format of the UAB is subject to change. Offsets into the returned buffer should be defined using the system macro ACNDF$.

5.41.9 GI.DEV - Get Device Information

The Get Device Information (GI.DEV) function returns information about a particular device. The device on which information is returned is determined by first performing a logical assignment (if required) and then following any redirection assignments. Device assignments are checked if the high bit in the flags byte is clear; otherwise, no check of device assignments is made.

Macro Call

GIN$ GI.DEV, buf, siz, dev, unt

Parameters

GI.DEV

GIN$ function code (6)

buf

Address of buffer to receive the unit information

siz

Buffer size in words

GIN$

dev

Device name (if blank, use task's TI:)

unt

Device unit number (if high bit clear, follow assignments)

Buffer Format

Word 0	Device-characteristics word: Bit 0—A logical assignment was followed Bit 1—Unit is allocated Bit 2—Unit is attached Bit 3—Unit has a labeled tape Bit 4—Unit is marked for dismount Bit 5—Unit is mounted foreign Bit 6—Unit is not mounted Bit 7—Unit or controller is off line Bit 8—Unit is off line Bit 9—Unit is redirected Bit 10—Unit is a public device Bit 11—Unit is attached for diagnostics Bit 12—Device controller is off line Bit 13—Unit is allocated by issuing task's TI: Bit 14—Unit is attached by issuing task Bit 15—Device driver is unloaded
Word 1	(Optional) First device-characteristics word
Word 2	(Optional) Unit Control Block (UCB) U.PRM disk size doubleword, first half
Word 3	(Optional) UCB U.PRM+2 disk size doubleword, second half
Word 4	(Optional) ASCII device name
Word 5	(Optional) (Low byte) Device logical unit number (High byte) LCB L.TYPE logical assignment type
Word 6	(Optional) Radix–50 attaching task name, first half
Word 7	(Optional) Radix–50 attaching task name, second half
Word 8	(Optional) ASCII device name of allocating terminal
Word 9	(Optional) Unit number of allocating terminal
Word 10	(Optional) Radix–50 Ancillary Control Processor (ACP) name, first half
Word 11	(Optional) Radix–50 ACP name, second half
Word 12	(Optional) ASCII volume name, first sixth
Word 13	(Optional) ASCII volume name, second sixth
Word 14	(Optional) ASCII volume name, third sixth
Word 15	(Optional) ASCII volume name, fourth sixth

Word 16 (Optional) ASCII volume name, fifth sixth

Word 17 (Optional) ASCII volume name, last sixth

Macro Expansion

```
GIN$    GI.DEV, DVBUF, DVSIZ, "TT, 1
.BYTE   169.,6
.WORD   GI.DEV
.WORD   DVBUF
.WORD   DVSIZ
.WORD   "TT
.WORD   1
```

DSW Return Codes

IS.SUC	Successful completion.
IE.ADP	Part of DPB is out of task's address space.
IE.IDU	The specified device does not exist, or device is a virtual terminal and issuing task is not parent or offspring.
IE.SDP	Invalid function code or the DPB size is invalid.

Notes

1. If the task has the "slave" attribute, logical assignments are not checked regardless of the setting of the high bit in the fourth parameter word.
2. Optional information is returned only if there is room in the buffer and the information is available.

5.41.10 GI.APR - Get System APRs

The Get System Active Page Registers (GI.APR) function returns information on the contents of the Page Address Registers (PARs) and Page Description Registers (PDRs) for all modes and spaces present on the host system.

Macro Call

GIN$ GI.APR, buf, siz

Parameters

GI.APR

GIN$ function code (9)

buf

Address of 97-word buffer to receive the APR information

siz

Buffer size in words

Buffer Format

Word 0	Buffer-characteristics word: Bit 0—Kernel D-space information is present Bit 1—User D-space information is present Bit 2—Supervisor mode information is present
Words 01-08	Kernel I-space PAR 0 to PAR 7
Words 09-16	Kernel I-space PDR 0 to PDR 7
Words 17-24	(Optional) Kernel D-space PAR 0 to PAR 7
Words 25-32	(Optional) Kernel D-space PDR 0 to PDR 7
Words 33-40	User I-space PAR 0 to PAR 7
Words 41-48	User I-space PDR 0 to PDR 7
Words 49-56	(Optional) User D-space PAR 0 to PAR 7
Words 57-64	(Optional) User D-space PDR 0 to PDR 7
Words 65-72	(Optional) Supervisor I-space PAR 0 to PAR 7
Words 73-80	(Optional) Supervisor I-space PDR 0 to PDR 7
Words 81-88	(Optional) Supervisor D-space PAR 0 to PAR 7
Words 89-96	(Optional) Supervisor D-space PDR 0 to PDR 7

Macro Expansion

```
GIN$    GI.APR, APBUF, APSIZ
.BYTE   169.,4
.WORD   GI.APR
.WORD   APBUF
.WORD   APSIZ
```

DSW Return Code

IS.SUC Successful completion.

Note

Bits set in the first buffer word indicate that sets of buffer words are valid.

5.41.11 GI.TSK - Find and Return Task Information

The Find and Return Task Information (GI.TSK) function returns information on a task which may have its Task Control Block (TCB) in secondary or primary pool.

Macro Call

GIN$ GI.TSK, buf, siz, nam1, nam2

Parameters

GI.TSK
GIN$ function code (10_{10})

buf
Address of buffer to receive the task information

siz
Buffer size in words

nam1
First half of Radix-50 task name

nam2
Second half of Radix-50 task name

Buffer Format

Word 0	Radix-50 task name, first half
Word 1	Radix-50 task name, second half
Word 2	(Optional) T.STAT TCB task status word
Word 3	(Optional) T.ST2 TCB task status word
Word 4	(Optional) T.ST3 TCB task status word
Word 5	(Optional) T.ST4 TCB task status word
Word 6	(Optional) Radix-50 task partition name, first half
Word 7	(Optional) Radix-50 task partition name, second half

Macro Expansion

```
GIN$    GI.TSK, GTBUF, GTSIZ, <<^R...>>, <<^RTSK>>
.BYTE   169.,6
.WORD   GI.TSK
.WORD   GTBUF
.WORD   GTSIZ
.WORD   ^R...
.WORD   ^RTSK
```

DSW Return Codes

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

Notes

1. If an immediate Radix–50 expression is used for the task name, it must be enclosed in double angle brackets (< < > >).
2. If the task name is in the form ...xxx, the multiuser task xxxTnn is searched. If the task is not found, the prototype task is searched.
3. If the task name is of the form xxx$$$, only the prototype task is searched.
4. If the task name is limited to three characters and a task is not found, an additional search is made for a multiuser task or prototype task.
5. Optional information is returned only if there is room in the buffer.

5.41.12 GI.UPD - Update UICs and Default Directory

The Update UICs and Default Directory (GI.UPD) function takes the default UIC and the protection UIC from the Unit Control Block (UCB) of the terminal and copies them into the header of the task. If the default directory of the task and the default directory of the terminal are different, GI.UPD sets the default of the task directory to the same default as the terminal directory. Then, it returns the same information as the GI.UIC function.

Macro Call

```
GIN$  GI.UPD, buf, siz
```

Parameters

GI.UPD

GIN$ function code (17_{10})

buf

Address of 5- or 32-word buffer to receive the information

siz

Buffer size in words

Buffer Format

Word 0	System UIC
Word 1	Library UIC
Word 2	H.DUIC for the requesting task
Word 3	H.CUIC for the requesting task
Word 4	Login UIC
Word 5	(Optional) Current terminal UIC
Word 6	(Optional) Radix–50 CLI name, first half
Word 7	(Optional) Radix–50 CLI name, second half

Word 8 (Optional) ASCII system name, first third

Word 9 (Optional) ASCII system name, middle third

Word 10 (Optional) ASCII system name, last third

Word 11 (Optional) Network UIC

Word 12 (Optional) System size in 32-word blocks

Macro Expansion

```
GIN$    GI.UPD, GABUF, GASIZ
.BYTE   169.,4
.WORD   GI.UPD
.WORD   GABUF
.WORD   GASIZ
```

DSW Return Code

IS.SUC Successful completion.

Note

The buffer size must be a minimum of 32 words for the optional information to be received.

GLUN$

5.42 Get LUN Information

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

FORTRAN Call

CALL GETLUN (lun,dat[,ids])

Parameters

lun

Logical unit number

dat

A 6-word integer array to receive the LUN information

ids

Directive status

Macro Call

GLUN$ lun,buf

Parameters

lun

Logical unit number

buf

Address of a 6-word buffer that will receive the LUN information

Buffer Format

Word 0 Name of assigned device

Word 1 Unit number of assigned device and flags byte (flags byte equals 200 if the device driver is resident or 0 if the driver is not loaded)

Word 2 First device-characteristics word:

- Bit 0 Record-oriented device (DV.REC,1=yes)[FD.REC][1]
- Bit 1 Carriage-control device (DV.CCL,1=yes)[FD.CCL]
- Bit 2 Terminal device (DV.TTY,1=yes)[FD.TTY]
- Bit 3 Directory (file-structured) device (DV.DIR,1=yes)[FD.DIR]
- Bit 4 Single-directory device (DV.SDI,1=yes)[FD.SDI]

[1] Bits with associated symbols defined in FCS have the symbols shown in square brackets. These symbols can be defined for use by a task by means of the FCSBT$ macro. See the *RSX-11M-PLUS and Micro/RSX I/O Operations Manual.*

	Bit 5	Sequential device (DV.SQD,1=yes)[FD.SQD]
	Bit 6	Mass-storage device (DV.MSD,1=yes)
	Bit 7	User-mode diagnostics supported (DV.UMD,1=yes)
	Bit 8	Device supports extended 22-bit UNIBUS controller (DV.EXT,DV.MBC,1=yes)
	Bit 9	Unit software write-locked (DV.SWL,1=yes)
	Bit 10	Input spooled device (DV.ISP,1=yes)
	Bit 11	Output spooled device (DV.OSP,1=yes)
	Bit 12	Pseudo device (DV.PSE,1=yes)
	Bit 13	Device mountable as a communications channel (DV.COM,1=yes)
	Bit 14	Device mountable as a Files-11 device (DV.F11,1=yes)
	Bit 15	Device mountable (DV.MNT,1=yes)
Word 3	Second device-characteristics word	
Word 4	Third device-characteristics word (words 3 and 4 are device driver specific)	
Word 5	Fourth device-characteristics word (normally buffer-size as specified in the MCR SET /BUF or DCL SET TERM/WIDTH command)	

Macro Expansion

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

Local Symbol Definitions

G.LULU Logical unit number (2)

G.LUBA Buffer address (2)

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

G.LUNA Device name (2)

G.LUNU Device unit number (1)

G.LUFB Flags byte (1)

G.LUCW Four device-characteristics words (8)

DSW Return Codes

IS.SUC Successful completion.

IE.ULN Unassigned LUN.

IE.ILU Invalid logical unit number.

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

If a spooled device is found in the redirection chain and the issuing task is not the despooler, the LUN information returned by the Executive is as follows:

Word 0 Name of assigned (spooled) device

Word 1 Unit number of assigned spooled device and flags byte

Word 2 Logical OR of the first device-characteristics word for the intermediate device and the output spool bit (spooled device first characteristics word, bit 11)

Word 3 Spooled device fourth device-characteristics word

Word 4 Not defined

Word 5 Intermediate device standard device buffer size

5.43 Get MCR Command Line

The Get MCR Command Line directive instructs the system to transfer an 80-byte command line to the issuing task.

When a task is installed with a task name of "...tsk" or "tskTn," where "tsk" consists of three alphanumeric characters and n is an octal terminal number, the MCR dispatcher requests the task's execution when you issue the following command from terminal number n:

> tsk command-line

A task invoked in this manner must execute a call to Get MCR Command Line, which results in the entire "command line" following the prompt being placed in an 80-byte command-line buffer. (The MCR dispatcher is described in the *RSX-11M-PLUS MCR Operations Manual.*)

FORTRAN Call

CALL GETMCR (buf[,ids])

Parameters

buf

An 80-byte array to receive the command line

ids

Directive status

Macro Call

GMCR$

Macro Expansion

```
GMCR$
.BYTE   127.,41.        ;GMCR$ MACRO DIC, DPB SIZE = 41(10) WORDS
.BLKW   40.             ;80(10)-CHARACTER MCR COMMAND LINE BUFFER
```

Local Symbol Definitions

G.MCRB MCR command-line buffer (80)

DSW Return Codes

+n Successful completion; n is the number of data bytes transferred, excluding the termination character. The termination character is, however, in the buffer. (If the command line came from a task being spawned, the termination character is the ESC key (33).)

IE.AST No MCR command line exists for the issuing task; that is, the task was not requested by a command line as follows:

> tsk command-line

or the task has already issued the Get MCR Command Line directive.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. The GMCR$S form of the macro is not supplied because the DPB receives the actual command line.
2. The CLI dispatcher processes all lines to:
   - Convert tabs to a single space
   - Convert multiple spaces to a single space
   - Convert lowercase characters to uppercase
   - Remove comments between exclamation points
   - Remove all trailing blanks

   The terminator (RET or ESC) is the last character in the line.
3. If the character before the terminator is a hyphen, there is at least one continuation line present. Therefore, you must issue another GMCR$ directive to obtain the rest of the command line.

5.44 Get Mapping Context

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 through a series of Create Address Window directives. The macro argument specifies the address of a vector that contains one Window Definition Block (WDB) for each window block allocated in the task's header, plus a terminator word.

For each window block in the task's header, the Executive sets up a WDB in the vector as follows:

- If the window block is unused (that is, if it does not correspond to an existing address window), the Executive does not record any information about that block in a WDB. Instead, the Executive uses the WDB to record information about the first block encountered that corresponds to an existing window. In this way, unused window blocks are ignored in the mapping context description returned by the Executive.
- If a window block describes an existing unmapped address window, the Executive fills in the offsets W.NID, W.NAPR, W.NBAS, and W.NSIZ with information sufficient to re-create the window. The window status word W.NSTS is cleared.
- If a window block describes an existing mapped window, 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.

Note that in no case does the Executive modify W.NSRB.

The terminator word, which follows the last WDB filled in, is a word equal to the negative of the total number of window blocks in the task's header. It is thereby possible to issue a TST or TSTB instruction to detect the last WDB used in the vector. The terminating word can also be used to determine the number of window blocks built into the task's header.

When Create Address Window directives are used 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.

FORTRAN Call

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

Parameters

imcx

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*24+1=193.

ids

Directive status

Macro Call

```
GMCX$  wvec
```

GMCX$

Parameter

wvec

The address of a vector of n Window Definition Blocks, followed by a terminator word; n is the number of window blocks in the task's header

Macro Expansion

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

Window Definition Block Parameters

Input parameters:

None

Output parameters:

Array Element	Offset	Meaning
iwdb(1) bits 0-7	W.NID	ID of address window
iwdb(1) bits 8-15	W.NAPR	Base APR of the window
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, if the window is unmapped, no change
iwdb(5)	W.NOFF	Offset, in 32-word blocks, from the start of the region at which mapping begins or, if the window is unmapped, no change
iwdb(6)	W.NLEN	Length, in 32-word blocks, of the area currently mapped within the region or, if the window is unmapped, no change
iwdb(7)	W.NSTS	Bit settings[1] in the window status word (all 0 if the window is not mapped):

Bit	Definition
WS.MAP	1 if the window is mapped
WS.WRT	1 if the window is mapped with write access

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Array Element	Offset	Meaning	
		Bit	**Definition**
		WS.SIS	1 if the window is mapped in supervisor-mode instruction space
		WS.UDS	1 if the window is mapped in user-mode data space
		WS.NBP	1 if the window was created with cache bypass disabled (on RSX–11M–PLUS multiprocessor systems only)
		WS.RCX	1 if cache bypass has been enabled for the current mapping of the window (on RSX–11M–PLUS multiprocessor systems only)

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 Definition

G.MCVA Address of the vector (wvec) containing the Window Definition Blocks and terminator word (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ADP Address check of the DPB or the vector (wvec) failed.

IE.SDP DIC or DPB size is invalid.

Note

Due to the use of WS.RCX to indicate cache-bypass state, you may need to do additional manipulation of the WDB before you issue a CRAW$ or MAP$ directive (on RSX–11M–PLUS multiprocessor systems only).

GPRT$

5.45 Get Partition Parameters

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

FORTRAN Call

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

Parameters

prt
Partition name

buf
A 3-word integer array to receive the partition parameters

ids
Directive status

Macro Call

GPRT$ [prt],buf

Parameters

prt
Partition name

buf
Address of a 3-word buffer

Buffer Format

Word 0 Partition physical base address expressed as a multiple of 32 words. (Partitions are always aligned on 32-word boundaries.) Therefore, a partition starting at 40000_8 will have 400_8 returned in this word.

Word 1 Partition size expressed as a multiple of 32 words.

Word 2 Partition flags word. This word is returned equal to 0 to indicate a system-controlled partition or equal to 1 to indicate a user-controlled partition.

Macro Expansion

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

Local Symbol Definitions

G.PRPN Partition name (4)

G.PRBA Buffer address (2)

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

G.PRPB Partition physical base address expressed as an absolute 32-word block number (2)

G.PRPS Partition size expressed as a multiple of 32-word blocks (2)

G.PRFW Partition flags word (2)

DSW Return Codes

Successful completion is indicated by a cleared Carry bit and the starting address of the partition is returned in the DSW. In unmapped systems, the address is physical. In mapped systems, the returned address is virtual and is always zero if it is not the task partition. Unsuccessful completion is indicated by a set Carry bit and one of the following codes in the DSW:

IE.INS Specified partition not in system.

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. For Executives that support the memory management directives, a variation of this directive exists called Get Region Parameters (see next section). When the first word of the 2-word partition name is 0, the Executive interprets the second word of the partition name as a region ID. If the 2-word name is 0,0, it refers to the task region of the issuing task.
2. Omitting the partition-name argument prt returns parameters for the issuing task's unnamed subpartition, not for the system-controlled partition.

GREG$

5.46 Get Region Parameters

The Get Region Parameters directive instructs the Executive to fill an indicated 3-word buffer with region parameters. If a region is not specified, the task region of the issuing task is assumed.

This directive is a variation of the Get Partition Parameters directive for Executives that support the memory management directives.

FORTRAN Call

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

Parameters

rid

Region id

buf

A 3-word integer array to receive the region parameters

ids

Directive status

Macro Call

```
GREG$  [rid],buf
```

Parameters

rid

Region id

buf

Address of a 3-word buffer

Buffer Format

Word 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.

Word 1 Region size expressed as a multiple of 32 words.

Word 2 Region flags word. This word is returned equal to 0 if the region resides in a system-controlled partition or equal to 1 if the region resides in a user-controlled partition.

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
```

Local Symbol Definitions

G.RGID Region ID (2)

G.RGBA Buffer address (2)

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)

DSW Return Codes

Successful completion is indicated by a cleared Carry bit and the starting address of the region is returned in the DSW. In unmapped systems, the returned address is physical. In mapped systems, the returned address is virtual and is always zero if it is not the task region. Unsuccessful completion is indicated by a set Carry bit and one of the following codes in the DSW:

IE.NVR Invalid region ID.

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

GSSW$S

5.47 Get Sense Switches ($S Form Recommended)

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

FORTRAN Call

CALL READSW (isw)

Parameter

isw

Integer to receive the console switch settings

The following FORTRAN call allows a program to read the state of a single switch:

Format

CALL SSWTCH (ibt,ist)

ibt

The switch to be tested (0 to 15)

ist

Test results where:

1 = switch on
2 = switch off

Macro Call

GSSW$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

Successful completion is indicated by a cleared Carry bit and the contents of the console switch register are returned in the DSW. Unsuccessful completion is indicated by a set Carry bit and one of the following codes in the DSW:

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.
2. On RSX-11M-PLUS multiprocessor systems, the value returned is that of the virtual switch register maintained by the MCR SWR command.

GTIM$

5.48 Get Time Parameters

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

FORTRAN Call

CALL GETTIM (ibfp[,ids])

Parameters

ibfp

An 8-word integer array

ids

Directive status

Macro Call

GTIM$ buf

Parameter

buf

Address of an 8-word buffer

Buffer Format

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

Macro Expansion

```
GTIM$   .DATBUF
.BYTE   61.,2           ;GTIM$ DIC, DPB SIZE = 2 WORDS
.WORD   DATBUF          ;ADDRESS OF 8(10)-WORD BUFFER
```

Local Symbol Definition

G.TIBA Buffer address (2)

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

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

DSW Return Codes

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

Note

The format of the time buffer is compatible with that of the buffers used with the Set System Time directive (STIM$).

GTSK$

5.49 Get Task Parameters

The Get Task Parameters directive instructs the system to fill an indicated 16-word buffer with parameters relating to the issuing task.

FORTRAN Call

CALL GETTSK (buf[,ids])

Parameters

buf

A 16-word integer array to receive the task parameters

ids

Directive status

Macro Call

GTSK$ buf

Parameter

buf

Address of a 16-word buffer

Buffer Format

Word 0	Issuing task's name (first half) in Radix-50
Word 1	Issuing task's name (second half) in Radix-50
Word 2	Partition name (first half) in Radix-50
Word 3	Partition name (second half) in Radix-50
Word 4	Undefined in RSX-11M-PLUS and Micro/RSX systems (this word exists for compatibility with RSX-11D and IAS systems)
Word 5	Undefined in RSX-11M-PLUS and Micro/RSX systems (this word exists for compatibility with RSX-11D and IAS systems)
Word 6	Run priority
Word 7	User Identification Code (UIC) of issuing task (in a multiuser protection system, the task's default UIC)[1]
Word 10	Number of logical I/O units (LUNs)
Word 11	Processor model number
Word 12	Undefined in RSX-11M-PLUS and Micro/RSX systems (this word exists for compatibility with RSX-11D and IAS systems)

[1]See note in RQST$ description on contents of words 07 and 17.

Word 13 (Address of task SST vector tables)[2]

Word 14 (Size of task SST vector table in words)[2]

Word 15 Size (in bytes) either of task's address window 0 in mapped systems or, if a task is running under I- and D-space, word 15 contains the size of window 1, the D-space root

Word 16 System on which task is running:

- 0 for RSX-11D
- 1 for RSX-11M
- 2 for RSX-11S
- 3 for IAS
- 4 for RSTS
- 5 for VAX-11 RSX
- 6 for RSX-11M-PLUS and Micro/RSX
- 7 for RT-11 Single Job Monitor
- 10 for RT-11 Foreground/Background and Extended Memory Monitor
- 11 for P/OS

Word 17 Protection UIC (in multiuser system, the login UIC)[1]

---

[1] See note in RQST$ description on contents of words 07 and 17.

[2] Words 13 and 14 will contain valid data if word 14 is not zero. If word 14 is zero, the contents of word 13 are meaningless.

Macro Expansion

```
GTSK$   DATBUF
.BYTE   63.,2           ;GTSK$ MACRO DIC, DPB = 2 WORDS
.WORD   DATBUF          ;ADDRESS OF 18-WORD BUFFER
```

Local Symbol Definition

G.TSBA Buffer address (2)

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

G.TSTN Task name (4)

G.TSPN Partition name (4)

G.TSPR Priority (2)

G.TSGC UIC group code (1)

G.TSPC UIC member code (1)

G.TSNL Number of logical units (2)

GTSK$

G.TSVA	Task's SST vector address (2)
G.TSVL	Task's SST vector length in words (2)
G.TSTS	Task size (2)
G.TSSY	System on which task is running (2)
G.TSDU	Protection UIC (2)

DSW Return Codes

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

5.50 Map Address Window

The Map Address Window directive maps an existing window to 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, as follows:

- The window size specified when the window was created
- 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. (Since the Executive returns the actual length mapped as an output parameter, 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), as follows:

- 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.
- If WS.64B = 1, the task can align on 32-word boundaries; you 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 implementations of RSX emulators. Therefore, you should write applications adaptable to either alignment requirement. The bit setting of WS.64B could be a parameter chosen at assembly time (by means of a prefix file), at task-build time (as input to the GBLDEF option), or at run time (by means of command input or by means of the G.TSSY field returned from the GTSK$ directive).

FORTRAN Call

CALL MAP (iwdb[,ids])

Parameters

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids

Directive status

MAP$

Macro Call

MAP$ wdb

Parameter

wdb
Window Definition Block address

Macro Expansion

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

Window Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
iwdb(1) bits 0-7	W.NID	ID of the window to be mapped.
iwdb(4)	W.NRID	ID of the region to which the window is to be mapped or 0 if the task region 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[1] in the window status word:

	Bit	Definition
	WS.BPS	Bypass cache unconditional (for RSX–11M–PLUS multiprocessor systems and systems that have the conditional assembly parameter C$$CBP defined in RSXMC.MAC; C$$CBP is also included in the RL02 ID and MICROD pregenerated systems)
	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

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Output parameters:

Array Element	Offset	Meaning
iwdb(6)	W.NLEN	Length of the area within the region actually mapped by the window
iwdb(7)	W.NSTS	Bit settings[1] in the window status word:

Bit	Definition
WS.UNM	1 if the window was unmapped first

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Local Symbol Definition

M.APBA Window Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.PRI Privilege violation.

IE.NVR Invalid region ID.

IE.NVW Invalid address window ID.

IE.ALG 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.HWR Region had a parity error or a load failure.

IE.ITS WS.RES was set and region is not resident.

IE.ADP Part of the DPB or WDB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. When the Map Address Window directive is issued, the task may be blocked until the region is loaded.
2. Bit WS.RES in word W.NSTS of the Window Definition Block, when set, specifies that the region should be mapped only if the region is resident.

MRKT$

5.51 Mark Time

The Mark Time directive instructs the system to declare a significant event after an indicated time interval. The interval begins when the task issues the directive; however, task execution continues during the interval. If an event flag is specified, the flag is cleared when the directive is issued and set when the significant event occurs. If an AST entry-point address is specified, an AST (see Section 2.3.3) occurs at the time of the significant event. When the AST occurs, the task's PS, PC, directive status, Wait-for mask words, and the event flag number specified in the directive are pushed onto the issuing task's stack. If neither an event flag number nor an AST service entry point is specified, the significant event still occurs after the indicated time interval. See the Notes.

FORTRAN Calls

Format

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

Parameters

efn
Event flag number

tmg
Time interval magnitude (see Note 5)

tnt
Time interval unit (see Note 5)

ids
Directive status

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

Format

CALL WAIT (tmg,tnt[,ids])

Parameters

tmg
Time interval magnitude (see Note 5)

tnt
Time interval unit (see Note 5)

ids
Directive status

Macro Call

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

Parameters

efn

Event flag number

tmg

Time interval magnitude (see Note 5)

tnt

Time interval unit (see Note 5)

ast

AST entry-point address

Macro Expansion

```
MRKT$   52.,30.,2,MRKAST
.BYTE   23.,5           ;MRKT$ MACRO DIC, DPB SIZE = 5 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52

.WORD   30.             ;TIME MAGNITUDE=30(10)
.WORD   2               ;TIME UNIT=SECONDS
.WORD   MRKAST          ;ADDRESS OF MARK TIME AST ROUTINE
```

Local Symbol Definitions

M.KTEF Event flag (2)

M.KTMG Time magnitude (2)

M.KTUN Time unit (2)

M.KTAE AST entry-point address (2)

DSW Return Codes

For CALL MARK and MRKT$:

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory.

IE.ITI Invalid time parameter.

IE.IEF Invalid event flag number (EFN <0, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

For CALL WAIT:

The following positive error codes are returned for ISA calls:

1	Successful completion
2	Insufficient dynamic storage
3	Specified task not installed
94	Invalid time parameters
98	Invalid event flag number
99	Part of DPB out of task's range
100	DIC or DPB size invalid

Notes

1. Mark Time requires dynamic memory for the clock queue entry.
2. If an AST entry-point address is specified, the AST service routine is entered with the task's stack in the following state:

SP+10	Event-flag mask word[1]
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)

   [1]The event-flag mask word preserves the Wait-for conditions of a task prior to AST entry. A task can, after an AST, return to a Wait-for state. Because these flags and the other stack data are in the user task, they can be modified. Such modification is strongly discouraged, however, since the task can easily fault on obscure conditions. For example, clearing the mask word results in a permanent Wait-for state.

   The event flag number must be removed from the task's stack before an AST Service Exit directive is executed.
3. If the directive is rejected, the specified event flag is not guaranteed to be cleared or set. Consequently, if the task indiscriminately executes a Wait-for directive and the Mark Time directive is rejected, the task may wait indefinitely. Care should always be taken to ensure that the directive was completed successfully.
4. If a task issues a Mark Time directive that specifies a common or group global event flag and then exits before the indicated time has elapsed, the event flag is not set.
5. The Executive returns the code IE.ITI (or 94) in the Directive Status Word if the directive specifies an invalid time parameter. The time parameter consists of two components: the time interval magnitude and the time interval unit, represented by the arguments tmg and tnt, respectively.

A legal magnitude value (tmg) is related to the value assigned to the time interval unit (tnt). The unit values are encoded as follows:

For an ISA FORTRAN call (CALL WAIT):

0 Ticks. A tick occurs for each clock interrupt and is dependent on the type of clock installed in the system.

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

For a programmable clock, a maximum of 1000 ticks per second is available (the exact rate is determined during system generation).

1 Milliseconds. The subroutine converts the specified magnitude to the equivalent number of system clock ticks. On systems with line-frequency clocks, millisecond Mark Time requests can only be approximations.

For all other FORTRAN and macro calls:

1 Ticks. See definition of ticks above.

For both types of FORTRAN calls and all macro calls:

2 Seconds

3 Minutes

4 Hours

The magnitude (tmg) is the number of units to be clocked. The following list describes the magnitude values that are valid for each type of unit. In no case can the value of tmg exceed 24 hours. The list applies to both FORTRAN and macro calls:

If tnt = 0, 1, or 2, tmg can be any positive value with a maximum of 15 bits.
If tnt = 3, tmg can have a maximum value of 1440_{10}.
If tnt = 4, tmg can have a maximum value of 24_{10}.

6. If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of event flags. The use count is run down in the following cases:
   - The Mark Time event occurs.
   - The Mark Time event is canceled.
   - The issuing task exits with the Mark Time event still on the clock queue.
7. The minimum time interval is 1 tick. If you specify a time interval of 0, it will be converted to 1 tick.

MSDS$

5.52 Map Supervisor D-Space

(RSX–11M–PLUS systems only.) The Map Supervisor D-Space directive allows the issuing task to change the mapping of its supervisor-mode D-space APRs. This directive also provides information about the current mapping of the task's supervisor-mode D-space APRs.

Tasks that do not use data space execute with instruction and data space overmapped. Tasks in which the Task Builder has separated instruction and data space are mapped separately (instruction and data space are not overmapped). The overall mapping structure for these tasks is as follows:

Window 0	Root I-space
Window 1	Task header, stack, and root D-space
Window 2	I-space of the read-only section if a multiuser task; memory-resident overlays if not a multiuser task
Window 3	D-space of the read-only section if a multiuser task; memory-resident overlays if not a multiuser task
Window 4	Memory-resident overlays

. .
. .
. .

When supervisor-mode library code is executing, the supervisor-mode I-space APRs map supervisor-mode instruction space. However, the supervisor-mode D-space APRs normally map user-mode data space. Code that resides in a supervisor-mode library can include data (such as error messages) within its own instruction space. The Map Supervisor D-Space directive allows such code to use the supervisor-mode D-Space APRs to map locations in supervisor-mode instruction space that contain data.

The Map Supervisor D-Space directive allows the issuing task to specify a 7-bit mask that determines the mapping of supervisor-mode D-space APRs. The mask value contains one bit for each APR, starting with APR 1. The bits control the value stored in the supervisor mapping control byte in the task header (H.SMAP).

This mask is stored in the high byte of the parameter. The low byte of the parameter is ignored. Since the high bit of the PSW may be set, the PSW is returned in the low byte. The mask is returned in the high byte. Note that although there are eight APRs, the mask is only seven bits because APR 0 cannot be changed. The mask position in the parameter is identical to the DSW return.

To provide for the case when a supervisor-mode library is being used by some tasks as a user-mode library, this directive does not change the task's mapping when it is issued from user mode. However, the DSW is still returned.

When the directive is successfully executed, the DSW provides information about the task's current mapping and mode. Specifying a negative mask value causes the directive to return information rather than change the mapping.

FORTRAN Call

Not supported

Macro Call

MSDS$ mask

Parameter

mask

A 7-bit mask with one bit corresponding to each supervisor-mode D-space APR. If the bit is set, the APR is mapped to supervisor-mode I-space. If the bit is clear, the APR is mapped to user-mode D-space. The seven bits are specified in bits 8 through 14 of the mask word.

Macro Expansion

```
MSDS$   mask
.BYTE   201.,2          ;MSDS$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   MASK
```

DSW Return Codes

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. When including data in a supervisor-mode library, the library may not overmap APR 0 with the supervisor-mode library. The Executive assumes it has access to the task's DSW regardless of the mode from which a directive is issued. Data must therefore be placed near the end of the library or mapped through a memory-resident overlay to force its mapping into some APR other than 0.

2. In the following example, a supervisor-mode library routine changes its mapping in order to access an error message (which is data):

```
MESSAG:     .ASCIZ      / ERROR IN INPUT DATA/
TST         (R0)        ; CHECK SOME PIECE OF USER DATA
BPL         10$         ; IF PLUS OK

MSDS$S      #100000     ; GET CURRENT STATUS OF MAPPING
MOV         $DSW,R0     ;
MOV         R0,-(SP)    ; SAVE CURRENT STATE FOR RESTORE OF MAPPING STATE

BIS         #400,R0     ; UPDATE MASK TO MAP APR1 TO SUPERVISOR MODE
MSDS$S      R0          ; MAP TO SUPERVISOR I-SPACE

MOV         #MESSAG,R1  ; POINT TO ERROR MESSAGE (WHICH IS DATA)
CALL        ERROR       ; ERROR IS A SUBROUTINE THAT HAS LOCAL ERROR
                        ; MESSAGES IN A SUPERVISOR-MODE LIBRARY
```

MSDS$

```
MOV         (SP)+,R0      ; GET OLD MAPPING STATUS
MSDS$S      R0            ; RESTORE OLD MAPPING STATUS

RETURN                    ; BACK TO USER
```

5.53 Move to/from User/Supervisor I/D-Space

The Move to/from User/Supervisor I/D-Space directive instructs the system to fetch data from a specified location in user-mode or supervisor-mode instruction space or data space, or to write the specified value in the specified location in the specified type of address space.

This directive allows you to access a single word of I-space as data without creating a D-space window. This function is primarily intended for use by debugging aids. Use of this directive in production code is not recommended since the directive is not optimized for performance.

FORTRAN Call

Not supported

Macro Call

MVTS$ action,addr, val
buff

Parameters

action

One of the following:

MV.TUI	Move to user I-space
MV.TUD	Move to user D-space
MV.TSI	Move to supervisor I-space
MV.TSD	Move to supervisor D-space
MV.FUI	Move from user I-space
MV.FUD	Move from user D-space
MV.FSI	Move from supervisor I-space
MV.FSD	Move from supervisor D-space

addr

Address of the location in the task

val

Value to be stored in the location (for the move-to operations)

buff

Buffer to receive the value fetched (for the move-from operations)

MVTS$

Macro Expansion

```
MVTS$   action,addr,val
.BYTE   203.,4          ;MVTS$ MACRO DIC, DPB SIZE = 4 WORDS
.WORD   action          ;THE OPERATION TO BE PERFORMED

.WORD   addr            ;ADDRESS OF THE TASK LOCATION
.WORD   val             ;VALUE TO BE WRITTEN (OR BUFFER IF MOVE-FROM)
```

Local Symbol Definitions

M.VTAC Action code

M.VTAD Address of location in I- or D-space to be moved to or from

M.VTBF Buffer address
or or
M.VTVA Value to be moved

DSW Return Codes

IE.PRI The issuing task does not have write access to the target address.

IE.ADP Part of the DPB is out of the issuing task's address space, the specified address is not mapped, or the buffer or the target address is not in the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.54 Parse FCS

The Parse FCS directive takes a File Control Services string and returns a filled-in parse block.

FORTRAN Call

CALL PRSFCS ([mod],[itbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk,prssz],[dfnbk,dfnsz],[rsmsk][,idsw])

Parameters

mod

Optional modifier for logical name table entries; allowable symbolic values are:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

itbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

prbuf

Array containing the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

prsz

Size (in bytes) of the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsbuf

Array containing the resulting file specification buffer

rssz

Size (in bytes) of the resulting file specification buffer

rslen

Integer to receive the resulting string size

PFCS$

prsblk
Array containing the parse block

prssz
Size (in bytes) of the parse block

dfnbk
Array containing the default name block; dfnbk and dfnsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

dfnsz
Size of the default name block; dfnbk and dfnsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsmsk
Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device is not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

idsw
Integer to receive the Directive Status Word.

Macro Call

PFCS$ [mod],[tbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk],[prssz],[dfnbk],[dfnsz][,rsmsk]

Parameters

mod
Optional modifier for logical name table entries; appropriate values are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

tbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

prbuf

Address of the primary file specification buffer

prsz

Size (in bytes) of the primary file specification buffer

rsbuf

Address of the resulting file specification buffer

rssz

Size (in bytes) of the resulting file specification buffer

rslen

Address of a word to receive the resulting string size

prsblk

Address of the parse block

prssz

Size (in bytes) of the parse block

dfnbk

Address of the default name block

dfnsz

Size of the default name block

rsmsk

Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device is not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

PFCS$

Macro Expansion

```
PFCS$ MOD,TBMSK,LUN,PRBUF,PRSZ,RSBUF,RSSZ,RSLEN,PRSBLK,PRSSZ,DFNBK,DFNSZ,RSMSK
.BYTE     207.,13.          ;PFCS$ MACRO DIC, DPB SIZE = 13(10) WORDS
.BYTE     8.                ;PFCS$ SUBFUNCTION

.BYTE     MOD               ;MODIFIER
.BYTE     LUN               ;LUN TO BE ASSIGNED
.BYTE     TBMSK             ;INHIBIT MASK

.WORD     PRBUF             ;PRIMARY FILE SPECIFICATION ADDRESS
.WORD     PRSZ              ;PRIMARY FILE SPECIFICATION LENGTH

.WORD     RSBUF             ;RESULTING FILE SPECIFICATION BUFFER ADDRESS
.WORD     RSSZ              ;RESULTING FILE SPECIFICATION BUFFER SIZE
.WORD     RSLEN             ;RESULTING FILE SPECIFICATION LENGTH

.WORD     PRSBLK            ;PARSE BLOCK ADDRESS
.WORD     PRSSZ             ;PARSE BLOCK LENGTH

.WORD     DFNBK             ;DEFAULT NAME BLOCK ADDRESS
.WORD     DFNSZ             ;DEFAULT NAME BLOCK LENGTH
.WORD     RSMSK             ;SUPPRESSION MASK
```

Local Symbol Definitions

F.LFUN Subfunction value (1)

F.LMOD Logical name modifier (1)

F.LLUN LUN number (1)

F.LTBL Table inhibit mask (1)

F.LPBF Address of the primary file specification buffer (2)

F.LPSZ Size of the primary file specification buffer in bytes (2)

F.LRBF Address of the resulting file specification buffer (2)

F.LRSZ Size of the resulting file specification buffer in bytes (2)

F.LRLN Length of the resulting file specification (2)

F.LPRS Address of parse block (2)

F.LPRZ Length of parse block (2)

F.LDBF Address of the default name block (2)

F.LDSZ Size of the default name block (2)

F.LMSK Resulting string suppression mask (2)

DSW Return Codes

IS.SUC Successful completion.

IE.IDU Invalid device or unit.

IE.ILU Invalid LUN.

IE.LNF Logical translation failed. The O$STAT word in the parse block contains the subcode for the failure. (See the description of the FSS$ directive.)

IE.LNL LUN in use.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.

IE.SDP DIC or DPB size is invalid.

Notes

The parse block has the following format:

1. O$STAT (status word). Indicates the status of the operation. This field can contain the following values:

   SU$SUC Success

   ER$NOD Error in node name (or imbalanced nodes for $RENAME)

   ER$DEV Bad device or inappropriate device type

   ER$DIR Error in directory name

   ER$FNM Error in file name

   ER$TYP Error in file type

   ER$VER Error in version number

   ER$ESS Expanded string area too short

   ER$XTR Extraneous file detected during parsing

   ER$BEQ Bad logical name equivalence string

   ER$FTB File specification became too big because of logical names

   ER$TRN Too many logical name translations

2. O$FLAG (flag word). The following flags indicate what was found in the file specification:

   FS$NOD Node present

   FS$DEV Device present

   FS$DIR Directory

   FS$QUO Quoted file name present

   FS$NAM File name present

   FS$TYP File type present

   FS$VER File version present

   FS$WCH Wildcard character present

   FS$WDI Wild directory

FS$WNA Wild file name

FS$WTY Wild file type

FS$WVE Wild file version

3. O$NODS: Length of the node specification.
4. O$NODA: Address of the node specification.
5. O$DEVS: Length of the device specification.
6. O$DEVA: Address of the device specification.
7. O$DIRS: Length of the directory specification.
8. O$DIRA: Address of the directory specification.
9. O$NAMS: Length of the file name specification.
10. O$NAMA: Address of the file name specification.
11. O$TYPS: Length of the file type specification.
12. O$TYPA: Address of the file type specification.
13. O$VERS: Length of the file version specification.
14. O$VERA: Address of the file version specification.
15. O$TRLS: Length of the trailing string.
16. O$TRLA: Address of the trailing string.
17. O$ACCS: Length of the access control specification.
18. O$ACCA: Address of the access control specification.
19. O$LTYP (logical type byte). The first element that could be a logical name. This field can contain the following words:

    P.LNON No logical name present

    P.LNAM File name may be a logical name

    P.LDEV Device name may be a logical name

    P.LNOD Node specification may be a logical name
20. O$PLEN: Length of the parse block.

The above offsets are defined by the macro LNBDF$, not by FSS$.

Although the entire parse block is 20 words long, the size of the parse block specified in the call (prssz) determines how much of the block is returned.

5.55 Parse RMS

The Parse RMS directive takes an RMS–11 string and returns a filled-in parse block.

FORTRAN Call

CALL PRSRMS ([mod],[itbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk,prssz],[dfbuf,dfsz],[rsmsk][,idsw])

Parameters

mod

Modifier for logical name table entries; accepted values are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

itbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

prbuf

Array containing the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

prsz

Size (in bytes) of the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsbuf

Array containing the resulting file specification buffer

rssz

Size (in bytes) of the resulting file specification buffer

rslen

Integer to receive the resulting string size

PRMS$

prsblk
Array containing the parse block

prssz
Size (in bytes) of the parse block

dfbuf
Address of the default file specification buffer; dfbuf and dfsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

dfsz
Size of the default file specification buffer; dfbuf and dfsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsmsk
Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device and directory are not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

idsw
Integer to receive the Directive Status Word.

Macro Call

PRMS$ [mod],[tbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk],[prssz],[dfbuf],[dfsz][,rsmsk]

Parameters

mod
Modifier for logical name table entries; accepted values are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

tbmsk
Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun
LUN to be assigned

prbuf
Address of the primary file specification buffer

prsz
Size (in bytes) of the primary file specification buffer

rsbuf
Address of the resulting file specification buffer

rssz
Size (in bytes) of the resulting file specification buffer

rslen
Address of a word to receive the resulting string size

prsblk
Address of the parse block

prssz
Size (in bytes) of the parse block

dfbuf
Address of the default file specification buffer

dfsz
Size (in bytes) of the default file specification buffer

rsmsk
Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device and directory are not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

PRMS$

Macro Expansion

```
PRMS$ MOD,TBMSK,LUN,PRBUF,PRSZ,RSBUF,RSSZ,RSLEN,PRSBLK,PRSSZ,DFBUF,DFSZ,RSMSK
.BYTE     207.,13.          ;PRMS$ MACRO DIC, DPB SIZE = 13(10) WORDS
.BYTE     7.                ;PRMS$ SUBFUNCTION

.BYTE     MOD               ;MODIFIER
.BYTE     LUN               ;LUN TO BE ASSIGNED
.BYTE     TBMSK             ;INHIBIT MASK

.WORD     PRBUF             ;PRIMARY FILE SPECIFICATION ADDRESS
.WORD     PRSZ              ;PRIMARY FILE SPECIFICATION LENGTH

.WORD     RSBUF             ;RESULTING FILE SPECIFICATION BUFFER ADDRESS
.WORD     RSSZ              ;RESULTING FILE SPECIFICATION BUFFER SIZE
.WORD     RSLEN             ;RESULTING FILE SPECIFICATION LENGTH

.WORD     PRSBLK            ;PARSE BLOCK ADDRESS
.WORD     PRSSZ             ;PARSE BLOCK LENGTH

.WORD     DFBUF             ;DEFAULT FILE SPECIFICATION ADDRESS
.WORD     DFSZ              ;DEFAULT FILE SPECIFICATION LENGTH
.WORD     RSMSK             ;SUPPRESSION MASK
```

Local Symbol Definitions

R.LFUN Subfunction value (1)

R.LMOD Logical name modifier (1)

R.LLUN LUN number (1)

R.LTBL Table inhibit mask (1)

R.LPBF Address of the primary file specification buffer (2)

R.LPSZ Size of the primary file specification buffer in bytes (2)

R.LRBF Address of the resulting file specification buffer (2)

R.LRSZ Size of the resulting file specification buffer in bytes (2)

R.LRLN Length of the resulting file specification (2)

R.LPRS Address of parse block (2)

R.LPRZ Length of parse block (2)

R.LDBF Address of the default file specification buffer (2)

R.LDSZ Size of the default file specification buffer in bytes (2)

R.LMSK Resulting string suppression mask (2)

DSW Return Codes

IS.SUC Successful completion.

IE.IDU Invalid device or unit.

IE.ILU Invalid LUN.

IE.LNF Logical translation failed. The O$STAT word in the parse block contains the subcode for the failure. (See the description of the FSS$ directive.)

IE.LNL LUN in use.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.

IE.SDP DIC or DPB size is invalid.

Notes

1. The parse block of this directive is the same as that listed for the Parse FCS (PFCS$) directive.
2. Although the entire parse block is 20 words long, the size of the parse block specified in the call (prssz) determines how much of the block is returned.

QIO$

5.56 Queue I/O Request

The Queue I/O Request directive instructs the system to place an I/O request for an indicated physical device unit into a queue of priority-ordered requests for that device unit. The physical device unit is specified as a logical unit number (LUN) assigned to the device.

The Executive declares a significant event when the I/O transfer completes. If the directive call specifies an event flag, the Executive clears the flag when the request is queued and sets the flag when the significant event occurs.

The I/O status block is also cleared when the request is queued and is set to the final I/O status when the I/O request is complete. If an AST service routine entry-point address is specified, the AST occurs upon I/O completion, and the task's Wait-for mask word, PS, PC, DSW, and the address of the I/O status block are pushed onto the task's stack.

The description below deals solely with the Executive directive. The device-dependent information can be found in the *RSX-11M-PLUS and Micro/RSX I/O Drivers Reference Manual.* See the Notes.

FORTRAN Call

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

Parameters

fnc

I/O function code[1]

lun

Logical unit number

efn

Event flag number

pri

Priority (ignored, but parameter must be present in call)

isb

A 2-word integer array to receive final I/O status

prl

A 6-word integer array containing device-dependent parameters to be placed in parameter words 1 through 6 of the DPB. Fill in this array by using the GETADR routine (see Section 1.5.2.4).

ids

Directive status

Macro Call

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

[1] I/O function code definitions are included in the *RSX-11M-PLUS and Micro/RSX I/O Drivers Reference Manual.*

Parameters

fnc

I/O function code[1]

lun

Logical unit number

efn

Event flag number

pri

Priority (ignored, but Q.IOPR byte must be present in DPB)

isb

Address of I/O status block

ast

Address of entry point of AST service routine

prl

Parameter list of the form <P1,...P6>

Macro Expansion

```
QIO$    IO.RVB,7,52.,,IOSTAT,IOAST,<IOBUFR,512.>
.BYTE   1,12.           ;QIO$ MACRO DIC, DPB SIZE = 12(10) WORDS
.WORD   IO.RVB          ;FUNCTION=READ VIRTUAL BLOCK

.WORD   7               ;LOGICAL UNIT NUMBER 7
.BYTE   52.,0           ;EFN 52(10), PRIORITY IGNORED

.WORD   IOSTAT          ;ADDRESS OF 2-WORD I/O STATUS BLOCK
.WORD   IOAST           ;ADDRESS OF I/O AST ROUTINE
.WORD   IOBUFR          ;ADDRESS OF DATA BUFFER

.WORD   512.            ;BYTE COUNT=512(10)

.WORD   0               ;ADDITIONAL PARAMETERS...
.WORD   0               ;...NOT USED IN...
.WORD   0               ;...THIS PARTICULAR...
.WORD   0               ;...INVOCATION OF QUEUE I/O
```

Local Symbol Definitions

Q.IOFN I/O function code (2)

Q.IOLU Logical unit number (2)

Q.IOEF Event flag number (1)

Q.IOPR Priority (1)

[1] I/O function code definitions are included in the *RSX-11M-PLUS and Micro/RSX I/O Drivers Reference Manual.*

Q.IOSB	Address of I/O status block (2)
Q.IOAE	Address of I/O-done AST entry point (2)
Q.IOPL	Parameter list (six words) (12)

DSW Return Codes

IS.SUC	Successful completion.
IE.UPN	Insufficient dynamic memory.
IE.ULN	Unassigned LUN.
IE.HWR	Device driver not loaded.
IE.PRI	Task other than despooler attempted a write-logical-block operation.
IE.ILU	Invalid LUN.
IE.IEF	Invalid event flag number (EFN <0, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).
IE.ADP	Part of the DPB or I/O status block is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

Notes

1. If the directive call specifies an AST entry-point address, the task enters the AST service routine with its stack in the following state:

SP+10	Event-flag mask word
SP+06	PS of task prior to AST
SP+04	PC of task prior to AST
SP+02	DSW of task prior to AST
SP+00	Address of I/O status block, or zero if none was specified in the QIO directive

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

2. If the directive is rejected, the specified event flag is not guaranteed to be cleared or set. Consequently, if the task indiscriminately executes a Wait-for or Stop-for directive and the QIO directive is rejected, the task may wait indefinitely. Care should always be taken to ensure that the directive was completed successfully.

3. Tasks (or regions) cannot normally be checkpointed with I/O outstanding for the following reasons:

   - If the QIO directive results in a data transfer, the data transfers directly to or from the user-specified buffer.
   - If an I/O status block address is specified, the directive status is returned directly to the I/O status block.

The Executive waits until a task has no outstanding I/O before initiating checkpointing in all cases except the one described below.

On systems that support buffered I/O, drivers that buffer I/O check for the following conditions for a task:

- That the task is checkpointable
- That checkpointing is enabled

If these conditions are met, the driver and/or the Executive buffers the I/O request internally and the task is checkpointable with this outstanding I/O. If the task also entered a Wait-for state when the I/O was issued (see the QIOW$ directive) or subsequently enters a Wait-for state, the task is stopped. Any competing task waiting to be loaded into the partition can checkpoint the stopped task, regardless of priority. If the stopped task is checkpointed, the Executive does not bring it back into memory until the stopped state is terminated by completion of buffered I/O or satisfaction of the Wait-for condition.

Not all drivers buffer I/O requests. The terminal driver is an example of one that does.

4. Any task that is linked to a common (read-only) area can issue QIO write requests from that area.
5. If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - The I/O is completed.
   - The I/O is killed by reassigning the specified LUN with the ALUN$ directive.
   - The I/O is killed by issuing the IO.KIL function for the specified LUN.
   - The task exits before I/O is completed.

5.57 Queue I/O Request and Wait

The Queue I/O Request and Wait directive is identical to the Queue I/O Request directive in all but one aspect: when the Wait variation of the directive specifies an event flag, the Executive automatically effects a Wait for Single Event Flag directive.

Consult the description of the Queue I/O Request directive for a definition of the parameters, the local symbol definitions, the DSW return codes, and explanatory notes.

FORTRAN Call

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

Parameters

fnc

I/O function code[1]

lun

Logical unit number

efn

Event flag number

pri

Priority (ignored, but parameter must be present in call)

isb

A 2-word integer array to receive final I/O status

prl

A 6-word integer array containing device-dependent parameters to be placed in parameter words 1 through 6 of the DPB

ids

Directive status

Macro Call

QIOW$ fnc,lun,[efn],[pri],[isb],[ast][,prl]

Parameters

fnc

I/O function code[1]

lun

Logical unit number

efn

Event flag number

[1] I/O function code definitions are included in the *RSX-11M-PLUS and Micro/RSX I/O Drivers Reference Manual.*

pri
Priority (ignored, but Q.IOPR byte must be present in DPB)

isb
Address of I/O status block

ast
Address of entry point of AST service routine

prl
Parameter list of the form <P1,...P6>

Macro Expansion

```
QIOW$   IO.RVB,7,52.,,IOSTAT,IOAST,<IOBUFR,512.>
.BYTE   3,12.           ;QIOW$ MACRO DIC, DPB SIZE = 12(10) WORDS
.WORD   IO.RVB          ;FUNCTION=READ VIRTUAL BLOCK

.WORD   7               ;LOGICAL UNIT NUMBER 7
.BYTE   52.,0           ;EFN 52(10), PRIORITY IGNORED

.WORD   IOSTAT          ;ADDRESS OF 2-WORD I/O STATUS BLOCK
.WORD   IOAST           ;ADDRESS OF I/O AST ROUTINE
.WORD   IOBUFR          ;ADDRESS OF DATA BUFFER

.WORD   512.            ;BYTE COUNT=512(10)

.WORD   0               ;ADDITIONAL PARAMETERS...
.WORD   0               ;...NOT USED IN...
.WORD   0               ;...THIS PARTICULAR...
.WORD   0               ;...INVOCATION OF QUEUE I/O
```

RCST$

5.58 Receive Data or Stop

The Receive Data or Stop directive instructs the system to dequeue a 13-word data block for the issuing task. The data block was queued for the task with a Send Data directive or a Send, Request, and Connect directive.

A 2-word task name of the sender (in Radix–50 format) and the 13-word data block are returned in an indicated 15-word buffer. The task name is contained in the first two words of the buffer.

If no data has been sent, the issuing task is stopped. In this case, the sender task is expected to issue an Unstop directive after sending data. A success status code of IS.SUC indicates that a packet has been received. A success status code of IS.SET indicates that the task was stopped and has been unstopped. The directive must be reissued to retrieve the packet.

When a slave task issues the Receive Data or Stop directive, it assumes the UIC (if it has no outstanding group global event flag context) and TI: of the task that sent the data.

FORTRAN Call

```
CALL RCST  ([rtname],ibuf[,idsw])
```

Parameters

rtname

Sender task name (if not specified, data may be received from any task)

ibuf

Address of a 15-word buffer to receive the sender task name and data

idsw

Integer to receive the Directive Status Word

Macro Call

```
RCST$  [tname],buf
```

Parameters

tname

Sender task name (if not specified, data may be received from any task)

buf

Address of a 15-word buffer to receive the sender task name and data

Macro Expansion

```
RCST$   ALPHA,TSKBUF
.BYTE   139.,4          ;RCST$ MACRO DIC, DPB SIZE = 4 WORDS
.RAD50  ALPHA           ;DATA SENDER TASK NAME
.WORD   TSKBUF          ;BUFFER ADDRESS
```

Local Symbol Definitions

R.CSTN Task name (4)

R.CSBF Buffer address (2)

DSW Return Codes

IS.SUC Successful completion.

IS.SET No data was received and the task was stopped. (Note that the task must be unstopped before it can see this status.)

IE.RSU The issuing task is a slave task with a group global context active, and the next packet received would have changed the task's group number.

IE.AST The issuing task is at AST state.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

On all Micro/RSX systems and those RSX–11M–PLUS systems that support variable send and receive directives (by means of the secondary-pool-support system generation option), the Receive Data or Stop directive is treated as a 13_{10}-word Variable Receive Data or Stop (VRCS$) directive.

RCVD$

5.59 Receive Data

The Receive Data directive instructs the system to dequeue a 13-word data block for the issuing task. The data block has been queued (FIFO) for the task by a Send Data directive.

A 2-word task name of the sender (in Radix–50 format) and the 13-word data block are returned in an indicated 15-word buffer. The task name is contained in the first two words of the buffer.

When a slave task issues the Receive Data directive, it assumes the UIC (if it has no outstanding group global event flag context) and TI: of the task that sent the data.

FORTRAN Call

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

Parameters

tsk

Sender task name (if not specified, data may be received from any task)

buf

A 15-word integer array for received data

ids

Directive status

Macro Call

RCVD$ [tsk],buf

Parameters

tsk

Sender task name (if not specified, data may be received from any task)

buf

Address of a 15-word buffer

Macro Expansion

```
RCVD$   ALPHA,DATBUF    ;TASK NAME AND BUFFER ADDRESS
.BYTE   75.,4           ;RCVD$ MACRO DIC, DPB SIZE = 4 WORDS

.RAD50  /ALPHA/         ;SENDER TASK NAME
.WORD   DATBUF          ;ADDRESS OF 15(10)-WORD BUFFER
```

Local Symbol Definitions

R.VDTN Sender task name (4)

R.VDBA Buffer address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ITS No data currently queued.

IE.RSU The issuing task is a slave task with a group global context active, and the next packet to be received would have changed the task's group number.

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. On all Micro/RSX systems and those RSX-11M-PLUS systems that support variable send and receive directives (by means of the secondary-pool-support system generation option), the Receive Data directive is treated as a 13_{10}-word Variable Receive Data (VRCD$) directive.
2. If the sending task specifies a common or group global event flag in the Send Data directive, the receiving task may use that event flag for synchronization. However, between the time the receiver issues this directive and then issues its next instruction, the sender can send data and set the event flag. If the next instruction is an Exit directive, any data sent during this time will be lost because the Executive flushes the task's receive list as part of exit processing. Therefore, use the Exit If directive or the Receive Data or Exit directive in order to avoid the race condition.

RCVX$

5.60 Receive Data or Exit

The Receive Data or Exit directive instructs the system to dequeue a 13-word data block for the issuing task. The data block has been queued (FIFO) for the task by a Send Data directive.

A 2-word task name of the sender (in Radix–50 format) and the 13-word data block are returned in an indicated 15-word buffer. The task name is contained in the first two words of the buffer.

If no data has been sent, a task exit occurs. To prevent the possible loss of send packets, you should not rely on I/O rundown to take care of any outstanding I/O or open files. The task should assume this responsibility.

When a slave task issues the Receive Data or Exit directive, it assumes the UIC (if it has no outstanding group global event flag context) and TI: of the task that sent the data.

FORTRAN Call

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

Parameters

tsk

Sender task name (if not specified, data may be received from any task)

buf

A 15-word integer array for received data

ids

Directive status

Macro Call

RCVX$ [tsk],buf

Parameters

tsk

Sender task name (if not specified, data may be received from any task)

buf

Address of a 15-word buffer

Macro Expansion

```
RCVX$   ALPHA,DATBUF     ;TASK NAME AND BUFFER ADDRESS
.BYTE   77.,4            ;RCVX$ MACRO DIC, DPB SIZE = 4 WORDS
```

```
.RAD50  /ALPHA/         ;SENDER TASK NAME
.WORD   DATBUF          ;ADDRESS OF 15(10)-WORD BUFFER
```

Local Symbol Definitions

R.VXTN Sender task name (4)

R.VXBA Buffer address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.RSU The issuing task is a slave task with a group global context active, and the next packet to be received would have changed the task's group number.

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. A FORTRAN program that issues the RECOEX call must first close all files by issuing CLOSE calls. See the *VAX FORTRAN User's Guide* or the *PDP-11 FORTRAN-77 User's Guide* for instructions concerning how to ensure that such files are closed properly if the task exits.

   To avoid the time overhead involved in the closing and reopening of files, the task should first issue the RECEIV call. If the directive status indicates that no data was received, then the task can close all files and issue the call to RECOEX. The following example illustrates the same overhead saving in MACRO-11:

```
RCVBUF: .BLKW   15.             ; Receive buffer

START:  RCVX$C  ,RCVBUF         ; Attempt to receive message
        CALL    OPEN            ; Call user subroutine to open files

PROC:             .
                  .
        Process packet of data
                  .
                  .

        RCVD$C  ,RCVBUF         ; Attempt to receive another message
        BCC     PROC            ; If CC successful receive

        CALL    CLOSE           ; Call user subroutine to close files
                                ; and prepare for possible task exit
        JMP     START           ; Make one last attempt at receiving
```

2. If no data has been sent—that is, if no Send Data directive has been issued—the task exits. Send packets may be lost if a task exits with outstanding I/O or open files (see third paragraph of directive description).

3. The Receive Data or Exit directive is useful in avoiding a possible race condition that can occur between two tasks communicating by the Send and Receive directives. The race condition occurs when one task executes a Receive directive and finds its receive queue empty, but before the task can exit, the other task sends it a message. The message is lost because the Executive flushes the receiver task's receive queue when it exits. This condition can be avoided by the receiving task's executing a Receive Data or Exit directive. If the

receive queue is found to be empty, a task exit occurs before the other task can send any data. Thus, no loss of data can occur.

4. On exit, the Executive frees task resources. In particular, the Executive:
   - Detaches all attached devices
   - Flushes the AST queue and despecifies all specified ASTs
   - Flushes the receive and receive-by-reference queues
   - Flushes the clock queue for outstanding Mark Time requests for the task
   - Closes all open files (files open for write access are locked)
   - Detaches all attached regions, except in the case of a fixed task
   - Runs down the task's I/O
   - Deaccesses the group global event flags for the task's group
   - Disconnects from interrupts
   - Flushes all outstanding CLI command buffers for the task
   - Returns a success status (EX$SUC) to any parent tasks
   - Marks for deallocation all virtual terminal units the task has created
   - Breaks the connection with any offspring tasks
   - Frees the task's memory if the exiting task was not fixed
5. If the task exits, the Executive declares a significant event.
6. On all Micro/RSX systems and those RSX-11M-PLUS systems that support variable send and receive directives (by means of the secondary-pool-support system generation option), the Receive Data or Exit directive is treated as a 13_{10}-word Variable Receive Data or Exit (VRCX$) directive.

5.61 Read All Event Flags

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

Note

This directive does not return group global event flags (event flags 65-96).

FORTRAN Call

A FORTRAN task can read only one event flag. The call is:

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

Parameters

efn

Event flag number

ids

Directive status

The Executive returns the status codes IS.SET (+02) and IS.CLR (00) for FORTRAN calls in order to report event-flag polarity.

Macro Call

```
RDAF$  buf
```

Parameter

buf

Address of a 4-word buffer

Buffer Format

Word 0 Task local flags 1-16

Word 1 Task local flags 17-32

Word 2 Task common flags 33-48

Word 3 Task common flags 49-64

Macro Expansion

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

Local Symbol Definition

R.DABA Buffer address (2)

RDAF$

DSW Return Codes

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

5.62 Read Event Flag

The Read Event Flag directive tests an indicated event flag and reports its polarity in the Directive Status Word.

FORTRAN Call

CALL READEF (iefn[,ids])

Parameters

iefn

Integer containing an event flag number

ids

Integer variable to receive the Directive Status Word

Macro Call

RDEF$ efn

Parameter

efn

Event flag number

Macro Expansion

```
RDEF$   6               ;RDEF$ MACRO DIC, DPB SIZE = 2 WORDS
.BYTE   37.,2           ;VARIABLE TO RECEIVE DSW
.WORD   6
```

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.DEEF Event flag number (length - 2 bytes)

DSW Return Codes

IS.CLR Flag was clear.

IS.SET Flag was set.

IE.IEF Invalid event flag number (event flag number <1 or >96).

IE.ADP Part of DPB is out of issuing task's address space.

IE.SDP DIC or DPB size is invalid.

RDXF$

5.63 Read Extended Event Flags

The Read Extended Event Flags directive instructs the system to read all local, common, and group global event flags for the issuing task and record their polarity in a 96-bit (6-word) buffer.

FORTRAN Call

A FORTRAN task can read only one event flag. The call is:

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

Parameters

efn

Event flag number

ids

Directive status

The Executive returns the status codes IS.SET (+02) and IS.CLR (00) for FORTRAN calls in order to report event-flag polarity.

Macro Call

```
RDXF$  buf
```

Parameter

buf

Address of a 6-word buffer

Buffer Format

Word 0	Task local flags 1-16
Word 1	Task local flags 17-32
Word 2	Task common flags 33-48
Word 3	Task common flags 49-64
Word 4	Task group global flags 65-80
Word 5	Task group global flags 81-96

Macro Expansion

```
RDXF$   FLGBUF
.BYTE   39.,3           ;RDXF$ MACRO DIC, DPB SIZE = 3 WORDS
.WORD   FLGBUF          ;ADDRESS OF 6-WORD BUFFER
```

Local Symbol Definition

R.DABA	Buffer address (2)

DSW Return Codes

IS.SUC	Successful completion.
IS.CLR	Group global event flags do not exist. Words 4 and 5 of the buffer contain zero.
IE.ADP	Part of the DPB or buffer is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

5.64 Recursive Translation of Logical Name

The Recursive Translation of Logical Name directive returns an equivalent string from a succession of logical translations of equivalence-name strings for the original user-specified logical name.

The RCTLON and RLON$ calls are the preferred calls to use on RSX–11M–PLUS and Micro/RSX operating systems. The RCTLOG and RLOG$ calls are provided for compatibility with the P/OS operating system. See the Note.

FORTRAN Calls

```
CALL RCTLON  ([mod],[itbmsk],[status],lns,lnssz,iens,ienssz,[rsize],[rtbmod][,idsw])
CALL RCTLOG  ([mod],[itbmsk],[status],lns,lnssz,iens,ienssz,[rsize],[rtbmod][,idsw])
```

Parameters

mod

Optional modifier of the logical name within a table. Ordinarily, no value would be specified to allow any defined logical name to be found.

itbmsk

Inhibit mask to prevent a logical name table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status

Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns

Character array containing the logical name string

lnssz

Size (in bytes) of the logical name string

iens

Character array buffer to receive the returned equivalence name string

ienssz
Size (in bytes) of the data area for the returned equivalence name string

rsize
Word to receive the size of the equivalence name string

rtbmod
Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

idsw
Integer to receive the Directive Status Word

Macro Calls

RLON$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]
RLOG$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]

Parameters

mod
Optional modifier to be matched against the logical name within a table. Ordinarily, no value would be specified to allow any logical in table to be found.

tbmsk
Inhibit mask to prevent a logical name table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status
Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns
Character array containing the original logical name string

lnssz
Size (in bytes) of the original logical name string

RLON$, RLOG$

ens
Character array buffer to receive the returned equivalence name string

enssz
Size (in bytes) of the data area for the returned equivalence name string

rsize
Word to receive the size of the equivalence name string

rtbmod
Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

Macro Expansion

```
RLON$ MOD,TBMSK,LNS,STATUS,LNSSZ,ENS,ENSSZ,RSIZE,RTBMOD
.BYTE     207.,10.          ;RLON$ MACRO DIC, DPB SIZE = 10(10) WORDS
.BYTE     14.               ;SUBFUNCTION VALUE (RLOG$ = 10(10))

.BYTE     MOD               ;LOGICAL NAME MODIFIER
.WORD     TBMSK             ;LOGICAL NAME TABLE INHIBIT MASK

.WORD     LNS               ;LOGICAL NAME STRING ARRAY

.WORD     STATUS            ;LOCATION OF LOGICAL NAME STATUS

.WORD     LNSSZ             ;SIZE (IN BYTES) OF LOGICAL NAME STRING

.WORD     ENS               ;RETURNED EQUIVALENCE NAME ARRAY
.WORD     ENSSZ             ;SIZE (IN BYTES) OF EQUIVALENCE NAME

.WORD     RSIZE             ;LOCATION OF SIZE FOR RETURNED EQUIVALENCE NAME

.WORD     RTBMOD            ;LOCATION OF LOGICAL TABLE NUMBER (LOWER BYTE) AND
                            ;MODIFIER VALUE OF LOCATED LOGICAL NAME (HIGHER BYTE)
```

Local Symbol Definitions

R.LENS Address of buffer for returned equivalence name (2)

R.LESZ Byte count of buffer for returned equivalence name (2)

R.LFUN Subfunction value (1)

R.LLNS Address of logical name string (2)

R.LLSZ Size (in bytes) of specified logical name (2)

R.LMOD Logical name modifier (1)

R.LRSZ Word for returned equivalence name size (2)

R.LRTM Word for returned table number and modifier (2)

R.LSTS Status returned from final logical translation (2)

R.LTBL Table inhibit mask (2)

DSW Return Codes

IS.SUC	Successful completion.
IE.ITN	Invalid table number specified.
IE.LNF	The specified logical name string was not found.
IE.ADP	Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.
IE.SDP	DIC or DPB size is invalid.

Note

The RCTLON and RLON$ calls are the preferred calls to use on RSX-11M-PLUS and Micro/RSX operating systems. The RCTLOG and RLOG$ calls are provided for compatibility with the P/OS operating system. When you use RCTLOG or RLOG$, the system performs the following actions:

- If a device name or node name ends with one or more colons, strips off one to two of the terminating colons.
- If a physical device name string is in the form ddnnn:, compresses any leading zeros. For example, DR005: becomes DR5.

RMAF$S

5.65 Remove Affinity ($S Form Recommended)

(RSX–11M–PLUS multiprocessor systems only.) The Remove Affinity directive removes the task's CPU affinity that was previously established by issuing a Set Affinity directive. Note that only the $S form is available for this directive.

FORTRAN Call

CALL RMAF [(idsw)]

Parameter

idsw

Integer to receive the Directive Status Word

Macro Call

RMAF$S

Macro Expansion

```
RMAF$S
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK

.BYTE   163.,1          ;RMAF$S MACRO DIC, DPB SIZE = 1 WORD
EMT     377             ;TRAP TO EXECUTIVE
```

Local Symbol Definitions

None

DSW Return Codes

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

Note

A task that is installed with task affinity must not issue this directive. Any attempt to do so results in an IE.ITS error returned.

5.66 Request and Pass Offspring Information

The Request and Pass Offspring Information directive instructs the system to request the specified task and to chain to it by passing any or all of the parent connections from the issuing task to the requested task. Optionally, the directive can pass a command line to the requested task. Only a privileged or CLI task may specify the UIC and TI: of the requested task.

FORTRAN Call

CALL RPOI (tname,[iugc],[iumc],[iparen],[ibuf],[ibfl],[isc],[idnam],[iunit],[itask],[ocbad][,idsw])

Parameters

tname

Name of an array containing the actual name (in Radix–50) of the task to be requested and optionally chained to

iugc

Name of an integer containing the group code number for the UIC of the requested target chain task

iumc

Name of an integer containing the member code number for the UIC of the requested target chain task

iparen

Name of an array (or I*4 integer) containing the Radix–50 name of the parent task. This is returned in the information buffer of the GTCMCI subroutine.

ibuf

Name of an array containing the command line text for the chained task

ibfl

Name of an integer containing the number of bytes in the command in the ibuf array

isc

Flag byte controlling the actions of this directive request when executed. The bit definitions of this byte (only the low-order byte of the integer specified in the call is ever used) are as follows:

RP.OEX	128.	Force this task to exit on successful execution of the RPOI$ directive.
RP.OAL	1	Pass all of this task's connections to the requested task (default is pass none).

Note

You cannot pass all connections if the target task is a CLI task.

RP.ONX	2	Pass the first connection in the queue, if there is one.

RPOI$

idnam
Name of an integer containing the ASCII name of the requested task's TI: (must be the name of a physical device)

iunit
Name of an integer containing the unit number of the requested task's TI:

itask
Name of an array containing the Radix–50 name the requested task is to run under.

Any task may specify a new name for the requested task as long as the requested task is not a CLI task.

The requested task (specified in the tname parameter) must be installed in the ...tsk format.

ocbad
Name of an integer containing the pool address of the parent OCB. This value may be obtained only in the information buffer of the GTCMCI subroutine, which only a CLI can issue. Therefore, only a CLI can specify this argument.

idsw
Name of an integer to receive the Directive Status Word

Macro Call

RPOI$ tname,,,[ugc],[umc],[parent],[bufadr],[buflen],[sc],[dnam],[unit],[task],[ocbad]

Parameters

tname
Name of the task to be chained to

ugc
Group code for the UIC of the requested task

umc
Member code for the UIC of the requested task

parent
Name of issuing task's parent task whose connection is to be passed

bufadr
Address of buffer to be given to the requested task

buflen
Length of buffer to be given to the requested task

sc

Flag bits controlling the execution of this directive. The flag bits are defined as follows:

RP.OEX	(200)	Force issuing task to exit.
RP.OAL	(1)	Pass all connections (default is pass none).

Note

You cannot pass all connections if the target task is a CLI task.

RP.ONX	(2)	Pass the first connection in the queue, if there is one.

dnam

ASCII name for TI: (must be the name of a physical device)

unit

Unit number of task TI:

task

Radix–50 name that the requested task is to run under. Any task may specify a new name for the requested task as long as the requested task is not a CLI task.

The requested task (specified in the tname parameter) must be installed in the ...tsk format.

ocbad

Address of OCB to pass (CLIs only)

Local Symbol Definitions

R.POTK	Radix–50 name of the task to be chained to (4)
R.POUM	UIC member code (1)
R.POUG	UIC group code
R.POPT	Name of parent whose OCB should be passed (4)
R.POOA	Address of OCB to pass (CLIs only) (2)
R.POBF	Address of command buffer (2)
R.POBL	Length of command (2)
R.POUN	Unit number of task TI: (1)
R.POSC	Flags byte (1)
R.PODV	ASCII device name for TI: (2)
R.POTN	Radix–50 name of task to be started (4)

RPOI$

Macro Expansion

```
RPOI$   tname,,,ugc,umc,ptsk,buf,buflen,sc,dev,unit,task,ocbad
.BYTE   11.,16.         ;RPOI$ MACRO DIC, DPB SIZE = 16(10) WORDS
.RAD50  /tname/         ;NAME OF TASK TO CHAIN TO

.BLKW   3               ;RESERVED

.BYTE   umc             ;UIC MEMBER CODE
.BYTE   ugc             ;UIC GROUP CODE

.RAD50  /ptsk/          :NAME OF TASK WHOSE OCB SHOULD BE PASSED
.WORD   ocbad           ;ADDRESS OF OCB

.WORD   buf             ;ADDRESS OF BUFFER TO SEND
.WORD   buflen          ;LENGTH OF BUFFER

.BYTE   unit            ;UNIT NUMBER OF TI: DEVICE
.BYTE   sc              ;PASS BUFFER AS SEND PACKET OR COMMAND CODE

.ASCII  /dev/           ;ASCII NAME OF TI: OF REQUESTED TASK
.RAD50  /task/          ;NAME THAT REQUESTED TASK IS TO RUN UNDER
```

DSW Return Codes

IE.UPN There was insufficient dynamic memory to allocate an Offspring Control Block, command-line buffer, Task Control Block, or Partition Control Block.

IE.INS The specified task was not installed, or it was a CLI but no command line was specified.

IE.ACT The specified task was already active and it was not a command line interpreter.

IE.IDU The specified virtual terminal unit does not exist or was not created by the issuing task.

IE.ITS A task that is not a CLI specified a CLI-only parameter or specified passing all connections to a CLI.

IE.NVR There is no Offspring Control Block from the specified parent task.

IE.ALG A CLI specified a parent name and an Offspring Control Block address that did not describe the same connection, or either a parent name or an Offspring Control Block address was specified and the pass-all-connections flag or the pass-next-connection flag was set.

IE.PNS The Task Control Block cannot be created in the same partition as its prototype.

IE.ADP Part of the DPB, exit status block, or command line is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.67 Request Task

The Request Task directive instructs the system to activate a task. The task is activated and subsequently runs contingent upon priority and memory availability. The Request Task directive is the basic mechanism used by running tasks to initiate other installed (dormant) tasks. The Request Task directive is a frequently used subset of the Run directive. See the Notes.

FORTRAN Call

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

Parameters

tsk

Task name

opt

A 4-word integer array containing the following information:

opt(1) Partition name, first half (ignored, but must be present)

opt(2) Partition name, second half (ignored, but must be present)

opt(3) Priority (ignored, but must be present)

opt(4) User Identification Code

ids

Directive status

Macro Call

RQST$ tsk,[prt],[pri][,ugc,umc]

Parameters

tsk

Task name

prt

Partition name (ignored, but must be present)

pri

Priority (ignored, but must be present)

ugc

UIC group code

umc

UIC member code

RQST$

Macro Expansion

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

Local Symbol Definitions

R.QSTN Task name (4)

R.QSPN Partition name (4)

R.QSPR Priority (2)

R.QSGC UIC group (1)

R.QSPC UIC member (1)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory.

IE.INS Task is not installed.

IE.ACT Task is already active.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. The requested task must be installed in the system.
2. If the partition in which a requested task is to run is already occupied, the Executive places the task in a queue of tasks waiting for that partition. The requested task then runs, depending on priority and resource availability, when the partition is free. Another possibility is that checkpointing may occur. If the current occupant or occupants of the partition are checkpointable, have checkpointing enabled, and are of lower priority than the requested task, they are written to disk when their current outstanding I/O completes; the requested task is then read into the partition.
3. Successful completion means that the task has been declared active, not that the task is actually running.
4. The requested task acquires the same TI: terminal assignment as that of the requesting task.
5. The requested task always runs at the priority specified in its task header.
6. A task that executes in a system-controlled partition requires dynamic memory for the Partition Control Block used to describe its memory requirements.

7. On systems that support multiuser protection, each active task has two UICs: a protection UIC and a default UIC. These are both returned when a task issues a Get Task Parameters directive (GTSK$). The UICs are used in the following ways:

   - The protection UIC determines the task's access rights for opening files and attaching to regions. When a task attempts to open a file, the system compares the task's protection UIC against the protection mask of the specified UFD. The comparison determines whether the task is to be considered for system, owner, group, or world access.
   - The default UIC is used by the File Control Services (FCS) to determine the default UFD when a file-open operation does not specify a directory and, on Micro/RSX systems, if there is no default directory string. (The default UIC has no significance when a task attaches to a region.)

   On multiuser protection systems, each terminal also has a protection UIC and a default UIC. If a terminal is nonprivileged, the protection UIC is the login UIC and the default UIC is the UIC specified in the last SET /UIC command issued. If no SET /UIC command has been issued, the default UIC is equal to the login UIC. If the terminal is privileged, both the protection and the default UICs are equal either to the UIC specified in the last SET /UIC command or to the login UIC if a SET /UIC command has not been issued.

   The system establishes a task's UICs when the task is activated. In general, when the MCR dispatcher or the MCR or DCL RUN command activates a task, the task assumes the protection and default UICs of the issuing terminal. However, if you specify the /UIC keyword to the MCR or DCL INSTALL or RUN command, the specified UIC becomes the default UIC for the activated task; and if the issuing terminal is privileged, the specified UIC becomes the activated task's protection UIC as well.

   The system establishes UICs in the same manner when one task issues a Request directive to activate another task. The protection and default UICs of the issuing task generally become the corresponding UICs of the requested task. However, if a nonprivileged task specifies a UIC in a Request directive, the specified UIC becomes only the default UIC for the requested task. If a privileged task specifies a UIC in a Request directive, the specified UIC becomes both the protection and default UIC for the requested task.

8. On RSX-11M-PLUS systems, if you are using named directory support, the requested task acquires the same default directory string as that of the requesting task. This string is used by the File Control Services (FCS) when a file-open operation does not specify a directory.

RREF$

5.68 Receive By Reference

The Receive By Reference directive requests the Executive to dequeue the next packet in the receive-by-reference queue of the issuing (receiver) task. Optionally, the task will exit if there are no packets in the queue. The directive may also specify that the Executive proceed to map the region referred to.

If successful, the directive declares a significant event.

Each reference in the task's receive-by-reference queue represents a separate attachment to a region. If a task has multiple references to a given region, it is attached to that region the corresponding number of times. Because region attachment requires system dynamic memory, the receiver task should detach from any region that it was already attached to in order to prevent depletion of the memory pool. That is, the task needs to be attached to a given region only once.

If the Executive does not find a packet in the queue and the task has set WS.RCX in the window status word (W.NSTS), the task exits. If WS.RCX is not set, the Executive returns the DSW code IE.ITS.

If the Executive finds a packet, 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 passed by the sender task (if any) are placed in the specified buffer. If the sender task did not pass on 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 to 1, the Executive transfers control to the Map Address Window directive to attempt to map the reference.

When a task that has unreceived packets in its receive-by-reference queue exits or is removed, the Executive removes the packets from the queue and deallocates them. Any related flags are not set.

FORTRAN Call

```
CALL RREF  (iwdb,[isrb][,ids])
```

Parameters

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

isrb

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.

ids

Directive status

Macro Call

RREF$ wdb

Parameter

wdb

Window Definition Block address

Macro Expansion

```
RREF$   WDBADR
.BYTE   81.,2           ;RREF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   WDBADR          ;WDB ADDRESS
```

Window Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
iwdb(1) bits 0-7	W.NID	ID of an existing window if region is to be mapped
iwdb(7)	W.NSTS	Bit settings[1] in the window status word:

	Bit	Definition
	WS.MAP	1 if received reference is to be mapped
	WS.RCX	1 if task exit desired when no packet 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

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Output parameters:

Array Element	Offset	Meaning
iwdb(4)	W.NRID	Region ID (pointer to attachment description)
iwdb(5)	W.NOFF	Offset word specified by sender task

Array Element	Offset	Meaning
iwdb(6)	W.NLEN	Length word specified by sender task
iwdb(7)	W.NSTS	Bit settings[1] in the window status word:

Bit	Definition
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
WS.RRF	1 if receive was successful
The Executive clears the remaining bits.	

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Local Symbol Definition

R.REBA Window Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IS.HWR Region has incurred a parity error.

IE.ITS No packet found in the receive-by-reference queue.

IE.ADP Address check of the DPB, WDB, or the receive buffer (W.NSRB) failed.

IE.SDP DIC or DPB size is invalid.

5.69 Receive By Reference or Stop

The Receive By Reference or Stop directive requests the Executive to dequeue the next packet in the receive-by-reference queue of the issuing (receiver) task. The task will stop if there are no packets in the queue. The directive may also specify that the Executive proceed to map the region referred to.

If successful, the directive declares a significant event.

Each reference in the task's receive-by-reference queue represents a separate attachment to a region. If a task has multiple references to a given region, it is attached to that region the corresponding number of times. Because region attachment requires system dynamic memory, the receiver task should detach from any region that it was already attached to in order to prevent depletion of the memory pool. That is, the task needs to be attached to a given region only once.

If the Executive finds a packet, 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 passed by the sender task (if any) are placed in the specified buffer. If the sender task did not pass on 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 to 1, the Executive transfers control to the Map Address Window directive to attempt to map the reference.

When a task that has unreceived packets in its receive-by-reference queue exits or is removed, the Executive removes the packets from the queue and deallocates them. Any related flags are not set.

FORTRAN Call

CALL RRST (iwdb,[isrb][,ids])

Parameters

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

isrb

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.

ids

Directive status

Macro Call

RRST$ wdb

RRST$

Parameter

wdb

Window Definition Block address

Macro Expansion

```
RRST$   WDBADR
.BYTE   213.,2          ;RRST$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   WDBADR          ;WDB ADDRESS
```

Window Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
iwdb(1) bits 0-7	W.NID	ID of an existing window if region is to be mapped
iwdb(7)	W.NSTS	Bit setting[1] in the window status word:

	Bit	Definition
	WS.MAP	1 if received reference is to be mapped
iwdb(8)	W.NSRB	Optional address of a 10-word buffer to contain the sender task name and additional information

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Output parameters:

Array Element	Offset	Meaning
iwbd(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[1] in the window status word:

	Bit	Definition
	WS.RED	1 if attached with read access
	WS.WRT	1 if attached with write access
	WS.EXT	1 if attached with extend access

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Array Element	Offset	Meaning	
		Bit	**Definition**
		WS.DEL	1 if attached with delete access
		WS.RRF	1 if receive was successful
		The Executive clears the remaining bits.	

Local Symbol Definition

R.RSBA Window Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IS.HWR Region has incurred a parity error.

IE.ADP Address check of the DPB, WDB, or the receive buffer (W.NSRB) failed.

IE.SDP DIC or DPB size is invalid.

RSUM$

5.70 Resume Task

The Resume Task directive instructs the system to resume the execution of a task that has issued a Suspend directive.

FORTRAN Call

CALL RESUME (tsk[,ids])

Parameters

tsk

Task name

ids

Directive status

Macro Call

RSUM$ tsk

Parameter

tsk

Task name

Macro Expansion

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

Local Symbol Definition

R.SUTN Task name (4)

DSW Return Codes

IS.SUC Successful completion.

IE.INS Task is not installed.

IE.ACT Task is not active.

IE.ITS Task is not suspended.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

Normally, the RSUM$ directive searches in primary pool for the Task Control Block (TCB) of the task to be resumed. On RSX–11M–PLUS and Micro/RSX systems, however, the TCB for the inactive version of a task is kept in secondary pool. RSUM$ does not search secondary pool when it is looking for the TCB of the specified task because it cannot resume a task that is not active. Therefore, when the task is not found in primary pool, RSUM$ returns the error message, "Task is not installed."

RUN$

5.71 Run Task

The Run Task directive causes a task to be requested at a specified future time and, optionally, to be requested periodically. The schedule time is specified in terms of delta time from issuance. If the smg, rmg, and rnt parameters are omitted, the Run directive is the same as the Request directive, except for the following:

- Run causes the task to become active 1 clock tick after the directive is issued.
- The system always sets the TI: device for the requested task to CO:.

See the Notes.

FORTRAN Call

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

Parameters

tsk

Task name

opt

A 4-word integer array:

opt(1)	Partition name, first half (ignored, but must be present)
opt(2)	Partition name, second half (ignored, but must be present)
opt(3)	Priority (ignored, but must be present)
opt(4)	User Identification Code

smg

Schedule delta magnitude

snt

Schedule delta unit (either 1, 2, 3, or 4)

rmg

Reschedule interval magnitude

rnt

Reschedule interval unit

ids

Directive status

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

Format

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

Parameters

tsk
Task name

smg
Schedule delta magnitude

snt
Schedule delta unit (either 0, 1, 2, 3, or 4)

ids
Directive status

Macro Call

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

tsk
Task name

prt
Partition name (ignored, but must be present)

pri
Priority (ignored, but must be present)

ugc
UIC group code

umc
UIC member code

smg
Schedule delta magnitude

snt
Schedule delta unit (either 1, 2, 3, or 4)

rmg
Reschedule interval magnitude

rnt
Reschedule interval unit

RUN$

Macro Expansion

```
RUN$    ALPHA,,,20,10,20.,3,10.,3
.BYTE   17.,11.          ;RUN$ MACRO DIC, DPB SIZE = 11(10) WORDS
.RAD50  /ALPHA/          ;TASK "ALPHA"

.WORD   0,0              ;PARTITION IGNORED
.WORD   0                ;PRIORITY IGNORED

.BYTE   10,20            ;UIC TO RUN TASK UNDER
.WORD   20.              ;SCHEDULE MAGNITUDE=20(10)
.WORD   3                ;SCHEDULE DELTA TIME UNIT=MINUTE (=3)

.WORD   10.              ;RESCHEDULE INTERVAL MAGNITUDE=10(10)
.WORD   3                ;RESCHEDULE INTERVAL UNIT=MINUTE (=3)
```

Local Symbol Definitions

R.UNTN Task name (4)

R.UNPN Partition name (4)

R.UNPR Priority (2)

R.UNGC UIC group code (1)

R.UNPC UIC member code (1)

R.UNSM Schedule magnitude (2)

R.UNSU Schedule unit (2)

R.UNRM Reschedule magnitude (2)

R.UNRU Reschedule unit (2)

DSW Return Codes

For CALL RUN and RUN$:

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory.

IE.ACT Multiuser task name specified.

IE.INS Task is not installed.

IE.PRI Nonprivileged task specified a UIC other than its own.

IE.ITI Invalid time parameter.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

For CALL START:

The following positive error codes are returned for ISA calls:

2	Insufficient dynamic storage.
3	Specified task not installed.
94	Invalid time parameter.
98	Invalid event flag number.
99	Part of DPB is out of task's address space.
100	DIC or DPB size is invalid.

Notes

1. On multiuser protection systems, a nonprivileged task cannot specify a UIC that is not equal to its own protection UIC. A privileged task can specify any UIC.
2. The target task must be installed in the system.
3. If there is not enough room in the partition in which a requested task is to run, the Executive places the task in a queue of tasks waiting for that partition. The requested task will then run, depending on priority and resource availability, when the partition is free. Another possibility is that checkpointing will occur. If the current occupant or occupants of the partition are checkpointable, have checkpointing enabled, are of lower priority than the requested task, or are stopped for terminal input, they will be written to disk when their current outstanding I/O completes. The requested task will then be read into the partition.
4. Successful completion means the task has been made active. It does not mean that the task is actually running.
5. Time Intervals

   The Executive returns the code IE.ITI in the DSW if the directive specifies an invalid time parameter. A time parameter consists of two components: the time interval magnitude and the time interval unit.

   A legal magnitude value (smg or rmg) is related to the value assigned to the time interval unit snt or rnt. The unit values are encoded as follows:

   For an ISA FORTRAN call (CALL START):

0	Ticks. A tick occurs for each clock interrupt and is dependent on the type of clock installed in the system. For a line-frequency clock, the tick rate is either 50 or 60 per second, corresponding to the power-line frequency. For a programmable clock, a maximum of 1000 ticks per second is available. (The exact rate is determined during system generation.)
1	Milliseconds. The subroutine converts the specified magnitude to the equivalent number of system clock ticks.

For all other FORTRAN and all macro calls:

1 Ticks. See definition of ticks above.

For both types of FORTRAN calls and all macro calls:

2 Seconds

3 Minutes

4 Hours

The magnitude is the number of units to be clocked. The following list describes the magnitude values that are valid for each type of unit. In no case can the magnitude exceed 24 hours. The list applies to both FORTRAN and macro calls:

If unit = 0, 1, or 2, the magnitude can be any positive value with a maximum of 15 bits.
If unit = 3, the magnitude can have a maximum value of 1440.
If unit = 4, the magnitude can have a maximum value of 24(10).

6. The schedule delta time is the difference in time from the issuance of the RUN$ directive to the time the task is to be run. This time may be specified in the range from 1 clock tick to 24 hours.
7. The reschedule interval is the difference in time from task initiation to the time the task is to be reinitiated. If this time interval elapses and the task is still active, no reinitiation request is issued. However, a new reschedule interval is started. The Executive will continually try to start a task, wait for the specified time interval, and then restart the task. This process continues until a CSRQ$ (Cancel Scheduled Initiation Requests) directive or an MCR or DCL CANCEL command is issued.
8. Run requires dynamic memory for the clock-queue entry used to start the task after the specified delta time. If the task is to run in a system-controlled partition, further dynamic memory is required for the task's dynamically allocated Partition Control Block (PCB).
9. If optional rescheduling is not desired, then the macro call should omit the arguments rmg and rnt.

5.72 Specify Command Arrival AST

The Specify Command Arrival AST directive instructs the system to enable or disable command arrival ASTs for the issuing CLI task. If command arrival ASTs are enabled, the Executive transfers control to a specified address when commands have been queued to the CLI.

Only CLI tasks can use this AST.

The format of the stack when the AST routine is entered is as follows:

SP+10 Zero since no event flags are involved

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 command buffer just queued

The AST routine must remove the command buffer address from the stack before issuing an ASTX$ directive.

The command buffer address may be used when issuing a GCCI$ directive.

FORTRAN Call

Not supported

Macro Call

SCAA$ [ast]

Parameter

ast

AST service-routine entry point. Omitting this parameter disables command arrival ASTs for the issuing task until the directive is respecified.

Macro Expansion

```
SCAA$   ast
.BYTE   173.,2          ;SCAA$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   ast             ;ADDRESS OF AST ROUTINE
```

Local Symbol Definition

S.CAAE Address of AST routine (2)

DSW Return Codes

IE.ITS ASTs are already not desired.

IE.AST Directive issued from AST state.

IE.PRV Issuing task is not a CLI.

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

5.73 Supervisor Call ($S Form Recommended)

(RSX–11M–PLUS systems only.) The Supervisor Call directive is issued by a task in user mode or supervisor mode to call a supervisor-mode library routine. Returning to the user mode from supervisor-mode routines entered with the SCAL$S directive (macro form) is effected by a completion routine that is executed in supervisor mode. Note that only the $S form is available for this directive.

Note

We strongly suggest using the Task Builder to resolve references to supervisor-mode routines rather than explicitly using the SCAL$S directive. Doing so allows you to take advantage of the CSM (Call Supervisor Mode) instruction, which is used by the Task Builder.

FORTRAN Call

Not supported

Macro Call

SCAL$S saddr,caddr[,err]

Parameters

saddr

Address of the called supervisor-mode routine

caddr

Address of the completion routine for return to the caller

err

Address of error routine (see Section 1.4.3 for more information)

Macro Expansion

```
SCAL$S  SRAD,CRAD,ERR
MOV     CRAD,-(SP)      ;COMPLETION ROUTINE ADDRESS
MOV     SRAD,-(SP)      ;SUPERVISOR ROUTINE ADDRESS

MOV     (PC)+,-(SP)
.BYTE   155.,3.         ;SCAL$ MACRO DIC, DPB SIZE = 3 WORDS

EMT     ^O<377>
BCC     .+6
CALL    ERR
```

Local Symbol Definitions

None

SCAL$S

DSW Return Codes

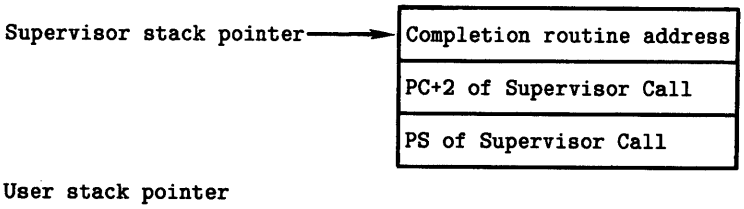
IS.SUC Successful completion.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

This directive transfers control to the specified routine in supervisor mode with all registers preserved and with the following stack:



The stack, as shown, represents the stack content immediately after issuing the Supervisor Call directive. The user stack pointer is not guaranteed to remain valid.

The supervisor stack is the user stack with three words pushed onto it. It is mapped in supervisor data space along with the rest of the user-mode mapping. Previous mode bits are set to the caller's mode. This is normally user mode, but it may be supervisor mode.

If there is insufficient stack space for the three words, the issuing task is aborted.

5.74 Set Command Line Interpreter

The Set Command Line Interpreter directive instructs the system to set up the specified CLI as the CLI for the indicated terminal. The issuing task must be privileged or a CLI.

If the restricted access flag (CP.RST) in the CLI status word is set, the issuing CLI task is the only CLI task that can set a terminal to that CLI.

FORTRAN Call

CALL SETCLI (icli,idev,iunit[,ids])

Parameters

icli
Name of a 2-word array element containing the name of the CLI the terminal is to be set to

idev
Name of an integer containing the ASCII name of the terminal to be set (default = TI:)

iunit
Name of an integer containing the unit number of the terminal

ids
Directive status

Macro Call

SCLI$ cli,[dev],[unit]

Parameters

cli
Name of the CLI the terminal is to be set to

dev
ASCII name of the terminal to be set (default = TI:)

unit
Unit number of terminal

Local Symbol Definitions

S.CIDV ASCII name of the terminal whose CLI is to be set

S.CIUN Octal unit number of terminal

S.CICN Radix-50 name of the CLI that the terminal is to be set to

SCLI$

Macro Expansion

```
SCLI$   cli,dev,unit
.BYTE   173.,5          ;SCLI$ MACRO DIC, DPB SIZE = 5 WORDS
.ASCII  /dev/           ;ASCII NAME OF TERMINAL TO BE SET

.WORD   unit            ;UNIT NUMBER
.RAD50  /cli/           ;CLI NAME
```

DSW Return Codes

IE.PRI Task not privileged or not a CLI. If CP.RST was set, task was not the CLI itself.

IE.IDU Device not a terminal or does not exist.

IE.INS Specified CLI does not exist.

IE.UPN Insufficient dynamic memory.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB length is invalid.

5.75 Send Data

The Send Data directive instructs the system to declare a significant event and to queue (FIFO) a 13-word block of data for a task to receive.

Note

When a local event flag is specified, the flag is set for the sending task.

When a common event flag is specified, the flag is set for all tasks.

When a group global event flag is specified, the flag is set for all tasks within the specified group.

For all event flags, a significant event is always declared.

FORTRAN Call

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

Parameters

tsk
Task name

buf
A 13-word integer array of data to be sent

efn
Event flag number

ids
Directive status

Macro Call

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

Parameters

tsk
Task name

buf
Address of a 13-word data buffer

efn
Event flag number

SDAT$

Macro Expansion

```
SDAT$   ALPHA,DATBUF,52.
.BYTE   71.,5           ;SDAT$ MACRO DIC, DPB SIZE = 5 WORDS
.RAD50  /ALPHA/         ;RECEIVER TASK NAME

.WORD   DATBUF          ;ADDRESS OF 13(10)-WORD BUFFER
.WORD   52.             ;EVENT FLAG NUMBER 52(10)
```

Local Symbol Definitions

S.DATN Task name (4)

S.DABA Buffer address (2)

S.DAEF Event flag number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.INS Receiver task is not installed.

IE.UPN Insufficient dynamic memory.

IE.IEF Invalid event flag number (EFN <0, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB or data block is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. Send Data requires dynamic memory.
2. If the directive specifies a local event flag, the flag is local to the sender (issuing) task. RSX-11M-PLUS and Micro/RSX systems do not allow one task to set or clear a flag that is local to another task.

   Normally, the event flag is used to trigger the receiver task into some action. For this purpose, the event flag must be common (33 through 64) or group global (65 through 96) rather than local. (Refer to the descriptions of the Receive Data directive and the Exit If directive.)
3. The Send Data directive is treated as a 13_{10}-word Variable Send Data (VSDA$) directive.

5.76 Set Default Directory

The Set Default Directory directive establishes, modifies, and deletes the default directory string.

FORTRAN Call

CALL SETDDS (mod,iens,ienssz[,idsw])

Parameters

mod

Modifier for the SDIR$ directive; specify one of the following values:

0	Modify task default
SD.LOG	Modify terminal default
SD.BYE	Delete terminal default
SD.TI	Set task default to terminal default

iens

Character array containing the default directory string

ienssz

Size (in bytes) of the default directory string

idsw

Integer to receive the Directive Status Word

Macro Call

SDIR$ { mod / ,ens,enssz / mod,ens,enssz } (must choose one of these options)

Parameters

mod

Modifier for the SDIR$ directive (must be selected if ,ens,enssz is not); specify one of the following values:

0	Modify task default
SD.LOG	Modify terminal default
SD.BYE	Delete terminal default
SD.TI	Set task default to terminal default

ens

Buffer address of the default directory string; if not specified, the default directory string is deleted (ens and enssz must be selected to modify the default)

enssz

Size (in bytes) of the default directory string (enssz and ens must be selected to modify the default)

Macro Expansion

```
SDIR$ MOD,ENS,ENSSZ
.BYTE   207.,5          ;SDIR$ MACRO DIC, DPB SIZE = 5 WORDS
.BYTE   3               ;SUBFUNCTION CODE FOR SET DEFAULT DIRECTORY

.BYTE   MOD             ;MODIFIER
.WORD   0               ;RESERVED

.WORD   ENS             ;BUFFER ADDRESS OF DEFAULT DIRECTORY STRING
.WORD   ENSSZ           ;BYTE COUNT OF DEFAULT DIRECTORY STRING
```

Local Symbol Definitions

S.DENS Address of default directory string buffer (2)

S.DESZ Byte count of the default directory string (2)

S.DFUN Subfunction code (1)

S.DMOD Modifier (1)

DSW Return Codes

IS.SUC Successful completion of service.

IS.SUP Successful completion of service. A new default directory string superseded a previously specified name string.

IE.LNF Default directory string does not exist.

IE.IBS The length of the default directory string is invalid. The string length must be greater than 0 but less than 12_{10}.

IE.ITN Illegal table number. The reserved word in the DPB was not a zero.

IE.UPN Insufficient dynamic storage is available to create the default directory string.

IE.ADP Part of the DPB or user buffer is out of the issuing task's address space, or you do not have proper access to that region.

IE.SDP DIC or DPB size is invalid, or an illegal subfunction code was specified.

Notes

In addition to the terminal default directory associated with each logged-in terminal, a default directory string is associated with each active task. The default directory string (DDS) is stored in a context block (CTX).

The following rules apply to default directory strings and their context blocks:

- Each logged-in terminal has a default directory string stored in a context block, referred to as the terminal_CTX. The context block is created by HELLO/LOGIN when you log in and is deleted by BYE when you log out. You can change the terminal_CTX by using either the

MCR SET /DEF or DCL SET DEFAULT command. The context block is pointed to from the terminal's Unit Control Block (UCB).

- Each active task has associated with it a default directory string referred to as the task_CTX. Exceptions to this rule are system tasks running from the console terminal (CO:), such as LDR, F11ACP, SHF, and so on. The task_CTX is pointed to from the Task Control Block (TCB).
- When a task is activated from a terminal, the terminal_CTX is propagated to the task_CTX.
- When a task issues the SDIR$ directive, the DDS from the task_CTX is modified. For HELLO/LOGIN and other CLI commands, the SD.LOG modifier should be used to indicate that the DDS in the terminal_CTX is to be modified. For BYE/LOGOUT, the SD.BYE modifier should be used to indicate that the terminal_CTX should be deleted. To set the task_CTX to be the same as the terminal_CTX, the SD.TI modifier should be used.
- When a task spawns an offspring task, the parent's task_CTX is propagated.
- When an entry is inserted into the clock queue for time-based schedule requests from a task, the issuing task's task_CTX is propagated to the clq_CTX (the context block for the clock queue). When an entry is inserted into the clock queue for time-based schedule requests from a terminal CLI command, the issuing terminal's terminal_CTX is propagated to the clq_CTX. When the time expires and the task is activated, the task_CTX is propagated from the clq_CTX.
- When a task sends a packet to a slave task, the sender's task_CTX is propagated to the packet_CTX (the context block for the packet). When the slave task issues a Receive Data (RCVD$) directive to get the packet, the receiver's task_CTX is propagated from the packet_CTX.

SDRC$

5.77 Send, Request, and Connect

The Send, Request, and Connect directive performs a Send Data to the specified task, requests the task if it is not already active, and then connects to the task. The receiver task normally returns status by an Emit Status or Exit with Status directive.

FORTRAN Call

```
CALL SDRC  (rtname,ibuf,[iefn],[iast],[iesb],[iparm][,idsw])
CALL SDRCN  (rtname,ibuf,[iefn],[iast],[iesb],[iparm][,idsw])
```

Parameters

rtname

Target task name of the offspring task to be connected

ibuf

Name of a 13-word send buffer

iefn

Event flag to be set when the offspring task exits or emits status

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL SDRCN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the iefn parameter above.

iparm

Name of a word to receive the status block address when an AST occurs

idsw

Integer to receive the Directive Status Word

Macro Call

```
SDRC$  tname,buf,[efn],[east],[esb]
```

Parameters

tname

Target task name of the offspring task to be connected

buf

Address of a 13-word send buffer

efn

The event flag to be cleared on issuance and set when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the efn parameter above.

Macro Expansion

```
SDRC$   ALPHA,BUFFR,2,SDRCTR,STBLK
.BYTE   141.,7          ;SDRC$ MACRO DIC, DPB SIZE = 7 WORDS
.RAD50  ALPHA           ;TARGET TASK NAME

.WORD   BUFFR           ;SEND BUFFER ADDRESS
.WORD   2               ;EVENT FLAG NUMBER = 2

.WORD   SDRCTR          ;ADDRESS OF AST ROUTINE
.WORD   STBLK           ;ADDRESS OF STATUS BLOCK
```

Local Symbol Definitions

S.DRTN	Task name (4)
S.DRBF	Buffer address (2)
S.DREF	Event flag (2)
S.DREA	AST routine address (2)
S.DRES	Status block address (2)

SDRC$

DSW Return Codes

IS.SUC	Successful completion.
IE.UPN	There was insufficient dynamic memory to allocate a send packet, Offspring Control Block, Task Control Block, or Partition Control Block.
IE.INS	The specified task is an ACP or has the no-send attribute.
IE.IEF	An invalid event flag number was specified (EFN <0, or EFN> 96 if group global event flags exist for the task or EFN> 64 if not).
IE.ADP	Part of the DPB or exit status block is not in the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

Notes

1. If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - Status is returned from the connected task.
   - The issuing task exits before status is returned.
2. The virtual mapping of the exit status block should not be changed while the connection is in effect. Doing so may result in obscure errors.
3. If the directive is rejected, the state of the specified event flag is indeterminate.

5.78 Send Data Request and Pass Offspring Control Block

The Send Data Request and Pass Offspring Control Block directive instructs the system to send a send-data packet for the specified task, chain to the requested task, and request it if it is not already active.

FORTRAN Call

CALL SDRP (task,ibuf,[ibfl],[iefn],[iflag],[iparen],[iocbad][,idsw])

Parameters

task

Name of an array (REAL,INTEGER,I*4) containing the Radix-50 name of the target task

ibuf

Name of an integer array containing the data to be sent

ibfl

Name of an integer containing the number of words (integers) in the array to be sent. This argument may be in the range of 1 to 255_{10}. If this argument is not specified, a default value of 13_{10} is assumed.

iefn

Name of an integer containing the number of the event flag that is to be set when this directive is executed successfully

iflag

Name of an integer containing the flag bits controlling the execution of this directive. They are defined as follows:

SD.REX	128.	Force this task to exit upon successful execution of this directive.
SD.RAL	1	Pass all connections to the requested task (default is pass none). If you specify this flag, do not specify the parent task name.

Note

The target task may not be a CLI task.

SD.RNX	2	Pass the first connection in the queue, if there is one, to the requested task. If you specify this flag, do not specify the parent task name.

iparen

Name of an array containing the Radix-50 name of the parent task whose connection should be passed to the target task. The name of the parent task was returned in the information buffer of the GTCMCI subroutine.

iocbad

Name of an integer containing the pool address of the OCB to pass. This value was returned in the information buffer of the GTCMCI subroutine. Only CLI tasks may specify this parameter.

SDRP$

idsw
Name of an integer to receive the contents of the Directive Status Word

Macro Call

SDRP$ task,bufadr,[buflen],[efn],[flag],[parent],[ocbad]

Parameters

task
Name of the task to be chained to

bufadr
Address of buffer to be given to the requested task

buflen
Length of buffer to be given to the requested task

efn
Event flag number

flag
Flag bits controlling the execution of this directive. The flag bits are defined as follows:

SD.REX	(200)	Force this task to exit upon successful completion of this directive.
SD.RAL	(1)	Pass all connections to the requested task (default is pass none). If you specify this flag, do not specify the parent task name.

Note

The target task may not be a CLI task.

SD.RNX	(2)	Pass the first connection in the queue, if there is one, to the requested task. If you specify this flag, do not specify the parent task name.

parent
Name of issuing task's parent task whose connection is to be passed. If not specified, all connections or no connections are passed, depending on the flag bit.

ocbad
Address of OCB to pass (CLI tasks only)

Macro Expansion

```
SDRP$ TASK,BUFADR,[BUFLEN],[EFN],[FLAG],[PARENT],[OCBAD]
.BYTE   141.,9.          ;SDRP$ MACRO DIC, DPB SIZE = 9(10) WORDS
.RAD50  /TASK/           ;TASK NAME IN RADIX-50

.WORD   BUFADR           ;BUFFER ADDRESS
.BYTE   EFN,FLAG         ;EVENT FLAG, FLAGS BYTE
.WORD   BUFLEN           ;BUFFER LENGTH
```

```
.RAD50  /PARENT/        ;PARENT TASK NAME
.WORD   OCBAD           ;ADDRESS OF OCB
```

Local Symbol Definitions

S.DRTK Radix-50 name of task to be chained to

S.DRAD Send data buffer address

S.DREF Event flag

S.DRFL Flag bits (see above)

S.DRBL Length of send-data packet (up to 256_{10} words)

S.DRPT Name of parent whose OCB should be passed

S.DROA Address of OCB to pass (CLIs only)

DSW Return Codes

IE.ITS A task that is not a CLI specified a CLI-only parameter or attempted to pass all connections to a CLI.

IE.NVR No Offspring Control Block from specified parent.

IE.ALG A CLI specified a parent name and an Offspring Control Block address that did not describe the same connection, or either a parent name or an OCB address was specified and the pass-all-connections flag was set.

IE.IBS Length of send packet is illegal. The send packet may be up to 256_{10} words long.

IE.UPN There was insufficient dynamic memory to allocate a send packet, Offspring Control Block, Task Control Block, or Partition Control Block.

IE.INS The specified task is an ACP or has the no-send attribute.

IE.IEF An invalid event flag number was specified (EFN <0, or EFN> 96 if group global event flags exist or EFN> 64 if not).

IE.ADP Part of the DPB or exit status block is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. If the directive is rejected, the state of the specified event flag is indeterminate.
2. If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - Status is returned from the connected tasks.
   - The issuing task exits before status is returned.

SETF$

5.79 Set Event Flag

The Set Event Flag directive instructs the system to set an indicated event flag, reporting the flag's polarity before setting it.

FORTRAN Call

CALL SETEF (efn[,ids])

Parameters

efn

Event flag number

ids

Directive status

Macro Call

SETF$ efn

Parameter

efn

Event flag number

Macro Expansion

```
SETF$   52.
.BYTE   33.,2           ;SETF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   52.             ;EVENT FLAG NUMBER 52(10)
```

Local Symbol Definition

S.ETEF Event flag number (2)

DSW Return Codes

IS.CLR Flag was clear.

IS.SET Flag was already set.

IE.IEF Invalid event flag number (EFN <1, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

Set Event Flag does not declare a significant event. It merely sets the specified flag.

5.80 Specify Floating Point Processor Exception AST

The Specify Floating Point Processor Exception AST directive instructs the system to record one of the following cases:

- Floating Point Processor exception ASTs for the issuing task are desired, and the Executive is to transfer control to a specified address when such an AST occurs for the task.
- Floating Point Processor exception ASTs for the issuing task are no longer desired.

When an AST service-routine entry-point address is specified, future Floating Point Processor exception ASTs will occur for the issuing task and control will be transferred to the indicated location at the time of the AST's occurrence. When an AST service entry-point address is not specified, future Floating Point Processor exception ASTs will not occur until the task issues a directive that specifies an AST entry point. See the Notes.

FORTRAN Call

Not supported

Macro Call

SFPA$ [ast]

Parameter

ast

AST service-routine entry-point address

Macro Expansion

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

Local Symbol Definition

S.FPAE AST entry address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory.

IE.ITS AST entry-point address is already unspecified or task was built without floating-point support (/FP switch not specified in Task Builder TSK file specification).

IE.AST Directive was issued from an AST service routine or ASTs are disabled.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

SFPA$

Notes

1. A Specify Floating Point Processor Exception AST requires dynamic memory.
2. The Executive queues Floating Point Processor exception ASTs when a Floating Point Processor exception trap occurs for the task. No future ASTs of this kind will be queued for the task until the first one queued has actually been effected (that is, terminated by an ASTX$ directive).
3. The Floating Point Processor exception AST service routine is entered with the task stack in the following state:

   SP+12 Event-flag mask word

   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 task must remove the floating exception code and address from the task's stack before an AST Service Exit directive is executed.
4. This directive cannot be issued either from an AST service routine or when ASTs are disabled.
5. This directive applies only to the Floating Point Processor.

5.81 Send Message

The Send Message directive instructs the system to create and send a formatted data packet to a system-defined target task. The only valid target for the Send Message directive is the Error Logger, and the formatted data packet must be an error log packet. The task that issues the SMSG$ directive must be privileged. The valid system-defined target identifier and its code are as follows:

TARGET INDENTIFIER	CODE
Error logging	SM.SER

FORTRAN Call

CALL SMSG (itgt,ibuf,ibufl,iprm,iprml[,ids])

Parameters

itgt
The name of the integer containing the target object (currently, only SM.SER is defined)

ibuf
The name of an integer array containing the data to be inserted into the formatted data packet

ibufl
The name of an integer containing the length of the ibuf array

iprm
The name of an integer array containing any additional parameters

iprml
The name of an integer containing the number of parameters in the iprm array

ids
The name of an optional integer to receive the directive status

Macro Call

SMSG$ tgt,buf,len, <pri,...,prn>

Parameters

tgt
Target identifier

buf
Address of the optional data buffer

len
Length in bytes of the optional data buffer

SMSG$

pri,...,prn

Target-specific (for the Error Logger) parameter list:

SMSG$ SM.SER,buf,len, <typ,sub,lun,msk>

typ

Error Logger packet type code

sub

Error Logger packet subtype code

lun

Logical unit number of the device

msk

Control mask word

The directive creates an error log packet of the specified type and subtype codes. If you specify a LUN, the directive also records information about the device to which the LUN refers. The control mask word sets flags to zero I/O and error counts on the device specified, as shown below:

Control-mask-word flag:

SM.ZER Zeroes device I/O and error counts for device specified by LUN

The directive also creates the following subpackets and places them in the error log packet in the order listed below:

1. Header subpacket. The header subpacket, which contains the type and subtype codes, the time-stamp, and the system identification, is always recorded.
2. Task subpacket. The task subpacket, which identifies the task that issued the directive, is always recorded.
3. Device subpacket. The device subpacket, which identifies the device, is recorded if the directive specifies a LUN argument.
4. Data subpacket. The data subpacket is recorded if the directive specifies an address and length of an optional data buffer.

Macro Expansion (with Error Logger target)

```
SMSG$   SM.SER,DATBUF,DATLEN,<PR1,PR2,PR3,PR4>
.BYTE   171.,8.          ;SMSG$ MACRO DIC, DPB SIZE = 8(10) WORDS
.WORD   SM.SER           ;TARGET IDENTIFIER - ERROR LOGGING

.WORD   DATBUF           ;DATA BUFFER ADDRESS
.WORD   DATLEN           ;DATA BUFFER LENGTH

.WORD   PR1              ;PARAMETER  1
.WORD   PR2              ;PARAMETER  2
```

```
.WORD   PR3             ;PARAMETER  3
.WORD   PR4             ;PARAMETER  4
```

Local Symbol Definitions

S.MTGT Target identifier (2)

S.MDBA Buffer address (2)

S.MDBL Buffer length (2)

S.MPRL Parameter list

DSW Return Codes

IS.SUC Successful completion.

IE.ILU Invalid LUN (error log target only).

IE.ULN Specified LUN is not assigned to a mass storage device.

IE.UPN Insufficient dynamic memory.

IE.INS Target task is not installed.

IE.ITS Invalid target identifier or invalid control mask.

IE.ADP Part of the DPB or data buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

SNXC$

5.82 Send Next Command

The Send Next Command directive allows a task that is servicing a CLI command to inform the system that the command execution is complete. This normally happens automatically when the task exits. This directive is not necessary if the task will exit when it completes the command; it is intended for tasks that do not exit at this point.

The task of concern here is the final task involved in the command processing. For example, a CLI that passes the command to another task using the RPOI$ or SDRP$ directive and exits need not issue an SNXC$ directive. If the CLI were to do all the processing necessary for a command, not pass it to another task, and go on to the next command, it would have to issue an SNXC$ directive.

Issuing this directive causes a prompt request to be generated if one would have occurred on task exit and will cause the terminal driver to send the next command to the dispatcher if the terminal is in serial-execution mode.

A nonprivileged task may specify only its TI:. A privileged task or a CLI task may specify any terminal. If no terminal is specified, the default is the issuing task's TI:.

FORTRAN Call

CALL SNXC ([dnam][,iunit][,idsw])

Parameters

dnam

Device name (ASCII); if not specified, TI: is used

iunit

Unit number of the terminal from which the command is to be sent

idsw

Integer to receive the Directive Status Word

Macro Call

SNXC$ [dnam][,unum]

Parameters

dnam

Device name (ASCII); if not specified, TI: is used

unum

Unit number of the terminal from which the command is to be sent

Macro Expansion

```
SNXC$   TT,3
.BYTE   127.,3          ;SNXC$ MACRO DIC, DPB SIZE = 3 WORDS

.ASCII  /TT/            ;ASCII DEVICE NAME
.BYTE   3,0             ;UNIT NUMBER IS 3
```

Local Symbol Definitions

S.NXDV Device name

S.NXUN Unit number

DSW Return Codes

IE.IDU The specified device does not exist or is not a terminal.

IE.PRI A nonprivileged task specified a terminal other than its own TI:.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

SPEA$

5.83 Specify Parity Error AST

The Specify Parity Error AST directive enables a task to specify an AST service routine to be entered if a hardware parity error occurs. If an AST address is not specified, any previously specified parity error AST is canceled. Upon entering the AST service routine, the stack contains the following information:

SP+62	Event flag mask word
SP+60	PS of task prior to AST
SP+56	PC of task prior to AST
SP+54	Task's Directive Status Word
SP+52	Contents of memory parity CSRs (hardware-dependent information)
SP+50	
SP+46	
SP+44	
SP+42	
SP+40	
SP+36	
SP+34	
SP+32	
SP+30	
SP+26	
SP+24	
SP+22	
SP+20	
SP+16	
SP+14	
SP+12	Contents of cache-control register
SP+10	Contents of memory system-error register
SP+06	Contents of high-error-address register
SP+04	Contents of low-error-address register
SP+02	Processor identification (single-processor system = 0)
SP+00	Number of bytes to add to SP to clean the stack (52)

FORTRAN Call

Not supported

Macro Call

SPEA$ [ast]

Parameter

ast

AST service-routine entry-point address

Macro Expansion

```
SPEA$   PTYERR
.BYTE   165.,2          ;SPEA$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   PTYERR          ;PARITY ERROR AST ROUTINE ADDRESS
```

Local Symbol Definition

S.PEAE Parity error AST routine address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic storage.

IE.ITS ASTs already not desired.

IE.AST Directive was issued from an AST service routine or ASTs are disabled.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

SPND$S

5.84 Suspend ($S Form Recommended)

The Suspend directive instructs the system to suspend the execution of the issuing task. A task can suspend only itself, not another task. The task can be restarted either by a Resume directive or by an MCR RESUME or DCL CONTINUE command.

FORTRAN Call

CALL SUSPND [(ids)]

Parameter

ids

Directive status

Macro Call

SPND$S [err]

Parameter

err

Error-routine address

Macro Expansion

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

Local Symbol Definitions

None

DSW Return Codes

IS.SPD	Successful completion (task was suspended).
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

Notes

1. A suspended task retains control of the system resources allocated to it. The Executive makes no attempt to free these resources until a task exits.
2. A suspended task is eligible for checkpointing unless it is fixed or declared to be noncheckpointable.
3. Because this directive requires only a 1-word DPB, the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

5.85 Specify Power Recovery AST

The Specify Power Recovery AST directive instructs the system to record one of the following cases:

- Power recovery ASTs for the issuing task are desired and control is to be transferred when a powerfail recovery AST occurs.
- Power recovery ASTs for the issuing task are no longer desired.

When an AST service-routine entry-point address is specified, future power recovery ASTs will occur for the issuing task and control will be transferred to the indicated location at the time of the AST's occurrence. When an AST service entry-point address is not specified, future power recovery ASTs will not occur until an AST entry point is again specified. See the Notes.

FORTRAN Call

To establish an AST:

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

Parameter

sub

Name of a subroutine to be executed upon power recovery. The PWRUP subroutine will effect the following call:

CALL sub (no arguments)

The subroutine is called as a result of a power recovery AST, and therefore may be controlled at critical points by using DSASTR (or INASTR) and ENASTR subroutine calls.

To remove an AST:

```
CALL PWRUP
```

Macro Call

```
SPRA$  [ast]
```

Parameter

ast

AST service-routine entry-point address

Macro Expansion

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

Local Symbol Definition

S.PRAE AST entry address (2)

SPRA$

DSW Return Codes

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

Notes

1. The Specify Power Recovery AST directive requires dynamic memory.
2. The Executive queues power recovery ASTs when the power-up interrupt occurs following a power failure. No future powerfail ASTs will be queued for the task until the first one queued has been effected.
3. The task enters the powerfail AST service routine with the task stack in the following state:

SP+06	Event-flag mask word
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 power recovery AST. Therefore, the AST Service Exit directive can be executed with the stack in the same state as when the AST was entered.
4. This directive cannot be issued either from an AST service routine or when ASTs are disabled.
5. Refer to Chapter 1 for a list of the restrictions on operations that may be performed in a FORTRAN AST routine.

5.86 Spawn

The Spawn directive requests a specified task for execution, optionally queuing a command line and establishing the task's TI: as a previously created virtual terminal unit or as a physical terminal.

When this directive is issued, an Offspring Control Block (OCB) is queued to the offspring TCB and a rundown count is incremented in the parent task's TCB. The rundown count is used to inform the Executive that the task is a parent task and has one or more offspring tasks and virtual terminals; cleanup is necessary if a parent task exits with active offspring tasks. The rundown count is decremented when the spawned task exits. The OCB contains the TCB address as well as sufficient information to effect all of the specified exit events when the offspring task exits.

If a command line is specified, it is buffered in the Executive pool and queued for the offspring task for subsequent retrieval by the offspring task with the Get MCR Command Line directive. The maximum command line length is 255_{10} characters.

If an AST address is specified, an exit AST routine is effected when the spawned task exits with the address of the task's exit status block on the stack. The AST routine must remove this word from the stack before issuing the AST Service Exit directive.

Special action is taken if the task being spawned is a command line interpreter (CLI), such as MCR or DCL. In this case, a command line must be specified, and both the OCB and the command line are queued for the interpreter task. MCR and DCL either handle commands directly or dispatch them to another task. In the case of direct execution of the command, the OCB may be used to immediately effect the proper exit conditions and return exit status by an Executive routine. If MCR or DCL dispatch another task, they simply move the OCB from their own OCB queue directly to the OCB queue of the dispatched task. They also queue the command line for the dispatched task as usual. At this point, the situation is exactly the same as if the SPWN$ directive had specified the dispatched task directly. No exit conditions occur until the dispatched task exits.

FORTRAN Call

CALL SPAWN (rtname,[iugc],[iumc],[iefn],[iast],[iesb],[iparm],[icmlin,icmlen],[iunit],[dnam][,idsw])
CALL SPAWNN (rtname,[iugc],[iumc],[iefn],[iast],[iesb],[iparm],[icmlin,icmlen],[iunit],[dnam][,idsw])

Parameters

rtname
Name (Radix-50) of the offspring task to be spawned

iugc
Group code number for the UIC of the offspring task

iumc
Member code number for the UIC of the offspring task

iefn
Event flag to be set when the offspring task exits or emits status

SPWN$

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL SPAWNN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status, as follows:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the iefn parameter above.

iparm

Name of a word to receive the status block address when the AST occurs

icmlin

Name of a command line to be queued for the offspring task

icmlen

Length of the command line; maximum length is 255_{10}

iunit

Unit number of terminal to be used as the TI: for the offspring task. If the optional dnam parameter is not specified, this parameter must be the unit number of a virtual terminal created by the issuing task. If a value of 0 is specified for the unit number, the TI: of the issuing task is propagated. A task must be a privileged task or must be a CLI task in order to specify a TI: other than the parent task's TI:.

dnam

Device name mnemonic (must be the name of a physical device). On RSX-11M-PLUS and Micro/RSX systems, if not specified, the virtual terminal specified by iunit is used as TI:.

idsw

Integer to receive the Directive Status Word

Macro Call

SPWN$ tname,,,[ugc],[umc],[efn],[east],[esb],[cmdlin,cmdlen],[unum],[dnam]

Parameters

tname

Name (Radix–50) of the offspring task to be spawned

ugc

Group code number for the UIC of the offspring task

umc

Member code number for the UIC of the offspring task

efn

The event flag to be cleared on issuance and set when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status, as follows:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the efn parameter above.

cmdlin

Address of a command line to be queued for the offspring task

cmdlen

Length of the command line; maximum length is 255_{10}

unum

Unit number of terminal to be used as the TI: for the offspring task. If the optional dnam parameter is not specified, this parameter must be the unit number of a virtual terminal created by the issuing task. If a value of 0 is specified for the unit number, the TI: of the issuing task is propagated. A task must be a privileged task or must be a CLI task in order to specify a TI: other than the parent task's TI:.

dnam

Device name mnemonic (must be the name of a physical device). If not specified, the virtual terminal specified by unum is used as TI:.

SPWN$

Macro Expansion

```
SPWN$   ALPHA,,,3,7,1,ASTRUT,STBLK,CMDLIN,72.,2
.BYTE   11.,13.          ;SPWN$ MACRO DIC, DPB SIZE = 13(10) WORDS
.RAD50  ALPHA            ;NAME OF TASK TO BE SPAWNED

.BLKW   3                ;RESERVED
.BYTE   7,3              ;UMC = 7, UGC = 3

.BYTE   1                ;EVENT FLAG NUMBER = 1
.BYTE   16.              ;EXIT STATUS BLOCK CONSTANT

.WORD   ASTRUT           ;AST ROUTINE ADDRESS
.WORD   STBLK            ;EXIT STATUS BLOCK ADDRESS

.WORD   CMDLIN           ;ADDRESS OF COMMAND LINE
.WORD   72.              ;COMMAND LINE LENGTH = 72(10) CHARACTERS

.WORD   2                ;VIRTUAL TERMINAL UNIT NUMBER = 2
```

Note

If a virtual terminal is not specified, one additional parameter (device name) can be added for a hardware terminal name. For example, TT2 (instead of VT2) would have the same macro expansion shown above, plus the following:

```
.ASCII  /TT/     ;ASCII DEVICE NAME
```

The DPB size will then be 14_{10} words.

Local Symbol Definitions

S.PWTN Task name (4)

S.PWXX Reserved (6)

S.PWUM User member code (1)

S.PWUG User group code (1)

S.PWEF Event flag number (2)

S.PWEA Exit AST routine address (2)

S.PWES Exit status block address (2)

S.PWCA Command line address (2)

S.PWCL Command line length (2)

S.PWVT Terminal unit number (2)

S.PWDN Device name (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN There was insufficient dynamic memory to allocate an Offspring Control Block, command line buffer, Task Control Block, or Partition Control Block.

IE.INS The specified task was not installed, or it was a command line interpreter but no command line was specified.

IE.ACT The specified task was already active and it was not a command line interpreter.

IE.PRI A nonprivileged task attempted to specify an offspring task's TI: to be different from its own.

IE.IDU The specified virtual terminal unit does not exist, or it was not created by the issuing task, or the specified TI: device is not a terminal.

IE.IEF Invalid event flag number (EFN <0, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB, exit status block, or command line is out of the issuing task's address space, or the command line is too long.

IE.SDP DIC or DPB size is invalid.

Notes

1. If the UIC is defaulted and the offspring task is not a command line interpreter (CLI), that task is requested to run under the UIC of the parent task. If the UIC is defaulted and the offspring task is a CLI and the CLI passes the specified command line to a dispatched task, the dispatched task will run under the UIC of its TI: terminal. See the notes for the Request Task (RQST$) directive for more information about task UICs.
2. If the specified event flag is group global, then the use count for the event flag's group is incremented to prevent premature elimination of event flags. The use count is run down when the following events occur:
   - Status is returned from the spawned task.
   - The issuing task exits before status is returned.
3. The virtual mapping of the exit status block should not be changed while the connection is in effect. Doing so may cause obscure errors.
4. The types of operations that a FORTRAN AST routine may perform are extremely limited. Refer to Chapter 1 for a list of the restrictions.

The following program illustrates the use of the FORTRAN-callable SPAWN routine and the mechanism for handling ASTs from a FORTRAN program:

```
      P R O G R A M    S P W A S T
C
C This program illustrates the use of the FORTRAN-callable
C SPAWN routine and the use of a FORTRAN subprogram at AST state.

C This example keeps "ITMAX" tasks active at any point in time
C without having several copies of each utility installed under
C different names. The input file consists of single line commands

C of up to 45 characters in length which invoke tasks in the system
C library UIC. The first three characters of the input command line

C are the name of the task to be invoked (ie, MAC).  The output file
C consists of a log file containing the command lines and the exit status
C of the program invoked.
```

```
C
C The above is accomplished as follows:
C
C       A command is read from the input file "CMDFIL.CMD" which has the
C form "NAM COMMAND," where NAM is the name of the task and COMMAND is the
C command to be passed to this task. This input command line is transformed
C into an MCR RUN command line such as
C       RUN $MAC/TASK=TSKnn/EST=NO/CMD="command"
C where nn is a number assigned by this task so that the target task name
C is both known and unique. The MCR dispatcher (MCR...) is spawned with this
C transformed command line, which in turn causes the MCR... task to dispatch
C a copy of ...MCR under the name MCRTnn to execute this command.  When
C this copy of ...MCR exits, an exit AST is serviced by this task which
C issues a "CONNECT" to the target task TSKnn. This method introduces a timing
C window such that the target task could exit before the CONNECT is made. In
C this case, an error message is written to the log file indicating that
C exit status could not be returned due to a connect failure.
C
C This nonprivileged FORTRAN IV-PLUS program is compiled and
C built as follows:
C
C MCR>F4P SPWAST,SPWAST/-SP=SPWAST
C MCR>TKB SPWAST/FP,SPWAST=SPWAST,LB:[1,1]F4POTS/LB
C
C
C Define data structures
C
C  The following variables are kept on a per active "invoked task" basis.
C  For lack of a better name, each respective entry is called a task
C  information block.
C
C  IESTAT(8,XXX)        IEXSAD(XXX)     ISTAT(XXX)      ICMDLN(45,XXX)
C

      PARAMETER ITMAX=3

      COMMON /KOM1/IESTAT(8,ITMAX),IEXSAD(ITMAX),ISTAT(ITMAX),IPARM,RTNAME(2)
      COMMON /KOM2/THISTK(16)
      COMMON /COMMAN/ICMDLN(45,ITMAX)


      INTEGER IESTAT  !exit status array for each task
      INTEGER IEXSAD  !array containing the address of each task's iestat
      INTEGER ISTAT   !array containing the status (active vs free) of
C                      each task information block
      INTEGER IPARM   !contains address of IESTAT at AST state
      INTEGER RTNAME  !contains the Radix-50 name of the target task to be
C                     !connected to at AST state
      INTEGER THISTK
      BYTE ICMDLN     !saved input command line per task
```

```
C
C  Local input buffer variables
C
        DIMENSION INPCOM(3)
        DIMENSION INPBUF(45)
        EQUIVALENCE (INPBUF(1),INPCOM(1))

        BYTE INPBUF     !INPUT BUFFER
        BYTE INPCOM     !COMPONENT NAME FIELD OF INPBUF

C
C  Local variables for SPAWN call
C
        EXTERNAL EXTAST         !define the name of the AST routine externally

        DIMENSION CMDLIN(79)    !maximum command line passed to is 79(10) bytes

        BYTE CMDLIN             !actual command line passed to MCR...
        INTEGER*4 DSPNAM        !variable containing Radix-50 task name of MCR...
        DATA    DSPNAM/6RMCR.../!fill in name of ...MCR at compile time

C
C  Local control variables
C
        INTEGER ITCNT   !count of number of free task information blocks
        LOGICAL EOF     !flag indicating EOF detected on command input file

C
C  Misc. local variables
C
        INTEGER IDSW    !integer to contain directive status

C
C Open files
C
        OPEN (UNIT=1,TYPE='OLD',READONLY,NAME='CMDFIL.CMD')
        OPEN (UNIT=2,TYPE='NEW',CARRIAGECONTROL='FORTRAN',NAME='CMDFIL.LOG')

C
C Initialize Variables
C
        ITCNT=ITMAX+1   !set current count of available task information blocks
        EOF=.FALSE.     !reset EOF flag

        CALL IRAD50(3,'TSK',RTNAME(1)) !setup first half of target task name
        CALL GETTSK(THISTK(1))         !determine this task's name so that
C                                       STOPing and UNSTOPing may be done

C
C Initialize the IEXSAD array such that each entry contains the address
C of the exit status block that has the corresponding index. This is
C necessary so that the correct exit status block may be determined at AST
C state.
C
        DO 5 I=1,ITMAX
        CALL GETADR(IEXSAD(I),IESTAT(1,I))
5       CONTINUE
```

SPWN$

```
C
C Read a command line from the input file and initialize a free task information
C block.
C

10      READ (1,900,END=30)I,INPBUF     !read input command line
        ITCNT=ITCNT-1                   !one less free block

        DO 20 K=1,ITMAX                 !search for the free block
        IF (ISTAT(K) .NE. 0) GOTO 20    !IF NE, block is in use

        ISTAT(K)=1              !ELSE found one, mark it in use
        DO 15 J=1,I             !save command line for output later
        ICMDLN(J,K)=INPBUF(J)

15      CONTINUE
        DO 16 J=I+1,45          !pad saved command line with spaces
        ICMDLN(J,K)="40

16      CONTINUE
        GOTO 40         !exit search loop

20      CONTINUE
30      EOF=.TRUE.      !set EOF flag
        GOTO 55         !continue to log exit status of what's currently
C                       !active

C
C Construct the actual command line specified in the SPAWN call
C

C   Write saved command line to TI: so that any MCR RUN error messages
C   have context.

40      WRITE(5,710)(ICMDLN(J,K),J=1,45)
710     FORMAT(1X,45A1)

        ENCODE(I+35,800,CMDLIN)INPCOM,K,(INPBUF(J),J=1,I)
800     FORMAT('RUN $',3A1,'/TASK=TSK',I1,'/EST=NO/CMD="',45A1)
        CMDLIN(I+32)="42         !add terminating quote
        CMDLIN(I+33)="15         !and terminator

C
C Spawn MCR... with the command line such as:
C
C       RUN $MAC/TASK=TSK1/EST=NO/CMD="MAC TEST1=TEST1"
C

C At this point, the second half of the Radix-50 target task name is calculated
C so that the first exit AST may issue a connect after ...MCR exits.

        RTNAME(2)=40*40*(30*+K) !calculate second half of Radix-50 task name

C Spawn the MCR dispatcher with the constructed command line. The dispatcher
C will then spawn a copy of ...MCR which will in turn process the RUN command.

45      CALL SPAWN(DSPNAM,,,1,EXTAST,IESTAT(1,K),IPARM,CMDLIN,I+33,0,,IDSW)

C  An error could be received from the SPAWN call. This could be due to a
C  variety of reasons, such as the task file specified was not found or there
C  was insufficient system resources at the time the Executive directive
```

```
C  was issued. Only the IE.RSU errors will be recovered by waiting for
C  a significant event and reissuing the call to SPAWN.

        IF(IDSW+1) 50,52,54             !check directive status returned
C
C Spawn error
C
50      IESTAT(1,K)=5                   !if mi, uncorrectable error mark status
        IESTAT(2,K)=IDSW                !save directive status returned for log
        ISTAT(K)=3                      !indicate status present
        GOTO 60                         !go write error to log file and clean up
C
C Spawn error due to insufficient resources
C
52      CALL WFSNE                      !wait for significant event
        GOTO 45                         !reissue SPAWN
C
C Spawn successful, wait till ...MCR exits and first AST has been serviced.
C
54      CALL WAITFR(1)                  !wait for ...MCR to exit
C
C Do not STOP if connect failed, just process task info block and continue.
C
        IF(IESTAT(1,K) .EQ. 6) GOTO 60  !exit status code of 6 indicates
C                                        connect failure
C
C At this point, a check is made to determine whether this task has
C completed its quest. If there is no more input and all task information
C blocks are free, then exit processing will be performed.
C
55      IF(EOF .AND. (ITCNT .EQ. ITMAX+1)) GOTO 500
C
C Next, if all the task information blocks are being used or if there
C is no more input to process, this task is stopped so as to lower its
C priority effectively to zero. This task will once again wake up when
C the connect AST unstops this task.
C
        IF(ITCNT .EQ. 1 .OR. (EOF)) CALL STOP
C
C Scan all the task information blocks to process task information blocks
C now waiting for clean up and log-file processing.
C
60      DO 70 K=1,ITMAX                 !search task information blocks for
C                                        the task(s) that exited
        IF (ISTAT(K) .NE. 3) GOTO 70    !if eq, then offspring task connect AST
C                                        has not occurred for this task
        WRITE (2,901) (ICMDLN(J,K),J=1,45)      !write cmdlin to log file
        GOTO (62,63,64,61,65,66,67),(IESTAT(1,K) .AND. "377)+1 !decode exit status
```

```
61      WRITE (2,902)(IESTAT(1,K) .AND. "377) !unknown exit status
        GOTO 68

62      WRITE (2,903)                         !EX$WAR -- warning
C                                             !or none returned
        GOTO 68

63      WRITE (2,904)                         !EX$SUC -- success
        GOTO 68

64      WRITE (2,905)                         !EX$ERR -- error
        GOTO 68

65      WRITE (2,906)                         !EX$SEV -- severe error
        GOTO 68

66      WRITE (2,907)IESTAT(2,K)              !internal -- SPAWN failure
        GOTO 68

67      WRITE (2,908)IESTAT(2,K)              !internal -- CONNECT failure

68      ISTAT(K)=0                            !free up task information block
        IESTAT(1,K)=0                         !initialize exit status
        ITCNT=ITCNT+1                         !adjust free task info block count

70      CONTINUE
        GOTO 10

900     FORMAT(Q,45A1)
901     FORMAT('$',45A1)

902     FORMAT('+','Unknown exit status =',I3)
903     FORMAT('+','<< Warning')

904     FORMAT('+','<< Success')
905     FORMAT('+','<< Error')

906     FORMAT('+','<< Severe error')

907     FORMAT('+','<< Spawn error, DSW =',I3)
908     FORMAT('+','<< Connect error, DSW =',I3)

C
C Exit cleanly by closing all files

C
500     CLOSE (UNIT=1)          !close input  file on LUN 1
        CLOSE (UNIT=2)          !close output file on LUN 2
        CALL EXIT               !exit

        END


        S U B R O U T I N E    E X T A S T

        PARAMETER ITMAX=3
        COMMON /KOM1/IESTAT(8,ITMAX),IEXSAD(ITMAX),ISTAT(ITMAX),IPARM,RTNAME(2)
        COMMON /KOM2/THISTK(16)

        INTEGER IESTAT  !exit status array for each task
        INTEGER IEXSAD  !array containing the address of each task's IESTAT

        INTEGER ISTAT   !array containing the status (active vs free) of
C                       !each task information block
```

```
        INTEGER IPARM   !contains address of IESTAT at AST state
        INTEGER RTNAME  !contains the Radix-50 name of the target task to be
C                       !connected to at AST state
        INTEGER THISTK
        EXTERNAL TSKEXT
C
C Using IPARM, which contains the address of the exit status block array,
C find the task information block by comparing this with the address of each
C exit status block array (contained in IEXSAD).
C
        DO 10 I=1,ITMAX
        IF (IEXSAD(I) .EQ. IPARM) GOTO 20 !found the task info block
10      CONTINUE
        GOTO 30
20      ISTAT(I)=2                      !indicate ...MCR has exited
C
C Try to connect to the target task:
C
        CALL CNCT(RTNAME(1),2,TSKEXT,IESTAT(1,I),IPARM,IDSW)
        IF(IDSW .EQ. 1) GOTO 30         !if EQ, then successful connect
        IESTAT(1,I)=6                   !else pass connect failed status
        IESTAT(2,I)=IDSW
        ISTAT(I)=3                      !mark task information block as done
30      RETURN                          !return from AST state (returns
                                        !to internal AST handler)
        END

        S U B R O U T I N E   T S K E X T
        PARAMETER ITMAX=3
        COMMON /KOM1/IESTAT(8,ITMAX),IEXSAD(ITMAX),ISTAT(ITMAX),IPARM,RTNAME(2)
        COMMON /KOM2/THISTK(16)
        INTEGER IESTAT  !exit status array for each task
        INTEGER IEXSAD  !array containing the address of each task's IESTAT
        INTEGER ISTAT   !array containing the status (active vs free) of
C                       !each task information block
        INTEGER IPARM   !contains address of IESTAT at AST state
        INTEGER RTNAME  !contains the Radix-50 name of the target task to be
C                       !connected to at AST state
        INTEGER THISTK  !this task's name (so that an UNSTOP may be performed)
C
C Find exit status block:
C
        DO 10 I=1,ITMAX
        IF (IEXSAD(I) .EQ. IPARM) GOTO 20 !found the task information block
10      CONTINUE
        GOTO 30
```

```
20      ISTAT(I)=3                      !indicate AST has been serviced
        CALL USTP(THISTK)               !UNSTOP  this task
30      RETURN                          !return from AST state (returns
                                        !to internal AST handler)
        END
```

5.87 Specify Receive Data AST

The Specify Receive Data AST directive instructs the system to record one of the following cases:

- Receive data ASTs for the issuing task are desired, and the Executive transfers control to a specified address when data has been placed in the task's receive queue.
- Receive data ASTs for the issuing task are no longer desired.

When the directive specifies an AST service-routine entry point, receive data ASTs for the task will subsequently occur whenever data has been placed in the task's receive queue; the Executive will transfer control to the specified address.

When the directive omits an entry-point address, the Executive disables receive data ASTs for the issuing task. Receive data ASTs will not occur until the task issues another Specify Receive Data AST directive that specifies an entry-point address. See the Notes.

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.

Macro Call

SRDA$ [ast]

Parameter

ast

AST service-routine entry-point address

Macro Expansion

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

Local Symbol Definition

S.RDAE AST entry address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory.

IE.ITS AST entry-point address is already unspecified.

IE.AST Directive was issued from an AST service routine or ASTs are disabled.

SRDA$

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. The Specify Receive Data AST directive requires dynamic memory.
2. The Executive queues receive data ASTs when a message is sent to the task. No future receive data ASTs will be queued for the task until the first one queued has been effected.
3. The task enters the receive data AST service routine with the task stack in the following state:

   SP+06 Event-flag mask word

   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 data AST. Therefore, the AST Service Exit directive must be executed with the stack in the same state as when the AST was effected.
4. This directive cannot be issued either from an AST service routine or when ASTs are disabled.

5.88 Specify Requested Exit AST

The Specify Requested Exit AST directive allows the task issuing the directive to specify the AST service routine to be entered if an attempt is made to abort the task by a directive or MCR or DCL ABORT command. This allows a task to enter a routine for cleanup instead of abruptly aborting.

If an AST address is not specified, any previously specified exit AST is canceled.

Privileged tasks enter the specified AST routine each time an abort request is issued. However, subsequent exit ASTs will not be queued until the first exit AST has occurred.

Nonprivileged tasks enter the specified AST routine only once. Subsequent attempts to abort the task will actually abort the task.

SREX$ is the preferred form of this directive. The differences are explained in Notes 1 and 2.

FORTRAN Calls

```
CALL SREA  (ast[,idsw])
```

Parameters

ast
Name of the externally declared AST subroutine

idsw
Name of an optional integer to receive the Directive Status Word

Format

```
CALL SREX  (ast,ipblk,ipblkl,[dummy][,idsw])
```

Parameters

ast
Name of the externally declared AST subroutine

ipblk
Name of an integer array to receive the trap-dependent parameters

ipblkl
Number of parameters to be returned into the ipblk array

dummy
Reserved for future use

idsw
Name of an optional integer to receive the Directive Status Word

Macro Calls

```
SREA$  [ast]
SREX$  [ast][,dummy]
```

SREA$, SREX$

Parameters

ast

AST service-routine entry-point address

dummy

Reserved for future use

Macro Expansions

```
SREA$   REQAST
.BYTE   167.,2          ;SREA$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   REQAST          ;EXIT AST ROUTINE ADDRESS

SREX$   REQAST
.BYTE   167.,3          ;SREX$ MACRO DIC, DPB SIZE = 3 WORDS

.WORD   REQAST          ;EXIT AST ROUTINE ADDRESS
.WORD   0               ;RESERVED FOR FUTURE USE
```

Note

The DPB length for the SREA$ form of the directive is two words. For the SREX$ form of the directive, it is three words.

Local Symbol Definition

S.REAE Exit AST routine address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic storage.

IE.AST Directive was issued from an AST service routine or ASTs are disabled.

IE.ITS ASTs already not desired, or a nonprivileged task attempted to respecify or cancel the AST after one had already occurred.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. The SREX$ form of the directive is recommended for tasks that want to handle all privileged and nonprivileged abortion attempts that do not violate multiuser protection checks. The issuing task can use the information returned on the stack for this version of the directive to decide how to handle the abortion attempt.

   After specifying a requested exit AST using the SREX$ form of the directive, the issuing task will enter the AST service routine if any attempt is made to abort the task. On systems with multiuser protection, nonprivileged abortion attempts must originate from the same TI: as that of the issuing task.

When the AST service routine is entered and the AST has been specified using the SREX$ version of the directive, the task's stack is in the following state:

SP+12 Event-flag mask word

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 Trap-dependent parameter

SP+00 Number of bytes to add to SP to clean stack (4)

The trap-dependent parameter is formatted as follows:

Bit 0 = 0 if the abortion attempt was privileged.
= 1 if the abortion attempt was nonprivileged.

Bit 1 = 0 if the ABRT$ directive was issued.
= 1 if the MCR or DCL ABORT command was used.

Bits 2-15 are reserved for future use.

The task must remove the trap-dependent parameters from the stack before an AST Service Exit directive is executed. The recommended method is to add the value stored in SP+00 to SP. This is also the only recommended way to access the non-trap-dependent parameters on the stack.

2. The SREA$ form of the directive is recommended for privileged tasks that do not want abortion attempts from a nonprivileged user's MCR or DCL ABORT command to be allowed but do not otherwise care about the nature of the abortion attempt. It is also recommended for any nonprivileged tasks that simply do not care about the nature of the abortion attempt.

   After specifying a requested exit AST using the SREA$ form of the directive, privileged tasks will enter the AST service routine if any of the following abortion attempts are made:

   - Any privileged ABRT$ directive or privileged MCR or DCL ABORT command
   - Any nonprivileged ABRT$ directive from the same TI: on systems with multiuser protection

   Nonprivileged tasks will enter the AST service routine for the abortion attempts listed above, plus the following:

   - Any nonprivileged MCR or DCL ABORT command from the same TI: on systems with multiuser protection

   When the AST service routine is entered, the task's stack is in the following state:

   SP+06 Event-flag mask word

   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 an AST specified by SREA$. Therefore, the AST Service Exit directive can be executed with the stack in the same state as when the AST was entered.

3. The event-flag mask word at the bottom of the stack preserves the Wait-for conditions of a task prior to AST entry. A task can, after an AST, return to a Wait-for state. Because these flags and other stack data are in the user task, they can be modified. However, modifying the stack data may cause unpredictable results. Therefore, such modification is not recommended.
4. If an SREX$ requested exit AST is not specified for a task, it is impossible to abort a privileged task from a nonprivileged terminal using either MCR or DCL on systems with multiuser protection.
5. The two forms of this directive should not be mixed in the same code since the stack format and the trap-dependent parameters differ. Any mismatch between the form of the directive and the AST routine will have unpredictable results.
6. See Chapter 1 for a list of restrictions on operations that can be performed in a FORTRAN AST routine.

5.89 Send By Reference

The Send By Reference directive inserts a packet containing a reference to a region into the receive-by-reference queue of a specified (receiver) task. The Executive automatically attaches the receiver task for each Send By Reference directive issued by the task to the specified region (the region identified in W.NRID of the Window Definition Block). The attachment occurs even if the receiver task is already attached to the region unless bit WS.NAT in W.NSTS of the Window Definition Block is set. The successful execution of this directive causes a significant event to occur.

The send packet contains the following information:

- A pointer to the created attachment descriptor, which becomes the region ID to be used by the receiver task
- The offset and length words specified in W.NOFF and W.NLEN of the Window Definition Block (which the Executive passes without checking)
- The receiver task's permitted access to the region, contained in the window status word W.NSTS
- The sender task name
- Optionally, the address of an 8-word buffer that contains additional information (if the packet does not include a buffer address, the Executive sends eight words of zero)

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 zero, the receiver task can default the permitted access to that of the sender task.

If the directive specifies an event flag, the Executive sets the flag in the sender task—when the receiver task acknowledges the reference—by issuing the Receive By Reference or the Receive By Reference or Stop 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.

FORTRAN Call

```
CALL SREF  (tsk,[efn],iwdb,[isrb][,ids])
```

Parameters

tsk

A single-precision, floating-point variable containing the name of the receiving task in Radix-50 format

efn

Event flag number

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

SREF$

isrb

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)

ids

Directive status

Macro Call

SREF$ task,wdb[,efn]

Parameters

task

Name of the receiver task

wdb

Window Definition Block address

efn

Event flag number

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
```

Window Definition Block Parameters

Input parameters:

Array Element	Offset	Meaning
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[1] in window status word (the receiver task's permitted access):

Bit	Definition
WS.RED	1 if read access is permitted
WS.WRT	1 if write access is permitted

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Array Element	Offset	Meaning	
		Bit	Definition
		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

S.RETN Receiver task name (4)

S.REBA Window Definition Block base address (2)

S.REEF Event flag number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN A send packet or an attachment descriptor could not be allocated.

IE.INS The sender task attempted to send a reference to an Ancillary Control Processor (ACP) task, or task not installed.

IE.PRI Specified access not allowed to sender task itself.

IE.NVR Invalid region ID.

IE.IEF Invalid event flag number (EFN <0, or EFN> 96 if group global event flags exist for the task or EFN> 64 if not).

IE.HWR Region had load failure or parity error.

IE.ADP The address check of the DPB, the WDB, or the send buffer failed.

IE.SDP DIC or DPB size is invalid.

Notes

1. For your convenience, 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 argument (efn) is at the end of the macro call. This arrangement is also compatible with the SDAT$ macro.
2. Because region attachment requires system dynamic memory, the receiver task should detach from any region to which it was already attached in order to prevent depletion of the memory pool. That is, the task needs to be attached to a given region only once.

3. If the specified event flag is group global, then the use count for the event flag's group is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - The packet is received.
   - The issuing task exits before the packet is received.

5.90 Specify Receive-By-Reference AST

The Specify Receive-By-Reference AST directive instructs the system to record one of the following cases:

- Receive-by-reference ASTs for the issuing task are desired, and the Executive transfers control to a specified address when such an AST occurs.
- Receive-by-reference ASTs for the issuing task are no longer desired.

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.

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. See the Notes.

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.

Macro Call

SRRA$ [ast]

Parameter

ast

AST service-routine entry-point address (0)

Macro Expansion

```
SRRA$   RECAST
.BYTE   21.,2           ;SRRA$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   RECAST          ;ADDRESS OF RECEIVE AST
```

Local Symbol Definition

S.RRAE AST entry address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic memory.

IE.ITS AST entry-point address is already unspecified.

IE.AST Directive was issued from an AST service routine or ASTs are disabled.

SRRA$

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. The Specify Receive-By-Reference AST directive requires dynamic memory.
2. The Executive queues receive-by-reference ASTs when a message is sent to the task. Future receive-by-reference ASTs will not be queued for the task until the first one queued has been effected.
3. The task enters the receive-by-reference AST service routine with the task stack in the following state:

   SP+06 Event-flag mask word

   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 directive must be executed with the stack in the same state as when the AST was effected.
4. This directive cannot be issued either from an AST service routine or when ASTs are disabled.

5.91 Set Affinity

(RSX-11M-PLUS multiprocessor systems only.) The Set Affinity directive can be issued by a task to select which CPU and UNIBUS runs to use during task execution.

Task CPU/UNIBUS affinity enables a task to select which CPU and UNIBUS runs to use for task execution when running on PDP-11 multiprocessor systems. You must be completely aware of the particular system hardware configuration in which the task will be executed before using these directives.

Task CPU/UNIBUS affinity can be established at three possible times, as follows:

- When the task is installed
- When the task is mapped into a device partition (which must have CPU/UNIBUS run affinity previously established)
- When set by the Set Affinity directive

When issued, the Set Affinity directive produces an affinity mask word that defines task CPU /UNIBUS affinity. One bit in the word is set to select one CPU on which the task will be run. One or more of 12 additional bits can be set to select one or more UNIBUS runs for peripheral device use during task execution.

Two directives support task affinity, as follows:

- Set Affinity. This directive accepts parameters that define the CPU and UNIBUS run mask for task execution. At assembly time, a 1-word mask is created consisting of the logical OR of all the parameters.
- Remove Affinity. This directive removes task CPU/UNIBUS affinity previously established by a Set Affinity directive.

A 1-word CPU/UNIBUS affinity mask defines directive parameters. Parameters enable specification of one of 4 (maximum) CPUs and one or more of 12 (maximum) UNIBUS runs. The affinity mask word consists of the logical OR of all the parameters. Only one parameter (cp or ub) is required. Directive parameters are assembled to produce the mask-word bit values shown as follows:

Directive Parameter	Mask-Word Function	Assembled Bit Value
CPA	Select CPU "A"	1
CPB	Select CPU "B"	2
CPC	Select CPU "C"	4
CPD	Select CPU "D"	10
UBE	Select UNIBUS run "E"	20
UBF	Select UNIBUS run "F"	40
UBH	Select UNIBUS run "H"	100

STAF$

Directive Parameter	Mask-Word Function	Assembled Bit Value
UBJ	Select UNIBUS run "J"	200
UBK	Select UNIBUS run "K"	400
UBL	Select UNIBUS run "L"	1000
UBM	Select UNIBUS run "M"	2000
UBN	Select UNIBUS run "N"	4000
UBP	Select UNIBUS run "P"	10000
UBR	Select UNIBUS run "R"	20000
UBS	Select UNIBUS run "S"	40000
UBT	Select UNIBUS run "T"	100000

FORTRAN Call

CALL STAF (iaff[,idsw])

Parameters

iaff

Affinity mask word

idsw

Integer to receive the Directive Status Word

Macro Call

STAF$ [cp!ub!ub...]

Parameters

cp

CPU selected (A through D, as previously listed)

ub

UNIBUS run or runs selected (E through T, as previously listed)

Macro Expansion

```
STAF$   CPB!UBF!UBJ
.BYTE   161.,2          ;STAF$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   242             ;AFFINITY MASK WORD ('OR' OF PARAMETERS)
```

Local Symbol Definition

S.AFAF Affinity mask word (2)

DSW Return Codes

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

Notes

1. A task that is installed with task affinity must not issue this directive. Any attempt to do so results in an IE.ITS error returned.
2. If this directive is issued with parameters that prevent the task from running, an IE.ITS error is returned.

STIM$

5.92 Set System Time

The Set System Time directive instructs the system to set the system's internal time to the specified time parameters. Optionally, the Set System Time directive returns the system's current internal time to the issuing task before setting it to the specified values.

All time parameters must be specified as binary numbers.

A task must be privileged to issue this directive.

Changing the system time does not affect the time-based entries in the clock queue. Although the actual system time changes, the time interval after which a time-based entry is to be dequeued remains the same. This behavior allows the proper time-synchronization of events to be maintained.

For example, if a task is scheduled to run one hour from the current time, it will still run after this interval even though the time might be changed from 11:27 to 11:37. The display of the entry in the clock queue (MCR CLQ or DCL SHOW CLOCK_QUEUE) shows the new time at which the task will run.

FORTRAN Call

```
CALL SETTIM  (ibufn[,ibufp][,ids])
```

Parameters

ibufn
An 8-word integer array—new time-specification buffer

ibufp
An 8-word integer array—previous time buffer

ids
Directive status

Macro Call

```
STIM$  bufn,[bufp]
```

Parameters

bufn
Address of new 8-word time-specification buffer

bufp
Address of an 8-word buffer to receive the previous system-time parameters

Buffer Format

Word 0 Year (since 1900)

Word 1 Month (1-12)

Word 2 Day (1-n, where n is the highest day possible for the given month and year)

Word 3 Hour (0-23)

Word 4 Minute (0-59)

Word 5 Second (0-59)

Word 6 Tick of second (0-n, where n is the frequency of the system clock minus one); if the next parameter (ticks per second) is defaulted, this parameter is ignored

Word 7 Ticks per second (must be defaulted or must match the frequency of the system clock); this parameter is used to verify the intended granularity of the "tick of second" parameter

Note

If any of the specified new time parameters are defaulted (equal to -1), the corresponding previous system-time parameters will remain unchanged and will be substituted for the defaulted parameters during argument validation.

Macro Expansion

```
STIM$   NEWTIM,OLDTIM
.BYTE   61.,3           ;STIM$ DIC, DPB SIZE = 3 WORDS
.WORD   NEWTIM          ;ADDRESS OF 8(10)-WORD INPUT BUFFER
.WORD   OLDTIM          ;ADDRESS OF 8(10)-WORD OUTPUT BUFFER
```

Local Symbol Definitions

S.TIBA Input buffer address (2)

S.TIBO Output buffer address (2)

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

S.TIYR Year (2)

S.TIMO Month (2)

S.TIDA Day (2)

S.TIHR Hour (2)

S.TIMI Minute (2)

S.TISC Second (2)

S.TICT Clock tick of second (2)

S.TICP Clock ticks per second (2)

DSW Return Codes

IS.SUC Successful completion.

IE.PRI The issuing task is not privileged.

IE.ITI One of the specified time parameters is out of range, or both the tick-of-second parameter and the ticks-per-second parameter were specified and the ticks-per-second parameter does not match the system's clock frequency. The system time at the moment the directive is issued (returned in the second buffer) can be useful in determining the cause of the fault if any of the specified time parameters were defaulted.

IE.ADP Part of the DPB or one of the buffers is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. Execution of this directive generates an error log packet and sends it to the Error Logger.
2. On a system with accounting active, this directive causes an accounting transaction that records both the old and new times.
3. The highest clock frequency supported by the operating system is 1000 Hz for a programmable clock. Note that as the clock frequency approaches this value, the maximum resolution for this directive becomes more time-critical. The accuracy of this directive depends upon the elapsed time between the moment that a new system time is specified and the time that the directive actually traps to the Executive.
4. The buffers used in this directive are compatible with those of the Get Time Parameters (GTIM$) directive.
5. The second buffer (previous time) is filled in only if the directive was successfully executed or if it was rejected with an error code of IE.ITI.

5.93 Stop for Logical OR of Event Flags

The Stop for Logical OR of Event Flags directive instructs the system to stop the issuing task until the Executive sets one or more of the indicated event flags from one of the following groups:

GR 0 Local flags 1-16

GR 1 Local flags 17-32

GR 2 Common flags 33-48

GR 3 Common flags 49-64

GR 4 Group global flags 65-80

GR 5 Group global flags 81-96

The task does not stop itself if any of the indicated flags are already set when the task issues the directive. This directive cannot be issued at AST state. See the Notes.

A task that is stopped for one or more event flags can become unstopped only by setting the specified event flag. It cannot become unstopped with the Unstop directive or with the MCR UNSTOP or DCL START command.

FORTRAN Call

```
CALL STLOR  (ef1,ef2,ef3...,efn)
CALL STLORS  (idsw,ef1,ef2,ef3...,efn)
```

Parameters

ef1...efn
List of event flag numbers

idsw
Integer to receive the Directive Status Word

Macro Call

```
STLO$  grp,msk
```

Parameters

grp
Desired group of event flags

msk
A 16-bit mask word

STLO$

Macro Expansion

```
STLO$   1,47
.BYTE   137.,3          ;STLO$ MACRO DIC, DPB SIZE = 3 WORDS

.WORD   1               ;GROUP 1 FLAGS (FLAGS 17-32)
.WORD   47              ;MASK WORD = 47 (FLAGS 17, 18, 19, 22)
```

Local Symbol Definitions

S.TLGR Group flags (2)

S.TLMS Mask word (2)

DSW Return Codes

IS.SUC Successful completion.

IE.AST The issuing task is at AST state.

IE.IEF An event flag group other than 0 through 5 was specified, or the event-flag mask word is zero.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. There is a one-to-one correspondence between bits in the mask word and the event flags in the specified group. That is, if group 1 were specified (as in the above macro expansion example), bit 0 in the mask word would correspond to event flag 17, bit 1 to event flag 18, and so forth.
2. The Executive does not arbitrarily clear event flags when Stop for Logical OR of Event Flags conditions are met. Some directives (Queue I/O Request, for example) implicitly clear a flag. Otherwise, they must be explicitly cleared by a Clear Event Flag directive.
3. The argument list specified in the FORTRAN or other high-level language call must contain only those event flag numbers that lie within one event flag group. If event flag numbers are specified that lie within more than one event flag group or if an invalid event flag is specified, a task abort is generated with an error code in a register (see Section 1.5.3).
4. Tasks stopped for event flag conditions cannot be unstopped by issuing the Unstop directive; tasks stopped in this manner can be unstopped only by meeting other event flag conditions.
5. The grp operand must always be of the form n regardless of the macro form used. In almost all other macro calls, numeric or address values for $S form macros have the following form:

   #n

   For STLO$S, this form of the grp argument would be as follows:

   n

6. If the specified event flag group is group global, the group's use count is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - The Stop-for condition is satisfied.
   - The issuing task exits before the Stop-for condition is satisfied.

STOP$S

5.94 Stop ($S Form Recommended)

The Stop directive stops the issuing task. This directive cannot be issued at AST state. A task stopped in this manner can be unstopped only by another task issuing an Unstop directive directed to the task, the task issuing an Unstop directive at AST state, or with the MCR UNSTOP or DCL START command.

FORTRAN Call

CALL STOP [(idsw)]

Parameter

idsw

Integer to receive the Directive Status Word

Macro Call

STOP$S

Macro Expansion

```
STOP$S
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK

.BYTE   131.,1          ;STOP$ MACRO DIC, DPB SIZE = 1 WORD
EMT     377             ;TRAP TO THE EXECUTIVE
```

Local Symbol Definitions

None

DSW Return Codes

IS.SET	Successful completion.
IE.AST	The issuing task is at AST state.
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

5.95 Stop for Single Event Flag

The Stop for Single Event Flag directive instructs the system to stop the issuing task until the specified event flag is set. If the flag is set at issuance, the task is not stopped. This directive cannot be issued at AST state.

A task that is stopped for one or more event flags can become unstopped only by setting the specified event flag. It cannot become unstopped by the Unstop directive or by the MCR UNSTOP or DCL START command.

FORTRAN Call

CALL STOPFR (iefn[,idsw])

Parameters

iefn

Event flag number

idsw

Integer to receive the Directive Status Word

Macro Call

STSE$ efn

Parameter

efn

Event flag number

Macro Expansion

```
STSE$   7
.BYTE   135.,2          ;STSE$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   7               ;LOCAL EVENT FLAG NUMBER = 7
```

Local Symbol Definition

S.TSEF Event flag number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.AST The issuing task is at AST state.

IE.IEF Invalid event flag number (EFN <1, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Note

If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of event flags. The use count is run down when the following events occur:

- The Stop-for condition is satisfied.
- The issuing task exits before the Stop-for condition is satisfied.

5.96 Specify SST Vector Table for Debugging Aid

The Specify SST Vector Table for Debugging Aid directive instructs the system to record the address of a table of SST service-routine entry points for use by an intratask debugging aid (ODT, for example).

To deassign the vector table, omit the parameters adr and len from the macro call.

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

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.

Macro Call

SVDB$ [adr][,len]

Parameters

adr

Address of the SST vector table

len

Length of (that is, number of entries in) the table in words

The vector table has the following format:

Word 0 Odd address of nonexistent memory error

Word 1 Memory protect violation

Word 2 T-bit trap or execution of a BPT instruction

Word 3 Execution of an IOT instruction

Word 4 Execution of a reserved instruction

Word 5 Execution of a non-RSX EMT instruction

Word 6 Execution of a TRAP instruction

Word 7 PDP–11/40 floating-point exception

A zero entry in the table indicates that the task does not want to process the corresponding SST.

SVDB$

Macro Expansion

```
SVDB$   SSTTBL,4
.BYTE   103.,3          ;SVDB$ MACRO DIC, DPB SIZE = 3 WORDS

.WORD   SSTTBL          ;ADDRESS OF SST TABLE
.WORD   4               ;SST TABLE LENGTH = 4 WORDS
```

Local Symbol Definitions

S.VDTA Table address (2)

S.VDTL Table length (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ADP Part of the DPB or table is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.97 Specify SST Vector Table for Task

The Specify SST Vector Table for Task directive instructs the system to record the address of a table of SST service-routine entry points for use by the issuing task.

To deassign the vector table, omit the parameters adr and len from the macro call.

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

FORTRAN Call

Neither the FORTRAN language nor the ISA standard permits direct linking to system-trapping mechanism. Therefore, this directive is not available to FORTRAN tasks.

Macro Call

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

Parameters

adr

Address of the SST vector table

len

Length of (that is, number of entries in) the table in words

The vector table has the following format:

Word 0	Odd address of nonexistent memory error
Word 1	Memory protect violation
Word 2	T-bit trap or execution of a BPT instruction
Word 3	Execution of an IOT instruction
Word 4	Execution of a reserved instruction
Word 5	Execution of a non-RSX EMT instruction
Word 6	Execution of a TRAP instruction
Word 7	PDP–11/40 floating-point exception

A zero entry in the table indicates that the task does not want to process the corresponding SST.

SVTK$

Macro Expansion

```
SVTK$   SSTTBL,4
.BYTE   105.,3          ;SVTK$ MACRO DIC, DPB SIZE = 3 WORDS

.WORD   SSTTBL          ;ADDRESS OF SST TABLE
.WORD   4               ;SET TABLE LENGTH = 4 WORDS
```

Local Symbol Definitions

S.VTTA Table address (2)

S.VTTL Table length (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ADP Part of the DPB or table is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

5.98 Switch State

The Switch State directive makes it possible for a privileged task which is not itself mapped to the Executive to map subroutines that require access to the Executive. For information on mapping the subroutines, see Notes 3 and 5 for the description of the Connect to Interrupt (CINT$) directive.

The directive maps the subroutine through APR 5 (that is, it uses virtual addresses 120000 through 137777_8). Therefore, the subroutine, and all data in the task referenced by the subroutine, must fall within the limits of 4K words of the base virtual address specified in the directive. The subroutine itself is executed as part of the SWST$ directive and is, therefore, in system state during its execution. Local data references must also be within the 4K-word limit.

FORTRAN Call

Not supported

Macro Call

SWST$ base,addr

base

The base virtual address within the task for mapping the subroutine through APR 5

addr

Virtual address of the subroutine to be executed in system state by the directive

Macro Expansion

```
SWST$   BASE,ADDR
.BYTE   175.,3          ;SWST$ MACRO DIC, DPB SIZE = 3 WORDS
.WORD   BASE            ;BASE VIRTUAL ADDRESS FOR MAPPING THE
                        ;SUBROUTINE THROUGH APR 5
.WORD   ADDR            ;VIRTUAL ADDRESS OF THE SUBROUTINE EXECUTED AT SYSTEM STATE
```

Local Symbol Definitions

S.WBAS Base virtual address for mapping the subroutine through APR 5

S.WADD Virtual address of the subroutine executed at system state

DSW Return Codes

IS.SUC Successful completion of service.

IE.PRI The issuing task is not privileged.

IE.MAP The specified system-state routine is more than 4K words from the specified base.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

SWST$

Notes

1. User-mode register contents are preserved across the execution of the kernel-mode subroutine. Contents of the user-mode registers are passed into the kernel-mode registers. Contents of the kernel-mode registers are discarded when the subroutine has completed execution.

2. User-mode registers appear at the following octal stack offsets during execution of the specified subroutine in kernel mode:

   User-mode R0 at S.WSR0 (=2) offset on kernel stack
   User-mode R1 at S.WSR1 (=4) offset on kernel stack
   User-mode R2 at S.WSR2 (=6) offset on kernel stack
   User-mode R3 at S.WSR3 (=10) offset on kernel stack
   User-mode R4 at S.WSR4 (=12) offset on kernel stack
   User-mode R5 at S.WSR5 (=14) offset on kernel stack

   If you want to return any register values to the user-mode registers, you must store the desired values on the stack using the above offsets.

   These offset values become valid when the subroutine is called, and remain valid as long as the stack pointer is not changed. Once the stack pointer changes, the offset values become invalid.

3. Virtual address values passed to system state in a register must be realigned through kernel APR 5. For example, if R5 contains address n and the base virtual address in the DPB is 1000_8, the value in R5 must be aligned using the following formula:

   n+120000+base virtual address

   The resulting value is n+121000.

4. The system-state subroutine should exit by issuing a return instruction. This causes a successful directive status to be returned as the directive is terminated.

Caution

Keep in mind that the memory management unit rounds the base address to the nearest 32-word boundary.

5.99 Test for Specified Task Feature

The Test for Specified Task Feature directive tests for the presence of a specific task software option, such as fast-mapping support or privilege status.

FORTRAN Call

CALL TFEA (isym,idsw)

Parameters

isym
Symbol for the specified task feature

idsw
Integer to receive the Directive Status Word

Macro Call

TFEA$ sym

Parameter

sym
Symbol for the specified task feature (see Table 5-2)

Table 5–2: Task Feature Symbols

Symbol	Value	Meaning
T2$WFR	1	Task in Wait-for state (1=YES)
T2$WFA	2	Saved T2$WFR on AST in progress
T2$SPN	3	Task suspended (1=YES)
T2$SPA	4	Saved T2$SPN on AST in progress
T2$STP	5	Task stopped (1=YES)
T2$STA	6	Saved T2$SPN [STP?] on AST in progress
T2$ABO	7	Task marked for abort (1=YES)
AT2$AFF	9.	Task is installed with affinity
T2$SIO	10.	Task stopped for buffered I/O
T2$SEF	12.	Task stopped for event flag or flags (1=YES)
T2$REX	13.	Requested exit AST specified
T2$CHK	14.	Task not checkpointable (1=YES)
T2$DST	15.	AST recognition disabled (1=YES)
T2$AST	16.	AST in progress (1=YES)

Table 5–2 (Cont.): Task Feature Symbols

Symbol	Value	Meaning
T3$GFL	17.	Group global event flag lock
T3$SWS	18.	Reserved for use by Software Services
T3$CMD	19.	Task is executing a CLI command
T3$MPC	20.	Mapping change with outstanding I/O
T3$NET	21.	Network protocol level
T3$ROV	22.	Task has resident overlays
T3$CAL	23.	Task has checkpoint space in image
T3$NSD	24.	Task does not allow Send Data
T3$RST	25.	Task is restricted (1=YES)
T3$CLI	26.	Task is a command line interpreter
T3$SLV	27.	Task is a slave task (1=YES)
T3$MCR	28.	Task requested as external MCR function
T3$PRV	29.	Task is privileged (1=YES)
T3$REM	30.	Remove task on exit (1=YES)
T3$PMD	31.	Dump task on synchronous abort (0=YES)
T3$ACP	32.	Ancillary Control Processor (1=YES)
T4$SNC	33.	Task uses commons for synchronization
T4$DSP	34.	Task was built for user I/D space
T4$PRV	35.	Task was privileged, but has cleared T3.PRV with GIN$ (may be resent with GIN$ if T4$PRV set)
T4$PRO	36.	TCB is (or should be) a prototype
T4$LDD	37.	Task's load device has been dismounted
T4$MUT	38.	Task is a multiuser task
T4$CTC	39.	Task has been processed by GIN$ ^C abort
T4$FMP	40.	Task has fast-mapping header extension

Macro Expansion

```
TFEA$   T4$FMP
.BYTE   209.,2          ;TFEA$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   T4$FMP          ;FEATURE IDENTIFIER
```

Local Symbol Definition

F.TEAF Feature identifier (2)

DSW Return Codes

IS.CLR Successful completion; feature not present.

IS.SET Successful completion; feature present.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

TLON$, TLOG$

5.100 Translate Logical Name String

The Translate Logical Name String directive returns the equivalence string previously associated with the specified logical name.

The TRALON and TLON$ calls are the preferred calls to use on RSX-11M-PLUS and Micro/RSX operating systems. The TRALOG and TLOG$ calls are provided for compatibility with the P/OS operating system. See the Note.

FORTRAN Calls

CALL TRALON ([mod],[tbmsk],[status],lns,lnssz,ens,ienssz,[rsize],[rtbmod][,idsw])
CALL TRALOG ([mod],[tbmsk],[status],lns,lnssz,ens,ienssz,[rsize],[rtbmod][,idsw])

Parameters

mod

Optional modifier of the logical name within a table. Ordinarily, no value would be specified to allow any defined logical name to be found.

tbmsk

Inhibit mask to prevent a logical name table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status

Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns

Character array containing the logical name string

lnssz

Size (in bytes) of the logical name string

ens

Character array buffer to contain the returned equivalence string

lenssz
Size (in bytes) of the data area for the returned equivalence name string

rsize
Word to receive the size of the returned equivalence name

rtbmod
Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

idsw
Integer to receive the Directive Status Word

Macro Calls

TLON$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]
TLOG$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]

Parameters

mod
Optional modifier to be matched against the logical name within a table. Ordinarily, no value would be specified to allow any logical name in table to be found.

tbmsk
Inhibit mask to prevent a table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified, the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status
Word to receive the logical status word:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns
Character array containing the original logical name string

lnssz
Size (in bytes) of the original logical name string

TLON$, TLOG$

ens
Character array to contain the returned equivalence string

enssz
Size (in bytes) of the data area for the returned equivalence name string

rsize
Word to receive the size of the returned equivalence name; this size is always the actual size of the equivalence name regardless of the string size specified with enssz

rtbmod
Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

Macro Expansion

```
TLON$ MOD,TBMSK,LNS,STATUS,LNSSZ,ENS,ENSSZ,RSIZE,RTBMOD
.BYTE     207.,10.        ;TLON$ MACRO DIC, DPB SIZE = 10(10) WORDS
.BYTE     13.             ;SUBFUNCTION VALUE (TLOG$ = 9(10))

.BYTE     MOD             ;LOGICAL NAME MODIFIER
.WORD     TBMSK           ;LOGICAL NAME TABLE INHIBIT MASK

.WORD     LNS             ;LOGICAL NAME STRING ARRAY

.WORD     STATUS          ;LOCATION OF LOGICAL NAME STATUS

.WORD     LNSSZ           ;SIZE (IN BYTES) OF LOGICAL NAME STRING

.WORD     ENS             ;RETURNED EQUIVALENCE NAME ARRAY
.WORD     ENSSZ           ;SIZE (IN BYTES) OF EQUIVALENCE NAME

.WORD     RSIZE           ;LOCATION OF SIZE FOR RETURNED EQUIVALENCE NAME

.WORD     RTBMOD          ;LOCATION OF LOGICAL TABLE NUMBER (LOWER BYTE) AND
                          ;MODIFIER VALUE OF LOCATED LOGICAL NAME (HIGHER BYTE)
```

Local Symbol Definitions

T.LENS Address of buffer for returned equivalence name (2)

T.LESZ Byte count of buffer for returned equivalence name (2)

T.LFUN Subfunction value (1)

T.LLNS Address of logical name string (2)

T.LLSZ Size (in bytes) of specified logical name (2)

T.LMOD Logical name modifier (1)

T.LRSZ Word for returned equivalence name size (2)

T.LRTM Word for returned table number and modifier (2)

T.LSTS Address of status block for LNB (2)

T.LTBL Table inhibit mask (2)

DSW Return Codes

IS.SUC	Successful completion.
IE.ITN	Invalid table number specified.
IE.LNF	The specified logical name string was not found.
IE.ADP	Part of the DPB or user buffer is out of the issuing task's address space, or you do not have the proper access to that region.
IE.SDP	DIC or DPB size is invalid.

Note

The TRALON and TLON$ calls are the preferred calls to use on RSX–11M–PLUS and Micro/RSX operating systems. The TRALOG and TLOG$ calls are provided for compatibility with the P/OS operating system. When you use TRALOG or TLOG$, the system performs the following actions:

- If a device name or node name ends with one or more colons, strips off one to two of the terminating colons.
- If a physical device name string is in the form ddnnn:, compresses any leading zeros. For example, DR005: becomes DR5.

ULGF$S

5.101 Unlock Group Global Event Flags ($S Form Recommended)

The Unlock Group Global Event Flags directive instructs the Executive to decrement the use count of the group global event flags for the issuing task's protection group UIC (H.CUIC+1). This unlocks flags that were locked by the Create Group Global Event Flags directive.

A task may unlock the event flags only once before locking them again.

The group global event flags are eliminated if the following conditions are satisfied:

- The use count in the Group Global Event Flag Control Block (GFB) is zero after this directive is issued.
- The GFB is marked for deletion.

FORTRAN Call

CALL ULGF [(ids)]

Parameter

ids

Directive status

Macro Call

ULGF$S [err]

Parameter

err

Error-routine address

Macro Expansion

```
ULGF$S   ERR
MOV      (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK

.BYTE    159.,1          ;ULGF$S MACRO DIC, DPB SIZE = 1 WORD
EMT      377             ;TRAP TO THE EXECUTIVE

BCC      .+6             ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR      PC,ERR          ;OTHERWISE, CALL ROUTINE "ERR"
```

Local Symbol Definitions

None

DSW Return Codes

IS.SUC Successful completion.

IE.RSU Event flags already unlocked from the issuing task.

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

UMAP$

5.102 Unmap Address Window

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.

FORTRAN Call

CALL UNMAP (iwdb[,ids])

Parameters

iwdb
An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids
Directive status

Macro Call

UMAP$ wdb

Parameter

wdb
Window Definition Block address

Macro Expansion

```
UMAP$   WDBADR
.BYTE   123.,2          ;UMAP$ MACRO DIC, DPB SIZE = 2 WORDS
.WORD   WDBADR          ;WDB ADDRESS
```

Window Definition Block Parameters

Input parameters:

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

Output parameters:

Array Element	Offset	Meaning	
iwdb(7)	W.NSTS	Bit settings[1]	in the window status word:
		Bit	**Definition**
		WS.UNM	1 if the window was unmapped successfully

[1] If you are a FORTRAN programmer, refer to Section 3.5.2 to determine the bit values represented by the symbolic names described.

Local Symbol Definition

U.MABA Window Definition Block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ITS The specified address window is not mapped.

IE.NVW Invalid address window ID.

IE.ADP DPB or WDB out of range.

IE.SDP DIC or DPB size is invalid.

USTP$

5.103 Unstop Task

The Unstop Task directive unstops the specified task that has stopped itself by either the Stop, the Receive by Reference or Stop, or the Receive Data or Stop directive. It does not unstop tasks stopped for event flags or tasks stopped for buffered I/O. If the Unstop directive is issued to a task previously stopped by means of the Stop or Receive or Stop directive while at task state and the task is presently at AST state, the task becomes unstopped only when it returns to task state.

It is considered the responsibility of the unstopped task to determine if it has been unstopped validly.

The Unstop directive does not cause a significant event.

FORTRAN Call

CALL USTP ([rtname][,ids])

Parameters

rtname

Name of the task to be unstopped (if not specified, CALL USTP will use the issuing task as its default)

ids

Integer to receive directive status information

Macro Call

USTP$ [tname]

Parameter

tname

Name of the task to be unstopped (if not specified, USTP$ will use the issuing task as its default)

Macro Expansion

```
USTP$   ALPHA
.BYTE   133.,3          ;USTP$ MACRO DIC, DPB SIZE = 3 WORDS
.RAD50  /ALPHA/         ;NAME OF TASK TO BE UNSTOPPED
```

Local Symbol Definition

U.STTN Task name (4)

DSW Return Codes

IS.SUC Successful completion.

IE.INS The specified task is not installed in the system.

IE.ACT	The specified task is not active.
IE.ITS	The specified task is not stopped, or it is stopped for event flags or buffered I/O.
IE.ADP	Part of the DPB is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

VRCD$

5.104 Variable Receive Data

The Variable Receive Data directive instructs the system to dequeue a variable-length data block for the issuing task. (The data block has been queued (FIFO) for the task by a Variable Send Data directive.) When a sender task is specified, only data sent by the specified task is received.

The buffer size can be 256_{10} words maximum. If no buffer size is specified (macro calls only), the buffer size is 13_{10} words. If a buffer size greater than 256_{10} is specified, an IE.IBS error is returned.

A 2-word sender task name (in Radix–50 form) and the data block are returned in the specified buffer, with the task name in the first two words. The two words are added to the buffer size you specify.

Variable-length data blocks are transferred from the sending task to the receiving task by means of buffers in secondary pool.

FORTRAN Call

CALL VRCD ([task],bufadr,buflen[,idsw])

Parameters

task
Sender task name

bufadr
Address of the buffer to receive the sender task name and data (must be word-aligned (INTEGER*2))

buflen
Length of the buffer

idsw
Integer to receive the Directive Status Word

If the directive was successful, it returns the number of words transferred into the user buffer. If the directive execution encountered an error, it returns the error code in the ids parameter.

Any error return of the form IE.XXX is a negative word value. If the status is positive, the value of the status word is the number of words transferred including the task name. For example, if you specify a buffer size of 13 in the VRCD$ call, the value returned in the Directive Status Word is 15 (13 words of data plus the 2 words needed to return the task name).

Macro Call

VRCD$ [task],bufadr[,buflen],[ti]

Parameters

task
Sender task name

bufadr
Buffer address

buflen
Buffer size in words

ti
TI: indicator (ignored on RSX systems)

Macro Expansion

```
VRCD$   SNDTSK,DATBUF,BUFSIZ,0
.BYTE   75.,6           ;VRCD$ MACRO DIC, DPB SIZE = 6 WORDS
.RAD50  /SNDTSK/        ;SENDER TASK NAME
.WORD   DATBUF          ;ADDRESS OF DATA BUFFER
.WORD   BUFSIZ          ;BUFFER SIZE
.WORD   0               ;TI: INDICATOR (IGNORED ON RSX SYSTEMS)
```

Local Symbol Definitions

R.VDTN Sender task name (4)

R.VDBA Buffer address (2)

R.VDBL Buffer length (2)

R.VDTI TI: indicator (ignored on RSX systems) (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ITS No data in task's receive queue or no data from specified task.

IE.RBS Receive buffer is too small.

IE.IBS Invalid buffer size specified (greater than 256_{10}).

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

VRCS$

5.105 Variable Receive Data or Stop

The Variable Receive Data or Stop directive instructs the system to dequeue a variable-length data block for the issuing task. (The data block has been queued (FIFO) for the task by a Variable Send Data directive.) If there is no such packet to be dequeued, the issuing task is stopped. In this case, another task (the sender task) is expected to issue an Unstop directive after sending the data. When stopped in this manner, the directive status returned is IS.SET, indicating that the task was stopped and that no data has been received. However, since the task must be unstopped in order to see this status, the task can now reissue the Variable Receive Data or Stop directive to actually receive the data packet.

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

The buffer size can be 256_{10} words maximum. If no buffer size is specified, the buffer size is 13_{10} words. If a buffer size greater than 256_{10} is specified, an IE.IBS error is returned.

A 2-word sender task name (in Radix-50 form) and the data block are returned in the specified buffer, with the task name in the first two words. The two words are added to the buffer size you specify.

Variable-length data blocks are transferred from the sending task to the receiving task by means of buffers in secondary pool.

FORTRAN Call

```
CALL VRCS  ([task],bufadr,[buflen][,idsw])
```

Parameters

task
Sender task name

bufadr
Address of the buffer to receive the sender task name and data

buflen
Length of the buffer

idsw
Integer to receive the Directive Status Word

If the directive was successful, it returns the number of words transferred into the user buffer. If the directive execution encountered an error, it returns the error code in the ids parameter.

Any error return of the form IE.XXX is a negative word value. If the status is positive, the value of the status word is the number of words transferred including the task name. For example, if you specify a buffer size of 13 in the VRCS$ call, the value returned in the directive status word is 15 (13 words of data plus the 2 words needed to return the task name).

Macro Call

```
VRCS$  [task],bufadr[,buflen],[ti]
```

Parameters

task
Sender task name

bufadr
Buffer address

buflen
Buffer size in words

ti
TI: indicator (ignored on RSX systems)

Macro Expansion

```
VRCS$   SNDTSK,DATBUF,BUFSIZ,0
.BYTE   139.,6          ;VRCS$ MACRO DIC, DPB SIZE = 6 WORDS
.RAD50  /SNDTSK/        ;SENDER TASK NAME

.WORD   DATBUF          ;ADDRESS OF DATA BUFFER
.WORD   BUFSIZ          ;BUFFER SIZE IN WORDS
.WORD   0               ;TI: INDICATOR (IGNORED ON RSX SYSTEMS)
```

Local Symbol Definitions

R.VSTN	Sender task name (4)
R.VSBA	Buffer address (2)
R.VSBL	Buffer size in words (2)
R.VSTI	TI: indicator (ignored on RSX systems) (2)

DSW Return Codes

IS.SUC	Successful completion.
IE.ITS	No data in task's receive queue or no data from specified task.
IE.RBS	Receive buffer is too small.
IE.IBS	Invalid buffer size specified (greater than 256_{10}).
IE.ADP	Part of the DPB or buffer is out of the issuing task's address space.
IE.SDP	DIC or DPB size is invalid.

VRCX$

5.106 Variable Receive Data or Exit

The Variable Receive Data or Exit directive instructs the system to dequeue a variable-length data block for the issuing task. (The data block has been queued (FIFO) for the task by a Variable Send Data directive.) When a sender task is specified, only data sent by the specified task is received.

A 2-word sender task name (in Radix–50 form) and the data block are returned in the specified buffer, with the task name in the first two words. The two words are added to the buffer size you specify.

If no data has been sent, a task exit occurs. To prevent the possible loss of send-data packets, you should not rely on I/O rundown to take care of any outstanding I/O or open files. The task should assume this responsibility.

The buffer size can be 256_{10} words maximum. If no buffer size is specified, the buffer size is 13_{10} words. If a buffer size greater than 256_{10} is specified, an IE.IBS error is returned.

Variable-length data blocks are transferred from the sending task to the receiving task by means of buffers in secondary pool.

FORTRAN Call

CALL VRCX ([task],bufadr,[buflen][,idsw])

Parameters

task
Sender task name

bufadr
Address of the buffer to receive the sender task name and data

buflen
Length of the buffer

idsw
Integer to receive the Directive Status Word

If the directive was successful, it returns the number of words transferred into the user buffer. If the directive execution encountered an error, it returns the error code in the ids parameter.

Any error return of the form IE.XXX is a negative word value. If the status is positive, the value of the status word is the number of words transferred including the task name. For example, if you specify a buffer size of 13 in the VRCX$ call, the value returned in the directive status word is 15 (13 words of data plus the 2 words needed to return the task name).

Macro Call

VRCX$ [task],bufadr[,buflen],[ti]

Parameters

task
Sender task name

bufadr
Buffer address

buflen
Buffer size in words

ti
TI: indicator (ignored on RSX systems)

Macro Expansion

```
VRCX$   SNDTSK,DATBUF,BUFSIZ,0
.BYTE   77.,6           ;VRCX$ MACRO DIC, DPB SIZE = 6 WORDS
.RAD50  /SNDTSK/        ;SENDER TASK NAME
.WORD   DATBUF          ;ADDRESS OF DATA BUFFER
.WORD   BUFSIZ          ;BUFFER SIZE IN WORDS
.WORD   0               ;TI: INDICATOR (IGNORED ON RSX SYSTEMS)
```

Local Symbol Definitions

R.VXTN Sender task name (4)

R.VXBA Buffer address (2)

R.VXBL Buffer size in words (2)

R.VXTI TI: indicator (ignored on RSX systems) (2)

DSW Return Codes

IS.SUC Successful completion.

IE.ITS No data in task's receive queue or no data from specified task.

IE.RBS Receive buffer is too small.

IE.IBS Invalid buffer size specified (greater than 256_{10}).

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

VSDA$

5.107 Variable Send Data

The Variable Send Data directive instructs the system to queue a variable-length data block for the specified task to receive.

The buffer size can be 256_{10} words maximum. If no buffer size is specified (macro calls only), the buffer size is 13_{10} words. If a buffer size greater than 256_{10} is specified, an IE.IBS error is returned.

When an event flag is specified, a significant event is declared if the directive is executed successfully. The indicated event flag is set for the sending task.

Variable-length data blocks are transferred from the sending task to the receiving task by buffers in secondary pool.

FORTRAN Call

CALL VSDA (task,bufadr,[buflen],[efn][,idsw])

Parameters

task
Receiver task name

bufadr
Array containing data to be sent (must be word-aligned (INTEGER*2))

buflen
Length (in words) of the array

efn
Event flag number

idsw
Integer to receive the Directive Status Word

Macro Call

VSDA$ task,bufadr,[buflen],[efn],[spri],[ti]

Parameters

task
Receiver task name

bufadr
Buffer address

buflen
Buffer size in words

efn
Event flag number

spri

Send priority (ignored on RSX systems)

ti

TI: indicator (ignored on RSX systems)

Macro Expansion

```
VSDA$   RECTSK,DATBUF,BUFSIZ,4,0,1
.BYTE   71.,8.          ;VSDA$ MACRO DIC, DPB SIZE = 8(10) WORDS

.RAD50  /RECTSK/        ;RECEIVER TASK NAME
.WORD   DATBUF          ;ADDRESS OF DATA BUFFER

.WORD   4               ;EVENT FLAG 4
.WORD   BUFSIZ          ;BUFFER SIZE

.WORD   0               ;SEND PRIORITY (IGNORED ON RSX SYSTEMS)
.WORD   1               ;TI: INDICATOR (IGNORED ON RSX SYSTEMS)
```

Local Symbol Definitions

S.DATN Sender task name (4)

S.DABA Buffer address (2)

S.DAEF Event flag number (2)

S.DABL Buffer length (2)

S.DASP Send priority (ignored on RSX systems) (2)

S.DATI TI: indicator (ignored on RSX systems) (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN Insufficient dynamic storage.

IE.INS Specified task not installed.

IE.IBS Invalid buffer size specified (greater than 256_{10}).

IE.IEF Invalid event flag number (EFN <0 or EFN> 96).

IE.ADP Part of the DPB or buffer is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

VSRC$

5.108 Variable Send, Request, and Connect

The Variable Send, Request, and Connect directive performs a Variable Send Data to the specified task, requests the task if it is not already active, and then connects to the task. The receiver task normally returns status by the Emit Status or the Exit with Status directive.

The buffer size can be 256_{10} words maximum. If no buffer size is specified, the buffer size is 13_{10} words. If a buffer size greater than 256_{10} is specified, an IE.IBS error is returned.

FORTRAN Call

```
CALL VSRC  (rtname,ibuf,[ibuflen],[iefn],[iast],[iesb],[iparm][,idsw])
CALL VSRCN (rtname,ibuf,[ibuflen],[iefn],[iast],[iesb],[iparm][,idsw])
```

Parameters

rtname
Target task name of the offspring task to be connected

ibuf
Name of send buffer

ibuflen
Length of the buffer

iefn
Event flag to be set when the offspring task exits or emits status

iast
Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL VSRCN)

iesb
Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the iefn parameter above.

iparm
Name of a word to receive the status block address when an AST occurs

idsw
Integer to receive the Directive Status Word

Macro Call

VSRC$ tname,buf[,buflen],[efn],[east],[esb]

Parameters

tname

Target task name of the offspring task to be connected

buf

Address of send buffer

buflen

Length of buffer

efn

The event flag to be cleared on issuance and set when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Note

The exit status block defaults to one word. To use the 8-word exit status block, you must specify the logical OR of the symbol SP.WX8 and the event flag number in the efn parameter above.

Macro Expansion

```
VSRC$   ALPHA,BUFFR,BUFSIZE,2,SDRCTR,STBLK
.BYTE   141.,8.          ;VSRC$ MACRO DIC, DPB SIZE = 8(10) WORDS

.RAD50  /ALPHA/          ;TARGET TASK NAME
.WORD   BUFFR            ;SEND BUFFER ADDRESS

.BYTE   2                ;EVENT FLAG NUMBER = 2
.BYTE   16.              ;EXIT STATUS BLOCK CONSTANT
```

VSRC$

```
.WORD   BUFSIZE         ;LENGTH OF BUFFER IN WORDS
.WORD   SDRCTR          ;ADDRESS OF AST ROUTINE
.WORD   STBLK           ;ADDRESS OF STATUS BLOCK
```

Local Symbol Definitions

V.SRTN Task name (4)

V.SRBF Buffer address (2)

V.SREF Event flag (2)

V.SRBL Buffer length (2)

V.SREA AST routine address (2)

V.SRES Status block address (2)

DSW Return Codes

IS.SUC Successful completion.

IE.UPN There was insufficient dynamic memory to allocate a send packet, Offspring Control Block, Task Control Block, or Partition Control Block.

IE.INS The specified task is an ACP or has the no-send attribute.

IE.IBS Invalid buffer size specified (greater than 256_{10}).

IE.IEF An invalid event flag number was specified (EFN <0, or EFN> 96 if group global event flags exist or EFN> 64 if not).

IE.ADP Part of the DPB or exit status block is not in the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. If the specified event flag is group global, the use count for the event flag's group is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - Status is returned from the connected task.
   - The issuing task exits before status is returned.
2. Changing the virtual mapping of the exit status block while the connection is in effect may result in obscure errors.

5.109 Wait for Significant Event ($S Form Recommended)

The Wait for Significant Event directive is used to suspend the execution of the issuing task until the next significant event occurs. It is an especially effective way to block a task that cannot continue because of a lack of dynamic memory, since significant events occurring throughout the system often result in the release of dynamic memory. The execution of a Wait for Significant Event directive does not itself constitute a significant event.

FORTRAN Call

CALL WFSNE

Macro Call

WSIG$S [err]

Parameter

err

Error-routine address

Macro Expansion

```
WSIG$S  ERR
MOV     (PC)+,-(SP)     ;PUSH DPB ONTO THE STACK

.BYTE   49.,1           ;WSIG$S MACRO DIC, DPB SIZE = 1 WORD
EMT     377             ;TRAP TO THE EXECUTIVE

BCC     .+6             ;BRANCH IF DIRECTIVE SUCCESSFUL
JSR     PC,ERR          ;OTHERWISE, CALL ROUTINE "ERR"
```

Local Symbol Definitions

None

DSW Return Codes

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

Notes

1. If a directive is rejected for lack of dynamic memory, this directive is the only technique available for blocking task execution until dynamic memory may again be available.
2. The wait state induced by this directive is satisfied by the first significant event to occur after the directive has been issued. The significant event that occurs may or may not be related to the issuing task.
3. Because this directive requires only a 1-word DPB, using the $S form of the macro is recommended. It requires less space and executes with the same speed as that of the DIR$ macro.

4. Significant events include the following:
   - I/O completion
   - Task exit
   - Execution of a Send Data directive
   - Execution of a Send Data Request and Pass OCB directive
   - Execution of a Send, Request, and Connect directive
   - Execution of a Send By Reference, Receive By Reference, or Receive By Reference or Stop directive
   - Execution of an Alter Priority directive
   - Removal of an entry from the clock queue (for instance, resulting from the execution of a Mark Time directive or the issuance of a rescheduling request)
   - Execution of a Declare Significant Event directive
   - Execution of the round-robin scheduling algorithm at the end of a round-robin scheduling interval
   - Execution of an Exit, Exit with Status, or Emit Status directive

5.110 Wait for Logical OR of Event Flags

The Wait for Logical OR of Event Flags directive instructs the system to block the execution of the issuing task until the Executive sets one or more of the indicated event flags from one of the following groups:

GR 0 Local flags 1-16

GR 1 Local flags 17-32

GR 2 Common flags 33-48

GR 3 Common flags 49-64

GR 4 Group global flags 65-80

GR 5 Group global flags 81-96

The task does not block itself if any of the indicated flags are already set when the task issues the directive. See the Notes.

FORTRAN Call

CALL WFLOR (ef1,ef2,ef3...,efn)
CALL WFLORS (idsw,ef1,ef2,ef3...,efn)

Parameters

ef1...efn
List of event flag numbers

idsw
Integer to receive the Directive Status Word

Macro Call

WTLO$ grp,msk

Parameters

grp
Desired group of event flags

msk
A 16-bit flag mask word

Macro Expansion

```
WTLO$   2,160003
.BYTE   43.,3           ;WTLO$ MACRO DIC, DPB SIZE = 3 WORDS

.WORD   2               ;GROUP 2 FLAGS (FLAGS 33-48)
.WORD   160003          ;EVENT FLAGS 33, 34, 46, 47, AND 48
```

Local Symbol Definitions

None

WTLO$

DSW Return Codes

IS.SUC Successful completion.

IE.IEF No event flag specified in the mask word or flag group indicator other than 0, 1, 2, 3, 4, or 5.

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

Notes

1. There is a one-to-one correspondence between bits in the mask word and the event flags in the specified group. That is, if group 1 were specified, then bit 0 in the mask word would correspond to event flag 17, bit 1 to event flag 18, and so forth.
2. The Executive does not arbitrarily clear event flags when Wait-for conditions are met. Some directives (Queue I/O Request, for example) implicitly clear a flag. Otherwise, they must be explicitly cleared by a Clear Event Flag directive.
3. The grp operand must always be of the form n regardless of the macro form used. In almost all other macro calls, numeric or address values for $S form macros have the following form:

   #n

   For WTLO$S, this form of the grp argument would be as follows:

   n
4. The argument list specified in the FORTRAN or other high-level language call must contain only those event flag numbers that lie within one event flag group. If event flag numbers are specified that lie within more than one event flag group or if an invalid event flag is specified, a task abort is generated with an error code in a register (see Section 1.5.3).
5. If the issuing task has outstanding buffered I/O when it enters the Wait-for state, it will be stopped. When the task is in a stopped state, it can be checkpointed by any other task regardless of priority. The task is unstopped when the following situations occur:
   - The outstanding buffered I/O completes.
   - The Wait-for condition is satisfied.
6. If the specified group of event flags is group global, the group's use count is incremented to prevent premature elimination of the event flags. The use count is run down when the following events occur:
   - The Wait-for condition is satisfied.
   - The issuing task exits before the Wait-for condition is satisfied.

5.111 Wait for Single Event Flag

The Wait for Single Event Flag directive instructs the system to block the execution of the issuing task until the indicated event flag is set. If the flag is set at issuance, task execution is not blocked.

FORTRAN Call

CALL WAITFR (efn[,ids])

Parameters

efn

Event flag number

ids

Directive status

Macro Call

WTSE$ efn

Parameter

efn

Event flag number

Macro Expansion

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

Local Symbol Definition

W.TSEF Event flag number (2)

DSW Return Codes

IS.SUC Successful completion.

IE.IEF Invalid event flag number (EFN <1, or EFN> 96 if group global event flags exist for the task's group or EFN> 64 if not).

IE.ADP Part of the DPB is out of the issuing task's address space.

IE.SDP DIC or DPB size is invalid.

WTSE$

Notes

1. If the issuing task has outstanding buffered I/O when it enters the Wait-for state, it will be stopped. When the task is in a stopped state, it can be checkpointed by any other task regardless of priority. The task is unstopped when the following situations occur:
   - The outstanding buffered I/O completes.
   - The Wait-for condition is satisfied.
2. If the specified event flag is group global, the group's use count is incremented to prevent premature elimination of event flags. The use count is run down when the following events occur:
   - The Wait-for condition is satisfied.
   - The issuing task exits before the Wait-for condition is satisfied.
3. Be aware of the following situation:

   If you have more than one task waiting for the same event flag and the task with the highest priority clears the event flag first, the remaining tasks will not be able to resume execution. This behavior is inherent in the way tasks execute by priority. (See Section 1.6.)

Appendix A
Summary of Directives

Abort Task (ABRT$)

FORTRAN Call

CALL ABORT (tsk[,ids])

tsk

Name (Radix–50) of the task to be aborted

ids

Directive status

Macro Call

ABRT$ tsk

tsk

Name (Radix–50) of the task to be aborted

Assign Channel (ACHN$)

FORTRAN Call

CALL ACHN ([mod],[itbmsk],lun,fsbuf,fssz[,idsw])

mod

Optional modifier to be matched against the logical name within a table. Ordinarily, no value will be specified to allow any logical name in table to be found.

itbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

fsbuf

Array containing the file specification buffer

fssz

Size (in bytes) of the file specification buffer

idsw

Integer to receive the Directive Status Word

Macro Call

ACHN$ [mod],[tbmsk],lun,fsbuf,fssz

mod

Optional modifier to be matched against the logical name within a table. Ordinarily, no value will be specified to allow any logical name in table to be found.

tbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

fsbuf

Address of file specification buffer

fssz

Size (in bytes) of the file specification buffer

Alter Priority (ALTP$)

FORTRAN Call

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

tsk
Active task name

ipri
A 1-word integer value equal to the new priority, from 1 to 250_{10}

ids
Directive status

Macro Call

ALTP$ [tsk][,pri]

tsk
Active task name

pri
New priority, from 1 to 250_{10}

Assign LUN (ALUN$)

FORTRAN Call

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

lun
Logical unit number

dev
Device name (format: 1A2)

unt
Device unit number

ids
Directive status

Macro Call

ALUN$ lun,dev,unt

lun
Logical unit number

dev
Device name (two uppercase characters)

unt
Device unit number

AST Service Exit (ASTX$S; $S form recommended)

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.

Macro Call

ASTX$S [err]

err
Error-routine address

Attach Region (ATRG$)

FORTRAN Call

CALL ATRG (irdb[,ids])

irdb
An 8-word integer array containing a Region Definition Block (see Section 3.5.1.2)

ids
Directive status

Macro Call

ATRG$ rdb

rdb
Region Definition Block address

Connect to Interrupt Vector (CINT$)

FORTRAN Call

Not supported

Macro Call

CINT$ vec,base,isr,edir,pri,ast

vec

Interrupt vector address; must be in the range 60_8 to highest vector specified during system generation, inclusive, and must be a multiple of 4

base

Virtual base address for kernel APR 5 mapping of the ISR and enable/disable interrupt routines

isr

Virtual address of the ISR or 0 to disconnect from the interrupt vector

edir

Virtual address of the enable/disable interrupt routine

pri

Initial priority at which the ISR is to execute

ast

Virtual address of an AST routine to be entered after the fork-level routine queues an AST

Clear Event Flag (CLEF$)

FORTRAN Call

CALL CLREF (efn[,ids])

efn

Event flag number

ids

Directive status

Macro Call

CLEF$ efn

efn

Event flag number

Create Logical Name (CLON$, CLOG$)

(CALL CRELON and CLON$ are the preferred calls to use on RSX-11M-PLUS and Micro/RSX systems. CALL CRELOG and CLOG$ are provided for compatibility with P/OS systems.)

FORTRAN Calls

```
CALL CRELON  ([mod],itbnum,lns,lnssz,iens,ienssz[,idsw])
CALL CRELOG  ([mod],itbnum,lns,lnssz,iens,ienssz[,idsw])
```

mod

Modifier of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB

itbnum

Logical name table number in the lower byte and the status byte in the upper byte, as follows:

Table number:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

Status:

LS.TRM	1	Terminal status. Iterative translations will not proceed beyond this logical name.
LS.PRV	2	Privileged status. Only privileged tasks may delete this logical name.

lns

Character array containing the logical name string

lnssz

Size (in bytes) of the logical name string

iens

Character array containing the equivalence string to be created

ienssz

Size (in bytes) of the data area for the equivalence string

idsw

Integer to receive the Directive Status Word

Macro Calls

```
CLON$  [mod], <prmlst> ,lns,lnssz,ens,enssz
CLOG$  [mod], <prmlst> ,lns,lnssz,ens,enssz
```

mod

Modifier of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB

<**prmlst**>

<[tbnum][,status]>

(Angle brackets not required if only tbnum is specified.)

tbnum

Logical name table number. The following are the symbolic offsets for the table:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

status

Logical status definition value. The following are the valid bits for the value:

LS.TRM	1	Terminal status. Iterative translations will not proceed beyond this logical name.
LS.PRV	2	Privileged status. Only privileged tasks may delete this logical name.

lns

Logical name string

lnssz

Size (in bytes) of the logical name string

ens

Equivalence name to be associated with logical name

enssz

Size (in bytes) of the equivalence name string

Cancel Mark Time Requests (CMKT$)

FORTRAN Call

CALL CANMT ([efn][,ids])

efn

Event flag number

ids

Directive status

Macro Call

CMKT$ [[efn],[ast],[err]]

efn

Event flag number

ast

Mark time AST address

err

Error-routine address

Connect (CNCT$)

FORTRAN Call

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

rtname

Name (Radix–50) of the offspring task to be connected

iefn

Event flag to be set when the offspring task exits or emits status

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL CNCTN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

iparm

Name of a word to receive the status block address when an AST occurs

ids

Integer to receive the Directive Status Word

Macro Call

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

tname

Name (Radix–50) of the offspring task to be connected

efn

The event flag to be cleared on issuance and set when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Checkpoint Common Region (CPCR$)

FORTRAN Call

CALL CPCR (name[,ids])

name

Name (Radix–50) of the common region to be checkpointed

ids

Directive status

Macro Call

CPCR$ name

name

Name of the common region to be checkpointed

Create Address Window (CRAW$)

FORTRAN Call

CALL CRAW (iwdb[,ids])

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids

Directive status

Macro Call

CRAW$ wdb

wdb
Window Definition Block address

Create Group Global Event Flags (CRGF$)

FORTRAN Call

CALL CRGF ([group][,ids])

group
Group number for the flags to be created. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

ids
Integer to receive the Directive Status Word

Macro Call

CRGF$ [group]

group
Group number for the flags to be created. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

Create Region (CRRG$)

FORTRAN Call

CALL CRRG (irdb[,ids])

irdb
An 8-word integer array containing a Region Definition Block (see Section 3.5.1.2)

ids
Directive status

Macro Call

CRRG$ rdb

rdb

Region Definition Block address

Create Virtual Terminal (CRVT$)

FORTRAN Call

CALL CRVT ([iiast],[ioast],[iaast],[imlen],iparm[,ids])

iiast

AST address at which input requests from offspring tasks are serviced

ioast

AST address at which output requests from offspring tasks are serviced

iaast

AST address at which the parent task may be notified of the completion of successful offspring attach and detach requests to the virtual terminal unit

imlen

Maximum buffer length allowed for offspring I/O requests

iparm

Address of 3-word buffer to receive information from the stack when an AST occurs

ids

Integer to receive the Directive Status Word containing the virtual terminal number

Macro Call

CRVT$ [iast],[oast],[aast],[mlen]

iast

AST address at which input requests from offspring tasks are serviced; if iast=0, offspring input requests are rejected with IE.IFC returned

oast

AST address at which output requests from offspring tasks are serviced; if oast=0, offspring output requests are rejected with IE.IFC returned

aast

AST address at which the parent task may be notified of the completion of successful offspring attach and detach requests to the virtual terminal unit; if aast=0, no notification of offspring attach/detach is returned to the parent task

mlen

Maximum buffer length (in bytes) allowed for offspring I/O requests (default and maximum values for this parameter are system generation options)

Cancel Scheduled Initiation Requests (CSRQ$)

FORTRAN Call

CALL CANALL (tsk[,ids])

tsk
Task name

ids
Directive status

Macro Call

CSRQ$ tsk

tsk
Scheduled (target) task name

Declare Significant Event (DECL$S; $S form recommended)

FORTRAN Call

CALL DECLAR ([,ids])

ids
Directive status

Macro Call

DECL$S [,err]

err
Error-routine address

Delete Logical Name (DLON$, DLOG$)

(CALL DELLON and DLON$ are the preferred calls to use on RSX–11M–PLUS and Micro/RSX systems. CALL DELLOG and DLOG$ are provided for compatibility with P/OS systems.)

FORTRAN Calls

CALL DELLON ([mod],itbnum,[lns],[lnssz][,idsw])
CALL DELLOG ([mod],itbnum,[lns],[lnssz][,idsw])

mod

Modifier of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB

itbnum

Logical name table number. The tables and their corresponding numbers are:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

lns

Character array name containing the logical name string

lnssz

Size (in bytes) of the logical name string

idsw

Integer to receive the Directive Status Word

Macro Calls

DLON$ [mod],tbnum,[lns],[lnssz]
DLOG$ [mod],tbnum,[lns],[lnssz]

mod

Modifier value of the logical name within a table; if not specified, the nonzero value reserved by the system (LB.LOC = 1) is placed in the DPB

tbnum

Logical name table number. The tables and their corresponding numbers are:

System	(LT.SYS)	0
Group	(LT.GRP)	1
Session	(LT.SES)	4
Task	(LT.TSK)	3

lns

Address of logical name string to be deleted

lnssz

Size (in bytes) of the logical name string

Disable AST Recognition (DSAR$S; $S form recommended)

FORTRAN Call

CALL DSASTR [(ids)]

ids
Directive status

Macro Call

DSAR$S [err]

err
Error-routine address

Disable Checkpointing (DSCP$S; $S form recommended)

FORTRAN Call

CALL DISCKP [(ids)]

ids
Directive status

Macro Call

DSCP$S [err]

err
Error-routine address

Detach Region (DTRG$)

FORTRAN Call

CALL DTRG (irdb[,ids])

irdb
An 8-word integer array containing a Region Definition Block (see Section 3.5.1.2)

ids
Directive status

Macro Call

DTRG$ rdb

rdb

Region Definition Block address

Eliminate Address Window (ELAW$)

FORTRAN Call

CALL ELAW (iwdb[,ids])

iwdb

An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids

Directive status

Macro Call

ELAW$ wdb

wdb

Window Definition Block address

Eliminate Group Global Event Flags (ELGF$)

FORTRAN Call

CALL ELGF ([group][,ids])

group

Group number of flags to be eliminated. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

ids

Integer to receive the Directive Status Word

Macro Call

ELGF$ [group]

group

Group number of flags to be eliminated. Only privileged tasks can specify group numbers other than the issuing task's group UIC. If the UIC is not specified, the task's protection UIC (H.CUIC+1) in the task's header is used.

Eliminate Virtual Terminal (ELVT$)

FORTRAN Call

CALL ELVT (iunum[,ids])

iunum
Virtual terminal unit number

ids
Integer to receive the Directive Status Word

Macro Call

ELVT$ unum

unum
Unit number of the virtual terminal to be eliminated. The task must provide this parameter after the virtual terminal is created.

Emit Status (EMST$)

FORTRAN Call

CALL EMST ([rtname],status[,ids])

rtname
Name of a task connected to the issuing task to which the status is to be emitted

status
A 16-bit quantity to be returned to the connected task

ids
Integer to receive the Directive Status Word

Macro Call

EMST$ [tname],status

tname
Name of a task connected to the issuing task to which the status is to be emitted

status
A 16-bit quantity to be returned to the connected task

Enable AST Recognition (ENAR$S; $S form recommended)

FORTRAN Call

CALL ENASTR [(ids)]

ids
Directive status

Macro Call

ENAR$S [err]

err
Error-routine address

Enable Checkpointing (ENCP$S; $S form recommended)

FORTRAN Call

CALL ENACKP [(ids)]

ids
Directive status

Macro Call

ENCP$S [err]

err
Error-routine address

Exit If (EXIF$)

FORTRAN Call

CALL EXITIF (efn[,ids])

efn
Event flag number

ids
Directive status

Macro Call

EXIF$ efn

efn
Event flag number

Task Exit (EXIT$S; $S form recommended)

FORTRAN Call

CALL EXIT (istat)

istat
A 16-bit quantity to be returned to the parent task

Macro Call

EXIT$S [err]

err
Error-routine address

Exit with Status (EXST$)

FORTRAN Call

CALL EXST (istat)

istat
A 16-bit quantity to be returned to the parent task

Macro Call

EXST$ status

status
A 16-bit quantity to be returned to the parent task

Extend Task (EXTK$)

FORTRAN Call

CALL EXTTSK ([inc][,ids])

inc
A positive or negative number equal to the number of 32-word blocks by which the task size is to be extended or reduced

ids
Directive status

Macro Call

EXTK$ [inc]

inc

A positive or negative number equal to the number of 32-word blocks by which the task is to be extended or reduced

Test for Specified System Feature (FEAT$)

FORTRAN Call

CALL FEAT (isym[,ids])

isym

Symbol for the specified system feature

ids

Directive status

Macro Call

FEAT$ sym

sym

Symbol for the specified system feature

File Specification Scanner (FSS$)

FORTRAN Call

CALL FSS (fsbuf,fssz,prsblk,prssz,[reserv][,idsw])

fsbuf

Array containing the file specification buffer

fssz

Size (in bytes) of the file specification buffer

prsblk

Array containing the parse block

prssz

Size (in bytes) of the parse block

reserv

Reserved parameter (must not be specified)

idsw

Integer to receive the Directive Status Word

Macro Call

FSS$ fsbuf,fssz,prsblk,prssz[,reserv]

fsbuf
Address of the file specification buffer

fssz
Size (in bytes) of the file specification buffer

prsblk
Address of the parse block

prssz
Size (in bytes) of the parse block

reserv
Reserved parameter (must be blank)

Get Command for Command Interpreter (GCCI$)

FORTRAN Call

CALL GTCMCI (icbf,icbfl,[iibuf],[iibfl],[iaddr],[incp][,ids])

icbf
Name of a byte array to receive the command

icbfl
Integer containing the size of the icbf array in bytes

iibuf
Name of an integer array to receive the optional information buffer

iibfl
Name of an integer containing the length of the optional information buffer. If you specify a length shorter than the information buffer, as much information as will fit in the specified length is returned.

iaddr
Name of an integer that contains the address in pool of the command desired. (This address was obtained by a previous CALL to GTCMCI with GC.CND specified.)

incp
Name of an integer containing a value indicating the action to take if there is no command queued:

GC.CCS (000) Return with Carry set (default)

GC.CEX (001) Force CLI to exit instead of returning

GC.CST (002) Force CLI to stop instead of returning

GC.CND (200) Copy command into buffer, but do not dequeue it from the list

ids

Integer to receive the Directive Status Word

Macro Call

GCCI$ cbuf,cbfl,[ibuf],[ibfl],[addr],[ncp]

cbuf

Address of buffer to receive command string

cbfl

Length of buffer; maximum buffer size is 266_{10}

ibuf

Address of buffer to receive information on the issuing terminal

ibfl

Length of buffer to receive information

addr

Address of command

ncp

Action to take if no command buffer is present:

GC.CCS (000) Return with Carry set (default)

GC.CEX (001) Force CLI to exit instead of returning

GC.CST (002) Force CLI to stop instead of returning

GC.CND (200) Copy command into buffer, but do not dequeue it from the list

Get Command Interpreter Information (GCII$)

FORTRAN Call

CALL GETCII (ibuf,ibfl,[icli],[idev],[iunit][,ids])

ibuf

Name of an integer array to receive the CLI information

ibfl

Length in bytes of the integer array to receive the CLI information

icli
Name of a 2-word array element containing the Radix-50 name of the CLI

idev
Name of an integer containing the ASCII name of the terminal (default = TI:)

iunit
Name of an integer containing the octal unit number of the terminal

ids
Directive status

Macro Call

GCII$ buf,bufl,cli,[dev],[unit]

buf
Address of buffer to receive information

bufl
Length of information buffer

cli
Name (Radix-50) of the CLI on which information is requested

dev
ASCII name of terminal whose CLI should be used (default = TI:)

unit
Octal unit number of terminal

Get Default Directory (GDIR$)

FORTRAN Call

CALL GETDDS (mod,iens,ienssz,[irsize][,idsw])

mod
Modifier for the GDIR$ directive; specify one of the following values:

0	Get task default
GD.LOG	Get terminal default

iens
Character array containing the default directory string

ienssz
Size (in bytes) of the default directory string

lrsize
Buffer address of the returned default directory string size

idsw
Integer to receive the Directive Status Word

Macro Call

GDIR$ [mod],ens,enssz[,rsize]

mod
Modifier for the GDIR$ directive; specify one of the following values:

0	Get task default
GD.LOG	Get terminal default

ens
Buffer address of the default directory string

enssz
Size (in bytes) of the default directory string buffer

rsize
Buffer address to which the size of the default directory string is returned

General Information (GIN$)

The following are the functions of the GIN$ directive:

GI.GAS - Get Assigned Device Name

Macro Call

GIN$ GI.GAS, buf, siz, dev, unt, udev, unum

GI.GAS
GIN$ function code (0)

buf
Address of 6-word buffer to receive the LUN information

siz
Buffer size in words

dev
Device name

unt
Device unit number

udev
Device name for which this assignment holds (if blank, get global assignment)

unum
Unit number of terminal for which this assignment holds (if high bit set, get login assignment)

GI.UIC - Get System UIC Information

Macro Call

GIN$ GI.UIC, buf, siz

GI.UIC
GIN$ function code (1)

buf
Address of 5- or 32-word buffer to receive the information

siz
Buffer size in words

GI.DEF - Set Task Default UIC

Macro Call

GIN$ GI.DEF, uic

GI.DEF
GIN$ function code (2)

uic
User Identification Code

GI.SPR - Set Task Privilege

Macro Call

GIN$ GI.SPR, flg

GI.SPR
GIN$ function code (7)

flg
New privilege bit in bit 0

GI.REN - Rename Task

Macro Call

GIN$ GI.REN, nam1, nam2

GI.REN
GIN$ function code (8)

nam1
Radix–50 task name, first half

nam2
Radix–50 task name, second half

GI.FMK - Get Feature Mask Words

Macro Call

GIN$ GI.FMK, buf, siz

GI.FMK
GIN$ function code (3)

buf
Address of 9-word buffer to receive the information

siz
Buffer size in words

GI.QMC - Queue MCR Command Line

Macro Call

GIN$ GI.QMC, buf, siz

GI.QMC
GIN$ function code (4)

buf
Address of buffer containing the MCR command line

siz
Buffer size in words

GI.UAB - Get User Account Block

Macro Call

GIN$ GI.UAB, buf, siz, dev, unt

GI.UAB
GIN$ function code (5)

buf
Address of buffer to receive the UAB information

siz
Buffer size in words

dev
Device name (if blank, use task's TI:)

unt
Device unit number

GI.DEV - Get Device Information

Macro Call

GIN$ GI.DEV, buf, siz, dev, unt

GI.DEV
GIN$ function code (6)

buf
Address of buffer to receive the unit information

siz
Buffer size in words

dev
Device name (if blank, use task's TI:)

unt
Device unit number (if high bit clear, follow assignments)

GI.APR - Get System APRs

Macro Call

GIN$ GI.APR, buf, siz

GI.APR
GIN$ function code (9)

buf
Address of 97-word buffer to receive the APR information

siz
Buffer size in words

GI.TSK - Find and Return Task Information

Macro Call

GIN$ GI.TSK, buf, siz, nam1, nam2

GI.TSK
GIN$ function code (10_{10})

buf
Address of buffer to receive the task information

siz
Buffer size in words

nam1
First half of Radix–50 task name

nam2
Second half of Radix–50 task name

GI.UPD - Update UICs and Default Directory

Macro Call

GIN$ GI.UPD, buf, siz

GI.UPD
GIN$ function code (17.)

buf
Address of 5- or 32-word buffer to receive the information

siz
Buffer size in words

Get LUN Information (GLUN$)

FORTRAN Call

CALL GETLUN (lun,dat[,ids])

lun
Logical unit number

dat
A 6-word integer array to receive LUN information

ids
Directive status

Macro Call

GLUN$ lun,buf

lun
Logical unit number

buf
Address of a 6-word buffer that will receive the LUN information

Get MCR Command Line (GMCR$)

FORTRAN Call

CALL GETMCR (buf[,ids])

buf
An 80-byte array to receive the command line

ids
Directive status

Macro Call

GMCR$

Get Mapping Context (GMCX$)

FORTRAN Call

CALL GMCX (imcx[,ids])

imcx

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*24+1=193.)

ids

Directive status

Macro Call

GMCX$ wvec

wvec

The address of a vector of n Window Definition Blocks, followed by a terminator word; n is the number of window blocks in the task's header

Get Partition Parameters (GPRT$)

FORTRAN Call

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

prt

Partition name

buf

A 3-word integer array to receive the partition parameters

ids

Directive status

Macro Call

GPRT$ [prt],buf

prt

Partition name

buf

Address of a 3-word buffer

Get Region Parameters (GREG$)

FORTRAN Call

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

rid
Region id

buf
A 3-word integer array to receive the region parameters

ids
Directive status

Macro Call

GREG$ [rid],buf

rid
Region id

buf
Address of a 3-word buffer

Get Sense Switches (GSSW$S; $S form recommended)

FORTRAN Call

CALL READSW (isw)

isw
Integer to receive the console switch settings

The following FORTRAN call allows a program to read the state of a single switch:

CALL SWITCH (ibt,ist)

ibt
The switch to be tested (0 to 15)

ist
Test results where:

1 = switch on
2 = switch off

Macro Call

GSSW$S [err]

err
Error-routine address

Get Time Parameters (GTIM$)

FORTRAN Call

CALL GETTIM (ibfp[,ids])

ibfp
An 8-word integer array

ids
Directive status

Macro Call

GTIM$ buf

buf
Address of an 8-word buffer

Get Task Parameters (GTSK$)

FORTRAN Call

CALL GETTSK (buf[,ids])

buf
A 16-word integer array to receive the task parameters

ids
Directive status

Macro Call

GTSK$ buf

buf
Address of a 16-word buffer

Inhibit AST Recognition (IHAR$S; $S form recommended)

FORTRAN Call

CALL INASTR [(ids)]

ids
Directive status

Macro Call

IHAR$S [err]

err
Error-routine address

Map Address Window (MAP$)

FORTRAN Call

CALL MAP (iwdb[,ids])

iwdb
An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids
Directive status

Macro Call

MAP$ wdb

wdb
Window Definition Block address

Mark Time (MRKT$)

FORTRAN Call

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

efn
Event flag number

tmg
Time interval magnitude

tnt
Time interval unit

ids
Directive status

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

CALL WAIT (tmg,tnt,ids)

tmg
Time interval magnitude

tnt
Time interval unit

ids
Directive status

Macro Call

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

efn
Event flag number

tmg
Time interval magnitude

tnt
Time interval unit

ast
AST entry-point address

Map Supervisor D-Space (MSDS$; RSX-11M-PLUS)

FORTRAN Call

Not supported

Macro Call

MSDS$ mask

mask
A 7-bit mask with one bit corresponding to each supervisor-mode D-space APR. If the bit is set, the APR is mapped to supervisor-mode I-space. If the bit is clear, the APR is mapped to user-mode D-space. The seven bits are specified in bits 8 through 14 of the mask word.

Move to/from User/Supervisor I/D-Space (MVTS$)

FORTRAN Call

Not supported

Macro Call

MVTS$ action,addr,val [or buff]

action
One of the following:

MV.TUI Move to user I-space

MV.TUD Move to user D-space

MV.TSI Move to supervisor I-space

MV.TSD Move to supervisor D-space

MV.FUI Move from user I-space

MV.FUD Move from user D-space

MV.FSI Move from supervisor I-space

MV.FSD Move from supervisor D-space

addr

Address of the location in the task

buf

Buffer to receive the value fetched (for the move-from operations)

val

Value to be stored in the location (for the move-to operations)

Parse FCS (PFCS$)

FORTRAN Call

CALL PRSFCS ([mod],[itbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk,prssz],[dfnbk,dfnsz],[rsmsk][,idsw])

mod

Optional modifier for logical name table entries; allowable symbolic offsets are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

itbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

prbuf

Array containing the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

prsz

Size (in bytes) of the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsbuf

Array containing the resulting file specification buffer

rssz

Size (in bytes) of the resulting file specification buffer

rslen

Integer to receive the resulting string size

prsblk

Array containing the parse block

prssz

Size (in bytes) of the parse block

dfnbk

Array containing the default name block; dfnbk and dfnsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

dfnsz

Size of the default name block; dfnbk and dfnsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsmsk

Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device is not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

idsw

Integer to receive the Directive Status Word.

Macro Call

PFCS$ [mod],[tbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk],[prssz],[dfnbk],[dfnsz][,rsmsk]

mod

Optional modifier for logical name table entries; allowable symbolic offsets are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

tbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

prbuf

Address of the primary file specification buffer

prsz

Size (in bytes) of the primary file specification buffer

rsbuf

Address of the resulting file specification buffer

rssz

Size (in bytes) of the resulting file specification buffer

rslen

Address of a word to receive the resulting string size

prsblk

Address of the parse block

prssz

Size (in bytes) of the parse block

dfnbk

Address of the default name block

dfnsz

Size of the default name block

rsmsk

Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device is not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

Parse RMS (PRMS$)

FORTRAN Call

CALL PRSRMS ([mod],[itbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk,prssz],[dfbuf,dfsz],[rsmsk][,idsw])

mod

Optional modifier for logical name table entries; allowable symbolic offsets are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

itbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun

LUN to be assigned

prbuf

Array containing the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

prsz

Size (in bytes) of the primary file specification buffer; prbuf and prsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsbuf

Array containing the resulting file specification buffer

rssz

Size (in bytes) of the resulting file specification buffer

rslen

Integer to receive the resulting string size

prsblk

Array containing the parse block

prssz

Size (in bytes) of the parse block

dfbuf

Address of the default file specification buffer; dfbuf and dfsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

dfsz

Size of the default file specification buffer; dfbuf and dfsz must both be specified or both omitted; if omitted, a comma between their positions must be present unless no other parameters follow

rsmsk

Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device and directory are not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

idsw

Integer to receive the Directive Status Word.

Macro Call

PRMS$ [mod],[tbmsk],[lun],prbuf,prsz,rsbuf,rssz,[rslen],[prsblk],[prssz],[dfbuf],[dfsz],[rsmsk]

mod

Optional modifier for logical name table entries; allowable symbolic offsets are as follows:

LB.LOC = 1
LB.LOG = 2

Specifying one of these values indicates that matches in the logical table are based on the exact value. Not specifying a value indicates that the system will look for the first matching logical block, regardless of the modifier value.

tbmsk

Inhibit mask to prevent a logical table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

lun
LUN to be assigned

prbuf
Address of the primary file specification buffer

prsz
Size (in bytes) of the primary file specification buffer

rsbuf
Address of the resulting file specification buffer

rssz
Size (in bytes) of the resulting file specification buffer

rslen
Address of a word to receive the resulting string size

prsblk
Address of the parse block

prssz
Size (in bytes) of the parse block

dfbuf
Address of the default file specification buffer

dfsz
Size (in bytes) of the default file specification buffer

rsmsk
Mask of fields in the resulting string to suppress before returning the string. The bits currently defined are the same as those for the flag word in the parse block. The bits are FS$NOD, FS$DEV, FS$DIR, FS$NAM, FS$TYP, and FS$VER. If the bit FS$NDF is set, the device and directory are not defaulted to and the LUN is not assigned. (FS$NDF has no meaning for the FSS$ directive.)

Queue I/O Request (QIO$)

FORTRAN Call

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

fnc
I/O function code

lun
Logical unit number

efn
Event flag number

pri
Priority (ignored, but parameter must be present in call)

isb
A 2-word integer array to receive final I/O status

prl
A 6-word integer array containing device-dependent parameters to be placed in parameter words 1 through 6 of the DPB. Fill in this array by using the GETADR routine (see Section 1.5.2.4).

ids
Directive status

Macro Call

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

fnc
I/O function code

lun
Logical unit number

efn
Event flag number

pri
Priority (ignored, but Q.IOPR byte must be present in DPB)

isb
Address of I/O status block

ast
Address of AST service-routine entry point

prl
Parameter list of the form <P1,...P6>

Queue I/O Request and Wait (QIOW$)

FORTRAN Call

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

fnc
I/O function code

lun
Logical unit number

efn
Event flag number

pri
Priority (ignored, but parameter must be present in call)

isb
A 2-word integer array to receive final I/O status

prl
A 6-word integer array containing device-dependent parameters to be placed in parameter words 1 through 6 of the DPB

ids
Directive status

Macro Call

QIOW$ fnc,lun,[efn],[pri],[isb],[ast][,prl]

fnc
I/O function code

lun
Logical unit number

efn
Event flag number

pri
Priority (ignored, but Q.IOPR byte must be present in DPB)

isb
Address of I/O status block

ast
Address of AST service-routine entry point

prl
Parameter list of the form <P1,...P6>

Receive Data or Stop (RCST$)

FORTRAN Call

CALL RCST ([rtname],ibuf[,ids])

rtname
Sender task name (if not specified, data may be received from any task)

ibuf
Address of a 15-word buffer to receive the sender task name and data

ids
Integer to receive the Directive Status Word

Macro Call

RCST$ [tname],buf

tname
Sender task name (if not specified, data may be received from any task)

buf
Address of a 15-word buffer to receive the sender task name and data

Receive Data (RCVD$)

FORTRAN Call

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

tsk
Sender task name (if not specified, data may be received from any task)

buf
A 15-word integer array for received data

ids
Directive status

Macro Call

RCVD$ [tsk],buf

tsk
Sender task name (if not specified, data may be received from any task)

buf
Address of a 15-word buffer

Receive Data or Exit (RCVX$)

FORTRAN Call

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

tsk
Sender task name (if not specified, data may be received from any task)

buf
A 15-word integer array for received data

ids
Directive status

Macro Call

RCVX$ [tsk],buf

tsk
Sender task name (if not specified, data may be received from any task)

buf
Address of a 15-word buffer

Read All Event Flags (RDAF$)

A FORTRAN task can read only one event flag. The call is:

FORTRAN Call

CALL READEF (efn[,ids])

efn
Event flag number

ids
Directive status

The Executive returns the status codes IS.SET (+02) and IS.CLR (00) for FORTRAN calls in order to report event-flag polarity.

Macro Call

RDAF$ buf

buf
Address of a 4-word buffer

Read Event Flag (RDEF$)

FORTRAN Call

CALL READEF (iefn[,ids])

iefn
Integer containing an event flag number

ids
Integer variable to receive the Directive Status Word

Macro Call

RDEF$ efn

efn
Event flag number

Read Extended Event Flags (RDXF$)

A FORTRAN task can read only one event flag. The call is:

FORTRAN Call

CALL READEF (efn[,ids])

efn
Event flag number

ids
Directive status

The Executive returns the status codes IS.SET (+02) and IS.CLR (00) for FORTRAN calls in order to report event-flag polarity.

Macro Call

RDXF$ buf

buf
Address of a 6-word buffer

Recursive Translation of Logical Name (RLON$, RLOG$)

(CALL RCTLON and RLON$ are the preferred calls to use on RSX-11M-PLUS and Micro/RSX systems. CALL RCTLOG and RLOG$ are provided for compatibility with P/OS systems.)

FORTRAN Calls

CALL RCTLON ([mod],[itbmsk],[status],lns,lnssz,iens,ienssz,[rsize],[rtbmod][,idsw])
CALL RCTLOG ([mod],[itbmsk],[status],lns,lnssz,iens,ienssz,[rsize],[rtbmod][,idsw])

mod

Optional modifier of the logical name within a table. Ordinarily, no value would be specified to allow any defined logical name to be found.

itbmsk

Inhibit mask to prevent a logical name table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status

Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns

Character array containing the logical name string

lnssz

Size (in bytes) of the logical name string

iens

Character array buffer to contain the returned equivalence name string

ienssz

Size (in bytes) of the data area for the returned equivalence name string

rsize

Word to receive the size of the returned equivalence name string

rtbmod

Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

idsw

Integer to receive the Directive Status Word

Macro Calls

RLON$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]
RLOG$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]

mod

Optional modifier to be matched against the logical name within a table. Ordinarily, no value will be specified to allow any logical name in table to be found.

tbmsk

Inhibit mask to prevent a logical name table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status

Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns

Character array containing the original logical name string

lnssz

Size (in bytes) of the original logical name string

ens

Character array buffer to receive the returned equivalence name string

enssz

Size (in bytes) of the data area for the returned equivalence name string

rsize

Word to receive the size of the equivalence name string

rtbmod

Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

Remove Affinity (RMAF$S; RSX-11M-PLUS multiprocessor systems; $S form recommended)

FORTRAN Call

CALL RMAF [(ids)]

ids

Integer to receive the Directive Status Word

Macro Call

RMAF$S

Request and Pass Offspring Information (RPOI$)

FORTRAN Call

CALL RPOI (tname,[iugc],[iumc],[iparen],[ibuf],[ibfl],[isc],[idnam],[iunit],[itask],[ocbad][,ids])

tname

Name of an array containing the actual name (in Radix-50) of the task to be requested and optionally chained to

iugc

Name of an integer containing the group code number for the UIC of the requested target chain task

iumc

Name of an integer containing the member code number for the UIC of the requested target chain task

iparen

Name of an array (or I*4 integer) containing the Radix-50 name of the parent task. This is returned in the information buffer of the GTCMCI subroutine.

ibuf

Name of an array containing the command line text for the chained task

ibfl

Name of an integer containing the number of bytes in the command in the ibuf array

isc

Flag byte controlling the actions of this directive request when executed. The bit definitions of this byte (only the low-order byte of the integer specified in the call is ever used) are as follows:

RP.OEX	128.	Force this task to exit on successful execution of the RPOI$ directive.
RP.OAL	1	Pass all of this task's connections to the requested task. (The default is none).
RP.ONX	2	Pass the first connection in the queue, if there is one.

idnam

Name of an integer containing the ASCII name of the requested task's TI: (must be the name of a physical device)

iunit

Name of an integer containing the unit number of the requested task's TI:

itask

Name of an array containing the Radix-50 name the requested task is to run under.

Any task may specify a new name for the requested task as long as the requested task is not a CLI task.

The requested task (specified in the tname parameter) must be installed in the ...tsk format.

ocbad

Name of an integer containing the internal pool address of the parent OCB. This value may be obtained only in the information buffer of the GTCMCI subroutine, which only a CLI can issue; therefore, only a CLI can specify this argument.

ids

Integer to receive the Directive Status Word

Macro Call

RPOI$ tname,,,,[ugc],[umc],[parent],[bufadr],[buflen],[sc],[dnam],[unit],[task],[ocbad]

tname

Name of task to be chained to

ugc

Group code for the UIC of the requested task

umc

Member code for the UIC of the requested task

parent

Name of issuing task's parent task whose connection is to be passed

bufadr

Address of buffer to be given to the requested task

buflen

Length of buffer to be given to the requested task

sc

Flag bits:

RP.OEX	(200)	Force issuing task to exit.
RP.OAL	(1)	Pass all connections (default is none.)
RP.ONX	(2)	Pass the first connection in the queue, if there is one.

dnam

ASCII name for TI: (must be the name of a physical device)

unit

Unit number of task's TI:

task

Radix-50 name of task to be started.

Any task may specify a new name for the requested task as long as the requested task is not a CLI task.

The requested task (specified in the tname parameter) must be installed in the ...tsk format.

ocbad

Address of OCB to pass (CLIs only)

Request Task (RQST$)

FORTRAN Call

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

tsk

Task name

opt

A 4-word integer array:

opt(1)	Partition name, first half (ignored, but must be present)
opt(2)	Partition name, second half (ignored, but must be present)
opt(3)	Priority (ignored, but must be present)
opt(4)	User Identification Code

ids

Directive status

Macro Call

RQST$ tsk,[prt],[pri][,ugc,umc]

tsk
Task name

prt
Partition name (ignored, but must be present)

pri
Priority (ignored, but must be present)

ugc
UIC group code

umc
UIC member code

Receive By Reference (RREF$)

FORTRAN Call

CALL RREF (iwdb,[isrb][,ids])

iwdb
An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

isrb
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.

ids
Directive status

Macro Call

RREF$ wdb

wdb
Window Definition Block address

Receive By Reference or Stop (RRST$)

FORTRAN Call

CALL RRST (iwdb,[isrb][,ids])

iwdb

An 8-word integer array containing a Window Definition Block

isrb

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.

ids

Directive status

Macro Call

RRST$ wdb

wdb

Window Definition Block address

Resume Task (RSUM$)

FORTRAN Call

CALL RESUME (tsk[,ids])

tsk

Task name

ids

Directive status

Macro Call

RSUM$ tsk

tsk

Task name

Run Task (RUN$)

FORTRAN Call

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

tsk

Task name

opt

A 4-word integer array:

opt(1) Partition name, first half (ignored, but must be present)

opt(2) Partition name, second half (ignored, but must be present)

opt(3) Priority (ignored, but must be present)

opt(4) User Identification Code

smg

Schedule delta magnitude

snt

Schedule delta unit (either 1, 2, 3, or 4)

rmg

Reschedule interval magnitude

rnt

Reschedule interval unit

ids

Directive status

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

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

tsk

Task name

smg

Schedule delta magnitude

snt

Schedule delta unit (either 0, 1, 2, 3, or 4)

ids

Directive status

Macro Call

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

tsk

Task name

prt

Partition name (ignored, but must be present)

pri

Priority (ignored, but must be present)

ugc
UIC group code

umc
UIC member code

smg
Schedule delta magnitude

snt
Schedule delta unit (either 1, 2, 3, or 4)

rmg
Reschedule interval magnitude

rnt
Reschedule interval unit

Specify Command Arrival AST (SCAA$)

FORTRAN Call

Not supported

Macro Call

SCAA$ [ast]

ast
AST service-routine entry point. Omitting this parameter disables command arrival ASTs for the issuing task until the directive is respecified.

Supervisor Call (SCAL$S; RSX-11M-PLUS systems; $S form recommended)

FORTRAN Call

Not supported

Macro Call

SCAL$S saddr,caddr[,err]

saddr
Address of the called supervisor-mode routine

caddr
Address of the completion routine for return to the caller

err
Address of error routine (see Section 1.4.3 for more information)

Set Command Line Interpreter (SCLI$)

FORTRAN Call

CALL SETCLI (icli,idev,iunit[,ids])

icli
Name of a 2-word array element containing the name of the CLI to which the terminal is to be set

idev
Name of an integer containing the ASCII name of the terminal to be set (default = TI:)

iunit
Name of an integer containing the unit number of the terminal

ids
Directive status

Macro Call

SCLI$ cli,[dev],[unit]

cli
Name of the CLI to which the terminal is to be set

dev
ASCII name of the terminal to be set (default = TI:)

unit
Unit number of the terminal

Send Data (SDAT$)

FORTRAN Call

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

tsk
Task name

buf
A 13-word integer array of data to be sent

efn
Event flag number

ids
Directive status

Macro Call

SDAT$ tsk,buf[,efn]

tsk

Task name

buf

Address of a 13-word data buffer

efn

Event flag number

Set Default Directory (SDIR$)

FORTRAN Call

CALL SETDDS (mod,iens,ienssz[,idsw])

mod

Modifier for the SDIR$ directive; specify one of the following values:

0	Modify task default
SD.LOG	Modify terminal default
SD.BYE	Delete terminal default
SD.TI	Set task default to terminal default

iens

Character array containing the default directory string

ienssz

Size (in bytes) of the default directory string

idsw

Integer to receive the Directive Status Word

Macro Call

SDIR$ { mod
,end,enssz
mod,ens,enssz } (must choose one of these options)

mod

Modifier for the SDIR$ directive; specify one of the following values:

0	Modify task default
SD.LOG	Modify terminal default
SD.BYE	Delete terminal default
SD.TI	Set task default to terminal default

ens

Buffer address of the default directory string; if not specified, the default directory string is deleted (ens and enssz must be selected to modify the default)

enssz

Size (in bytes) of the default directory string (enssz and ens must be selected to modify the default)

Send, Request, and Connect (SDRC$)

FORTRAN Call

```
CALL SDRC  (rtname,ibuf,[iefn],[iast],[iesb],[iparm][,ids])
CALL SDRCN  (rtname,ibuf,[iefn],[iast],[iesb],[iparm][,ids])
```

rtname

Target task name of the offspring task to be connected

ibuf

Name of a 13-word send buffer

iefn

Event flag to be set when the offspring task exits or emits status

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL SDRCN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

iparm

Name of a word to receive the status block address when an AST occurs

ids

Integer to receive the Directive Status Word

Macro Call

SDRC$ tname,buf,[efn],[east],[esb]

tname

Target task name of the offspring task to be connected

buf

Address of a 13-word send buffer

efn

The event flag to be cleared on issuance and when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Send Data Request and Pass Offspring Control Block (SDRP$)

FORTRAN Call

CALL SDRP (task,ibuf,[ibfl],[iefn],[iflag],[iparen],[iocbad][,ids])

task

Name of an array (REAL, INTEGER, I*4) containing the Radix–50 name of the target task

ibuf

Name of an integer array containing data to be sent

ibfl

Name of an integer containing the number of words (integers) in the array to be sent. This argument may be in the range of 1 to 255_{10}. If this argument is not specified, a default value of 13_{10} is assumed.

iefn

Name of an integer containing the number of the event flag to be set when this directive is executed successfully

iflag

Name of an integer containing the flag bits controlling the execution of this directive. They are defined as follows:

SD.REX	128.	Force this task to exit upon successful execution of this directive
SD.RAL	1	Pass all connections to the requested task (default is pass none); if you specify this flag, do not specify the parent task name
SD.RNX	2	Pass the first connection in the queue, if there is one, to the requested task; if you specify this flag, do not specify the parent task name

iparen

Name of an array containing the Radix–50 name of the parent task whose connection should be passed to the target task. The name of the parent task was returned in the information buffer of the GTCMCI subroutine.

iocbad

Name of an integer containing the pool address of the OCB to pass. This value was returned in the information buffer of the GTCMCI subroutine. Only CLI tasks may specify this parameter.

ids

Name of an integer to receive the contents of the Directive Status Word

Macro Call

SDRP$ task,bufadr,[buflen],[efn],[flag],[parent],[ocbad]

task

Name of task to be chained to

bufadr

Address of buffer to be given to the requested task

buflen

Length of buffer to be given to the requested task

efn

Event flag number

flag

Flag bits controlling the execution of this directive (see iflag, above, for the definitions of the bits)

parent

Name of issuing task's parent task whose connection is to be passed. If not specified, all connections or no connections are passed, depending on the flag bit.

ocbad

Address of OCB to pass (CLIs only)

Set Event Flag (SETF$)

FORTRAN Call

CALL SETEF (efn[,ids])

efn
Event flag number

ids
Directive status

Macro Call

SETF$ efn

efn
Event flag number

Specify Floating Point Processor Exception AST (SFPA$)

FORTRAN Call

Not supported

Macro Call

SFPA$ [ast]

ast
AST service-routine entry-point address

Send Message (SMSG$)

FORTRAN Call

CALL SMSG (itgt,ibuf,ibufl,iprm,iprml[,ids])

itgt
Name of an integer containing the target object (currently, only SM.SER is defined)

ibuf
Name of an integer array containing the data to be inserted into the formatted data packet

ibufl
Name of an integer containing the length of the ibuf array

iprm
Name of an integer array containing any additional parameters

iprml
Name of an integer containing the number of parameters in the iprm array

ids
Name of an optional integer to receive the directive status

Macro Call

SMSG$ tgt,buf,len, <pri,...,prn>

tgt
Target identifier

buf
Address of an optional data buffer

len
Length in bytes of the optional data buffer

pri,...,prn
Target-specific (for the Error Logger) parameter list:

SMSG$ SM.SER,buf,len, <typ,sub,lun,mask>

typ
Error Logger packet code

sub
Error Logger packet subtype code

lun
Logical unit number of the device

msk
Control mask word

Send Next Command (SNXC$)

FORTRAN Call

CALL SNXC ([dnam][,iunit][,ids])

dnam
Device name (ASCII); if not specified, TI: is used

iunit
Unit number of the terminal from which the command is to be sent

ids
Integer to receive the Directive Status Word

Macro Call

SNXC$ [dnam][,unum]

dnam
Device name (ASCII); if not specified, TI: is used

unum
Unit number of the terminal from which the command is to be sent

Specify Parity Error AST (SPEA$)

FORTRAN Call

Not supported

Macro Call

SPEA$ [ast]

ast
AST service-routine entry-point address

Suspend (SPND$S; $S form recommended)

FORTRAN Call

CALL SUSPND [(ids)]

ids
Directive status

Macro Call

SPND$S [err]

err
Error-routine address

Specify Power Recovery AST (SPRA$)

FORTRAN Call

To establish an AST:

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

sub

Name of a subroutine to be executed upon power recovery. The PWRUP subroutine will effect the following call:

CALL sub (no arguments)

The subroutine is called as a result of a power recovery AST, and therefore may be controlled at critical points through the use of DSASTR (or INASTR) and ENASTR subroutine calls.

To remove an AST:

CALL PWRUP

Macro Call

SPRA$ [ast]

ast

AST service-routine entry-point address

Spawn (SPWN$)

FORTRAN Call

CALL SPAWN (rtname,[iugc],[iumc],[iefn],[iast],[iesb],[iparm],[icmlin,icmlen],[iunit],[dnam][,ids])
CALL SPAWNN (rtname,[iugc],[iumc],[iefn],[iast],[iesb],[iparm],[icmlin,icmlen],[iunit],[dnam][,ids])

rtname

Name (Radix-50) of the offspring task to be spawned

iugc

Group code number for the UIC of the offspring task

iumc

Member code number for the UIC of the offspring task

iefn

Event flag to be set when the offspring task exits or emits status

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL SPAWNN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

iparm
Name of a word to receive the status block address when the AST occurs

icmlin
Name of a command line to be queued for the offspring task

icmlen
Length of the command line; maximum length is 255_{10}

iunit
Unit number of terminal to be used as the TI: for the offspring task. If the optional dnam parameter is not specified, this parameter must be the unit number of a virtual terminal created by the issuing task; if a value of 0 is specified, the TI: of the issuing task is propagated. A task must be a privileged task or a CLI task in order to specify a TI: other than the parent task's TI:.

dnam
Device name mnemonic (must be the name of a physical device). If not specified, the virtual terminal specified by iunit is used as TI:.

ids
Integer to receive the Directive Status Word

Macro Call

SPWN$ tname,,,[ugc],[umc],[efn],[east],[esb],[cmdlin,cmdlen],[unum],[dnam]

tname
Name (Radix–50) of the offspring task to be spawned

ugc
Group code number for the UIC of the offspring task

umc
Member code number for the UIC of the offspring task

efn
The event flag to be cleared on issuance and set when the offspring task exits or emits status

east
Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

cmdlin

Address of a command line to be queued for the offspring task

cmdlen

Length of the command line; maximum length is 255_{10}

unum

Unit number of terminal to be used as the TI: for the offspring task. If the optional dnam parameter is not specified, this parameter must be the unit number of a virtual terminal created by the issuing task; if a value of 0 is specified, the TI: of the issuing task is propagated. A task must be a privileged task or a CLI task in order to specify a TI: other than the parent task's TI:.

dnam

Device name mnemonic (must be the name of a physical device). If not specified, the virtual terminal specified by unum is used as TI:.

Specify Receive Data AST (SRDA$)

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.

Macro Call

SRDA$ [ast]

ast

AST service-routine entry-point address

Specify Requested Exit AST (SREA$; SREX$)

FORTRAN Call

CALL SREA (ast[,ids])

ast
Name of the externally declared AST subroutine

ids
Name of an optional integer to receive the Directive Status Word

Format

CALL SREX (ast,ipblk,ipblkl,[dummy][,ids])

ast
Name of the externally declared AST subroutine

ipblk
Name of an integer array to receive the trap-dependent parameters

ipblkl
Number of parameters to be returned into the ipblk array

dummy
Reserved for future use

ids
Name of an optional integer to receive the Directive Status Word

Macro Call

SREA$ [ast]
SREX$ [ast][,dummy]

ast
AST service-routine entry-point address

dummy
Reserved for future use

Send By Reference (SREF$)

FORTRAN Call

CALL SREF (tsk,[efn],iwdb,[isrb][,ids])

tsk
A single-precision, floating-point variable containing the name of the receiving task in Radix-50 format

efn
Event flag number

iwdb
An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

isrb
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)

ids
Directive status

Macro Call

SREF$ task,wdb[,efn]

task
Name of the receiver task

wdb
Window Definition Block address

efn
Event flag number

Specify Receive-By-Reference AST (SRRA$)

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.

Macro Call

SRRA$ [ast]

ast
AST service-routine entry-point address (0)

Set Affinity (STAF$; RSX-11M-PLUS multiprocessor systems)

FORTRAN Call

CALL STAF (iaff[,ids])

iaff
Affinity mask word

ids
Integer to receive the Directive Status Word

Macro Call

STAF$ [cp!ub!ub...]

cp
CPU selected (A through D)

ub
UNIBUS run or runs selected (E through T)

Set System Time (STIM$)

FORTRAN Call

CALL SETTIM (ibufn[,ibufp][,ids])

ibufn
An 8-word integer array—new time-specification buffer

ibufp
An 8-word integer array—previous time buffer

ids
Directive status

Macro Call

STIM$ bufn,[bufp]

bufn
Address of the new 8-word time-specification buffer

bufp
Address of the 8-word buffer to receive the previous system-time parameters

Stop for Logical OR of Event Flags (STLO$)

FORTRAN Call

CALL STLOR (ef1,ef2,ef3...,efn)
CALL STLORS (idsw,ef1,ef2,ef3...,efn)

idsw
Integer to receive the Directive Status Word

ef1...efn
List of event flag numbers

Macro Call

STLO$ grp,msk

grp
Desired group of event flags

msk
A 16-bit mask word

Stop (STOP$S; $S form recommended)

FORTRAN Call

CALL STOP [(ids)]

ids
Integer to receive the Directive Status Word

Macro Call

STOP$S

Stop for Single Event Flag (STSE$)

FORTRAN Call

CALL STOPFR (iefn[,ids])

iefn
Event flag number

ids
Integer to receive the Directive Status Word

Macro Call

STSE$ efn

efn
Event flag number

Specify SST Vector Table for Debugging Aid (SVDB$)

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.

Macro Call

SVDB$ [adr][,len]

adr
Address of the SST vector table

len
Length of (that is, number of entries in) the table in words

Specify SST Vector Table for Task (SVTK$)

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.

Macro Call

SVTK$ [adr][,len]

adr
Address of the SST vector table

len
Length of (that is, number of entries in) the table in words

Switch State (SWST$)

FORTRAN Call

Not supported

Macro Call

SWST$ base,addr

base
The base virtual address within the task for mapping the subroutine through APR 5

addr
Virtual address of the subroutine to be executed in system state by the directive

Test for Specified Task Feature (TFEA$)

FORTRAN Call

CALL TFEA (isym,idsw)

isym

Symbol for the specified task feature

idsw

Integer to receive the Directive Status Word

Macro Call

TFEA$ sym

sym

Symbol for the specified task feature

Translate Logical Name (TLON$, TLOG$)

(CALL TRALON and TLON$ are the preferred calls to use on RSX-11M-PLUS and Micro/RSX systems. CALL TRALOG and TLOG$ are provided for compatibility with P/OS systems.)

FORTRAN Calls

CALL TRALON ([mod],[tbmsk],[status],lns,lnssz,ens,ienssz,[rsize],[rtbmod][,idsw])
CALL TRALOG ([mod],[tbmsk],[status],lns,lnssz,ens,ienssz,[rsize],[rtbmod][,idsw])

mod

Optional modifier of the logical name within a table. Ordinarily, no value would be specified to allow any defined logical name to be found.

tbmsk

Inhibit mask to prevent a logical name table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status

Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns

Character array containing the logical name string

lnssz

Size (in bytes) of the logical name string

ens

Character array buffer to contain the returned equivalence string

ienssz

Size (in bytes) of the data area for the returned equivalence name string

rsize

Word to receive the size of the returned equivalence name

rtbmod

Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

idsw

Integer to receive the Directive Status Word

Macro Calls

TLON$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]
TLOG$ [mod],[tbmsk],[status],lns,lnssz,ens,enssz,[rsize],[rtbmod]

mod

Optional modifier to be matched against the logical name within a table. Ordinarily, no value would be specified to allow any logical name in table to be found.

tbmsk

Inhibit mask to prevent a table from being searched. The following symbol bit definitions, when set, prevent a particular table from being searched:

System	(IN.SYS)	10
Group	(IN.GRP)	4
Session	(IN.SES)	20
Task	(IN.TSK)	1

If no mask is specified (or a value of 0 is specified), the tables are searched in the following order: task, session, group, system. The tables are searched in this order for each iteration. The values remain constant for all iterations of a logical name translation.

status

Word to receive the logical status associated with the located logical name:

LS.TRM	1	Terminal status bit. Indicates the last logical name in list required no further translation.
LS.PRV	2	Privileged status. Last logical name in list can be deleted only by a privileged task.

lns
Character array containing the original logical name string

lnssz
Size (in bytes) of the original logical name string

ens
Character array to contain the returned equivalence string

enssz
Size (in bytes) of the data area for the returned equivalence name string

rsize
Word to receive the size of the returned equivalence name; this size is always the actual size of the equivalence name regardless of the string size specified with enssz

rtbmod
Word to receive, in the lower byte, the table number and, in the higher byte, the modifier value of the located logical name

Unlock Group Global Event Flags (ULGF$; $S form recommended)

FORTRAN Call

CALL ULGF [(ids)]

ids
Directive status

Macro Call

ULGF$S [err]

err
Error-routine address

Unmap Address Window (UMAP$)

FORTRAN Call

CALL UNMAP (iwdb[,ids])

iwdb
An 8-word integer array containing a Window Definition Block (see Section 3.5.2.2)

ids
Directive status

Macro Call

UMAP$ wdb

wdb
Window Definition Block address

Unstop Task (USTP$)

FORTRAN Call

CALL USTP ([rtname][,ids])

rtname
Name of the task to be unstopped (if not specified, CALL USTP will use the issuing task as its default)

ids
Integer to receive directive status information

Macro Call

USTP$ [tname]

tname
Name of the task to be unstopped (if not specified, USTP$ will use the issuing task as its default)

Variable Receive Data (VRCD$)

FORTRAN Call

CALL VRCD ([task],bufadr,buflen[,idsw])

task
Sender task name

bufadr
Address of the buffer to receive the sender task name and data (must be word-aligned (INTEGER*2))

buflen
Length of the buffer

idsw
Integer to receive the Directive Status Word

Macro Call

VRCD$ [task],bufadr[,buflen],[ti]

task
Sender task name

bufadr
Buffer address

buflen
Buffer size in words

ti
TI: indicator (ignored on RSX systems)

Variable Receive Data or Stop (VRCS$)

FORTRAN Call

CALL VRCS ([task],bufadr,[buflen][,ids])

task
Sender task name

buf
Address of the buffer to receive the sender task name and data

buflen
Length of the buffer

ids
Integer to receive the Directive Status Word

Macro Call

VRCS$ [task],bufadr[,buflen],[ti]

task
Sender task name

bufadr
Buffer address

buflen
Buffer size in words

ti
TI: indicator (ignored on RSX systems)

Variable Receive Data or Exit (VRCX$)

FORTRAN Call

CALL VRCX ([task],bufadr,[buflen][,ids])

task
Sender task name

bufadr
Address of the buffer to receive the sender task name and data

buflen
Length of the buffer

ids
Integer to receive the Directive Status Word

Macro Call

VRCX$ [task],bufadr[,buflen],[ti]

task
Sender task name

bufadr
Buffer address

buflen
Buffer size in words

ti
TI: indicator (ignored on RSX systems)

Variable Send Data (VSDA$)

FORTRAN Call

CALL VSDA ([task],bufadr,[buflen],[efn][,idsw])

task
Receiver task name

bufadr
Address of the buffer to receive the sender task name and data (must be word-aligned (INTEGER*2))

buflen
Length of the buffer

efn
Event flag number

idsw
Integer to receive the Directive Status Word

Macro Call

VSDA$ [task],bufadr,[buflen],[efn],[spri],[ti]

task
Receiver task name

bufadr
Buffer address

buflen
Buffer size in words

efn
Event flag number

spri
Send priority (ignored on RSX systems)

ti
TI: indicator (ignored on RSX systems)

Variable Send, Request, and Connect (VSRC$)

FORTRAN Call

CALL VSRC (rtname,ibuf,[ibuflen],[iefn],[iast],[iesb],[iparm][,idsw])
CALL VSRCN (rtname,ibuf,[ibuflen],[iefn],[iast],[iesb],[iparm][,idsw])

rtname
Target task name of the offspring task to be connected

ibuf
Name of send buffer

ibuflen
Length of the buffer

iefn
Event flag to be set when the offspring task exits or emits status

iast

Name of an AST routine to be called when the offspring task exits or emits status (ignored for CALL VSRCN)

iesb

Name of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

iparm

Name of a word to receive the status block address when an AST occurs

idsw

Integer to receive the Directive Status Word

Macro Call

VSRC$ tname,buf[,buflen],[efn],[east],[esb]

tname

Target task name of the offspring task to be connected

buf

Address of send buffer

buflen

Length of the buffer

efn

The event flag to be cleared on issuance and set when the offspring task exits or emits status

east

Address of an AST routine to be called when the offspring task exits or emits status

esb

Address of an 8-word status block to be written when the offspring task exits or emits status:

Word	0	Offspring-task exit status
Word	1	TKTN abort code
Words	2-7	Reserved

Wait for Significant Event (WSIG$; $S form recommended)

FORTRAN Call

CALL WFSNE

Macro Call

WSIG$S [err]

err
Error-routine address

Wait for Logical OR of Event Flags (WTLO$)

FORTRAN Call

CALL WFLOR (ef1,ef2,ef3...,efn)
CALL WFLORS (idsw,ef1,ef2,ef3...,efn)

ef1...efn
List of event flag numbers taken as the set of flags to be specified in the directive

idsw
Integer to receive the Directive Status Word

Macro Call

WTLO$ grp,msk

grp
Desired group of event flags

msk
A 16-bit flag mask word

Wait for Single Event Flag (WTSE$)

FORTRAN Call

CALL WAITFR (efn[,ids])

efn
Event flag number

ids
Directive status

Macro Call

WTSE$ efn

efn

Event flag number

Appendix B
Standard Error Codes

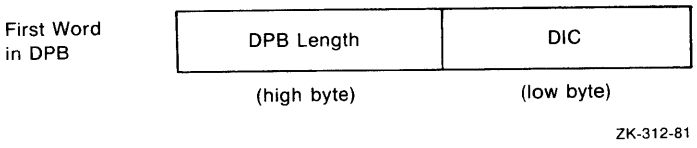
The symbols listed below are associated with the directive status codes returned by the RSX-11M-PLUS and Micro/RSX Executive. They are determined (by default) at task-build time. To include these in a MACRO-11 program, use the following two lines of code:

```
.MCALL  DRERR$
DRERR$

;
; Standard error codes returned by directives in the Directive Status
; Word (DSW)
;
   IS.CLR   +00    Event flag was clear
   IS.SUC   +01    Operation complete, Success
   IS.SET   +02    Event flag was set

;
;
;
   IE.UPN   -01.   Insufficient dynamic storage
   IE.INS   -02.   Specified task not installed
   IE.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.ITN   -20.   Invalid table number
IE.LNF   -21.   Logical name not found

;
;
;
IE.AST   -80.   Directive issued/not issued from AST
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 (>255.)

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 (>250.)

IE.ILU   -96.   Invalid LUN

IE.IEF   -97.   Invalid event flag number (>64.)
IE.ADP   -98.   Part of DPB out of user's space
IE.SDP   -99.   DIC or DPB size invalid
```

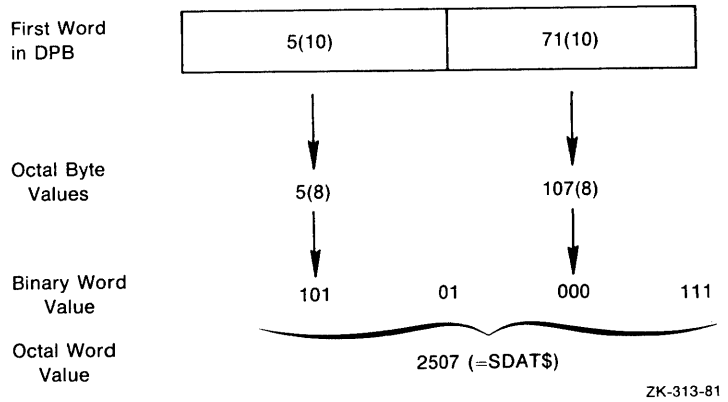
Appendix C
Directive Identification Codes

Directive Identification Codes (DICs) are used to identify each directive. The DIC appears in the low byte of the first (or only) word in the Directive Parameter Block (DPB). The DPB length (in words) appears in the high byte of the first DPB word. Thus, both bytes make up the word format shown below:



The remainder of this appendix contains a listing of directives arranged in numerical sequence, according to the octal value for the first DPB word. In addition, the DIC and DPB lengths are included as decimal values as they appear in Chapter 5.

This list can be used as a software debugging aid to quickly identify directives based on the octal value of the first word in a DPB. An example for the SDAT$ directive is provided below, illustrating the manner in which the octal value is obtained:



Octal Value for DPB First Word	Directive (Macro Call)	Decimal DIC	Values for DPB Length
433	CMKT$	27.	1.
443	DECL$S	35.	1.
455	SPND$S	45.	1.
461	WSIG$S	49.	1.
463	EXIT$S	51.	1.
537	DSCP$S	95.	1.
541	ENCP$S	97.	1.
543	DSAR$S or IHAR$S	99.	1.
545	ENAR$S	101.	1.
563	ASTX$S	115.	1.
575	GSSW$S	125.	1.
603	STOP$S	131.	1.
637	ULGF$S	159.	1.
643	RMAF$S	163.	1.
1015	STAF$	13.	2.
1025	SRRA$	21.	2.
1035	EXST$	29.	2.
1037	CLEF$	31.	2.
1041	SETF$	33.	2.
1045	RDEF$	37.	2.
1047	RDAF$	39.	2.
1051	WTSE$	41.	2.
1065	EXIF$	53.	2.
1067	CRRG$	55.	2.
1071	ATRG$	57.	2.
1073	DTRG$	59.	2.
1075	GTIM$	61.	2.
1077	GTSK$	63.	2.
1121	RREF$	81.	2.

Octal Value for DPB First Word	Directive (Macro Call)	Decimal DIC	Values for DPB Length
1153	SRDA$	107.	2.
1155	SPRA$	109.	2.
1157	SFPA$	111.	2.
1161	GMCX$	113.	2.
1165	CRAW$	117.	2.
1171	MAP$	121.	2.
1173	UMAP$	123.	2.
1207	STSE$	135.	2.
1227	ELVT$	151.	2.
1235	CRGF$	157.	2.
1237	ELGF$	159.	2.
1241	STAF$	161.	2.
1245	SPEA$	165.	2.
1247	SREA$	167.	2.
–	GIN$[1]	169.	–
1255	SCAA$	173.	2.
1261	FEAT$	177.	2.
1311	MSDS$	201.	2.
1321	TFEA$	209.	2.
1325	RRST$	213.	2.
1405	GLUN$	5.	3.
1431	CSRQ$	25.	3.
1433	CMKT$	27.	3.
1447	RDXF$	39.	3.
1453	WTLO$	43.	3.
1457	RSUM$	47.	3.
1475	STIM$	61.	3.
1523	ABRT$	83.	3.

[1] The octal value for the DPB first word and the value for the DPB length depend upon the macro expansion for each individual function.

Octal Value for DPB First Word	Directive (Macro Call)	Decimal DIC	Values for DPB Length
1531	EXTK$	89.	3.
1547	SVDB$	103.	3.
1551	SVTK$	105.	3.
1577	SNXC$	127.	3.
1605	USTP$	133.	3.
1611	STLO$	137.	3.
1617	CNCT$	143.	3.
1633	SCAL$S	155.	3.
1647	SREX$	167.	3.
1657	SWST$	175.	3.
1715	CPCR$	205.	3.
2007	ALUN$	7.	4.
2011	ALTP$	9.	4.
2101	GPRT$ or GREG$	65.	4.
2113	RCVD$	75.	4.
2115	RCVX$	77.	4.
2213	RCST$	139.	4.
2223	EMST$	147.	4.
2313	MVTS$	203.	4.
2427	MRKT$	23.	5.
2505	SREF$	69.	5.
2507	SDAT$	71.	5.
2625	CRVT$	149.	5.
2655	SCLI$	173.	5.
2717	ACHN$	207.	5.
2717	DLON$ or DLOG$	207.	5.
2717	SDIR$	207.	5.
3113	VRCD$	75.	6.
3115	VRCX$	77.	6.

Octal Value for DPB First Word	Directive (Macro Call)	Decimal DIC	Values for DPB Length
3213	VRCS$	139.	6.
3317	GDIR$	207.	6.
3413	RQST$	11.	7.
3577	GCCI$	127.	7.
3601	CINT$	129.	7.
3615	SDRC$	141.	7.
3655	GCII$	173.	7.
3717	CLON$ or CLOG$	207.	7.
3717	FSS$	207.	7.
4107	VSDA$	71.	8.
4215	VSRC$	141.	8.
4253	SMSG$	171.	8.
4615	SDRP$	141.	9.
5317	RLON$ or RLOG$	207.	10.
5317	TLON$ or TLOG$	207.	10.
5421	RUN$	17.	11.
6001	QIO$	1.	12.
6003	QIOW$	3.	12.
6413	SPWN$	11.	13.
6717	PFCS$	207.	13.
6717	PRMS$	207.	13.
7013	SPWN$	11.	14.
10013	RPOI$	11.	16.
24577	GMCR$	127.	41.

Index

A

C

D

E

F

G

H

I

L

M

P

Q

R

S

T

U

V

W

RSX–11M–PLUS and Micro/RSX
Executive Reference Manual
AA–JS17A–TC

READER'S COMMENTS

Your comments and suggestions are welcome and will help us in our continuous effort to improve the quality and usefulness of our documentation and software.

Remember, the system includes information that you read on your terminal: help files, error messages, prompts, and so on. Please let us know if you have comments about this information, too.

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.

_____

_____

_____

_____

_____

_____

What kind of user are you? ____ Programmer ____ Nonprogrammer

Years of experience as a computer programmer/user: _____

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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ---

RSX-11M-PLUS and Micro/RSX
Executive Reference Manual
AA-JS17A-TC

READER'S COMMENTS

Your comments and suggestions are welcome and will help us in our continuous effort to improve the quality and usefulness of our documentation and software.

Remember, the system includes information that you read on your terminal: help files, error messages, prompts, and so on. Please let us know if you have comments about this information, too.

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.

_____

_____

_____

_____

_____

_____

What kind of user are you? ___ Programmer ___ Nonprogrammer

Years of experience as a computer programmer/user: _____

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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35
110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ---