

May 1979

This manual contains general information on run-time systems and describes RSTS/E Monitor and RSX Emulator directives for the assembly-language programmer.

**RSTS/E
System Directives
Manual**

Order No. AA-D748A-TC

OPERATING SYSTEM AND VERSION: RSTS/E V7.0

SOFTWARE VERSION: RSTS/E V7.0

To order additional copies of this document, contact the Software Distribution Center, Digital Equipment Corporation, Maynard, Massachusetts 01754

digital equipment corporation • maynard, massachusetts

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

The software described in this document is furnished under a license and may only be used or copied 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 or its affiliated companies.

Copyright© 1979 Digital Equipment Corporation

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

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10

CONTENTS

PREFACE	xi
CHAPTER 1 INTRODUCTION	1-1
1.1 DISCUSSION OF RUN-TIME SYSTEMS	1-1
1.2 DISCUSSION OF JOB	1-3
1.3 ORGANIZATION OF THIS MANUAL	1-4
CHAPTER 2 GENERAL RSTS/E ENVIRONMENT	2-1
2.1 HOW RSTS/E ALLOCATES MEMORY -- ACTUAL AND VIRTUAL ADDRESSING	2-1
2.2 JOB SPACE -- HIGH SEGMENT AND LOW SEGMENT ..	2-4
2.3 IMPORTANT INSTALLATION OPTIONS	2-6
2.3.1 The "Disappearing" RSX Run-Time System	2-6
2.3.2 Resident Libraries	2-7
2.4 LOW-SEGMENT DETAILS -- FIRST 1000 BYTES OF THE LOW SEGMENT	2-8
2.5 HIGH-SEGMENT DETAILS -- PSEUDO VECTORS	2-16
2.5.1 Run-Time System Capability and Default Definitions	2-19
2.5.2 Synchronous System Trap Addresses	2-24
2.5.3 Asynchronous System Trap Addresses	2-26
2.5.4 Entry Points	2-28
CHAPTER 3 GENERAL MONITOR DIRECTIVES	3-1
3.1 INTRODUCTION	3-1
3.1.1 Summary of General Monitor Directives	3-1
3.1.2 Prefix File COMMON.MAC	3-3
3.1.3 Error Mnemonics: Link-File ERR.STB	3-7
3.1.4 Programming Hints	3-7
3.2 CALFIP -- Call the File Processor	3-9
3.2.1 ASSFQ (Assign a Device)	3-10
3.2.2 CLSFQ (Close a Channel)	3-13
3.2.3 CRBFQ (Create a Binary (Executable) File and Open It on a Channel ..	3-16

3.2.4	CREFQ (Create a File and Open It on a Channel)	3-22
3.2.5	CRTFQ (Create and Open a Temporary File)	3-30
3.2.6	DALFQ (Deassign All Devices)	3-35
3.2.7	DEAFQ (Deassign a Device)	3-37
3.2.8	DIRFQ (Get Directory Information)	3-40
3.2.9	DLNFQ (Delete a File)	3-48
3.2.10	ERRFQ (Return Error Message Text)	3-51
3.2.11	LOKFQ (Disk File/Wildcard Lookup)	3-54
3.2.12	OPNFQ (Open a File/Device on a Channel)	3-64
3.2.13	RENFQ (Rename a File)	3-71
3.2.14	RSTFQ (Reset a Channel)	3-74
3.2.15	UUOFQ (Hook to File Processor) ...	3-77
3.3	.CCL -- Check String for CCL Command	3-78
3.4	.CHAIN -- Execute Under Same RTS	3-83
3.5	.CLEAR -- Clear Keyword Bits	3-84
3.6	.CORE -- Change Memory Size	3-86
3.7	.DATE -- Return Current Date and Time	3-89
3.8	.ERLOG -- Log an Error from RTS	3-91
3.9	.EXIT -- Exit to System Default RTS	3-92
3.10	.FSS -- Check File Specification String	3-93
3.11	.LOGS -- Check for Logical Devices	3-105
3.12	.MESAG -- Message Send/Receive	3-112
3.12.1	Declare Receiver Subfunction	3-113
3.12.2	Remove Receiver Subfunction	3-114
3.12.3	Send Local Data Message Subfunction	3-115
3.12.4	Send Connect Initiate Subfunction	3-117
3.12.5	Send Connect Confirm Subfunction	3-122
3.12.6	Send Connect Reject Subfunction ..	3-124
3.12.7	Send Network Data Message Subfunction	3-126
3.12.8	Send Interrupt Subfunction	3-129
3.12.9	Send Link Service Subfunction	3-132
3.12.10	Send Disconnect Subfunction	3-135
3.12.11	Send Link Abort Subfunction	3-137
3.12.12	Receive Subfunction	3-139

3.13	.NAME -- Install Program Name with Monitor	3-159
3.14	.PEEK -- Look at Monitor Memory	3-161
3.15	.PLAS -- Access Resident Library	3-163
3.15.1	ATRFQ (Attach Resident Library) ..	3-164
3.15.2	CRAFQ (Create Address Window)	3-168
3.15.3	DTRFQ (Detach Region)	3-174
3.15.4	ELAFQ (Eliminate Address Window) ..	3-178
3.15.5	MAPFQ (Map Address Window)	3-182
3.15.6	UMPFQ (Unmap Address Window)	3-187
3.16	.POSTN -- Return Current Horizontal Position	3-191
3.17	.READ -- Read Data from File or Device	3-193
3.18	.RSX -- Execute Job and Disappear (RSX only)	3-199
3.18.1	No Requests for Extra Space	3-199
3.18.2	Request for Mapping a Window in APR 7	3-200
3.18.3	Request to Extend the User Job Image	3-200
3.19	.RTS -- Pass Control to New RTS	3-201
3.20	.RUN -- New Program to Run	3-207
3.21	.SET -- Set Keyword Bits	3-211
3.22	.SLEEP -- Suspend Job	3-213
3.23	.SPEC -- Special Functions for I/O	3-215
3.24	.STAT -- Return Job Statistics	3-226
3.25	.TIME -- Return Timing Information	3-228
3.26	.TTAPE -- Enter Tape Mode	3-230
3.27	.TTDDT -- Disable Full-Line Buffering	3-231
3.28	.TTECH -- Undo .TTAPE or .TTNCH	3-232
3.29	.TTNCH -- Stop Echo	3-233
3.30	.TTRST -- Restart Output	3-234
3.31	.ULOG -- Assign/Reassign/Deassign Device or Enter/Deassign User Logical	3-235
3.31.1	UU.ASS (Assign/Reassign a Device, or Enter User Logical	3-236

3.31.2	UU.DEA -- Deassign a Device or User Logical	3-243
3.31.3	UU.DAL -- Deassign All Devices and User Logicals	3-247
3.32	.UUO -- Execute BASIC-PLUS SYS Call	3-250
3.32.1	UU.ACT (Accounting Information Dump)	3-253
3.32.2	UU.ASS (Assign/reassign Device)....	3-254
3.32.3	UU.ATR (Read/Write File Attributes)	3-255
3.32.4	UU.ATT (Attach/Reattach Job)	3-257
3.32.5	UU.BCK (Change File Statistics) ...	3-259
3.32.6	UU.BYE (Logout)	3-260
3.32.7	UU.CCL (CCL Command Add/Delete) ..	3-261
3.32.8	UU.CHE (Enable/Disable Disk Caching)	3-263
3.32.9	UU.CHU (Change Password/Quota, Disable Terminal, Kill Job)	3-265
3.32.10	UU.CLN (Clean Up a Disk Pack)	3-268
3.32.11	UU.DAL (Deassign All Devices)	3-269
3.32.12	UU.DAT (Change System Date/Time)	3-270
3.32.13	UU.DEA (Deassign Device)	3-271
3.32.14	UU.DET (Detach)	3-272
3.32.15	UU.DIE (Special Shutup Logout) ...	3-273
3.32.16	UU.DIR (Directory Lookup)	3-274
3.32.17	UU.DLU (Delete User Account)	3-278
3.32.18	UU.ERR (Return Error Messages) ...	3-279
3.32.19	UU.FCB (Get Open Channel Statistics (FCB/DDB))	3-281
3.32.20	UU.FIL (File Placement and Modification)	3-283
3.32.21	UU.HNG (Hang Up a Dataset)	3-285
3.32.22	UU.JOB (Job Creation)	3-286
3.32.23	UU.LIN (Login)	3-288
3.32.24	UU.LOG (Set Number of Logins)	3-290
3.32.25	UU.LOK (Disk Directory Lookup by Filename/Wildcard Lookup)	3-292
3.32.26	UU.MNT (Disk Pack and Terminal Status)	3-296
3.32.27	UU.NAM (Associate a Run-Time System with a File)	3-298
3.32.28	UU.NLG (Disable Further Logins) ..	3-299
3.32.29	UU.PAS (Create User Account)	3-301
3.32.30	UU.POK (Poke Memory)	3-302
3.32.31	UU.PPN (Wildcard PPN Lookup)	3-303
3.32.32	UU.PRI (Change Priority/Run Burst/ Size)	3-305
3.32.33	UU.RAD (Read or Read-and-Reset Accounting Data)	3-306
3.32.34	UU.RTS (Add/Remove/Load/Unload Run- Time System or Resident Library..	3-308
3.32.35	UU.SLN (System Logical Names)	3-317
3.32.36	UU.SWP (Add/Remove System Files) ..	3-320

3.32.37	UU.SYS (Return Job Status Information)	3-322
3.32.38	UU.TB1 (Get Monitor Tables -- Part I)	3-325
3.32.39	UU.TB2 (Get Monitor Tables -- Part II)	3-327
3.32.40	UU.TRM (Set Terminal Characteristics)	3-329
3.32.41	UU.YLG (Enable Logins)	3-331
3.32.42	UU.ZER (Zero Device)	3-333
3.33	.WRITE -- Write Data to File or Device	3-334
CHAPTER 4	RSX RUN-TIME SYSTEM ENVIRONMENT	4-1
4.1	INTRODUCTION	4-1
4.1.1	Advantage: Transportable Code ...	4-1
4.1.2	General Services	4-2
4.1.3	RSX Directive Emulation within the RSTS/E Monitor	4-2
4.2	SYSTEM MACRO LIBRARY	4-3
4.3	DIRECTIVE PROCESSING	4-3
4.4	DIRECTIVE FORMS -- \$, \$C, \$S -- AND THEIR EXPANSIONS	4-5
4.4.1	\$ Form (and DIR\$ Directive)	4-6
4.4.2	\$C Form	4-8
4.4.3	\$S Form	4-9
4.5	FIRST 1000 BYTES OF LOW SEGMENT FOR RSX	4-10
CHAPTER 5	RSX EMULATOR DIRECTIVES	5-1
5.1	INTRODUCTION	5-1
5.2	ABRT\$ -- Abort	5-4
5.3	ALUN\$ -- Assign Logical Unit Number	5-5
5.4	ASTX\$ -- AST Service Exit	5-7
5.5	ATRG\$ -- Attach Resident Library	5-9
5.6	CRAW\$ -- Create Address Window	5-12
5.7	DTRG\$ -- Detach Resident Library	5-17
5.8	ELAW\$ -- Eliminate Address Window	5-19
5.9	EXIT\$ -- Task Exit	5-21

5.10	EXST\$ -- Exit with Status	5-22
5.11	EXTK\$ -- Extend Task	5-23
5.12	GLUN\$ -- Get LUN Information	5-25
5.13	GMCR\$ -- Get MCR (CCL) Command Line	5-27
5.14	GPRT\$ -- Get Partition (Job) Parameters	5-28
5.15	GTIM\$ -- Get Time Parameters	5-30
5.16	GTSK\$ -- Get Task (Job) Parameters	5-32
5.17	MAP\$ -- Map Address Window	5-34
5.18	QIO\$ and QIOW\$ -- Queue I/O Request (and Wait)	5-37
5.19	SCCA\$\$ -- Specify Control/C AST	5-44
5.20	SFPA\$ -- Specify Floating Point Processor Exception Address	5-45
5.21	SVDB\$ -- Specify SST Vector Table for Debugging Aid	5-46
5.22	SVTK\$ -- Specify SST Vector Table for Task ..	5-48
5.23	UMAP\$ -- Unmap an Address Window	5-50
5.24	WSIG\$ -- Wait for Significant Event Flag ...	5-52
5.25	WTSE\$ -- Wait for Single Event Flag	5-53
APPENDIX A	FULL LIST OF ERRORS	A-1
APPENDIX B	DISK DEVICE SIZES	B-1
APPENDIX C	SUPPLEMENTARY RSX DIRECTIVES FOR RESIDENT LIBRARIES	C-1
C.1	RDB Directives	C-1
C.2	WDB Directives	C-2

TABLES

TABLE	3-1	Summary of General Monitor Directives	3-2
	3-2	Data Input with .READ	3-194
	3-3	Special Functions for Magtape	3-221
	3-4	.UUO Subfunctions -- Calls to the File Processor (FIP)	3-251
	3-5	Data Output with .WRITE	3-335
	5-1	Vertical Format Control Characters	5-40

FIGURES

FIGURE		
2-1	How an Actual Address is Formed	2-2
2-2	Memory Mapping with the APRS	2-3
2-3	Job Area in Virtual Memory	2-5
2-4	First 1000 Bytes of Low Segment	2-9
2-5	Format of Pseudo-Vector Region of High Segment	2-18
4-1	General Form of the Directive Parameter Block	4-3
4-2	Example of RSX Directive Forms	4-5
4-3	First 1000 Bytes of Low Segment for RSX	4-11



PREFACE

This manual describes directives to the RSTS/E Monitor and the RSX Emulator that can be used in MACRO programs. We assume here that you are familiar with the MACRO-11 Assembly Language. MACRO is the standard assembler for Digital Equipment Corporation PDP-11 computers, and is available under various operating systems for the PDP-11. The syntax is basically the same for all operating systems, and is described in the

PDP-11 MACRO-11 Language Reference Manual

Useful information about the RSTS/E system is given in the

RSTS/E System User's Guide

BASIC-PLUS Language Manual

RSTS/E Programming Manual

RSTS/E Programmer's Utilities Manual

RSTS/E Task Builder Reference Manual

These manuals are liberally referenced here.

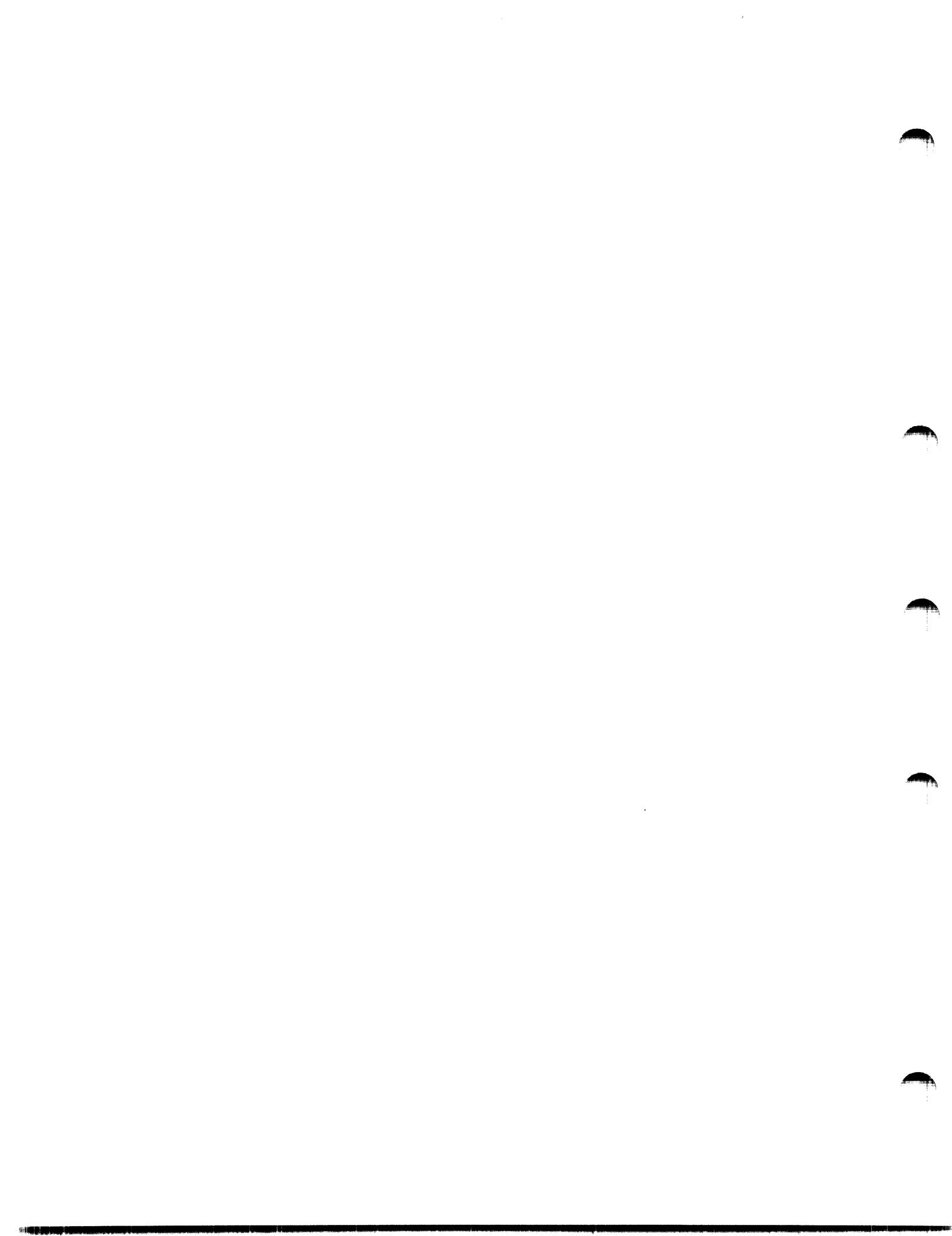
The System Manager sets certain parameters which affect the Monitor, and so affect the Monitor directives. Where relevant, the

RSTS/E System Generation Manual

RSTS/E System Manager's Guide

are also referenced here.

Specific page references are not given, since these may change with updates. Items referred to are indexed in the manual cited.



CHAPTER 1

INTRODUCTION

As an assembly-language programmer, you should be aware that a MACRO Assembler is available to RSTS/E jobs under the RSX Run-Time System and also under the RT11 Run-Time System. Thus, you will be using either the RSX Run-Time System or the RT11 Run-Time System to assemble and, in most cases*, run your program. (If you wish to modify a run-time system, or code your own, your code will, when installed, be running under the direct control of the Monitor.) In any case, to use the Monitor directives described in this manual, you will need to know something about the Monitor, run-time systems, and jobs in general under RSTS/E.

1.1 DISCUSSION OF RUN-TIME SYSTEMS

There are several ways to look at run-time systems. From the system design viewpoint, a run-time system is a way to implement code which, when it is resident in memory, can be shared by many users. In a time-sharing system such as RSTS/E, this is an important consideration. Run-time systems are normally implemented as "pure code"; that is, as a series of instructions and fixed data only, containing no variable data. Such re-entrant code saves space, since many jobs can use it, and time, since run-time systems need not be copied to and from disk in the way that user programs are swapped in and out of memory. At least one run-time system is permanently resident. It is called the system default run-time system, since the Monitor will execute it for a job when no specific run-time system has been asked for. Other run-time systems are loaded when requested, remain as long as necessary, and are removed when they are no longer in use. Since they contain no variable data, they need not be swapped out to disk; they are simply reloaded when they are needed again.

*A set of run-time systems are based on the RSX Run-Time System. These derivatives of RSX include BASIC2, BP2COM, RMS11, and other run-time systems used by optional languages such as RPGII. A MACRO routine assembled under the RSX Run-Time System can be run under these derivatives.

To a programmer, a run-time system can be regarded as:

1. A logical division of the RSTS/E system according to capabilities provided. For example:
 - The BASIC-PLUS Run-Time System provides a BASIC-PLUS interpreter which will translate and execute BASIC-PLUS statements as they are typed, or store them as a file on disk for later execution.
 - The RSX Run-Time System simulates parts of the RSX-11M operating system environment. With the RSX Run-Time System and its derivatives, you can compile programs in COBOL or BASIC-PLUS-2, and assemble MACRO-11 programs with the MAC assembler. You can then link such compiled or assembled object modules with the Task Builder (TKB) to form a file which is executable under the RSX Run-Time System.
 - The RT11 Run-Time System provides an environment similar to that available under the RT11 operating system. It provides a MACRO-11 assembler (called MACRO) and a linker (called LINK) to link compiled or assembled object modules to form an executable file.
2. A physical division of the RSTS/E system into units which, although sharable, usually take up space in the user's work area. As described in Chapter 2, the RSTS/E Monitor regards the run-time system as part of the 32K words allocated for each user's job space in memory.
 - The BASIC-PLUS Run-Time System requires 16K words of the user's job space. In most installations, this is the system default run-time system.
 - The RSX Run-Time System requires 4K words of the user's job space. In most installations, this is a non-resident run-time system. It is re-entrant and can be used by more than one user at a time. It is loaded into memory from disk when called for, and remains as long as anyone is using it. It is removed when it is no longer needed, and reloaded as necessary. An installation option allows the RSX Run-Time System to "disappear" in certain situations, such that an executing program can use the space normally taken by the RSX Run-Time System.
 - The RT11 Run-Time System requires 4K words of the user's job space and is non-resident in most installations. It is also re-entrant, and it is loaded, removed, and reloaded as necessary.

(Other run-time systems are available for RSTS/E. The BASIC-PLUS Run-Time System is mentioned here for comparison, as it is the system default run-time system in most RSTS/E installations. The RSX and RT11 Run-Time Systems both provide access to a MACRO Assembler.)

To a MACRO programmer, the run-time system provides an interface to the Monitor. For example, the RSTS/E Monitor contains I/O routines and drivers. A MACRO program can issue direct Monitor calls to perform I/O. A program can also use different calls which are recognized by the RSX Run-Time System at execution time. Code within the run-time system then analyzes the operation desired and performs one or more of the direct calls to the Monitor.

So, there are various aspects to consider in choosing the run-time system under which you wish to code MACRO-11. If you are coding a subroutine to be called by another program or programs, all modules will have to be compiled or assembled, linked, and run under the same run-time system (RSX or RT11).

The Monitor directives available vary, also. One set of directives are processed directly by the Monitor (Chapter 3). These directives will be compiled properly by the MACRO Assembler under either the RSX or RT11 Run-Time System. They will execute properly as part of a user program or as part of a run-time system. Another set of directives will be compiled correctly only by the MACRO Assembler under the RSX Run-Time System, and will execute properly only within a user program operating under control of the RSX Run-Time System (Chapter 5).

In other words, if you wish to modify one of the existing run-time systems, or code your own run-time system, you could use either MACRO Assembler, with the general Monitor directives described in Chapter 3. You could not, however, use the RSX directives described in Chapter 5.

Furthermore, if you are modifying or coding your own run-time system, you must store the run-time system file (the product of assembling and linking) on the system disk in account [0,1] in "saved image library" format. The routines which accomplish this are MAKSIL (for run-time systems assembled with MAC and linked with TKB) and SILUS (for run-time systems assembled with MACRO and linked with LINK).

1.2 DISCUSSION OF JOB

Like run-time systems, a "job" can be viewed from several angles. A job is a unit of work to the RSTS/E Monitor, generally associated with activity at a terminal on the system. Suppose, for example, that a user types a line at a logged-out terminal. The Monitor creates a job, assigning a job number and allocating internal tables for bookkeeping. The Monitor then passes control to a "new-user" entry point in the system default run-time system. The system default run-time system has code at this entry point which causes the LOGIN

utility to be loaded from the system disk and executed. LOGIN actually analyzes what was typed, and performs the normal log-in dialogue. When LOGIN exits, control returns to the system default run-time system, which waits for further input from the terminal. The Monitor regards the execution of the run-time system and LOGIN (and whatever else occurs at the terminal until it is logged out) as the same "job". (If the sequence was not a valid log-in, LOGIN exits with the job still logged out. The Monitor destroys "the job" and releases the job number.)

As a MACRO programmer, your awareness of the concept of "job" will probably center around the amount of memory RSTS/E provides for work space for a job, and the fact that the run-time system can take part of this work space. The allocation of work space is described in Chapter 2.

1.3 ORGANIZATION OF THIS MANUAL

Chapter 3 describes the general Monitor directives which can be used in programs compiled under either the RSX or RT11 Run-Time System. Chapter 5 describes the directives processed by the RSX Emulator in the RSX Run-Time System. Each of these chapters is preceded by a chapter giving relevant information on the general environment for the directive set.

CHAPTER 2

GENERAL RSTS/E ENVIRONMENT

To understand how and why one copy of a run-time system, shared by many users, can still take up space in each user's work area, we must go into some background on memory accessing in the PDP-11 (Section 2.1) and how RSTS/E uses it to define a job space, or work area in memory, for each user to run programs (Section 2.2). Section 2.3 briefly describes the special-case "disappearing" RSX Run-Time System, and resident libraries. Sections 2.4 and 2.5 give specifics on certain areas in the job space that are used by the Monitor, the run-time system, and the user program.

2.1 HOW RSTS/E ALLOCATES MEMORY -- ACTUAL AND VIRTUAL ADDRESSING

All RSTS/E systems use the memory management feature available on PDP-11/34, 40, 45, 50, 55, 60, and 70 computers. This feature extends the addressable memory range of the PDP-11 processor by using hardware registers called Active Page Registers (APRs).

The PDP-11 processor handles 16-bit operand addresses, allowing reference to 32K words. (Remember that the PDP-11 is byte-addressable, so the address range is from 0 through $2^{16}-1 = 64K$ bytes, or 32K words.) With the memory management unit, a 16-bit address is treated as a relocatable (virtual) address which is combined with information in an APR to form an 18-bit (22-bit, for the PDP-11/70) actual address.

The Processor Handbook for the PDP-11 processors tell how the APRs function in detail. Briefly, an APR consists of two 16-bit registers. These registers define a "page" of contiguous memory. The Page Address Register (PAR) defines an actual memory location where the page begins. The Page Descriptor Register (PDR) defines, among other things, the maximum length of the page and how it can be accessed (read/write, read-only, and so forth).

Figure 2-1 shows how a virtual address and a Page Address Register are combined to form an actual address in physical memory. The 16-bit virtual address defines which one of eight Active Page Registers is to be used, and a byte offset within the page. The Page Address Register

of the indicated APR is handled as though it contains bits 6-17 (6-21 for the PDP-11/70) of an 18-bit (or 22-bit) physical address, defining the start of the page.

In the figure, the virtual address of 072322 (octal) identifies APR 3, and byte 12322 of the page defined by APR 3. The PAR of APR 3 indicates a starting address of 146000 for the page. The actual address obtained is 146000+012322, or 160322.

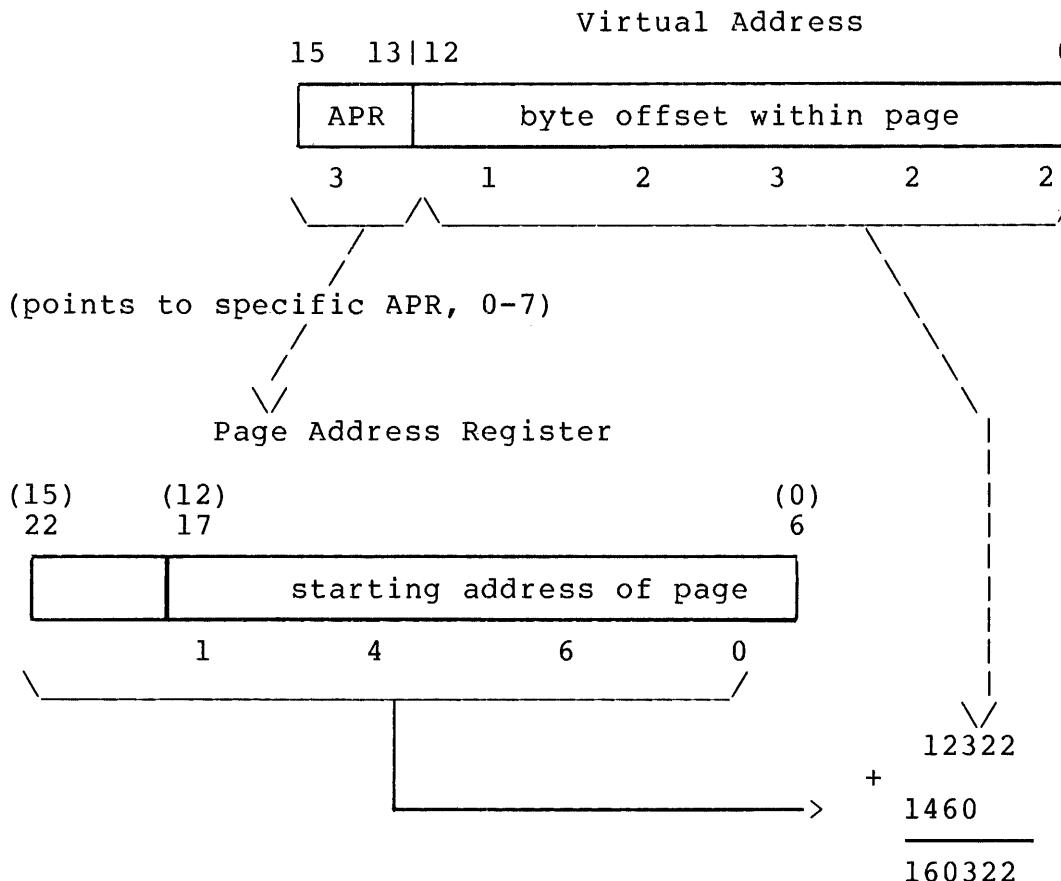


Figure 2-1 How an Actual Address is Formed

The byte offset field in the virtual address is 13 bits long. The maximum size of a page, then, is 2^{13} bytes, or 4096 words. In other words, one APR can "map" a virtual address range of up to 4K words into an equal extent of actual memory.

The memory management unit on the PDP-11 consists of two sets of APRs*, eight in each set. Since each APR can map a 4K segment of virtual memory to actual memory, each set of APRs can provide access to 32K words of actual memory.

*The PDP-11/45 and PDP-11/70 have three sets of APRs; the additional set for "supervisor mode" mapping.

The Monitor uses one set, called the "kernel mode" APRs, to map itself in physical memory. It uses the other set, called the "user mode" APRs, to map the job that is active during the current time slice of time-shared processing. Figure 2-2 illustrates the concept of mapping through the APRs.

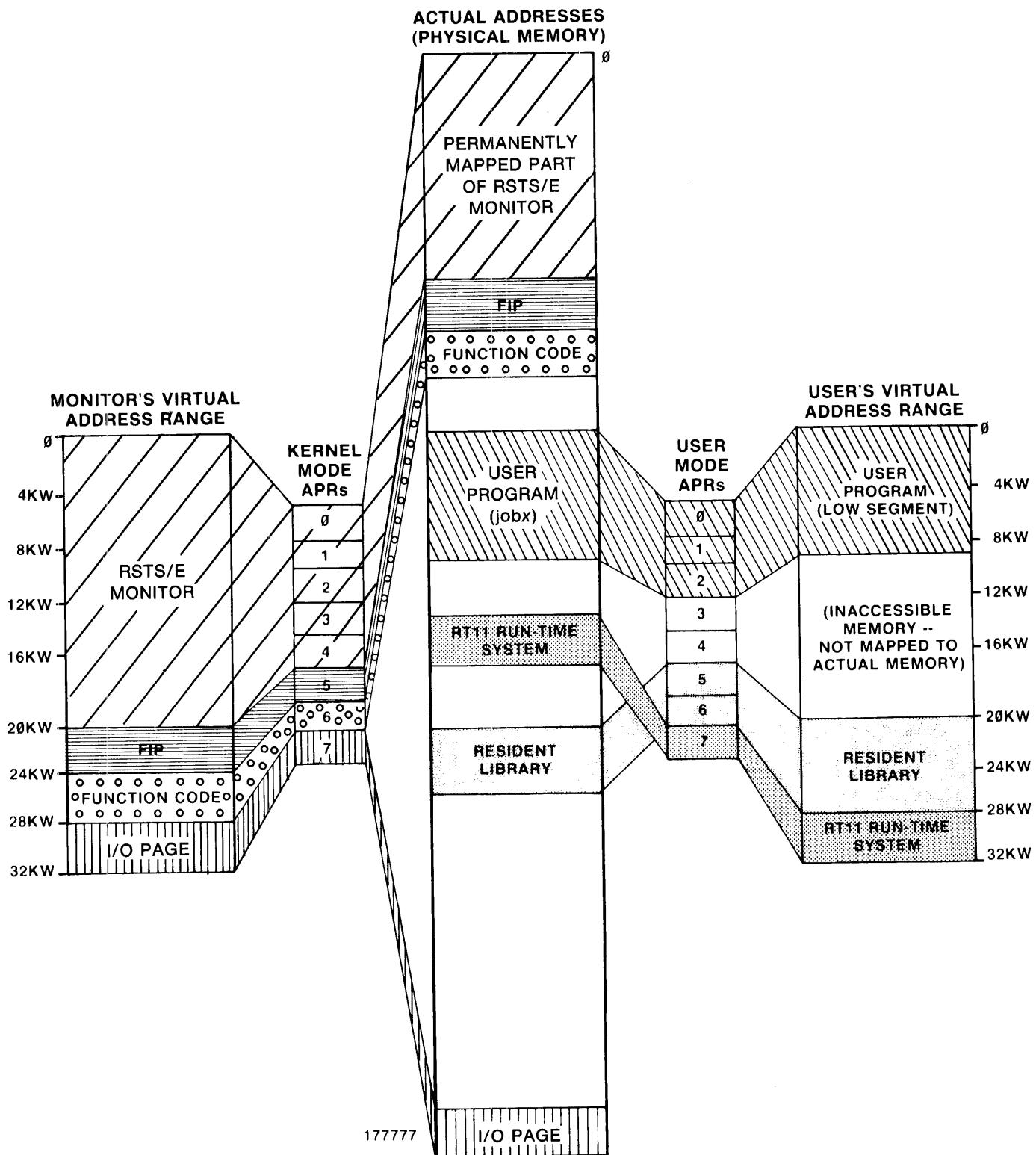


Figure 2-2 Memory Mapping with the APRs

2.2 JOB SPACE -- HIGH SEGMENT AND LOW SEGMENT

The RSTS/E Monitor is designed to handle work requested by a user through an interface: the run-time system. For example, the BASIC-PLUS, RSX, and RT11 run-time systems (available as part of a RSTS/E system) each provide their own keyboard monitor, to accept and process user commands. These run-time systems also contain code to handle their own set of directives, accepting and expanding user program calls to the Monitor. For example, the RSX run-time system provides I/O calls to the Monitor (phrased in terms of logical units and records) which the run-time system itself translates and executes as the more device-oriented calls handled directly by the Monitor.

Thus, the run-time system communicates with both the user program and the Monitor. Execution control passes back and forth between these three entities; data is passed between them using established ranges of virtual addresses that are the same for each job. The Monitor, then, needs to be able to access both the run-time system and the user job image during any given time slice. It does this by setting up the run-time system as part of the 32K words accessible through the eight user APRs.

The Monitor assigns an area for the run-time system in the high portion of virtual address space, called the "high segment". The low portion of virtual address space or "low segment" belongs to the "user job image"; that is, to the utility program, compiler, assembler, or executable user program that is currently being executed for the job. (As part of its housekeeping for each job, the Monitor keeps track of where the currently appropriate run-time system is, where the user job image is, and what the values were in the Program Counter register (PC), Program Status Word (PSW), and other job-context information at the end of the last time slice. Before the next time slice for the job, then, the Monitor simply loads the APRs with the correct values for the job, and loads the PC, PSW, and so forth, so that execution continues where it left off.)

In any case, the high segment, or run-time system, takes some multiple of 4K words of virtual address space, due to the APR mapping as discussed in Section 2.1. The BASIC-PLUS Run-Time System, for example, may take from 13 to 16K words of physical memory, depending on options selected when the system is installed. Even though the physical memory required may only be 13K words, it still requires four APRs to map this range, leaving four APRs, or a maximum of 16K words, for a user program running under the BASIC-PLUS Run-Time System.

The Monitor uses certain areas within the high segment and the low segment to get information from the job defining what work the Monitor is to do for it, and to pass information back to the job. Figure 2-3 illustrates the job area in virtual addresses. The first 1000 (octal) bytes are used to pass information between the Monitor, the run-time system, and the user job image for certain types of Monitor directives. The "pseudo-vector" region in high virtual memory is used by the Monitor to determine, for example, where control is to be passed when a job is initially entered. The run-time system sets this area with entry points and values to define itself to the Monitor.

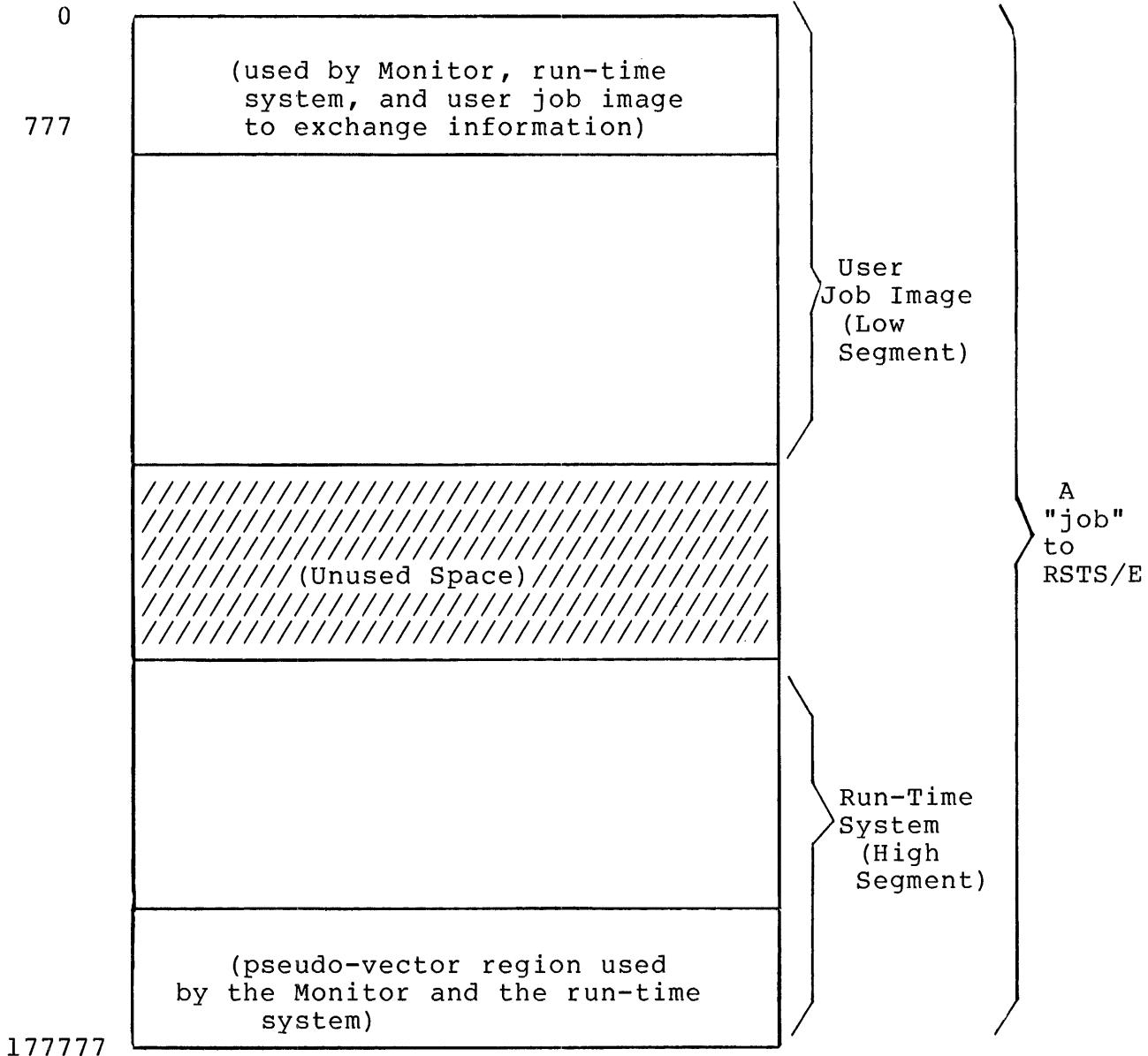


Figure 2-3 Job Area in Virtual Memory

Some detail on these areas are given in the following subsections of this chapter. Read Section 2.4 if you are interested in using the general Monitor directives described in Chapter 3. The RSX Run-Time System directives set up the first 1000 bytes of memory for you if you are only using the directives described in Chapter 5. Similarly, Section 2.5 will be of interest mainly if you wish to code your own run-time system or modify one of the existing ones, and need to know about the pseudo-vector region.

2.3 IMPORTANT INSTALLATION OPTIONS

2.3.1 The "Disappearing" RSX Run-Time System

When a RSTS/E system is generated, the system manager has the option of installing the "emulation" code of the RSX Run-Time System as a part of the Monitor. When this is done, the emulation code -- the part that processes the RSX directives in Chapter 5 and handles traps -- is permanently resident in memory. It is not permanently mapped; the Monitor maps this section of code using kernel mode APR 5 when requested to do so by the "user command processing" code in the RSX Run-Time System.

The whole RSX Run-Time System still exists as a file that is loaded from disk when necessary, remains as long as anyone is using it, and is removed when it is no longer needed. When someone at a terminal types a command to run a program which was, for example, assembled with MAC and linked with TKB, the Monitor passes control to the RSX Run-Time System to load the program. Once the program is loaded and ready for execution, however, the whole RSX Run-Time System is no longer needed. The emulation portion of the code, which must be present to process the execution of RSX directives and traps that might occur during execution of the program, is in the RSTS/E Monitor. The RSX Run-Time system passes control to the Monitor, with a request to "disappear" from the high segment of the user job space. When it gives control to the Monitor, the RSX Run-Time System passes on any requests from the user or from the program itself to make use of the high-segment space which is freed when the run-time system disappears.

The Monitor decreases the count of current users of the RSX Run-Time System, and frees the physical memory taken by the full RSX Run-Time System if no one else is using it and it was not installed as permanently resident. The Monitor then maps the RSX emulation code (using kernel mode APR 5), sets up the user APRs and other registers according to the information passed to it by the RSX Run-Time System, and passes control to the program.

Up to this point, the program itself is still limited to 28K words -- user APR 7 was needed for the RSX Run-Time System to process the command and load the program. Now, however, the program can execute directives to expand itself -- up to 31K words.* Or, it can access resident libraries of routines or data with user APR 7.

*Internal structures inherent in the RSTS/E Monitor limit the expansion to 31K words, rather than 32K words.

2.3.2 Resident Libraries

The system manager also has the option of installing the capability to handle resident libraries in the RSTS/E system.

A resident library, like a run-time system, can be shared by many user programs. In fact, resident libraries in RSTS/E have many parallels with run-time systems, both in concept and implementation.

The underlying concept of both run-time systems and resident libraries is "sharable". The difference lies in their purpose. Run-time systems allow user programs to share code which extends the capabilities of the Monitor. A user program runs under the control of a run-time system. A resident library, on the other hand, extends the capabilities of the user program. A user program can pass control to a routine within a resident library, or access data in a resident library.

As with run-time systems, a file is defined as a resident library by the system manager with UTILITY (RSTS/E System Manager's Guide). A resident library can be defined as permanently resident; that is, it will always be in physical memory. Or, a resident library can be defined such that the Monitor loads it from disk when necessary (when a job requests its use, or "attaches" to the resident library). The resident library will then remain in memory as long as at least one job is attached to it. It will be removed when no jobs are using it and the space is needed for something else.

You can access a resident library of sharable routines or data most easily by using the Task Builder (TKB).* The Task Builder options allow you to link the resident library to your executable program in such a way that your program can reference mnemonic entry points and locations in the resident library.

However, system directives also exist (used by the Task Builder itself) by which a user program (user job image) can attach itself to a resident library, create a window of virtual addresses to refer to locations in the library, and map the virtual addresses to all or some portion of the memory occupied by the resident library.

*See the description of the COMMON, LIBR, RESCOM, and RESLIB options in the RSTS/E Task Builder Reference Manual.

As with run-time systems, APRs are used to create the window of addresses which are mapped to actual memory locations. So, windows to access resident libraries take up space in the job area in 4K word units. Such windows cannot overlap the user job image (low segment), and cannot overlap the run-time system (high segment) unless it is the RSX Run-Time System which has been installed so that it can "disappear". When an installation chooses to use the RMS routines as a resident library, it must also choose to install RSX so that it can disappear. The RMS library requires user APRs 6 and 7.

The illustration of memory mapping in Figure 2-2, earlier in the chapter, shows a resident library mapped as part of a job running with the RT11 Run-Time System.

2.4 LOW-SEGMENT DETAILS -- FIRST 1000 BYTES OF THE LOW SEGMENT

The first 1000 (octal) bytes of virtual address space in the low segment have special meaning attached to them by the Monitor. This space is automatically allocated by the RSX task builder and RT11 linker; relocatable addresses assigned by these programs always begin at location 1000 (octal) unless you request otherwise. If you wish to use the general Monitor directives described in Chapter 3, your program must fill parts of this area with information for the Monitor, and the Monitor passes information back in this area. Rather than use octal addresses, you can use the COMMON.MAC prefix file, described in Section 3.1.2, to assign mnemonic names to commonly used addresses and offsets. COMMON.MAC does not allocate space, but rather assigns mnemonic names to areas within the first 1000 (octal) bytes of virtual address space. Using the mnemonics assigned with COMMON.MAC will make the code more readable and easier to maintain.

The general regions in this area are shown in Figure 2-4. Note that a Run-Time System may use some of the areas differently when it assumes control. The RSX Run-Time System, for example, uses the memory labeled "default SP stack area" in Figure 2-4 as a table of logical units. The Task Builder automatically generates a "user stack" after the first 1000 bytes of virtual address space. (Section 4.5 briefly describes how RSX uses the first 1000 bytes.)

If you use the general directives in Chapter 3, you should only reference the areas that are shown with mnemonics (provided by COMMON.MAC). The mnemonics to the right in Figure 2-4 are assigned through COMMON.MAC.

(controlled solely by job -- user job image or run-time system)	0
used by Monitor for job context information to make job swappable	60
used by Monitor for hardware floating point context information to make job swappable	110
default SP stack area	170
keyword	400 KEY USRSP
file request queue block	402 FIRQB
transfer request block	442 XRB
	460 CORCMN
core common area	
(controlled solely by job)	660
user-assignable project,programmer number	734 USRPPN
user-assignable default protection code	736 USRPRT
user logical device name table	740 USRLOG
	776

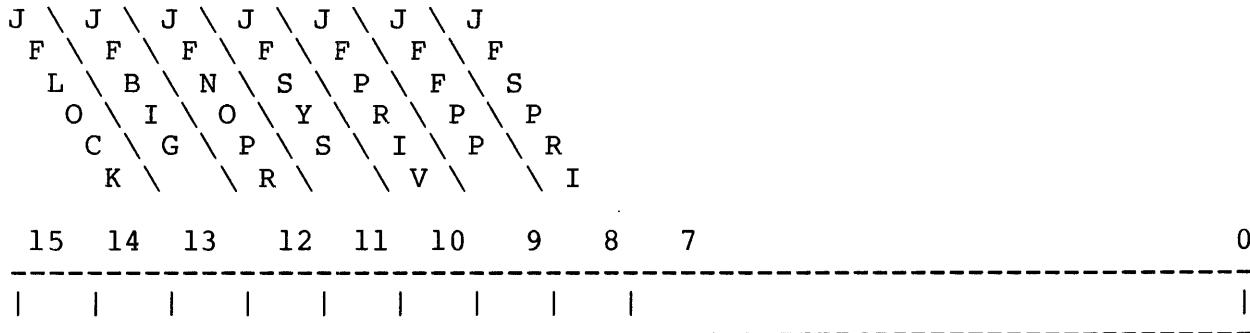
Figure 2-4 First 1000 Bytes of Low Segment

A general description of the named areas follows. The general Monitor calls in Chapter 3 describe specific formats for the areas the calls use.

KEY

The keyword defines the job's status in the time-sharing environment, for example, the job's privilege. Bits in the keyword can be set and cleared by the Monitor or by the job (either the run-time system or the user job image). The job can manipulate some bits in the keyword with the .SET and .CLEAR directives (Sections 3.21 and 3.5).

The keyword is "refreshed" by the Monitor at certain points, for example, when a run-time system is entered with the intent to load and execute a program file in the user job image (.RUN directive, Section 3.20). For a keyword refresh, the Monitor clears bit 15 and bits 7-0, and sets the remaining bits to indicate the job's current status. Only seven bits are significant to the Monitor. The rest can be used by the job in whatever manner it wishes.



- JFLOCK Set to 1 indicates that the job does not wish to be swapped. This bit can be .SET and .CLEARed. When set, the only normal condition which will cause the job to be swapped is when the job asks for a memory size expansion (see .CORE, Section 3.6), and there is not enough room to do the expansion where the job now is in memory.
- JFBIG Set to 1 indicates that the job can exceed its private memory maximum (see .CORE directive, Section 3.6). This bit is implicitly set every time a privileged program is run (by either a privileged or a non-privileged job). It can be .SET and .CLEARed.
- JFNOPR Set to 1 indicates that the job is not yet logged in. It is an informational bit and can only be altered by the Monitor when the job is logged in.
- JFSYS Set to 1 indicates that the job is currently running with temporary privileges. It is set by the Monitor when a non-privileged job runs a privileged program. Once the program is run, the job can regain temporary

privilege by .SETting this bit, and drop privilege temporarily by .CLEARing it. (This bit is never 1 when JFPRIV, below, is 1.)

JFPRIV	Set to 1 indicates the job is permanently privileged; it is logged in under a privileged account. It is set only by the Monitor; however, if the job issues a .CLEAR for this bit, temporary privilege (see JFSYS, above) is permanently lost.*
JFFPP	Set to 1 indicates that the contents of the hardware floating point unit (if any) should be part of the context of this job. That is, information in the floating point registers should be saved and restored along with the rest of the user job image during swapping. Any job using the hardware floating point unit should set this bit. It can be .SET and .CLEARed.
JFSPRI	Set to 1 indicates that the job is running with a special run priority -- at 1/2 level higher than normal. This bit can be .SET and .CLEARed.

USRSP

This mnemonic is assigned the value 400 (by COMMON.MAC). The Monitor automatically loads this value into the stack pointer register (R6) when a job is created. R6 is also reset to this value under certain conditions, effectively establishing a default user stack area (since R6 is considered the Stack Pointer or SP) for the job beginning at word 376. The user stack area ends at location 170 (octal). Any attempt to push the stack past location 170 results in a "stack overflow" error that is handled by the run-time system (see P.BAD description in Section 2.5).

A job may change R6 if it wishes. However, any attempt to reset R6 to any location between 0 and 167 (octal) causes a "stack overflow" error. Also, the Monitor resets R6 to 400 when a run-time system is entered via a .RUN, .CCL, or .RTS directive (Sections 3.20, 3.3, and 3.19, respectively), and when certain catastrophic errors occur, such as a fatal disk error while the user job image was being swapped (see Section 2.5, P.BAD description).

The .RUN, .CCL, and .RTS directives return control to a run-time system (implicitly, the user job image execution has not begun, or has exited). Similarly, it is extremely unlikely that a run-time system would wish to return control to a user job image after a catastrophic error has occurred. Thus, if you are coding a program to run as a user job image, you need not worry about the Monitor's resetting of R6. If you are coding or modifying a run-time system, however, this may be of interest to you -- take note.

*All privileged utilities which can be executed by non-privileged users -- like SYSTAT for example -- clear this bit before exiting, so that the temporary privilege set up for the job cannot be used further.

FIRQB

The FIRQB (file request queue block) is the main communication area between the Monitor and the job for Monitor directives that involve file or device operations such as open, close, and so forth. Either the run-time system or the user job image may use this area. If, for example, you use the general Monitor directives described in Chapter 3, your MACRO program must store values in the FIRQB before issuing some of the directives. If you choose to use the directives in either the RSX or RT11 Run-Time Systems, code within the run-time system will intercept the request, set up the FIRQB and other relevant areas, and then call the Monitor to handle the request.

The general format of the FIRQB, with all possible mnemonics assigned by COMMON.MAC, is shown below:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	FIRQB
3 FQFUN	CALFIP/.UUO subfnc. job number * 2	2	FQJOB
5 FQSIZM	MSB of file size channel number *2	4	FQFIL FQERNO
7	project number programmer number	6	FQPPN
11	file name (2 words in RAD50 format)	10	FQNAM1
13		12	
15	file extension (1 word in RAD50 format)	14	FQEXT
17	(least significant bits of) file size	16	FQSIZ
21	buffer length	20	FQBUFL FQNAM2
23	mode	22	FQMODE
25	status flags	24	FQFLAG
27 FQPROT	protection code <>0, prt. code real	26	
31	device name (2 ASCII characters)	30	FQDEV
33	<>0,unit no. real device unit no.	32	FQDEVN
35	cluster size	34	FQCLUS
37	number of entries in directory lookup	36	FQNENT

XRB

The XRB is the main communication area between the Monitor and the user for Monitor directives handling file or device I/O. It is also the area in which the Monitor stores information requested by straightforward information-request calls. As with the FIRQB, the general Monitor directives described in Chapter 3 require that you store and retrieve information directly to and from the XRB; the RSX and RT11 Run-Time Systems handle more general directives which they translate to a call or calls using the XRB. The general format of the XRB, with all possible mnemonics assigned by COMMON.MAC, follows.

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	buffer size in bytes	0	XRLEN
3	bytes actually transferred	2	XRBC
5	buffer address	4	XRLOC
7	XRBLKM MSB of block number channel number * 2	6	XRCI
11	least significant bits of block number	10	XRBLK
13	wait time for terminals	12	XRTIME
15	device modifier	14	XRMOD XRBSIZ

CORCMN

The core common area (CORCMN) is used as a common data exchange area when it is necessary to exchange lengthy data (usually strings) between the Monitor and the job, or between programs running under the same job number.

For example, the Monitor uses CORCMN to pass a string to the job which is the full name of a command which has been recognized as a valid CCL command. The Concise Command Language (CCL) of RSTS/E allows users to type one-line commands to call utilities which might otherwise require several input lines from the terminal. For example,

```
PIP FILE1.=FILE2.
```

Ready

instead of

```
RUN $PIP
*FILE1.=FILE2.
*(CTRL-Z)
```

Ready

To centralize decoding, the Monitor analyzes CCL commands by comparing them to those defined by the system manager when the system starts up timesharing. A job can ask the Monitor to analyze a string to see if it is an acceptable command with the .CCL directive (Section 3.3). If it is, the Monitor passes control to the run-time system associated with that CCL command, and passes the command and any arguments on to the job in the CORCMN area.

The general format of the CORCMN area is:

byte 1 of string	no. of bytes in string	460	CORCMN
byte 3 of string	byte 2 of string	462	
.			
.			
.			
(up to 127. bytes of data)			

USRPPN, USRPRT, USRLOG

The job can set these areas (see .ULOG, Section 3.31) to the assigned project, programmer number (USRPPN), default protection code (USRPRT), and assigned logical device names (USRLOG) which the Monitor will then use when an .FSS directive (Section 3.10) is executed. The .FSS directive causes the Monitor to convert a filename string to the standard RSTS/E file specification format, that is, to the FIRQB format. The formats for these areas follow.

The .ULOG and .FSS directives also allow you to define and use some non-standard area to contain these values, as described in Sections 3.10 and 3.31. However, the .ULOG directive will set up 18 words in the same basic format; the .FSS directive will expect these values in the same relative locations.

USRPPN

A non-zero value in this word is interpreted as a project,programmer number (high byte = project number, low byte = programmer number). The Monitor will use this value to translate a commercial at sign character (@) encountered in a file specification string for .FSS. If this word is zero, an .FSS will produce an error if an @ character appears in the string to be translated.

USRPRT

A non-zero value in this word is interpreted as a protection code to be used as a default if no protection code is specified in a file specification string translated by an .FSS directive. The value of the protection code should be in the high byte; the low byte should be non-zero, to indicate an explicit protection code. (Protection code values may range from 0 through 377 (octal);

the meanings associated with various values are described in the RSTS/E System User's Guide.)

A 0 in this word indicates that there is no default protection code for this user. Therefore, the system default protection code (normally 60. decimal) would be used.

USRLOG

This area holds the user's private logical device name table. It consists of 16.words, allowing either three or four logical names to be associated with devices. (If a project, programmer number is associated with a logical name, only three logical names can be assigned. If no project, programmer number is associated with any logical name, four logical names can be assigned.) The .FSS directive uses this table for logical-to-physical device translation; the user logicals here will supercede any system-wide logical names defined by the system manager.

The format of each entry in the first 12.words is as follows:

Octal Offset			Octal Offset
1			0
	logical device name, RAD50 format		2
3			
5	physical device name, 2 ASCII chars.	4	
7	unit number real unit number	6	

Offset Meaning

- | | |
|---|---|
| 0 | This 2-word area contains the logical device name in RAD50 format. If the first word is zero, then this entry is currently unused, and the remaining three words of the entry are random. |
| 4 | This word holds the physical device name as 2 ASCII characters. This physical device name is the one to be substituted for the given logical device name in an .FSS directive. |
| 6 | The low byte of this word contains the unit number of the physical device. The high byte is set to a non-zero value to indicate an explicit device number. The unit number defines the particular unit of the physical device substituted for the given logical device name in an .FSS directive. If the entire word is zero, then no unit number is associated with the device (for example, SY:). |

The last four words of the USRLOG area will be the same as described above, if no project, programmer number is associated with a logical name. If a project, programmer number is associated with a logical name, the format of the last four words is as follows:

Octal Offset		Octal Offset
1	-1 (flag for associated ppn's)	0
3	ppn for name at USRLOG - USRLOG+3	2
5	ppn for name at USRLOG+4 - USRLOG+7	4
7	ppn for name at USRLOG+10 - USRLOG+13	6

(Note: The .ULOG directive will automatically set up these areas--see Section 3.31.)

2.5 HIGH-SEGMENT DETAILS -- PSEUDO VECTORS

The Monitor and the run-time system use this area to communicate with each other. The general layout of this area is shown in Figure 2-5. As with the low 1000 bytes of virtual address space, the file COMMON.MAC assigns mnemonic names to locations in this area. These names are shown to the right in Figure 2-5. Each of the areas is described in detail in the text following. If you wish to modify or code your own run-time system, the format and meaning of these areas will be of considerable interest. Otherwise, you might wish to examine them simply to get an idea of the type of communication between the run-time system and the Monitor.

In general, the pseudo-vector region contains:

1. Values and flags which define the capabilities of the run-time system to the Monitor. For example, one flag indicates whether the run-time system can handle user-typed commands -- a keyboard monitor capability.
2. Addresses pointing to locations within the run-time system where the Monitor is to pass control when certain conditions occur. These addresses fall into three categories:
 - Addresses for Synchronous System Traps.* Control is passed to these locations when the job executes an instruction which causes a trap to the Monitor. The Monitor passes control to the run-time system along with the contents of the program counter (PC) and program status word (PSW). The term

"synchronous" is used in the sense that the trap occurs at the same time as (is a direct result of) some instruction executed by the job. These traps may or may not indicate an error. For example, if the job executes an instruction with an odd address, control is passed to one of these trap addresses. If the job simply executes a BPT instruction, control is passed to another of these addresses.

- Addresses for Asynchronous System Traps.*

Control is passed to these locations (a) as a result of some event external to the execution of the job (for example, the user types a CTRL-C at the terminal), or (b) as a result of some internal but

asynchronous process (such as an error in the PDP-11/45 hardware floating point unit, whose execution overlaps that of the PDP-11 central processor.) When such conditions occur, control is passed to the Monitor, which passes control on to the run-time system, along with the contents of the PC and PSW. In the case of an 11/45 floating point trap, the Monitor also passes along the floating exception code (FEC) and floating exception address (FEA). For the asynchronous traps, the PC and PSW do not refer to the instruction which caused the trap, but to the instruction which was executing in the central processor when the trap occurred.

- Entry Point Addresses. The Monitor passes control to the run-time system at entry-point addresses when some major transition point is reached for the job. For example, when the user types a RUN or CCL command at the terminal, the Monitor passes control to an entry point in the appropriate run-time system, to load and execute the requested program.

The pseudo-vector region is described in detail below.

NOTE: Normally, you would code the contents of the pseudo-vector region as part of the run-time system file. Please note, however, that the UTILITY routine's ADD command, used to define a file as an auxiliary run-time system, has switches which will cause the Monitor to override certain portions of the pseudo-vector region and use values assigned in the ADD. For example, one bit in one word of the pseudo-vector region states whether the run-time system is read/write or read-only when it is loaded in memory. Normally, this would be read-only, but for

*The term "pseudo-vector" arises from the relationship of some of these (one-word) trap addresses in the pseudo-vector region to the (two-word) vector addresses in kernel-mode memory set up to handle error traps and interrupts in the PDP-11. When the RSTS/E Monitor receives control as a result of a trap to certain of these vector addresses, it passes control on to the run-time system at addresses specified in the pseudo-vector region.

debugging a run-time system with ODT (which allows you to change memory), the run-time system must be read/write. The /RW switch in the ADD command of UTILITY lets you tell the Monitor that until further notice, this run-time system is read/write, regardless of what is specified in the pseudo-vectors. The UTILITY routine and its ADD command are described in the RSTS/E System Manager's Guide.

flags describing the run-time system	177732	P.FLAG	P.OFF
normal executable file extension	177734	P.DEXT	
(former use now obsolete--reserved word)	177736	P.ISIZ	
minimum size, in K words, of user job image	177740	P.MSIZ	
trap address for FIS hardware floating pt. option	177742	P.FIS	
crash entry point (default run-time system only)	177744	P.CRAS	
start entry point (default run-time system only)	177746	P.START	
entry point for new user	177750	P.NEW	
entry point for new user with program to run	177752	P.RUN	
trap address for various "bad" errors	177754	P.BAD	
trap address for BPT instruction and T-bit traps	177756	P.BPT	
trap address for IOT instruction	177760	P.IOT	
trap address for non-Monitor EMT instructions	177762	P.EMT	
trap address for all TRAP instructions	177764	P.TRAP	
trap address for FPP or FPU floating pt. unit	177766	P.FPP	
trap address when user types one CTRL-C	177770	P.CC	
trap address when user types two CTRL-Cs	177772	P.2CC	
maximum size (in K words) of user job image	177774	P.SIZE	
/////////(reserved for future use)//////////	177776		

Figure 2-5 Format of Pseudo-Vector Region of High Segment

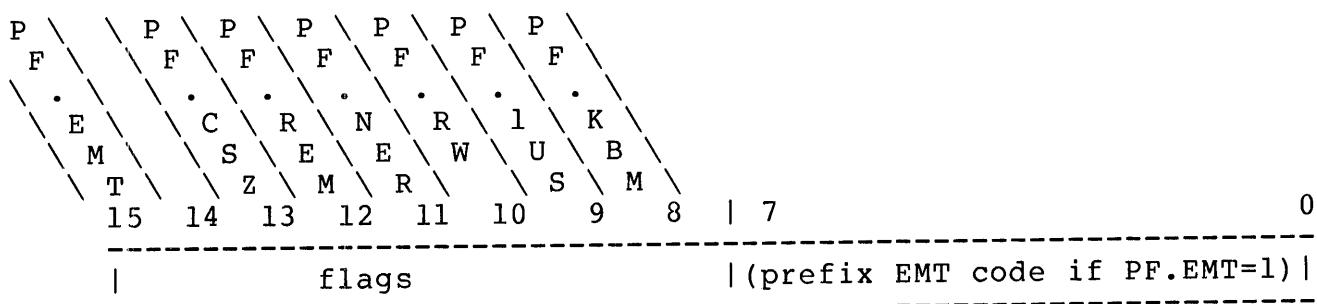
2.5.1 Run-Time System Capability and Default Definitions

P.OFF

This mnemonic is simply used to define the first word of the pseudo-vector region. It is currently set equivalent to 177732, the same as P.FLAG.

P.FLAG

The Monitor expects this word to be set with flags which define the capabilities of the run-time system.



PF.EMT

This bit is set to 1 to indicate that the run-time system wishes to handle a call that would normally be handled by the Monitor. To understand how the bit works, we must first describe what normally happens when a Monitor directive is translated and executed.

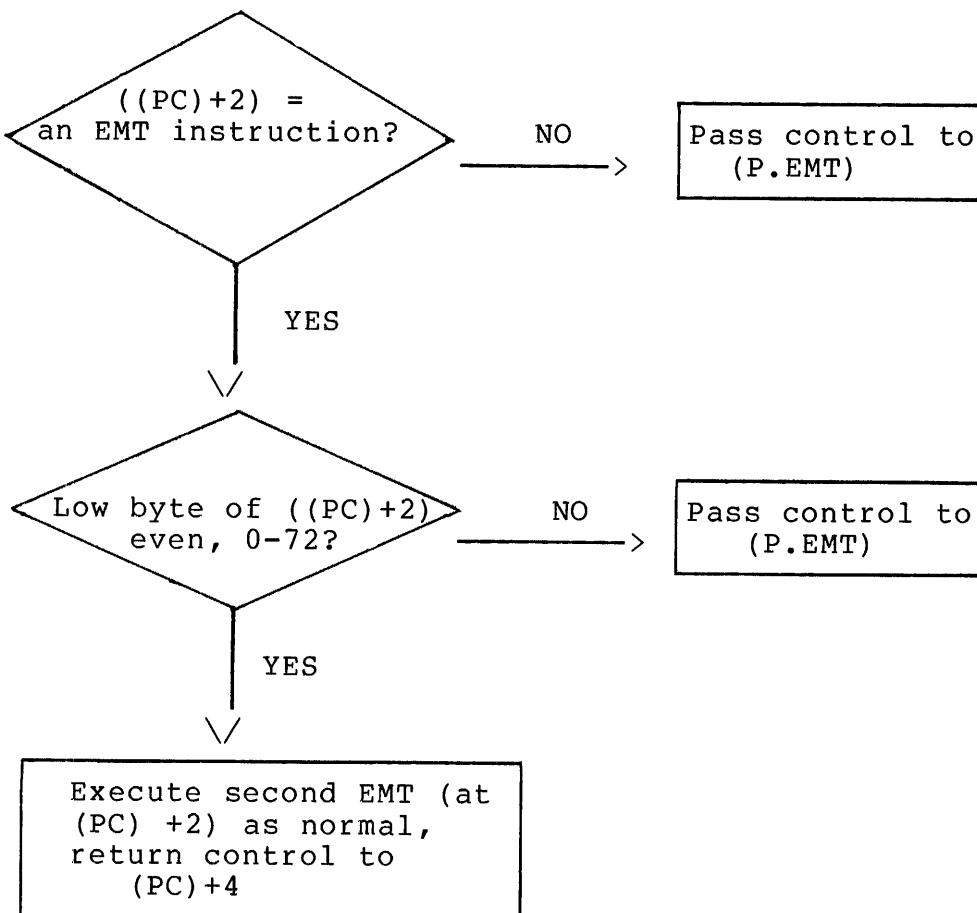
All the Monitor directives described in this manual are translated to EMT (Emulator Trap) instructions. The direct Monitor calls (Chapter 3) are one-for-one translations; that is, one call is translated to one EMT. The code to process the call is in the Monitor itself. The RSX Emulator calls may be translated to more than one instruction, but the code always contains an EMT. The direct Monitor calls, furthermore, are translated to an EMT with a low byte which is an even number, within the range 0 - 72 (octal). When such an instruction is executed, control transfers directly to the Monitor, the call is processed, and control returns to the instruction following the EMT.

An EMT instruction with an odd value in the low byte, in the range 1-71, or any value in the range 73-377, also transfers control to the Monitor. The Monitor examines the low byte, discovers that the EMT is not one of its "own", and transfers control to the run-time system at the entry point defined by location P.EMT in the pseudo-vector region.

Now -- the PF.EMT bit is set to 1 to indicate that the run-time system wishes to process EMTs which are normally processed by the Monitor, that is, with an even low byte in the range 0 - 72. When PF.EMT is set to 1, all EMTs will cause control to pass to the run-time system at entry P.EMT except those immediately preceded by a "special prefix" EMT -- an EMT whose low byte is equal to the low byte of P.FLAG.

Specifically, when PF.EMT = 1, the Monitor handles all EMT instructions as follows:

1. Any EMT whose low byte is not equal to the low byte of P.FLAG will cause control to pass through the Monitor (unprocessed except for examination), back to the run-time system at the address contained in the P.EMT word.
2. An EMT whose low byte is equal to the low byte of P.FLAG will cause control to pass to the Monitor, which looks at the word following the EMT with the special code. (That is, at the word in location $(PC)+2$.) Action is taken according to the value of this word:



In other words, special processing is done by the run-time system for all EMTs except those preceded by a "special prefix" EMT.

PF.CSZ

The Monitor preallocates space for a user job image executed as a result of a .RUN directive based on information provided by the run-time system the image is executing under. When this bit is set to 1, the Monitor will preallocate this space based on the size of the file referenced in the .RUN directive, as follows:

$$\text{space (in K words)} = (\text{filesize} + 3)/4$$

where filesize is the number of 512-byte blocks required for the file on disk. (The division by 4 is performed because there are 4 512-byte blocks in 1 K word. The addition of 3 "rounds up" any fraction of the integer divide to the next whole integer.)

When PF.CSZ is set to 0, the Monitor preallocates the space for the image according to the value specified in the P.MSIZ word of the pseudo-vector region.

PF.REM

When this bit is set to 1, the Monitor will immediately remove the run-time system from memory when no job is using it. When this bit is 0, the Monitor will leave the run-time system in memory until the space is actually needed by something else.

PF.NER

When this bit is set to 1, the Monitor will not log errors occurring within the run-time system to the system error log.

PF.RW

When this bit is set to 1, the Monitor will map the run-time system as read/write (recall the read/write feature of the Page Descriptor Register of an APR, Section 2.1). This is a useful feature when debugging a run-time system. In normal operation, this bit should be set to 0, indicating that the run-time system is to be mapped read-only.

PF.1US

When this bit is set to 1, the Monitor will allow only one job to use the run-time system. That is, it will not be handled as sharable code.

PF.KBM

When this bit is set to 1, the Monitor expects that the run-time system can function as a keyboard monitor. The run-time system can function as a "private default" run-time system only when this bit is set. (See the .RTS directive, Section 3.19, for a discussion of private default run-time systems.)

P.FLAG COMBINATIONS

The PF.1US, PF.RW, PF.NER, and PF.REM bits are useful flags when you are debugging a run-time system. PF.1US limits access to the run-time system to one user; PF.RW is necessary if you wish to use the ODT routine to change memory. PF.NER keeps the run-time system from logging useless errors while debugging, and PF.REM ensures that the run-time system will be reloaded each time it is used. (Otherwise, an old copy might still remain in memory when you really wanted to debug a new copy.)

P.DEXT

This word can be set to three Radix-50 characters which the Monitor will use as a default runnable file extension, as follows. If a .RUN (Section 3.20) is executed with no file extension given, the Monitor will scan its list of installed run-time systems (in the order they were installed*). For the first run-time system in the list (the system default run-time system), the Monitor looks for a file with the filename given in the .RUN and an extension which is the default extension for the run-time system. If such a file is found, it is set up for the .RUN. If no such file is found, the Monitor searches for a file with the given filename and the next run-time system's default runnable file extension, and so forth. Note that the order in which the file extensions are chosen does not depend in any way on the run-time system executing the .RUN.

For example, the BASIC-PLUS Run-Time System fills this word with .BAC; RT11, with .SAV, and RSX, with .TSK. If the run-time system has no runnable file extension, this word should be set to 0.

*The system manager installs run-time systems with the ADD command of UTILITY, as described in the RSTS/E System Manager's Guide. The order of installation shows up in the display produced by the SYSTAT utility.

P.MSIZ

This word gives the minimum allowable size for a user job image, in K words, for this run-time system. The Monitor uses this value as a check when the job issues a .CORE directive (Section 3.6) to change the size of the user job image in memory. The value of P.MSIZ must be an integer between 1 and 28, inclusive.

P.SIZE

This word contains the maximum size (in K words) that a user job image can be for this run-time system. The Monitor uses this value as a check when a job issues a .CORE directive (Section 3.6) to change the size of the user job image in memory. P.SIZE must be an integer between 1 and 28, inclusive. The effective upper limit is 32 minus the size of the run-time system rounded up to a multiple of 4. (Remember that the APR mapping requires that space for the run-time system be allocated in units of 4K words.) Thus, a run-time system that required 5K words could set an upper limit here of 24 (32-8). It could set P.SIZE to some smaller value, however.

The RSX Run-Time System, when the Emulator is installed as part of the Monitor, is an exception to this rule. As described in Section 2.3, an executing program can expand to 31K words. So P.SIZE in this particular case is set to 31.

2.5.2 Synchronous System Trap Addresses

P.FIS

The Monitor interprets this word as the trap address for the hardware floating point instruction set available on the PDP-11/40. Whenever an instruction from this set is executed which causes a trap to the kernel mode vector at 244 (octal), the Monitor passes control on to the run-time system at the location specified by the contents of the P.FIS word.

This trap pushes two words onto the user's SP stack: the contents of the PC and PS registers at the time of the exception:

```
SP -> (PC) at the time of the trap  
      (PS) at the time of the trap  
      word to which SP pointed before the trap
```

Whatever action the run-time system wishes to take for this trap should be done at the location specified by the contents of P.FIS. (A return from interrupt (RTI) instruction will return control to the point where it was when the trap occurred.)

P.BAD -- Synchronous Traps

The Monitor passes control on to the run-time system at the location specified by the contents of P.BAD when any of the following synchronous traps occur:

1. Memory management unit exception (trapped to kernel mode vector at 250 (octal)).
2. The job tries to execute a reserved instruction (trapped to kernel mode vector at 10 (octal)).
3. The job issues an instruction with an odd address (trapped to kernel mode vector at 4 (octal)).

This trap pushes two words onto the user's SP stack: the contents of the PC and PS registers at the time of the trap.

```
SP -> (PC) at the time of the trap.  
      (PS) at the time of the trap.  
      word to which SP pointed before the trap
```

An error code is returned in the first byte of the FIRQB so that the run-time system can determine which error occurred. The codes are:

B.4	Odd address
B.10	Reserved instruction
B.250	Memory management unit exception

The run-time system is responsible for processing these errors in whatever manner it sees fit. An RTI instruction can be used to return control to the point where it left off when the trap occurred. Note that some asynchronous traps also use this address (Section 2.5.3).

P.BPT

This word contains the trap address for a BPT instruction, and for T-bit traps. When the job issues a BPT instruction, or a T-bit trap occurs (to the kernel mode vector at 14 (octal)), the Monitor passes control on to the run-time system for the job at the address specified by the contents of this word.

The trap pushes two words onto the user's SP stack: the contents of the PC and PS registers.

SP → (PC) at the time of the trap.
(PS) at the time of the trap.
word to which SP pointed before the trap.

The run-time system would process these traps in any fashion it sees fit at the location specified by the contents of P.BPT. The RTI or RTT instructions can be used to return control to the user's program at the point where it was when the trap occurred.

P.IOT

This word contains the trap address for an IOT instruction. Whenever the job issues an IOT instruction (trapped to kernel mode vector at 20 (octal)), the Monitor passes control on to the run-time system at the address specified by the contents of this word. This trap pushes two words onto the user's SP stack: the contents of the PC and PS registers at the time of the trap.

SP → (PC) at the time of the trap.
(PS) at the time of the trap.
word to which SP pointed before the trap.

The run-time system can process the trap in any fashion it sees fit. An RTI instruction can be used to return control to the point where it was when the trap occurred.

P.EMT

This word contains the location to which control is transferred for non-Monitor EMT instructions; that is, for EMT instructions whose low byte is odd within the range 0 - 72 (octal), or any value in the range 73-377. If the PF.EMT bit is set in the P.FLAG word in the pseudo-vector region, control is transferred here for all EMT instructions except those preceded by the "special prefix" EMT, as described previously.

The trap pushes two words onto the user's SP stack: the contents of the PC and PS registers at the time of the trap.

SP → (PC) at the time of the trap.
(PS) at the time of the trap.
word to which SP pointed before the trap.

The run-time system is responsible for processing the EMT as it sees fit. The RTI instruction can be used to return control to the point where it was when the trap occurred.

P.TRAP

This is the location to which control is transferred for all TRAP instructions (operation codes 104400 through 104777, inclusive). Whenever the job executes such an instruction (trapped to kernel mode vector 34 (octal)), the Monitor passes control on to the run-time system at the location specified by the contents of this word.

The trap pushes two words onto the user's SP stack: the contents of the PC and PS registers at the time of the trap.

SP → (PC) at the time of the trap.
(PS) at the time of the trap.
word to which SP pointed before the trap.

The run-time system is responsible for processing the trap as it sees fit. An RTI instruction will return control to the point where it was when the trap occurred.

2.5.3 Asynchronous System Trap Addresses

P.FPP

This location is the trap address for the FPP (or FPU) hardware floating point unit (the PDP-11/45 and 11/70 asynchronous unit and/or its equivalents.) Whenever the unit takes an exception trap (to kernel mode vector at 244 (octal)), the Monitor passes control to the run-time system at the location specified by the contents of this word. Since the Floating Exception Code (FEC) and Floating Error Address (FEA) of this unit are not otherwise accessible, the Monitor pushes these two values onto the user's SP stack, in addition to the contents of the PC and PS registers at the time of the interrupt.

SP → Floating Point Exception Code (FEC).
Floating Error Address (FEA).
(PC) at the time of the trap.
(PS) at the time of the trap.
word to which SP pointed before the trap.

The run-time system can process the trap as appropriate, clean the stack (pop the FEC and FEA), and issue an RTI instruction to return control to the user's program at the point where it was when the trap occurred.

P.CC

This is the location to which control passes when the user types a CTRL/C. ("The user" in this case is any terminal input being accepted by this job, on any channel.)

The Monitor inhibits further programmed output for the job (CTRL/O effect), and cancels any pending character output. The user's SP stack is modified at entry as follows:

```
SP -> (PC) at the time of the trap.  
      (PS) at the time of the trap.  
      word to which SP pointed before the trap.
```

The run-time system can process the CTRL/C as desired. All run-time systems supplied with a RSTS/E system abort the job, unless the user job image has indicated that it wishes to handle CTRL/C traps itself (see SCCA\$ directive, Section 5.19).

P.2CC

This is the trap address taken when the user types a second CTRL/C before the run-time system has been able to respond to the first CTRL/C typed. (That is, the Monitor has received two CTRL/C's before it has been able to pass control on to the run-time system at (P.CC) in the time-sharing environment.)

As with one CTRL/C, when this point is reached, further programmed output has been canceled (CTRL/O effect), and any pending character output has been canceled. Two words are pushed onto the user's SP stack:

```
SP -> (PC) at the time of the trap.  
      (PS) at the time of the trap.  
      word to which SP pointed before the trap.
```

The run-time system can process the condition as desired (BASIC-PLUS exits immediately, returning control to the P.NEW entry point in the system default run-time system). An RTI instruction would return control to the point where the program left off, but this will annoy the user who typed the two CTRL-Cs expecting to get out.

P.BAD -- Asynchronous Traps

The Monitor passes control to the location specified by P.BAD whenever any of the following asynchronous errors occur:

1. The user's SP stack overflowed.
2. A fatal disk error occurred when the job was swapped. The original contents of the user job image are lost.
3. A memory parity fault occurred in the user job image. The original contents of the user job image are lost.

None of these errors are really recoverable; an error is returned in the first byte of the FIRQB to indicate which error occurred, the keyword (KEY) is refreshed, and the contents of the general registers (R0 through R5) are random. The SP (stack pointer) is reset to the value USRSP.

The error codes returned in the first byte of the FIRQB are:

- | | |
|--------|---------------------------------|
| B.STAK | The user's SP stack overflowed. |
| B.SWAP | Fatal disk error on swap. |
| B.PRTY | Memory parity fault. |

NOTE: Control is also transferred to P.BAD for some synchronous traps, as described in Section 2.5.2.

2.5.4 Entry Points

P.CRAS

This word is a special entry point used only by the system default run-time system. Control passes to this location in the system default run-time system only when the whole system is restarted after a crash.

P.START

This word is a special entry point used only by the system default run-time system. Control passes to this location only when the system starts up from a normal startup procedure.

P.NEW

The Monitor passes control to this entry point under the assumption that "new user" or "next request" processing is to be done, as opposed to the P.RUN entry point, where it is known that a specific program is to be run under this run-time system. This entry point is commonly used as the entry point to switch back to a job's "private default" run-time system. (The .EXIT directive (Section 3.9) passes control to this entry point in the system default run-time system. The .RTS

directive (Section 3.19) can be used to pass control to this entry point in a job's private default run-time system, or a specifically named run-time system.)

By examining the keyword (KEY) and the XRB, the run-time system can determine how and by whom it was entered at P.NEW, if this is significant. (Run-time systems which do not have keyboard monitors would probably wish to simply .EXIT to the system default run-time system here.) The three conditions under which control is passed to the P.NEW entry are:

1. Brand New Job on the System. In this case, JFNOPR (bit 12. in KEY) is set to 1 (the job is not yet logged in), and the words at location XRB+2 and XRB+4 are 0 (the Monitor requested the entry, not a run-time system). This indicates that the Monitor has passed control to this location having received terminal input over channel 0 in a "logged out" state (only occurs for the system default run-time system).

The run-time system should run some predetermined program to read (.READ directive, Section 3.17) the fully assembled line which the Monitor has buffered. The BASIC-PLUS Run-Time System loads and executes the file SY:[1,2]LOGIN.BAC (the LOGIN utility) in this case.

2. Switch to This Run-Time System when Job Logged Out. In this case, JFNOPR (bit 12. in KEY) is set, and the name of the calling run-time system* is given as two words of RAD50 in locations XRB+2 and XRB+4.

For this case, the run-time system should issue its "logged out" prompt message. For example, should LOGIN not recognize the assembled line that it read (in the case 1 example, above), it prints its own message -- Please say HELLO -- and exits, causing control to return to the P.NEW entry point in the system default run-time system. The BASIC-PLUS Run-Time System prints "Bye" and returns control to the Monitor, which in this case destroys the job.

3. Switch to This Run-Time System when Job Logged In. In this case, JFNOPR (bit 12. in KEY) will be clear (0), and the name of the calling run-time system* is given as two words of RAD50 in locations XRB+2 and XRB+4.

For this situation, the run-time system should issue its "logged in" prompt and attempt to read the next command from the terminal open on channel 0. BASIC-PLUS prints "Ready" at this point and waits for further input.

*The calling run-time system is the run-time system under whose control the directive was issued that caused the switch.

The Monitor usually* does some housekeeping for the job at the time the P.NEW entry point is entered. Specifically, the word at location FIRQB+FQJOB is always set to two times the job number assigned by the Monitor when the job was created, the keyword (KEY) is refreshed with current information about the job, as described in Section 2.4. Furthermore, the stack pointer, hardware register R6, is reset to 400 (see USRSP description in Section 2.4), and all the general registers (R0 through R5) contain random values.

The following information exists in the XRB at the time the P.NEW entry point is entered:

XRB ON P.NEW ENTRY

Offset Octal Mnemonic		Offset Octal Mnemonic
1	1 for switch without housekeeping; else 0	0
3		2
5	name of run-time system that called this one (RAD50)	4
7	-1 if calling RTS = new RTS; else 0	6
11		10
13	whatever was here when switch was made	12
15		14

- XRB+0 This word contains a value of 1 if control was transferred by an .RTS directive using the "switch without changing job context" option (Section 3.19).
- XRB+2 The two words beginning here contain the name of the calling run-time system, in RAD50 format. If control has been transferred here directly by the Monitor (case 1 above), these two words will contain 0.
- XRB+6 This word contains -1 if the calling run-time system is the same run-time system that now has control. This word is 0 otherwise, that is, if the "caller" is not the "callee".
- XRB+10 The contents of the next three words will be the same as they were when the switch occurred. That is, data can be passed from run-time system to run-time system

*This housekeeping is not done if a specific request is made to pass control to a run-time system without changing the job-context information. (See .RTS, Section 3.19).

here. If control has been transferred to P.NEW directly by the Monitor (case 1 above), these three words will be 0.

P.RUN

The Monitor passes control to this entry point when an executable program is to be run for a job under control of this run-time system. This can occur as the result of either a .RUN directive (in which a job has directly asked for a file to be run), or a .CCL directive (in which a job has asked the Monitor to check a string to see if it is a valid Concise Command Language (CCL) command, and if so, execute the appropriate file.)

The file to be run has been opened by the Monitor on channel 15 (decimal), and is a disk file. The file has not been read; it is up to the run-time system to load and execute the file.

As with P.NEW, the Monitor has "refreshed" the keyword (KEY), set R6 to 400, and left R0-R5 containing random values. In addition, the Monitor sets the JFBIG bit in KEY if the program to be run is a privileged program. If the program to be run is privileged and the caller is normally unprivileged, the Monitor sets the JFSYS bit in KEY.

Data is passed to the run-time system in the XRB, FIRQB, and KEY areas of the user job image (low segment.)

XRB ON P.RUN ENTRY

Offset Octal Mnemonic		Offset Octal Mnemonic
1	flag bits describing entry conditions	0
3		2
5	. name of run-time system which issued the call to this one	4
7	random value	6
11		10
13	same value as when the caller issued the .RUN or .CCL	12
15		14

XRB+0

This word contains flag bits that describe the entry conditions. (The STATUS variable in BASIC-PLUS returns these values.)

Bit	Meaning
15	Set to 1 indicates the entry was made as the result of a .CCL directive. 0 indicates the entry was made as the result of a .RUN directive.
14	Set to 1 indicates the caller issued a .CCL directive with a "/DETACH" switch, with the intent that this run-time system run the file in detached mode. It is up to the run-time system to take action on this flag. (Detaching a job can be done with the UUDET subfunction of the .UUO directive, as described in Section 3.32.)
13	Set to 1 indicates that the caller issued a directive with a /SIZE switch; that is, that the file is to be run at a specific size. When this bit is set, bits 0-7 of this word indicate the desired size (see below). It is up to the run-time system to set the size as indicated (see .CORE, Section 3.6.)
12-8	Reserved for future use.
7-0	If this byte is 0, then no special size for this program run is desired. If greater than zero, this byte indicates the size (in K words) that the program should be run at. If less than zero, the absolute value of this byte indicates an increment (in K words) to the size that the program would normally run at.
XRB+2	These two words contain the name of the run-time system under which the .RUN or .CCL directive to this run-time system was issued, in RAD50 format.
XRB+6	The contents of this word are random.
XRB+10	The three words beginning here contain the same information that they held when the job issued the .RUN or .CCL directive.

FIRQB ON P.RUN ENTRY

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	job number * 2	2 FQJOB
5	(always 0)	4
7	project number programmer number	6 FQPPN
11	name of file to be executed (in RAD50 format)	10 FQNAM1
13		12
15	extension of file to be executed (RAD50)	14 FQEXT
17	size of the file, in 512-byte blocks	16 FQSIZ
21	default buffer size for disk	20 FQBUFL
23		22
25	device description	24 FQFLAG
27 FQPROT	protection code cluster size	26
31	device name (2 ASCII characters)	30 FQDEV
33	flag byte device unit number	32 FQDEVN
35	file identification index	34 FQCLUS
37	entry parameter	36 FQNENT

- FIRQB+FQJOB The job number, times two, assigned by the Monitor when this job was created.
- FIRQB+FQPPN The project,programmer number for the file that is to be run.
- FIRQB+FQNAM1 The filename of the file that is to be run, as two words in RAD50 format.
- FIRQB+FQEXT The extension of the file that is to be run, as one word in RAD50 format.
- FIRQB+FQSIZ The size of the file, in 512-byte blocks.
- FIRQB+FQBUFL The recommended size, in bytes, for the buffer size in a .READ for this (disk) file.

FIRQB+FQFLAG	Flag bits defining the device. They are set to indicate that this is a disk file. (See the FQFLAG description in the open function of the CALFIP directive, Section 3.2.12.)
FIRQB+FQPROT-1	The file cluster size, modulo 256. (That is, a file cluster size of 256. is indicated by a zero byte here.) (This byte is the same as the FQCLUS value supplied in the "open" functions of the CALFIP directive, Section 3.2, except that it is returned in a byte instead of a word.)
FIRQB+FQPROT	The protection code of the file.
FIRQB+FQDEV	The device name of the disk device, as two ASCII characters.
FIRQB+FQDEVN	The unit number of the disk device is in this byte.
FIROB+FQDEVN+1	The low-order two bits of this byte are set to indicate whether or not the device is part of the public structure:
	Bit 0 = 0 The device is in the public structure.
	Bit 0 = 1 The device is a private disk.
	Bit 1 = 0 A specific device was not specified in the open.
	Bit 1 = 1 A specific device was specified in the open.
FIRQB+FQCLUS	The file identification index of this file. This word is significant mainly in that it can be used in place of the file name in subsequent opens of the file on disk. You can open the file with the OPNFQ subfunction of CALFIP (Section 3.2.12) using an explicit project,programmer number in FIRQB+FQPPN, a zero word in FIRQB+FQNAM1, and the file identification index in FIRQB+FQNAM1+2.
FIRQB+FQNENT	Parameter word from the caller. The .RUN directive, which causes entry at P.RUN in a run-time system, allows the caller to specify a parameter word to be passed to the run-time system. Bit 15 of this word may or may not be the same as the caller passed, however. If the caller was privileged, whatever the caller specified for bit 15 is passed. If the caller was non-privileged, the Monitor automatically sets this bit to 0. The intent of this bit is to "pass on" temporary privilege. If this bit is set, the run-time system should retain temporary privilege (the JFSYS bit in KEY is now set and should remain set). If this bit is off, the run-time system should consider whether or not to retain temporary privilege (if the JFSYS bit is on, perhaps it should be turned off).

For example, the BASIC-PLUS Run-Time System uses the FIRQB+FQNENT word as follows:

Temporary privilege is kept (JFSYS, if set, is left set) if:

1. Bit 15 of FIRQB+FQNENT is set, or
2. Bit 15 of FIRQB+FQNENT is clear, but the remaining bits are also 0.

If bit 15 of FIRQB+FQNENT is clear, and the remaining bits of are non-zero, temporary privilege is permanently dropped (by issuing .CLEAR for both JFSYS and JFPRIV).

In either case, the contents of bits 14-0 of FIRQB+FQNENT are used as the line number where execution is to be started for the program. A value of 0 in bits 14-0 means to start at the first executable line. Thus, clearing JFSYS (dropping temporary privilege) when execution is to start at other than the first line protects privileged programs from being started at odd locations to bypass their normal privilege checking.

Finally, if JFSYS was set upon entry (whether or not it gets immediately cleared), then the program will be destroyed upon completion (normal or abnormal). This protects a privileged program from being examined when it is not running.



CHAPTER 3

GENERAL MONITOR DIRECTIVES

3.1 INTRODUCTION

This chapter describes the general directives to the RSTS/E Monitor. These directives are available to the MACRO programmer under both the RSX and RT11 Run-Time Systems. They are EMT instructions that are processed directly by the Monitor. They are not examined or processed by a run-time system.

3.1.1 Summary Of General Monitor Directives

A summary of the directives is given in Table 3-1. Detailed descriptions are given in Sections 3.2 - 3.33, arranged alphabetically by mnemonic name.

Some directives which cause a change in Run-Time System or change in job size should not be executed by a program (user job image) running under the RT11 Run-Time System, or unpredictable results may occur. These directives are marked with an asterisk (*) below. (Note, however, that these are not restrictions for assembling using MACRO -- the assembler for the RT11 Run-Time System. If you are coding a run-time system, you can use these directives and assemble under either MACRO or MAC.)

Table 3-1: Summary of General Monitor Calls

Name	EMT Code (Octal)	Description
CALFIP	0	Call the File Processor portion of the RSTS/E Monitor. Includes "housekeeping" functions for file/device I/O such as open, close, and so forth.
.READ	2	Read from a previously opened file or device.
.WRITE	4	Write to a previously opened file or device.
.CORE*	6	Change memory size allocated for user job image.
.SLEEP	10	Sleep job for n seconds.
.PEEK	12	Peek at the Monitor's memory (privileged).
.SPEC	14	Special function.
.TTAPE	16	Enter tape mode.
.TTECH	20	Enable echo on a channel.
.TTNCH	22	Disable echo on a channel.
.TTDDT	24	Enter ODT submode on a channel.
.TTRST	26	Cancel CTRL/O effect.
.TIME	30	Get timing information.
.POSTN	32	Get device's horizontal position.
.DATE	34	Get current date.
.SET	36	Set keyword bits.
.STAT	40	Get statistics for job.
.RUN*	42	Run new program (user job image).
.NAME	44	Install a new program name.
.EXIT*	46	Exit to system default run-time system.
.RTS*	50	Switch to new run-time system.
.ERLOG	52	Log an error from run-time system.
.LOGS	54	Check for logical devices.
.CLEAR	56	Clear keyword bits.
.MESAG	60	Message send/receive.
.CCL*	62	Check string to see if Concise Command Language.
.FSS*	64	Scan a string for valid RSTS/E file specification.
.UUO	66	Execute Monitor FIP call (access to BASIC-PLUS SYS calls to FIP.)
.CHAIN	70	Execute user job image under same run-time system.
.PLAS	72	Access a shared library.
.RSX*	74	Used only by RSX Run-Time System to "disappear", if possible.
.ULOG	76	Assign/reassign/deassign device or user logical.

3.1.2 Prefix File COMMON.MAC

The Monitor directives described in this chapter require that you pass parameters to the Monitor in the FIRQB and XRB; values are also returned to your program in these areas. The file COMMON.MAC, provided with all RSTS/E kits, relates mnemonics to often-used addresses, offset values, and function codes, eliminating the need for octal coding and addressing. These mnemonics are used in the directive descriptions which follow, and their use is recommended for readable, maintainable code.

3.1.2.1 How To Assemble With COMMON.MAC

COMMON.MAC is a prefix file; it is assembled with your other MACRO source files under either the RSX or RT11 Run-Time Systems. For example, under the RT11 Run-Time System, the sequence

```
RUN $MACRO  
*OBJ,OBJ=COMMON,SRC1,SRC2
```

would assemble the files COMMON.MAC, SRC1.MAC, and SRC2.MAC into the object module file OBJ.OBJ with an assembly listing file OBJ.LST.

Similarly, under the RSX Run-Time System, the sequence

```
RUN $MAC  
MAC>OBJ,OBJ=COMMON,SRC1,SRC2
```

would assemble the files COMMON.MAC, SRC1.MAC, and SRC2.MAC into the object module file OBJ.TSK with an assembly listing file OBJ.LST.

3.1.2.2 Macros Provided In COMMON.MAC

In addition to providing mnemonics, the COMMON.MAC file contains macros which can be used in programs assembled under either the RSX or RT11 Run-Time Systems, as long as COMMON.MAC is assembled with the source, as described previously.

TITLE name,desc,nn,date,editors

The TITLE macro sets a title (.TITLE) from the name and description (desc) parameters, and builds an identification (.IDENT) from the specified number nn. For Version 7.0 of RSTS/E, the IDENT is "070.nn". Descriptive information is placed in the table of contents as follows:

EDIT: nn	DATE: date	BY: editors
--------------------	----------------------	-----------------------

ORG section[,offset]

ORG defines the origin address of a program section. The first occurrence of an ORG with a given section name causes all instructions requiring memory space following the ORG to be assigned consecutive relocatable addresses starting with 0, or, if an offset is given, with the octal address given. Following occurrences of an ORG with the same section name causes resumption of addressing wherever it left off before, because of an intervening ORG.

The ORG macro also defines a symbol with the same name as the section at the first relative location within the section. Every invocation of ORG also defines (or redefines) the section to be returned to by the macro UNORG (see below).

DEFORG section

The DEFORG macro is the same as the ORG macro except that the symbol at relative 0 (the section name) is declared as a global symbol. By convention, the module that defines the section (rather than just uses it) issues the DEFORG macro.

TMPORG section[,offset]

TMPORG is the same as ORG except that it doesn't define (or redefine) the section to be re-entered by the UNORG macro. In this way, the module can temporarily enter a new section and then return to the main section using UNORG without having to know the main section name.

UNORG

The UNORG macro will re-enter the section most recently declared in an ORG or DEFORG macro.

INCLUDE namel[,name2,...]

The INCLUDE macro indicates that the module issuing the INCLUDE requires the named modules (namel, etc.). The name(s) are assumed to be declared with DEFORG(s) in the required modules.

INCLUDE declares the section names listed as global symbols, and issues the macro directive .SBTTL with the heading "INCLUDE FROM LIBRARY "name"" to be inserted in the assembly listing table of contents. INCLUDE documents the named sections as required by this section.

.DSECT

[start][,cref]

The .DSECT macro starts a dummy program section (with the MACRO directive .ASECT) at relocatable address 0, or at the address given by the optional argument start. If the cross-reference (cref) parameter is given (non-blank), the program section is included in the cross-reference listing, if one was requested.

The .DSECT macro is used in the file COMMON.MAC to define the system parameters and offsets.

For example, coding of the form:

```
.DSECT 177776,NOCREF
        .BLKW -1
P.SIZE: .BLKW -1
P.2CC:  .BLKW -1
P.CC:   .BLKW -1
        .
        .
        .
```

is used in COMMON.MAC to assign the proper values to the mnemonics in the pseudo-vector region.

.BSECT

[HIGH][,cref]

The .BSECT macro is like the .DSECT macro except that the default starting address is 1 instead of 0. If the argument HIGH is used, the starting address is 400 (octal). This corresponds to generating bit values. The .BSECT macro is used in COMMON.MAC to define mnemonics for bit locations. For example,

```
.BSECT HIGH,NOCREF
JFSPRI: .BLKB .
JFPP:   .BLKB .
        .
        .
        .
```

is used to assign the mnemonics to the bit locations in the keyword (KEY).

.EQUATE

symbol,value

.EQUATE defines the given symbol to have the supplied value (which may be an expression) by using the equivalent of:

```
.DSECT value
.symbol:
UNORG
```

.BLKWO

[amount][,value]

This macro is similar to the MACRO directive .BLKW. The default for amount is 1, but amount can be any expression. While .BLKW just reserves space, .BLKWO fills in the space with zeros. If the optional value is used, that value is filled in instead of 0.

GLOBAL

name1[,name2,...]

GLOBAL declares the name symbols as external global symbols.

RETURN

[register]

This macro generates an RTS PC by default, but can generate any other RTS instruction by specifying an explicit register.

JMPX

label

JMPX is just like the JMP instruction but will also declare the label to be an external global (that is, jump external).

CALL

subroutine[,register[,argument list]]

CALL is used instead of JSR PC, to call subroutines. If an explicit register is specified, then the call is JSR on that register. If an argument list is specified, it generates a list of .WORD arguments in-line with the subroutine call.

CALLR

subroutine

CALLR is equivalent to a CALL to a subroutine immediately followed by a RETURN. CALLR generates a JMP instruction.

CALLX

subroutine

CALLX is just like CALL, but also declares the subroutine name as an external global symbol.

CALLRX

CALLRX is just like CALLR except that the subroutine name is declared as an external global symbol. That is, call and return are both external.

3.1.3 Error Mnemonics: Link-File ERR.STB

When the Monitor processes the directives described in this chapter, any errors that it detects are passed back to the job in the first byte of the FIRQB as a binary value. The ERR.STB file, provided with all RSTS/E kits, relates mnemonic values to these binary codes, so that you do not have to analyze and process errors in octal. The descriptions in this chapter all refer to the mnemonics provided by ERR.STB. (Appendix A lists all possible errors, the ERR.STB mnemonics, and the error text produced by the BASIC-PLUS Run-Time System when these errors occur.)

The symbols will be automatically resolved at link time. Or, you can include ERR.STB with the files you link with either TKB (the task-builder for the RSX Run-Time System) or LINK (the linker for the RT11 Run-Time System). For example, under the RSX Run-Time System,

```
RUN $TKB  
TKB>IMG1,MPl,SF1=ERR.STB,MAIN
```

would link ERR.STB and MAIN.OBJ to produce the executable file IMG1.TSK, a memory allocation file MPl.MAP, and a symbol definition file SF1.STB.

Similarly, under the RT11 Run-Time System, the sequence

```
RUN $LINK  
*IMG1,MPl,SF1=ERR.STB,MAIN
```

would link ERR.STB and MAIN.OBJ to produce the executable file IMG1.SAV, a memory allocation file MPl.MAP, and a symbol definition file SF1.STB.

3.1.4 Programming Hints

Preset the FIRQB and XRB to 0

The Monitor directives in this chapter pass information to the Monitor in the FIRQB and XRB areas of the low 1000 bytes of memory. It is usually a good idea to clear the entire FIRQB and XRB before issuing a call, to ensure that no extraneous information has been left in the areas (from data returned on a call, for example) that could affect how the call executes. In some cases, however, you may wish to leave the FIRQB and XRB alone. The .FSS call, for example, scans a string, and if it is a valid file specification, returns to the FIRQB the information needed to open the file with the CALFIP call. You would NOT want to clear the FIRQB before opening the file with CALFIP!

Data Returned to FIRQB and XRB

If a call completes without error, the Monitor sets byte 0 of the FIRQB to zero. If an error occurs, the Monitor sets byte 0 of the FIRQB to an error code. Likewise, the Monitor always sets the byte at FIRQB+2 to the current job number times two when a call completes.

In some circumstances, it may be useful to know what happens to the "data passed" when a call completes. (Is the filename still there? and so forth.) The areas not specified as containing "Data Returned" are left alone. If an error occurs, the data returned may or may not have replaced the data passed. It depends on how far processing for the call got before the error occurred.

Channel Numbers for I/O

Directives which handle input/output use a "channel number" to refer to a device. In device or file opens, a channel number is related to a specific device defined in the call. Directives which transfer data (.READ, .WRITE, .MESAG) can then refer to a channel number rather than define a device or file.

Valid channel numbers range from 0 through 15. (decimal). Channel 0 is the job's terminal; for example, a .WRITE to channel 0 will write to the terminal which is running the job. Channel 0 is always open. Similarly, the Monitor opens a file to be run on channel 15. when control transfers to the P.RUN entry point in a run-time system. Thus, user jobs may define and use channels 1 - 14.

Directives that Do I/O

The CALFIP subfunctions OPNFQ, CREFQ, CRBFQ, and CRTFQ open a file or device, and relate the specified channel number to that file or device. OPNFQ opens a file or device for input; CREFQ "creates" a file, that is, opens a file or device for output. CRBFQ creates a binary (executable) output file on disk, and CRTFQ creates a temporary file on disk.

The directives .READ and .WRITE transfer data between memory and a device or file specified by channel number.

The CLSFQ (close) and RSTFQ (reset) subfunctions of CALFIP close a device or file, and free the associated channel number so it can be used with another device or file.

Directives that Support I/O

The "file string scan" (.FSS) directive is useful for programs which process files specified by a terminal user. The .FSS directive examines a string of characters and if it is a valid RSTS/E file specification, converts it to the FIRQB format used to open a file. Thus, you can accept a typed string from the job's terminal, and use .FSS to convert the string to the FIRQB format to do I/O on the file.

The .LOGS directive examines a string, and if it is a system logical device name defined by the system manager, returns the actual device name and unit number that corresponds to the logical name.

The LOKFQ subfunction of CALFIP can be used to search for disk files which meet "wildcard" file specifications. For example, you could search an account on disk for all files with names beginning with the characters DD.

3.2 CALFIP -- Call the File Processor

Form:

CALFIP

Function:

The CALFIP directive to the RSTS/E Monitor handles "housekeeping" necessary for input/output on RSTS/E. For example, CALFIP allows you to open a channel for file or device I/O.

The particular function desired is selected by setting a function field in the FIRQB (at offset FQFUN). Other parameters are also passed to the Monitor in the FIRQB, depending on the function requested. The functions are described in following subsections; a summary is given below:

FQFUN Value (Octal)	Mnemonic	Action Performed (BASIC-PLUS Equivalent)
0	CLSFQ	Close an open channel (CLOSE)
2	OPNFAQ	Open a channel (OPEN FOR INPUT)
4	CREFAQ	Create/extend/open a channel (file-structured OPEN FOR OUTPUT)
6	DLNFAQ	Delete a file by name (KILL)
10	RENFAQ	Rename a file (NAME...AS)
12	DIRFAQ	Get directory information
14	UUOFQ	Process UUO
16	ERRFAQ	Get error message text
20	RSTFAQ	Reset (close) all channels (except channel 0)
22	LOKFQ	Lookup a file
24	ASSFAQ	Assign a device
26	DEAFQ	Deassign a device
30	DALFAQ	Deassign all devices
32	CRTFAQ	Create/extend/open a temporary file on disk (file-structured OPEN FOR OUTPUT, space deallocated on close)
34	CRBFQ	Create/extend/open a compiled image file on disk (file-structured OPEN FOR OUTPUT, protection code bit 6 always set)

3.2.1 ASSFQ (Assign a Device)

Form:

```
MOVB #ASSFQ,FIRQB+FQFUN  
.  
.  
.  
(Set up FIRQB to define device)  
.  
.  
.  
CALFIP
```

Function:

The ASSFQ subfunction reserves a physical device for a job or transfers assignment of a currently owned device to another job.

Data Passed:

FIRQB

		Offset	Octal Mnemonic
1			0
3	FQFUN	ASSFQ (= octal 24)	2
5			4
7			6
11	(must = 0)	=0,assign;<>0,jobno.	10
13			12
15	DOS or ANS (1 word RAD50) or 0 (magtape)		14
17			16
21			20
23	100001 for snagging assign/reassign;else0		22
25			24
27			26
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no. real device unit number	32	FQDEVN
35			34
37			36

FIRQB+FQFUN	The function code ASSFQ (octal value = 24).
FIRQB+10	This byte is set to 0 to indicate an assign; if non-zero, it is used as the job number to which the device is to be reassigned. The high byte of this word (FIRQB+11) must be set to 0.
FIRQB+14	When the device is magtape, this word can contain either DOS or ANS in RAD50 format, to indicate DOS or ANSI label format for the magtape drive. Or it can be set to 0, to indicate the system default for the drive.
FIRQB+22	This word is set to 100001 to assign a device that is currently assigned to another user. This is a "snagging" assign, and is available only to privileged users. If this is not desired, this word must be zero.
FIRQB+FQDEV	Device name, as two ASCII characters.
FIRQB+FQDEVN	The device unit number is passed here in binary. A non-zero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no unit number.

Data Returned:

Other than a possible error in byte 0 of the FIRQB, no data is returned by the ASSFQ subfunction.

Errors:

For Assign (byte at FIRQB+10 = 0):

NODEVC	The device name specified at FIRQB+FQDEV is not a valid device name.
NOTAVL	The device and unit specified exists on the system, but the attempt to reserved it is prohibited because: <ul style="list-style-type: none"> 1. The device is currently reserved by another job. 2. Ownership of the device requires privilege that the user does not have. 3. The device or its controller has been disabled by the system manager. 4. The device is a keyboard line for a pseudo keyboard only.

For Reassign (byte at FIRQB+10 <> 0):

INUSE	The device specified is currently open or has an open file.
NODEVC	The device name is a logical device name for which a physical device is not currently assigned.
NOTAVL	(See description for Assign, above.)
BDNERR	The job number specified does not exist.

Example:

The following code reassigns magtape unit 0 (MT0:) to job 12.

```
MAGT: .ASCII  /MT/
      MOVB #ASSFQ,FIRQB+FQFUN ;SET FUNCTION CODE
      MOVB #12.,FIRQB+10 ;ASSIGN TO JOB 12
      CLRB FIRQB+11 ;CLEAR BYTE 11
      MOV #^RANS,FIRQB+14 ;ANSI-LABEL TAPE
      CLR FIRQB+22 ;NO SNAGGING REASSIGN
      MOV MAGT,FIRQB+FQDEV ;MAGTAPE DEVICE
      CLRB FIRQB+FQDEVN ;UNIT NO. 0
      MOVB #377,FIRQB+FQDEVN+1 ;UNIT NO. REAL
      CALFIP
      TSTB FIRQB ;ANY ERRORS?
      BNE ERRTN ;BRANCH TO PROCESS ERROR
```

3.2.2 CLSFQ (Close a Channel)

Form/Example:

```
CHANO=8.          ;SET VALUE FOR CHANNEL
MOVB   #CLSFQ,FIRQB+FQFUN ;SET FUNCTION CODE IN FIRQB
MOVB   #CHANO*2,FIRQB+FQFIL ;SET CHANNEL 8 FOR CLOSE
CALFIP          ;EXECUTE MONITOR DIRECTIVE
```

Function:

The CLSFQ function closes a channel. The specific action taken depends on the device or file which was previously opened on the channel, whether it was opened for input or output, and the mode with which it was opened. For example, closing a channel on which a magtape was opened for input in file-structured mode will cause the Monitor to position the tape at the end-of-file, and free the user buffer space allocated when the channel was opened. (Actions taken on closing various devices/files are described in the RSTS/E Programming Manual.)

Requesting CLSFQ for a channel which is not currently open will simply return with no error indicated.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	FQFIL
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQFUN The function code CLSFQ (octal value = 0).

FIRQB+FQFIL Channel number times 2; defines the channel to be closed.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the CLSFQ function of CALFIP.

Errors:

All possible errors with the CLSFQ function of CALFIP are device-dependent; see Appendix A for a full list of errors.

Example:

The following MACRO code closes the file or device on channel 12.

```
MOVB      #CLSFQ,FIRQB+FQFUN ;SET FUNCTION CODE IN FIRQB
MOVB      #12.*2,FIRQB+FQFIL ;SET CHANNEL 12 FOR CLOSE
CALFIP
TSTB      FIRQB             ;EXECUTE MONITOR DIRECTIVE
BNE       ERRTN            ;TEST BYTE 0 FOR ERROR
                           ;BRANCH TO PROCESS ERROR
```

3.2.3 CRBFQ (Create a Binary (Executable) File and Open It on a Channel)

Form:

```
MOVB #CRBFQ,FIRQB+FQFUN ;SET FUNCTION CODE
.
.
.
(set parameters in FIRQB appropriate to device)
.
.
.
CALFIP
```

Function:

The CRBFQ function is used to create and open a binary (executable) file. It is identical to the CREFQ function (Section 3.2.4), except that (1) the protection code is automatically set to indicate an executable file, and (2) the file must be opened on a disk device.

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3	FQFUN	CRBFQ (= octal 34)	2	
5	FQSIZM	(must = 0) channel no. * 2	4	FQFIL
7		project number programmer number	6	FQPPN
11		file name (2 words in RAD50 format)	10	FQNAM1
13			12	
15		file extension (1 word in RAD50 format)	14	FQEXT
17		file size (in 512-byte blocks)	16	FQSIZ
21			20	
23		mode	22	FQMODE
25			24	
27	FQPROT	file protection <>0,prot. code real	26	
31		device name (2 ASCII characters)	20	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35		file cluster size	34	FQCLUS
37		device cluster number for first block	36	FQNENT

- FIRQB+FQFUN The function code CRBFQ (octal value = 34).
- FIRQB+FQFIL Channel number times 2; defines the channel upon which the file is to be opened.
- FIRQB+FQSIZM On other types of "create" opens, this byte contains the most significant bits of the file size. Executable (binary) files cannot be greater than 65,535. (decimal) blocks, however, so this byte must always be passed as zero.
- FIRQB+FQPPN The project,programmer number (ppn) with which the file is to be created. The project number is in the high byte (FQPPN+1), and the programmer number in the low byte (FQPPN). A value of 0 in both bytes defaults to the ppn under which the calling program is running.

FIRQB+FQNAME	The filename created, as two words of RAD50 data.
FIRQB+FQEXT	The file extension, as one word of RAD50 data.
FIRQB+FQSIZ	The desired file size, in 512.-byte blocks. The file is pre-extended to the specified size; that is, the space for the file is allocated on the open, rather than as the file is written.
FIRQB+FQMODE	The mode with which the file is to be opened; values and actions taken are as described for the MODE modifier in file-structured OPEN FOR OUTPUT statements for disk, as described in the RSTS/E Programming Manual.
FIRQB+FQPROT	File protection code; values for this field define read/write and execute access to the created file (see RSTS/E System User's Guide). If you wish a default protection code, then set a full word of 0's at FIRQB+FQPROT-1. In this case, either the system default for protection code will be used, or if the CRBFQ will be deleting a previously existing file with the same filename extension, ppn, and device, the file protection code of the previously existing file will be used.
	To assign a specific file protection code, a non-zero value is passed in byte FIRQB+FQPROT-1 (by convention, 377 octal), and the specific file protection code in byte FIRQB+FQPROT. Bit 6 is automatically set, indicating that the file is executable. Meanings for protection codes for executable files are described in the RSTS/E System User's Guide.
FIRQB+FQDEV	The device name is passed here as two ASCII characters; it must be a disk device. If this word is 0, the public disk structure is assumed.
FIRQB+FQDEVN	The device unit number is passed here in binary. A non-zero value in FQDEVN+1 indicates an explicit device unit number. A zero value in FQDEVN+1 indicates no unit number.
FIRQB+FQCLUS	This parameter has the same function as the CLUSTERSIZE option in BASIC-PLUS. A description of the CLUSTERSIZE option for disk is given in the BASIC-PLUS Language Manual.
FIRQB+FQNENT	Device cluster number for placement of block 1 of the file. When creating a new file, block 1 of the file can be placed on a particular block by specifying the disk device cluster number in this word. If this word is zero, no placement is done. If it is non-zero, the Monitor will try to place the file at the specified device cluster, or as near after it as possible. If the first block of the file can be placed at or after the specified device cluster number, the Monitor sets a

bit in the file's entry in the User File Directory (an internal Monitor directory). If the first block of the file cannot be placed at or after the specified device cluster number, the file is placed at the lowest free block on the disk, the bit in the file's entry in the User File Directory is not set, and no error is returned.

Data Returned:

FIRQB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1	/ /		0
3	/ / / / / / / / / / / / current job no. * 2	2	FQJOB
5 FQSIZM	(always 0) channel no. * 2	4	FQFIL
7	project number programmer number	6	FQPPN
11	file name (2 words in RAD50 format)	10	FQNAM1
13		12	
15	file extension (1 word in RAD50 format)	14	FQEXT
17	size of the file, in 512.-byte blocks	16	FQSIZ
21	reasonable buffer size for device	20	FQBUFL
23	(as passed)	22	FQMODE
25	device description	24	FQFLAG
27 FQPROT	protection code clustersize, mod256	26	
31	device name (2 ASCII characters)	30	FQDEV
33	flag byte device unit number	32	FQDEVN
35	file identification index	34	
37	(as passed)	36	FQNENT

FIRQB+FQJOB Current job number times two.

FIRQB+FQFIL Channel number times two; defines the channel on which the file is open.

FIRQB+FQPPN The project,programmer number under which the file is open. An actual project,programmer number is returned here even if this word was passed as 0.

FIRQB+FQNAM1	The filename created, as two words of RAD50 data.
FIRQB+FQEXT	The file extension created, as one word of RAD50 data.
FIRQB+FQSIZ	The size to which the file was pre-extended, in 512.-byte blocks.
FIRQB+FQBUFL	Reasonable buffer size for disk reads and writes, in bytes. (Always 512. for disk.)
FIRQB+FQFLAG	Description of the device just opened (the same information as the BASIC-PLUS STATUS variable). The low byte contains the device's handler index; always 0 (DSKHND) for disk. The high byte contains a set of status flags; irrelevant here since the device is always disk. (See the OPNFQ subfunction if you are interested in these settings.)
FIRQB+FQPROT-1	The file clustersize, modulo 256. That is, a file clustersize of 256. is indicated by a zero byte here. This is the same as the value passed at FIRQB+FQCLUS, except that it is returned in a byte instead of a word.
FIRQB+FQPROT	The protection code of the file. Bit 7 is 0, bit 6 is 1; bits 5 - 0 are as passed.
FIRQB+FQDEV	The device name of the disk device, as two ASCII characters. The actual device name is returned here, even if this word was passed as 0.
FIRQB+FQDEVN	The device unit number. The actual unit number is returned here, even if FIRQB+FQDEVN+1 was passed as 0.
FIRQB+34	The file identification index of this file. This word is significant mainly in that it can be used in place of the filename in subsequent opens of the file on disk. You can open the file with the OPNFQ subfunction of CALFIP using an explicit project, programmer number in FIRQB+FQPPN, a zero word in FIRQB+FQNAM1, and the file identification index in FIRQB+FQNAM1+2.

Errors:

NOTCLS	The specified channel is already open. It must be closed before it can be opened again.
PRVIOL	The specified device is not a disk device. The CRBFQ function can only be executed for a disk device.
xxxxx	Other errors are device-dependent. See Appendix A for a full list of possible error codes.

Example:

The following MACRO code sets up the FIRQB for the CRBFQ function of CALFIP. The ppn is set to 2,210; the filename and extension are set to FILNAM.EXT. The protection code is set such that the file is read/write-protected against everyone but the caller (user with ppn 2,210), and execute-protected against all but the caller and those in the caller's project (users with project number=2). The file is opened on disk unit 2 (DK2:). Filesize and clustersize are not specified. The cluster size defaults to the pack cluster size and the file size is not preallocated.

```
DK:      .ASCII   /DK/
        MOVB    #CRBFQ,FIRQB+FQFUN      ;SET FUNCTION CODE
        MOVB    #4*2,FIRQB+FQFIL       ;SET CHANNEL = 4
        MOVB    #2,FIRQB+FQPPN+1      ;SET PROJECT NUMBER =2
        MOVB    #210.,FIRQB+FQPPN     ;SET PROG. NO.=210.
        MOV     #^RFIL,FIRQB+FQNAM1    ;SET FILE NAME AND
        MOV     #^RNAM,FIRQB+FQNAM1+2  ;EXTENSION TO
        MOV     #^REXT,FIRQB+FQEXT    ;;"FILNAM.EXT"
        MOVB    #<8.+16.+32.>,FIRQB+FQPROT ;SET PROTECTION CODE
        MOVB    #377,FIRQB+FQPROT-1    ;SET PROTECTION CODE REAL
        MOV     DK,FIRQB+FQDEV        ;SET DEVICE TO DISK,
        MOVB    #2,FIRQB+FQDEVN       ;UNIT 2
        MOVB    #377,FIRQB+FQDEVN+1   ;(EXPLICIT DEVICE NO.)
CALFIP
```

3.2.4 CREFQ (Create a File and Open It on a Channel)

Form:

```
MOVB      #CREFQ,FIRQB+FQFUN    ;SET FUNCTION CODE  
.  
.  
(set parameters appropriate to device)  
. .  
.  
CALFIP
```

Function:

The CREFQ function performs the same action as a file-structured OPEN FOR OUTPUT statement in BASIC-PLUS. Parameters defining the device, filename and extension, protection code, mode, file size, and cluster size can be used by setting values in the FIRQB. The choice depends upon the device.

For example, CREFQ with a filename and extension on a magtape device with mode = 128 (decimal) would cause an "open for append" operation. A search for an existing file with the specified name and on the specified device would be made; the file would have to be the last file on the tape. When found, the tape would be positioned after the last record in the file, ready for data to be written and appended. The RSTS/E Programming Manual describes the file-structured OPEN FOR OUTPUT operation for the various devices.

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//	0	
3	FQFUN	CREFQ (= octal 4) //	2	
5	FQSIZM	MSB of file size channel no. * 2	4	FQFIL
7		project number programmer number	6	FQPPN
11			10	FQNAM1
13		filename (2 words in RAD50 format)	12	
15		file extension (1 word in RAD50 format)	14	FQEXT
17		LSB (least significant bits of) file size	16	FQSIZ
21		//	20	
23		mode	22	FQMODE
25		//	24	
27	FQPROT	file protection <>0, prot.code real	26	
31		device name (2 ASCII characters)	20	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35		file cluster size	34	FQCLUS
37		device cluster number for first block	36	FQNENT

- FIRQB+FQFUN The function code CREFQ (octal value = 4).
- FIRQB+FQFIL Channel number times 2; defines the channel upon which the file is to be opened.
- FIRQB+FQSIZM For large disk files (greater than 65,535 blocks), this byte contains the most significant bits of the file size. See FIRQB+FQSIZ, below, for a discussion of the entire 24-bit field is used for large files on disk.
- FIRQB+FQPPN The project,programmer number (ppn) with which the file is to be created. The project number is in the high byte (FIRQB+FQPPN+1), and the programmer number in the low byte (FIRQB+FQPPN). A value of 0 defaults to the ppn under which the calling program is running.

FIRQB+FQNAME	The filename to create, as two words of RAD50 data.
FIRQB+FQEXT	The file extension, as one word of RAD50 data.
FIRQB+FQSIZ	The desired file size, in 512.-byte blocks. This parameter is relevant only for disk and ANSI magtape files. For disk files, this word forms the least significant bits of the file size. It is combined with the byte at FIRQB+FQSIZM to indicate the file size. The disk file is automatically preextended to the indicated size. (That is, the space for the file is allocated on the open, not as the file is written.) For ANSI magtape, this word has the same function as the FILESIZE option in BASIC-PLUS, as described in the <u>RSTS/E Programming Manual</u> .
FIRQB+FQMODE	The mode with which the file is to be opened; values and actions taken for specific devices are as described for the MODE modifier for file-structured OPEN FOR OUTPUT statements in the <u>RSTS/E Programming Manual</u> .
FIRQB+FQPROT	File protection code; values for this field define subsequent read and write access to the opened file (see <u>RSTS/E System User's Guide</u>). If you wish a default protection code, then set a full word of 0's at FIRQB+FQPROT-1. In this case, either the system default for protection code will be used, or, if the CREFQ will be deleting a previously existing file with the same filename, extension, ppn, and device, the file protection code of the previously existing file will be used. To assign a specific file protection code, a non-zero value is passed in byte FIRQB+FQPROT-1 (by convention, 377 octal), and the specific file protection code in byte FIRQB+FQPROT. This allows an explicit file protection code of 0. Bit 6 is always cleared by the CREFQ function, regardless of whether or not it is set in FIRQB+FQPROT. Bit 6 is the "compiled file" bit; compiled files should be opened with the CRBFQ function (Section 3.2.3).
FIRQB+FQDEV	The device name is passed here as two ASCII characters. A zero word indicates the public disk structure.
FIRQB+FQDEVN	The device unit number is passed here in binary. A nonzero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no unit number.
FIRQB+FQCLUS	This parameter has the same function as the CLUSTERSIZE option in BASIC-PLUS. It is relevant only for disk files and ANSI magtape files. A description of the CLUSTERSIZE option for disk is given in the <u>BASIC-PLUS Language Manual</u> ; for ANSI magtape, in the <u>RSTS/E Programming Manual</u> .

FIRQB+FQENT For disk files, the device cluster number for placement of block 1 of the file. When creating a new file, block 1 of the file can be placed on a particular block by specifying the disk device cluster number in this word. If this word is zero, no placement is done. If it is non-zero, the Monitor will try to place the file at the specified device cluster, or as near after it as possible. If the first block of the file can be placed at or after the specified device cluster number, the Monitor sets a bit in the file's entry in the User File Directory (an internal Monitor directory). If the first block of the file cannot be placed at or after the specified device cluster number, the file is placed at the lowest free block on the disk, the bit in the file's entry in the User File Directory is not set, and no error is returned.

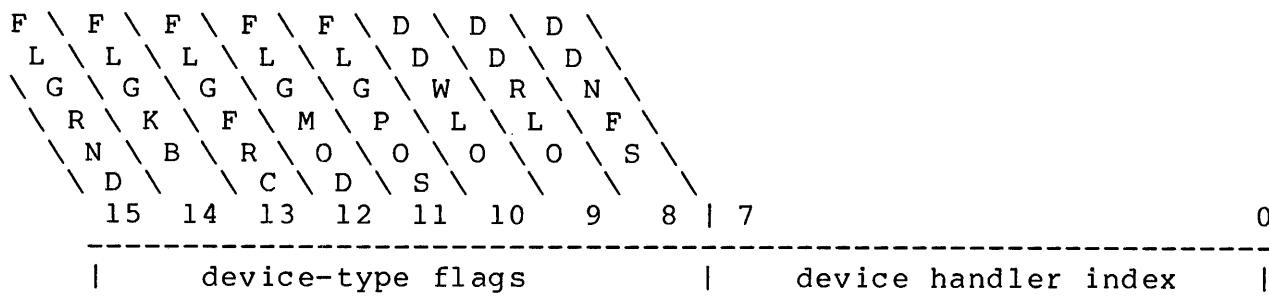
Data Returned:

FIRQB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1	/ /	0	
3	/ / / / / / / / / / / / current job no. * 2	2	FQJOB
5 FQSIZM	MSB of file size channel no. * 2	4	FQFIL
7	project number programmer number	6	FQPPN
11	filename (2 words in RAD50 format)	10	FQNAM1
13		12	
15	file extension (1 word in RAD50 format)	14	FQEXT
17	LSB (least significant bits) of file size	16	FQSIZ
21	reasonable buffer size for device	20	FQBUFL
23	(as passed)	22	FQMODE
25	device description	24	FQFLAG
27 FQPROT	protection code clustersize, mod256	26	
31	device name (2 ASCII characters)	30	FQDEV
33	flag byte device unit number	32	FQDEVN
35	file identification index	34	
37	(as passed)	36	FQENT

NOTE: For non-disk device, the relevant information returned with the CREFQ subfunction is in the two words at FIRQB+FQBUFL and FIRQB+FQFLAG. All other words are simply returned as passed.

- FIRQB+FQJOB Current job number times two.
- FIRQB+FQFIL Channel number times two; defines the channel on which the file is open.
- FIRQB+FQSIZM For large files, this byte contains the most significant bits of the size to which the file was pre-extended, in 512.-byte blocks. This byte is combined with the word at FIRQB+FQSIZ to form a 24-bit field giving the file size.
- FIRQB+FQPPN The project,programmer number under which the file is open. An actual project,programmer number is returned here even if this word was passed as 0.
- FIRQB+FQNAM1 The filename created, as two words of RAD50 data.
- FIRQB+FQEXT The file extension created, as one word of RAD50 data.
- FIRQB+FQSIZ The size to which the file was pre-extended, in 512.-byte blocks.
- FIRQB+FQBUFL Reasonable buffer size for this device, in bytes. If you are doing device-independent I/O (that is, if you do not wish to keep track of which device is being opened, and perform specific opens, reads, and writes, depending on the device), this value is the Monitor's "best guess" for a buffer size to use in subsequent .READS and .WRITES on the opened channel. (See Sections 3.17 and 3.33 on .READ and .WRITE.)
- FIRQB+FQFLAG Description of the device just opened (same information as the BASIC-PLUS STATUS variable). The low byte contains the device's handler index. There is one unique handler index for all device types. The high byte contains a set of status flags to allow for device-independent I/O routines.



High Byte -- Device-Type Flags:

The bits in the high byte of the flags word are set to indicate the type of file or device just opened:

FLGRND = 1 The device or file is random-access.
= 0 The device or file is sequential.

FLGKB = 1 The file/device is a terminal-type file/device (or is generically a terminal).
= 0 The file/device is not a terminal-type file/device.

FLGFRC = 1 The file/device is byte-oriented. That is, .READs and .WRITEs handle data in byte units.
= 0 The file/device is block-oriented. .READs and .WRITEs handle data in block units.

FLGMOD = 1 The file/device accepts modifiers in .READs and .WRITEs (Sections 3.17 and 3.33, XRB+XRMOD).
= 0 The file/device does not accept modifiers in .READs and .WRITEs.

FLGPOS = 1 The file/device keeps track of its horizontal position, and expands characters such as TAB into whatever is appropriate for the file/device. You can determine the current horizontal position with the .POSTN directive.
= 0 The file/device does not keep track of its horizontal position.

DDWLO = 1 The file/device has been write-locked (via the protection code or mode value in the open), or is generically a write-only device.
= 0 The file/device is not write-locked.

DDRLO = 1 The file/device has been read-locked (via the protection code in the open), or is generically a read-only device.
= 0 The file/device is not read-locked.

DDFNS = 1 The file/device is non-file structured (or is generically not a file-structured device).
= 0 The file/device is file-structured.

Low Byte -- Device Handler Index:

Bits 0-7 of the flags word contain a handler index which indicates the generic kind of device. Currently defined values are:

Octal Value	Symbol	Meaning
0	DSKHND	All disks
2	TTYHND	All terminals
4	DTAHND	DECtape
6	LPTHND	All line printers
10	PTRHND	Paper tape reader
12	PTPHND	Paper tape punch
14	CDRHND	Card reader
16	MTAHND	Magtape
20	PKBHND	Pseudo-keyboards
22	RXDHND	Floppy disks
24	RJEHND	2780 remote job entry
26	NULHND	The null device
30	DMCHND	The DMC11 DDCMP interface
32	AUDHND	Auto-dial interface
34	PLTHND	X-Y Plotter
36	DT2HND	DECtape II
40	KMCHND	KMC11
42	IBMHND	IBM interconnect

FIRQB+FQPROT-1 The file clustersize, modulo 256. That is, a file clustersize of 256. is indicated by a zero byte here. This is the same as the value passed at FIRQB+FQCLUS, except that it is returned in a byte instead of a word.

FIRQB+FQPROT The protection code of the file. Bits 6 and 7 are 0; bits 5 - 0 are as passed.

FIRQB+FQDEV The device name of the disk device, as two ASCII characters. The actual device name is returned here, even if this word was passed as 0.

FIRQB+FQDEVN The device unit number. The actual unit number is returned here, even if FIRQB+FQDEVN+1 was passed as 0.

FIRQB+34 The file identification index of this file. This word is significant mainly in that it can be used in place of the filename in subsequent opens of the file on disk. You can open the file with the OPNFQ subfunction of CALFIP using an explicit project, programmer number in FIRQB+FQPPN, a zero word in FIRQB+FQNAM1, and the file identification index in FIRQB+FQNAM1+2.

Errors:

NOTCLS The specified channel is already open. It must be closed before it can be opened again.

xxxxx Other errors are device-dependent. See Appendix A for a full list of errors.

Example:

The following MACRO code sets up the FIRQB for a CREFQ which opens a file called FILE01.LST on the public disk structure. (We assume here that previous code has filled the FIRQB with zeros, so that the words at FIRQB+FQDEV and FIRQB+FQDEVN are 0, indicating the public disk. Similarly, the mode (FIRQB+FQMODE) and cluster size (FIRQB+FQCLUS) are zero, indicating normal read/write, and the default cluster size.) A protection code of 56 is assigned (write protected against all but the owner, read-protected against all but those in the owner's project).

```
MOVB        #CREFQ,FIRQB+FQFUN        ;SET FUNCTION CODE IN FIRQB
MOVB        #5*2,FIRQB+FQFIL        ;SET CHANNEL = 5
MOV         #^RFIL,FIRQB+FQNAM1      ;SET FILENAME
MOV         #^RE01,FIRQB+FQNAM1+2     ;AND EXTENSION TO
MOV         #^RLST,FIRQB+FQEXT        ;FILE01.LST
MOVB        #56.,FIRQB+FQPROT        ;SET PROTECTION CODE
MOVB        #377,FIRQB+FQPROT-1      ;EXPLICIT PROT. CODE
CALFIP
```

3.2.5 CRTFQ (Create and Open a Temporary File)

Form:

```
MOV B      #CRTFQ,FIRQB+FQFUN  
.  
.  
(set appropriate parameters)  
.  
.  
CALFIP
```

Function:

The CRTFQ function can be used to create and open a temporary file on disk. The file is "temporary" only in that the Monitor generates a filename and extension for the file which is recognized by the LOGOUT utility. LOGOUT destroys such files when the user logs out; the Monitor does not inherently destroy temporary files created with CRTFQ.

Most parameters relevant on an open are defaulted. The programmer does not define or refer to the file by name; subsequent read and write operations refer to the channel on which the temporary file is open.

In addition to the function code, you specify a channel, and if desired, a file size and/or file cluster size. If an explicit cluster size is specified, it will be used. A specified file size may or may not be used. The Monitor will attempt to reuse an existing temporary file for the same job by simply re-opening that file. In this case, the file's size will be the size of the previous file. If no previous temporary file for the job exists, or if one does exist but is already in use by somebody else, then a new file will be created, with the specified file size. A new file will also be created if an explicit cluster size is given.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	CRTFQ (= octal 32)	2
5	FQSIZM	MSB of file size channel no. *2	4 FQFIL
7		6	
11		10	
13		12	
15		14	
17		16 FQSIZ	LSB of file size (no. of 512-byte blocks)
21		20	
23		22 FQMODE	mode
25		24	
27		26	
31		30 FQDEV	device name (2 ASCII characters)
33		32 FQDEVN	<>0, unit no. real device unit number
35		34 FQCLUS	file cluster size
37		36 FQNENT	device cluster number for first block

FIRQB+FQFUN The function code CRTFQ (octal value = 32).

FIRQB+FQFIL Channel number times 2; defines the channel upon which the file is to be opened.

FIRQB+FQSIZM For large disk files (greater than 65,535. blocks), this byte contains the most significant bits of the file size. See FIRQB+FQSIZ, below, for a description of how the entire 24.-bit field is used.

FIRQB+FQSIZ This word contains the least significant bits of the file's size in 512.-byte blocks. (It is combined with the byte at FIRQB+FQSIZM.) The file size may or may not

be used. If a temporary file already exists which is not in use, the Monitor will use the space allocated to the previous temporary file. However, if cluster size is specified, a new file will be created, and the file size indicated by the 24-bit file size will be used.

FIRQB+FQMODE	The mode with which the file is to be opened. Values are as described for the MODE modifier in OPEN FOR OUTPUT statements for disk, as described in the <u>RSTS/E Programming Manual</u> . The only relevant modes are for creating a tentative file, creating a contiguous file, creating a conditionally contiguous file, and for data caching. All other mode bits are ignored.
FIRQB+FQDEV	The device name is passed here as two ASCII characters; it must be a disk device. A value of 0 in this word indicates "SY", public disk.
FIRQB+FQDEVN	The device unit number is passed here in binary. A non-zero value in the high byte (FIRQB+FQDEVN+1) indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no unit number.
FIRQB+FQCLUS	File cluster size. Performs the same function as the CLUSTERSIZE option in BASIC-PLUS.
FIRQB+FQNENT	The device cluster number for placement of block 1 of the file on disk. When creating a new file, block 1 of the file can be placed on a particular block by specifying the disk device cluster number in this word. If this word is zero, no placement is done. If it is non-zero, the Monitor will try to place the file at the specified device cluster, or as near after it as possible. If the first block of the file can be placed at or after the specified device cluster number, the Monitor sets a bit in the file's entry in the User File Directory (an internal Monitor directory). If the first block of the file cannot be placed at or after the specified device cluster number, the file is placed at the lowest free block on the disk, the bit in the file's entry in the User File Directory is not set, and no error is returned.

Data Returned:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3		current job no. * 2	2	FQJOB
5	FQSIZM	MSB of file size channel no. * 2	4	FQFIL
7		project number programmer number	6	FQPPN
11		file name (2 words in RAD50 format)	10	FQNAM1
13			12	
15		file extension (1 word in RAD50 format)	14	FQEXT
17		LSB (least significant bits) of file size	16	FQSIZ
21		reasonable buffer size for device	20	FQBUFL
23		(as passed)	22	FQMODE
25		device description	24	FQFLAG
27	FQPROT	protection code clustersize, mod256	26	
31		device name (2 ASCII characters)	30	FQDEV
33		flag byte device unit number	32	FQDEVN
35		file identification index	34	
37		(as passed)	36	FQNENT

FIRQB+FQJOB Current job number times two.

FIRQB+FQFIL Channel number times two; defines the channel on which the file is open.

FIRQB+FQSIZM For large disk files (>65,535. blocks), this byte contains the most significant bits of the file size in 512.-byte blocks. It is combined with the word at FIRQB+FQSIZ to form a 24-bit field giving the file size.

FIRQB+FQPPN The project,programmer number under which the file is open. An actual project,programmer number is returned here even if this word was passed as 0.

FIRQB+FQNAM1 The filename created, as two words of RAD50 data.

FIRQB+FQEXT	The file extension created, as one word of RAD50 data.
FIRQB+FQSIZ	The size to which the file was pre-extended, in 512.-byte blocks.
FIRQB+FQBUFL	Reasonable buffer size for disk reads and writes, in bytes. (Always 512. for disk.)
FIRQB+FQFLAG	Description of the device just opened (the same information as the BASIC-PLUS STATUS variable). The low byte contains the device's handler index; always 0 (DSKHND) for disk. The high byte contains a set of status flags; irrelevant here since the device is always disk. (See the OPNFQ subfunction if you are interested in these settings.)
FIRQB+FQPROT-1	The file clustersize, modulo 256. That is, a file clustersize of 256. is indicated by a zero byte here. This is the same as the value passed at FIRQB+FQCLUS, except that it is returned in a byte instead of a word.
FIRQB+FQPROT	The protection code of the file.
FIRQB+FQDEV	The device name of the disk device, as two ASCII characters. The actual device name is returned here, even if this word was passed as 0.
FIRQB+FQDEVN	The device unit number. The actual unit number is returned here, even if FIRQB+FQDEVN+1 was passed as 0.
FIRQB+34	The file identification index of this file. This word is significant mainly in that it can be used in place of the filename in subsequent opens of the file on disk. You can open the file with the OPNFQ subfunction of CALFIP using an explicit project, programmer number in FIRQB+FQPPN, a zero word in FIRQB+FQNAM1, and the file identification index in FIRQB+FQNAM1+2.

Errors:

xxxxx All errors for this directive are device-dependent. See Appendix A for a full list of errors.

Example:

The following MACRO code opens a temporary file on channel 13. (We assume that the FIRQB has been initialized to all zeroes. Hence, the temporary file will be created on the public disk structure.)

```

MOV B #CRTFQ,FIRQB+FQFUN ;SET FUNCTION CODE
MOV B #13.*2,FIRQB+FQFIL ;SET CHANNEL TO 13
CALFIP

```

3.2.6 DALFQ (Deassign All Devices)

Form:

```
MOVB    #DALFQ, FIRQB+FQFUN  
CALFIP
```

Function:

DALFQ deassigns all devices currently assigned to the job.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	DALFQ (= octal 30)
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQFUN The function code DALFQ (octal value = 30).

Data Returned:

No data is returned by the DALFQ subfunction of CALFIP.

Errors:

No errors are possible with DALFQ.

Example:

See "Form", above.

3.2.7 DEAFQ (Deassign a Device)

Form:

```
MOVB    #DEAFQ,FIRQB+FQFUN
.
.
.
(Define device to be deassigned in FIRQB)
.
.
.
CALFIP
```

Function:

The DEAFQ subfunction deassigns a device from the current job (releases it for use by other jobs).

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11	(must = 0)	10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no.real device unit number	32	FQDEVN
35		34	
37		36	

- FIRQB+FQFUN The DEAFQ function code (octal value = 26).
- FIRQB+10 The word beginning at this location must be set to 0.
- FIRQB+FQDEV The name of the device to be deassigned, as two ASCII characters.
- FIRQB+FQDEVN The device unit number is passed here in binary. A non-zero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no device unit number.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the DEAFQ subfunction of CALFIP.

Errors:

NODEVC The device or its type specified at FIRQB+FQDEV and FIRQB+FQDEVN is not part of your system configuration.

Example:

The following code deassigns LP: from the current job.

```
MOVB            #DEAFQ,FIRQB+FQFUN        ;SET FUNCTION CODE
CLR            FIRQB+10                    ;CLEAR WORD 10
MOV            "#LP,FIRQB+FQDEV        ;DEVICE=LINE PRINTER
CLR            FIRQB+FQDEVN            ;LINE PRINTER IS LP:
CALFIP
```

3.2.8 DIRFQ (Get Directory Information)

Form:

```
MOVB    #DIRFQ,FIRQB+FQFUN
.
.
.
(Define device for which directory wanted)
.
.
.
CALFIP
```

Function:

The DIRFQ subfunction of CALFIP returns directory information about a disk, DECtape, or magtape file. Two forms of the call are available. One leaves a magtape file positioned at the end-of-file, and returns the size of the file as part of the directory information. The second form, for magtape only, leaves a magtape file positioned at the beginning of the file for which data is returned, and does not return the size of the file.

Data Passed -- Directory Lookup on Index:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//	0	
3 FQFUN		DIRFQ (= octal 12) //	2	
5		index of file to read	4	
7		project number programmer number	6 FQPPN	
11		//	10	
13		//	12	
15		//	14	
17		//	16	
21		//	20	
23		//	22	
25		//	24	
27		//	26	
31		device name (2 ASCII characters)	30 FQDEV	
33		<>0, unit no. real device unit number	32 FQDEVN	
35		//	34	
37		//	36	

FIRQB+FQFUN The function code DIRFQ (octal value = 12).

FIRQB+4 The index of the file to read. If this word is zero, the Monitor returns data for the first file in the directory. For some positive value n, the Monitor returns data for the n+1 file in the directory. For magtape, a value of 0 will cause the Monitor to rewind the tape before it gets the information for the first file (by reading the label record of the file). The Monitor will then space the tape forward to the next end-of-file record and calculate the number of records in the file. The tape will be left in that position. A non-zero value will perform the same action, except that the tape is not rewound.

For DECtape, the first call issued must have a value of zero in this word to read the directory blocks from the tape before reading the first file. Subsequent calls with this word non-zero read the directory from the BUFF.SYS file. (Directory information for DECtape is kept in this system file on disk to speed up DECtape file processing, as described in the RSTS/E Programming Manual.)

FIRQB+FQPPN

The project,programmer number of the directory to look up, for disk or magtape. (The Monitor does not use these bytes if the device is DECtape, but simply returns information for each file read on the device.)

If this word is zero and the device is disk, this directive returns information for the project,programmer number under which this job is being executed.

If this word is zero and the device is magtape, this directive returns information for each file read, regardless of the project,programmer number under which it was written.

FIRQB+FQDEV

The device name, as two ASCII characters. Must be disk, magtape, or DECtape. If this word is zero, the public disk structure (SY:) is used.

FIRQB+FQDEVN

The device unit number is passed here in binary. A non-zero value in FQDEVN+1 indicates an explicit device unit number. A zero value in FQDEVN+1 indicates no unit number.

Data Passed -- Special Magtape Lookup:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3 FQFUN	DIRFQ (= octal 12)	//////////	2	
5	index of file to read		4	
7	must = 177777 octal for magtape lookup		6	
11		//////////	10	
13		//////////	12	
15		//////////	14	
17		//////////	16	
21		//////////	20	
23		//////////	22	
25		//////////	24	
27		//////////	26	
31	MT or MM (2 ASCII characters)		30 FQDEV	
33	<>0,unit no.real	device unit number	32 FQDEVN	
35		//////////	34	
37		//////////	36	

FIRQB+FQFUN The function code DIRFQ (octal value = 12).

FIRQB+4 Index number of the file to be read. If this word is zero, information will be returned for the first file in the directory. If this word is some positive value n, information will be returned for the n+1 file in the directory. A value of 0 will cause the Monitor to rewind the tape before getting information from the first file (by reading the label record). It will then simply backspace the tape one record, leaving it positioned at the beginning of file. (This action leaves the tape positioned such that an open on this file will succeed on a single read from tape.) A non-zero value will not cause the tape to be rewound; the next record is read (it must be a label) and the tape backspaced one record. The normal action in searching a tape for specific files to be read is to

execute this directive with value of 0 first. If the file is one to be read, open the file requesting no rewind, process the file, and close it to position the tape at the end-of-file. If the file is not the one to be read, space the tape forward to the next end-of-file. (This can be done in MACRO with the .SPEC directive, Section 3.2.3). Then issue this call with a non-zero value in the word beginning at FIRQB+4, and continue the process.

FIRQB+6 This word must be set to 177777 (octal) for the special magtape lookup operation.

FIRQB+FQDEV The device name (MT or MM) as two ASCII characters.

FIRQB+FQDEVN The device unit number, in binary. A non-zero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no unit number.

Data Returned (Both):

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3		////////// current job no. * 2	2	FQJOB
5		(same as data passed)	4	
7		project number programmer number	6	FQPPN
11		file name (2 words in RAD50 format)	10	FQNAM1
13			12	
15		file extension (1 word in RAD50 format)	14	FQEXT
17		LSB of file length in blocks(not magtape)	16	FQSIZ
21		MSB of file length protection code	20	
23		date of last access	22	
25		date of creation	24	
27		time of creation	26	
31		(same as data passed)	30	FQDEV
33			32	FQDEVN
35		file cluster size (disk only)	34	FQCLUS
37		USTAT byte no. entries ret.	36	

FIRQB+FQJOB The current job number times two.

FIRQB+FQPPN The project,programmer number of the file. The project number is contained in the high byte (FIRQB+FQPPN+1) the programmer number in the low byte (FIRQB +FQPPN).

FIRQB+FQNAM The filename, as two words in RAD50 format.

FIRQB+FQEXT The file extension, as one word in RAD50 format.

FIRQB+FQSIZ The least significant bits of the file size, in 512.-byte blocks. This word is combined with the byte at FIRQB+21 to form a 24-bit value giving the file's size. (This word is not returned with the special magtape directory lookup.)

FIRQB+20	This byte contains the protection code of the file.
FIRQB+21	This byte contains the most significant bits of the file's size in 512.-byte blocks (see FIRQB+FQSIZ). (This byte is not returned with the special magtape directory lookup.)
FIRQB+22	The date the file was last accessed, in system internal format: (year -1970 * 1000.) + day-within-year (See the .DATE directive, Section 3.7, for a discussion of the system internal format for dates, if necessary.)
FIRQB+24	The date the file was created, in system internal format.
FIRQB+26	The time that the file was created, in system internal format: minutes before midnight, where midnight = 1440. (See the .DATE directive for a discussion of the system internal format for time, if necessary.)
FIRQB+FQCLUS	For disk devices, this word contains the file cluster size. For tape, not used.
FIRQB+36	Number of entries returned: for disk, 8; for tape, 6.
FIRQB+37	Internal flag information:
Bit	Meaning
002	Placed file.
004	Some job has write access now.
010	File is open in update mode.
020	File is contiguous; no extend available.
040	No delete or rename allowed.
100	MFD type entry.
200	File is marked for deletion.

Errors:

NOSUCH	The account (project,programmer number) does not exist on the device specified, or no more files exist on the account (the index number is greater than the number of files on the account). Or, for the special magtape lookup, no more files exist on the tape.
DEVNFS	The device specified is not a file-structured device.
xxxxx	Other errors are device-dependent. See Appendix A for a full list of errors.

Example:

The following code searches the disk directory to examine files for user [2,101].

```
LOOP:    MOVB      #DIRFQ,FIRQB+FQFUN      ;SET FUNCTION CODE
          CLR       FIRQB+4
          MOV       #<2.*400+101.>,FIRQB+FQPPN ;SET PROJ.,PROG.NO.
          CLR       FIRQB+FQDEV        ;SEARCH SYSTEM DISK
          CLR       FIRQB+FQDEVN
          CALFIP
          .
          .
          .
          (Error processing, examine filename, process file)
          .
          .
          .
          .
          INCB      FIRQB+4      ;INCREMENT INDEX
          JMP       LOOP        ;GO BACK FOR NEXT ROUND
```

3.2.9 DLNFQ (Delete a File)

Form:

```
MOV B      #DLNFQ,FIRQB+FQFUN
.
.
.
(Define file to be deleted in FIRQB)
.
.
.
CALFIP
```

Function:

The DLNFQ function of CALFIP deletes a file from disk or DECTape. The Monitor's internal data on the location of the file are destroyed, and the file's space on the device is made available for general use.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	DLNFQ (=octal 6)	2	
5		4	
7	project number programmer number	6 FQPPN	
11		10 FQNAM1	
13	file name (2 words in RAD50 format)	12	
15		14 FQEXT	
17		16	
21		20	
23		22	
25		24	
27		26	
31	device name (2 ASCII characters)	30 FQDEV	
33	<>0, unit no. real device unit number	32 FQDEVN	
35		34	
37		36	

FIRQB+FQFUN The function code DLNFQ (octal value = 6).

FIRQB+FQPPN The project, programmer number (ppn) of the file to be deleted. The project number is in the high byte (FIRQB+FQPPN+1), and the programmer number is in the low byte (FIRQB+FQPPN). A value of 0 defaults to the ppn under which the calling program is running.

FIRQB+FQNAM1 The name of the file to be deleted, as two words of RAD50 data.

FIRQB+FQEXT The file extension, as one word of RAD50 data.

FIRQB+FQDEV The name of the device containing the file to be deleted, as two ASCII characters; it must be a disk or DECtape device. A value of 0 in this word indicates the public disk structure.

FIRQB+FQDEVN The device unit number is passed here in binary. A non-zero value in FQDEVN+1 indicates an explicit device unit number. A zero value in FQDEVN+1 indicates no unit number.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the DLNFQ function of CALFIP.

Errors:

NOSUCH The file specified in the data passed cannot be found.

PRVIOI Protection violation. An attempt was made to delete a file write-protected against the caller, or which is marked for no delete.

Example:

The following code deletes the file MYFILE.LST from the public structure on the user's account. (Assume that the FIRQB has been filled with zeros previously).

```
MOV B      #DLNFQ,FIRQB+FQFUN
MOV      #^RMYF,FIRQB+FQNAM1      ;SET FILE NAME
MOV      #^RILE,FIRQB+FQNAM1+2    ;AND EXTENSION
MOV      #^RLST,FIRQB+FQEXT      ;TO "MYFILE.LST"
CALFIP
```

3.2.10 ERRFQ (Return Error Message Text)

Form/Example:

```
MOVB    #ERRFQ,FIRQB+FQFUN  
MOVB    #ERR,FIRQB+FQERNO
```

Function:

The ERRFQ subfunction of CALFIP returns error message text from the system error message file or from the default error message file if an error message file is not currently installed. The text is associated with the value of the error code passed as byte 4 of the FIRQB. This call returns the full RSTS/E error message text associated with errors returned in byte 0 of the FIRQB on all the Monitor directives.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	FQERNO
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQFUN The function code ERRFQ (octal value = 16).

FIRQB+FQERNO The error code value (in binary) for which the corresponding error message text is to be returned.

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	KB*2(-KB*2+1, dtch.) job number * 2	2	FQJOB
5		4	
	error message -- padded with NULs to 28 characters (ASCII format)		
37		36	

FIRQB+FQJOB The current job number times two.

FIRQB+3 If the job is attached, two times the currently attached keyboard number. If detached, the one's complement of two times the currently detached keyboard number.

FIRQB+4 The error message text begins in this byte. The text is padded with octal 000 bytes to 28 characters, if necessary.

Errors:

No errors are returned with the ERRFQ subfunction of CALFIP.

Example:

See "Form/Example", above.

3.2.11 LOKFQ (Disk File/Wildcard Lookup)

Form:

```
MOVB    #LOKFQ,FIRQB+FQFUN
.
.
.
(Set up FIRQB to define file/wildcard)
.
.
.
CALFIP
```

Function:

The LOKFQ subfunction of CALFIP will do one of two things. It will look for a file on disk by name, returning directory information (date of creation, and so forth). Or, it will perform a "wildcard" file search. For example, with a filename and extension in the FIRQB of *.TXT it will search an account for a file with any filename and an extension of .TXT. By incrementing an index and re-executing LOKFQ, you can search through an entire directory for all such files.

Data Passed -- Disk Directory Lookup:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3 FQFUN	LOKFQ (=octal 22)		2	
5	(must equal 177777 octal)		4	
7	project number	programmer number	6 FQPPN	
11		file name (2 words in RAD50 format)	10 FQNAM1	
13			12	
15		file extension (1 word in RAD50 format)	14 FQEXT	
17			16	
21			20	
23			22	
25			24	
27			26	
31		device name (disk) (2 ASCII characters)	30 FQDEV	
33		<>0, unit no. real device unit number	32 FQDEVN	
35			34	
37			36	

FIRQB+FQFUN The function code LOKFQ (octal value = 22).

FIRQB+4 The word beginning at this location must be set to 177777 (octal) to indicate the disk directory lookup by filename option of LOKFQ.

FIRQB+FQPPN The project,programmer number (ppn) for the file to be looked up. A value of 0 for this word defaults to the ppn under which the calling program is running.

FIRQB+FQNAM1 The filename to be looked up; two words in RAD50 format.

FIRQB+FQEXT The extension of the file to be looked up; one word in RAD50 format.

FIRQB+FQDEV The device name (must be disk), as two ASCII characters. If both bytes are 0, the public disk structure (SY:) is used.

FIRQB+FQDEVN The disk device unit number is passed here in binary. A non-zero value in FQDEVN+1 indicates an explicit device unit number. A zero value in FQDEVN+1 indicates the system default.

Data Returned -- Disk Directory Lookup by Filename:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	/ / / / / / / / / / / / job number * 2	2	FQJOB
5	same as data passed (177777 octal)	4	
7	project number programmer number	6	FQPPN
11		10	FQNAM1
13	file name (2 words in RAD50 format)	12	
15	file extension (1 word in RAD50 format)	14	FQEXT
17	LSB of file length in 512-byte blocks	16	FQSIZ
21	MSB of file length protection code	20	
23	date of last access	22	
25	date of creation	24	
27	time of creation	26	
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no. real device unit number	32	FQDEVN
35	file cluster size	34	FQCLUS
37	file identification index	36	

FIRQB+FQJOB	The current job number times two.
FIRQB+4	The word at this location is the same as the data passed; in this case, 177777 (octal).
FIRQB+FQPPN	The project, programmer number of the file (same as data passed).
FIRQB+FQNAM1	The filename, two words in RAD50 format (same as data passed).
FIRQB+FQEXT	The file extension, one word in RAD50 format (same as data passed).
FIRQB+FQSIZ	This word contains the least significant bits of the file's size in 512.-byte blocks. This word is combined with the byte at FIRQB+21 to form a 24-bit field giving the file size.
FIRQB+20	The file's protection code, in binary, is returned in this byte.
FIRQB+21	This byte contains the most significant bits of the file size in 512.-byte blocks. It is combined with the word at FIRQB+FQSIZ to form a 24-bit field giving the file size.
FIRQB+22	This word contains the date the file was last accessed, in system internal format: (year - 1970 *1000.) + day-within-year
FIRQB+24	This word contains the date the file was created, in system internal format (see FIRQB+22).
FIRQB+26	This word contains the time the file was created, in system internal format: minutes until midnight, with 1440 = midnight.
FIRQB+FQDEV	The device name, as two ASCII characters. Always a specific name, even if 0 was passed here.
FIRQB+FQDEVN	The device unit number, in binary. A specific number is always returned here; FIRQB+FQDEVN+1 is always non-zero.
FIRQB+FQCLUS	The file cluster size is returned in this word.

FIRQB+36

The file identification index is returned in this word. This word can be used instead of the filename to open a file on disk with the OPNFQ subfunction of CALFIP. If you specify an explicit project, programmer number at FIRQB+FQPPN, a zero word at FIRQB+FQNAM1, and the file identification index at FIRQB+FQNAM1+2, the OPNFQ will open the desired file. (The file identification index is used by utilities which access software subroutines in the RMS libraries, for example.)

Errors:

- BADNAM The filename in bytes FIRQB+10 through FIRQB+13 is missing.
- NOSUCH The device specified at FIRQB+30 is not disk, or the file specified does not exist on the specified disk.

Example:

The following code looks for the file MATRIX.DAT on the system disk.

```
MOV        #LOKFQ, FIRQB+FQFUN        ;SET FUNCTION CODE
MOV        #177777, FIRQB+4            ;FILENAME LOOKUP
CLR        FIRQB+FQPPN                ;CALLER'S ACCOUNT
MOV        #^RMAT, FIRQB+FQNAM1      ;SET FILENAME
MOV        #^RRIX, FIRQB+FQNAM1+2    ;AND EXTENSION
MOV        #^RDAT, FIRQB+FQEEXT     ;TO "MATRIX.DAT"
CLR        FIRQB+FQDEV                ;SYSTEM DISK
CLR        FIRQB+FQDEVN              ;DEVICE
CALFIP
```

Data Passed -- Disk Wild Card Directory Lookup:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3 FQFUN	LOKFQ (= octal 22) //////////	2
5	index: n means search for n+1 occurrence	4
7	project number programmer number	6 FQPPN
11		10 FQNAM1
13	wild card file name specification (2 words in RAD50 format)	12
15	wild card file extension (1 word RAD50)	14 FQEEXT
17	//////////	16
21	//////////	20
23	//////////	22
25	//////////	24
27	//////////	26
31	device name (disk) (2 ASCII characters)	30 FQDEV
33	<>0,device no. real device unit number	32 FQDEVN
35	//////////	34
37	//////////	36

- FIRQB+FQFUN The function code LOKFQ (octal value = 22).
- FIRQB+4 An index number specifying the occurrence of the filename meeting the wildcard specifications. A value of 0 in this word causes the Monitor to search for the first filename in the directory which meets the specification. A value of 1, the second filename, and so forth.
- FIRQB+FQPPN The project,programmer number of the account whose directory of disk files is to be searched.
- FIRQB+FQNAM1 The wildcard filename specification, as two words in RAD50 format. An * character can replace the entire filename, or a ? character can replace any character in the filename. For example, a filename of FILE??

would cause the Monitor to search the directory for any filename beginning with the characters FILE. An * character indicates that the filename does not matter in the search.

FIRQB+FQEXT

The wildcard file extension specification, as one word in RAD50 format. An * character can replace the entire file extension, or a ? character can replace any character in the extension. For example, a file extension of BA? causes the Monitor to search the directory for any file extension beginning with the characters BA. An * character indicates that the file extension does not matter in the search.

FIRQB+FQDEV

The name of the device to be searched (must be disk). A value of 0 in this word indicates the public disk structure (SY:).

FIRQB+FQDEVN

This byte contains the device unit number in binary. A non-zero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates the system default.

Data Returned: Disk Wild Card Directory Lookup:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	job number * 2	2 FQJOB
5	(same as data passed)	4
7	project number programmer number	6 FQPPN
11	file name (2 words in RAD50 format)	10 FQNAM1
13		12
15	file extension (1 word in RAD50 format)	14 FQEXT
17	LSB of file length in 512-byte blocks	16 FQSIZ
21	MSB of file length protection code	20
23	date of last access (disk only)	22
25	date of creation	24
27	time of creation	26
31	(same as data passed)	30 FQDEV
33		32 FQDEVN
35	file cluster size (disk only)	34 FQCLUS
37	USTAT byte	36

- FIRQB+FQJOB The current job number times two.
- FIRQB+FQPPN The project, programmer number of the file (same as data passed).
- FIRQB+FQNAM1 The actual filename of a file meeting the wildcard specification in the data passed. Two words in RAD50 format.
- FIRQB+FQEXT The actual file extension of a file meeting the wildcard specification in the data passed. One word in RAD50 format.

FIRQB+FQSIZ	This word contains the least significant bits of the file's size in 512.-byte blocks. This word is contained with the byte at FIRQB+21 to form a 24-bit field giving the file's size.
FIRQB+20	The file's protection code, in binary, is returned in this byte.
FIRQB+21	This byte contains the most significant bits of the file's size in 512.-byte blocks. The byte is combined with the word at FIRQB+FQSIZ to give the file's size.
FIRQB+22	This word contains the date the file was last accessed, in system internal format: (year - 1970 *1000.) + day-within-year
FIRQB+24	This word contains the date the file was created, in system internal format (see FIRQB+22).
FIRQB+26	This word contains the time the file was created, in system internal format: minutes until midnight, with 1440 = midnight.
FIRQB+FQDEV	The device name, as two ASCII characters. Always a specific name, even if 0 was passed here.
FIRQB+FQDEVN	The device unit number, in binary. A specific number is always returned here; FIRQB+FQDEVN+1 is always non-zero.
FIRQB+FQCLUS	The file cluster size is returned in this word.
FIRQB+37	Internal flag information:
Bit	Meaning
002	Placed file.
004	Some job has write access now.
010	File is open in update mode.
020	File is contiguous; no extend available.
040	No delete or rename allowed.
100	MFD type entry.
200	File is marked for deletion.

Errors:

BADNAM	No file specification appears at FIRQB+FQNAM1.
NOSUCH	The device specified at FIRQB+FQDEV and FIRQB+FQDEVN is not a disk, or no match exists for the occurrence specified in the word at FIRQB+4.

Example:

The following code asks the Monitor to search the directory for account [2,130] for the first occurrence of a file specification beginning with the letter X.

```
MOVB    #LOKFQ,FIRQB+FQFUN      ;SET FUNCTION CODE
CLR     FIRQB+4                 ;LOOK FOR FIRST OCCURRENCE
MOV     #<2.*400+130.>,FIRQB+FQPPN ;SET PROJ.,PROG. NO.
MOV     #^RX??,FIRQB+FQNAM       ;SET X AS FIRST CHARACTER
MOV     #^R???,FIRQB+FQNAM+2    ;REMAINING CHARACTERS WILD
MOV     #^R*,FIRQB+FQEEXT      ;EXTENSION IS WILD
CLR     FIRQB+FQDEV            ;DEVICE=PUBLIC STRUCTURE
CLR     FIRQB+FQDEVN
CALFIP
```

3.2.12 OPNFQ (Open a File/Device on a Channel)

Form:

```
MOVB      #OPNFQ,FIRQB+FQFUN
.
.
.
(Set parameters appropriate to file or device)
.
.
.
CALFIP
```

Function:

The OPNFQ function has the same effect as an OPEN FOR INPUT statement in BASIC-PLUS; it opens a device or already-existing file on a channel. Parameters defining the device, filename and extension, protection code, and mode are passed to the Monitor in the FIRQB. If a filename is given in the FIRQB, a file-structured open for input is performed. If no filename is given, a non-file structured open for input is performed. The RSTS/E Programming Manual describes file- and non-file-structured open for input and the actions taken for the mode parameter (MODE modifier in BASIC-PLUS) for each device.

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//	0	
3	FQFUN	OPNFQ (=octal 2) //	2	
5		// channel no. * 2	4	FQFIL
7		project number programmer number	6	FQPPN
11			10	FQNAM1
13		file name (2 words in RAD50 format)	12	
15		file extension (1 word in RAD50 format)	14	FQEXT
17		//	16	
21		//	20	
23		mode	22	FQMODE
25		//	24	
27		//	26	
31		device name (2 ASCII characters)	30	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35		//	34	
37		//	36	

- FIRQB+FQFUN The function code OPNFQ (octal value = 2).
- FIRQB+FQFIL Channel number times 2; defines the channel upon which the file is to be opened.
- FIRQB+FQPPN The project,programmer number (ppn) of the file to be opened. The project number is in the high byte (FQPPN+1), and the programmer number in the low byte (FQPPN). A value of 0 defaults to the ppn under which the calling program is running. (Not used for non-file-structured open.)
- FIRQB+FQNAM1 The filename to be opened, as two words of RAD50 data. May also be specified as a word of 0 followed by the file identification index (see LOKFQ, Section 3.2.11.).

(Must be two words of zero for non-file-structured open.)

FIRQB+FQEXT The file extension, as one word of RAD50 data. (Must be zero for non-file-structured open.)

FIRQB+FQMODE The mode with which the file is to be opened; values and actions taken are as described for the MODE modifier in OPEN FOR INPUT statements for disk, as described in the RSTS/E Programming Manual.

FIRQB+FQDEV The device name is passed here as two ASCII characters. A value of zero indicates "SY", public disk.

FIRQB+FQDEVN The device unit number is passed here in binary. A non-zero value in the high byte of this word (FIRQB+FQDEVN+1) indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no unit number.

Data Returned:

FIRQB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1		0	
3	current job no. * 2	2	FQJOB
5 FQSIZM	MSB of file size channel no * 2	4	FQFIL
7	project number programmer number	6	FQPPN
11	file name (2 words of RAD50)	10	FQNAM1
13		12	
15	file extension (1 word of RAD50)	14	FQEXT
17	LSB (least significant bits of) file size	16	FQSIZ
21	reasonable buffer size for device	20	FQBUFL
23	(as passed)	22	FQMODE
25	device description	24	FQFLAG
27 FQPROT	protection code clustersize, mod 256	26	
31	device name (2 ASCII characters)	30	FQDEV
33	flag byte, device unit number	32	FQDEVN
35	file identification index	34	FQCLUS
37		36	

NOTE: For non-disk devices, the relevant information returned with the OPNFQ subfunction is in the two words at FIRQB+FQBUFL and FIRQB+FQFLAG. All other words are simply returned as passed.

FIRQB+FQJOB The current job number times two.

FIRQB+FQFIL Channel number times two; defines the channel on which the file is open.

FIRQB+FQSIZM For large disk files (>65,535. blocks), this byte contains the most significant bits of the file's size in 512.-byte blocks. This byte is combined with the word at FIRQB+FQSIZ to form a 24-bit field giving the file size.

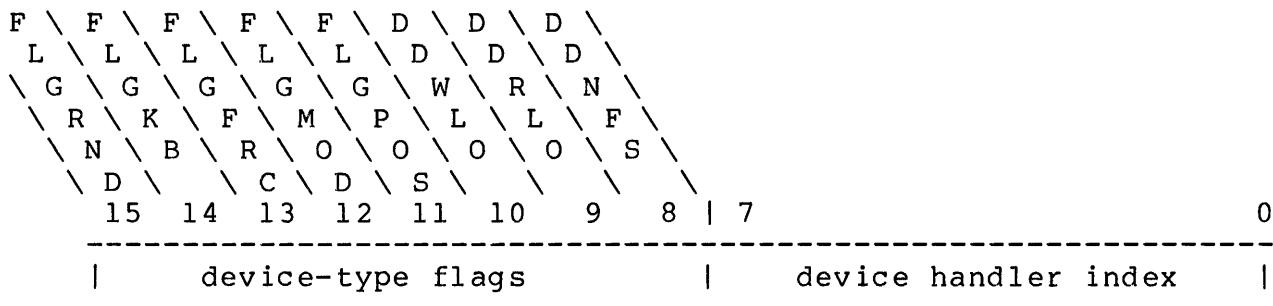
FIRQB+FQPPN The project,programmer number under which the file is open. An actual project,programmer number is returned here even if this word was passed as zero.

FIRQB+FQNAM1 The filename, as two words of RAD50 data.

FIRQB+FQEXT The file extension, as one word of RAD50 data.

FIRQB+FQBUFL Reasonable buffer size for this device, in bytes. If you are doing device-independent I/O (that is, if you do not wish to keep track of which device is being opened, and perform specific opens, reads, and writes, depending on the device), this value is the Monitor's "best guess" for a buffer size to use in subsequent .READs and .WRITEs on the opened channel. (See .READ and .WRITE, Sections 3.17 and 3.33.)

FIRQB+FQFLAG Description of the device just opened. The low byte contains the device's handler index. There is one unique handler index for all device types. The high byte contains a set of status flags to allow for device-independent I/O routines.



High Byte -- Device-Type Flags:

The bits in the high byte of the flags word are set to indicate the type of file or device just opened:

FLGRND = 1 The device or file is random-access.
= 0 The device or file is sequential.

FLGKB = 1 The file/device is a terminal-type file/device (or is generically a terminal).
= 0 The file/device is not a terminal-type file/device.

FLGFRC = 1 The file/device is byte-oriented. That is,
 .READs and .WRITES handle data in byte
 units.
 = 0 The file/device is block-oriented. .READs
 and .WRITES handle data in block units.

FLGMOD = 1 The file/device accepts modifiers in .READs
 and .WRITES (Sections 3.17 and 3.33,
 XRB+XRMOD).
 = 0 The file/device does not accept modifiers
 in .READs and .WRITES.

FLGPOS = 1 The file/device keeps track of its
 horizontal position, and expands such
 characters as TAB into whatever is
 appropriate for the file/device. You can
 determine the current horizontal position
 with the .POSTN directive.
 = 0 The file/device does not keep track of its
 horizontal position.

DDWLO = 1 The file/device has been write-locked (via
 the protection code in the open), or is
 generically a write-only device.
 = 0 The file/device is not write-locked.

DDRLO = 1 The file/device has been read-locked (via
 the protection code in the open), or is
 generically a read-only device.
 = 0 The file/device is not read-locked.

DDFNS = 1 The file/device is non-file structured (or
 is generically not a file-structured
 device).
 = 0 The file/device is file-structured.

Low Byte -- Device Handler Index:

Bits 0-7 of the flags word contain a handler index
 which indicates the generic kind of device.

Octal Value	Symbol	Meaning
0	DSKHND	All disks
2	TTYHND	All terminals
4	DTAHND	DECtape
6	LPTHND	All line printers
10	PTRHND	Paper tape reader
12	PTPHND	Paper tape punch
14	CDRHND	Card reader
16	MTAHND	Magtape
20	PKBHND	Pseudo-keyboards

22	RXDHND	Floppy disks
24	RJEHND	2780 remote job entry
26	NULHND	The null device
30	DMCHND	The DMC11 DDCMP interface
32	AUDHND	Auto-dial interface
34	PLTHND	All X-Y plotters
36	DT2HND	DECtape II
40	KMCHND	KMC11
42	IBMHND	IBM interconnect

FIRQB+FQPROT-1 The file clustersize, modulo 256 (decimal). That is, a file clustersize of 256. is indicated by a zero byte here.

FIRQB+FQPROT The protection code of the file.

FIRQB+FQDEV The device name, as two ASCII characters. (For disk, the actual device name is returned, even if a zero word was passed in this word.)

FIRQB+FQDEVN The device unit number. (For disk, the actual unit number is returned here, even if FIRQB+FQDEVN+1 was passed as 0.)

FIRQB+FQCLUS The file identification index of this file. This word is significant mainly in that it can be used in place of the filename in subsequent opens of the file on disk. You can open the file with the OPNFQ subfunction by using an explicit project,programmer number in FIRQB+FQPPN, a zero word in FIRQB+FQNAM, and the file identification index in FIRQB+FQNAM+2.

Errors:

NOTCLS The specified channel is already open. It must be closed before it can be opened again.

xxxxx All other possible errors are device-dependent. See Appendix A for a full list of errors.

Example:

The following MACRO code sets up the FIRQB for the OPNFQ function. A non-file-structured open of magtape unit 2 is done, on channel 3.

```
MAG:    .ASCII  /MT/
.
.
.
MOV     #OPNFQ,FIRQB+FQFUN      ;SET FUNCTION CODE
MOV     #3*2,FIRQB+FQFIL       ;SET CHANNEL = 3
MOV     MAG,FIRQB+FQDEV        ;SET DEVICE = MT
MOV     #377,FIRQB+FQDEVN+1   ;SET FLAG DEVICE NO. EXPLICIT
MOV     #2,FIRQB+FQDEVN       ;SET DEVICE NO. = 2
CALFIP
```

3.2.13 RENFQ (Rename a File)

Form:

```
MOVB      #RENFQ,FIRQB+FQFUN
.
.
.
(Define file to be renamed and new name in FIRQB)
.
.
.
CALFIP
```

Function:

The RENFQ function renames an existing file on disk or DECTape, and, if requested, deletes any existing file with the new name.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 RENFQ (=octal 10)	2
5		4	
7		6 FQPPN	
11		10 FQNAM1	
13		12	
15		14 FQEEXT	
17		16 FQSIZ	
21		20 FQNAM2	
23		22	
25		24	
27	FQPROT	file protection word=0 if no change	26
31		30 FQDEV	
33		32 FQDEVN	
35		34	
37		36	

FIRQB+FQFUN The function code RENFQ (octal value = 10).

FIRQB+FQPPN The project,programmer number (ppn) of the existing file to be renamed. The project number is in the high byte (FIRQB+FQPPN+1) and the programmer number is in the low byte (FIRQB+FQPPN). A value of 0 defaults to the ppn under which the calling program is running.

FIRQB+FQNAM1 The old (existing) name for the file, as two words of RAD50 data.

FIRQB+FQEEXT The old extension, as one word of RAD50 data.

FIRQB+FQSIZ	This word is set to -1 to indicate that any existing file on the specified device with the new name is to be deleted. If any other value is given here, and a file already exists with the new name, the RENFQ function will return an error.
FIRQB+FQNAM2	The new filename and extension are given here, as three words of RAD50 data. If these three words are all 0, the old file name will be used. That is, you can use RENFQ to change the protection code on a file by setting FQPROT, and leaving these three words 0.
FIRQB+FQPROT	The new protection code for the file, if any, is specified in this byte. To retain the old protection code, this entire word (FIRQB+FQPROT-1 and FIRQB+FQPROT) must be zero. If the word is non-zero, the high byte will be used as the new protection code.
FIRQB+FQDEV	The device name is passed here as two ASCII characters; it must be a disk or DECtape device. A value of 0 in this word indicates the public disk structure.
FIRQB+FQDEVN	The device unit number is passed here in binary. A non-zero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates the system default.

Data Returned:

Other than a possible error in byte 0 of the FIRQB), no data is returned by the RENFQ function.

Errors:

FIEEXIST	The new filename specified already exists.
NOSUCH	The old file specified cannot be found.

Example:

The following code renames the file OLDNAM.TXT to NEWNAM.TXT on the public disk structure under the caller's account. (Assume that the FIRQB has been filled with zeros.) FQSIZ is set to -1, so any existing file named NEWNAM.TXT will be deleted.

```

MOV     #RENFQ,FIRQB+FQFUN    ;SET FUNCTION CODE
MOV     #^ROLD,FIRQB+FQNAM1  ;SET OLD FILE NAME
MOV     #^RNAM,FIRQB+FQNAM1+2 ;AND EXTENSION
MOV     #^RTXT,FIRQB+FQEXT   ;TO "OLDNAM.TXT"
MOV     #-1,FIRQB+FQSIZ      ;DELETE EXISTING FILE
MOV     #^RNEW,FIRQB+FQNAM2  ;SET NEW FILE NAME
MOV     #^RNAM,FIRQB+FQNAM2+2 ;AND EXTENSION
MOV     #^RTXT,FIRQB+FQNAM2+4 ;TO "NEWNAM.TXT"
CALFIP

```

3.2.14 RSTFQ (Reset a Channel)

Form:

```
MOV B      #RSTFQ, FIRQB+FQFUN
.
.
.
(Define channel to be reset)
.
.
.
CALFIP
```

Function:

The RSTFQ function closes a channel (or all channels, or all channels except one) without performing any of the normal "cleanup" operations. For example, no trailer tape is written to paper tape punch, no form feed is given on the line printer, no trailer labels are written to magtape. This function is useful as a backup to a normal close operation. If a normal close fails, RSTFQ will close the channel regardless. Also, you can use RSTFQ to close a channel on which a tentative file is open if you don't want to make the file permanent. (Tentative files are described in the RSTS/E Programming Manual.) The RSTFQ directive functions the same as a CLOSE statement with a negative channel number in BASIC-PLUS.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	RSTFQ (= octal 20)
5		4	FQFIL
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQFUN The function code RSTFQ (octal value = 20).

FIRQB+FQFIL Three possibilities:

1. Reset one channel. FIRQB+FQFIL contains the channel number times two of the channel to be reset.
2. Reset all channels. FIRQB+FQFIL is set to zero.
3. Reset all but one channel. FIRQB+FQFUN is set to -(channel number times two); that is, the negative of two times the channel number to remain open.

Data Returned:

No data is returned with the RSTFQ function.

Errors:

No errors are possible with the RSTFQ function. If the channel or channels specified are not open, the call simply returns without error.

Example:

The following code resets all channels currently open for the job, except channel 2.

```
MOVB    #RSTFQ,FIRQB+FQFUN      ;SET FUNCTION CODE
MOVB    #-4,FIRQB+FQFIL        ;SET ALL CHANNELS BUT 2
CALFIP
```

Form:

```
MOVB    #LOKFQ,FIRQB+FQFUN
.
.
.
(Set up FIRQB for UUOFQ subfunction)
.
.
.
CALFIP
```

Function:

The UUOFQ subfunction of CALFIP performs the same operations as the .UUO Directive (Section 3.32). The same subfunctions are available, and a similar format is used for the data passed. The data returned is in the same format as for the .UUO Directive.

In general, the data passed for the UUOFQ subfunction of CALFIP is moved down one byte from the .UUO subfunction format. The value UUOFQ is stored in byte 3 in the FIRQB, the UU.xxx code is stored in byte 4 in the FIRQB, whatever is shown in byte 4 for .UUO is stored in byte 5 for UUOFQ, and so forth.

The UUOFQ subfunction of CALFIP is not recommended (although it does work!), because values which are normally set up as whole words must be stored as a high-byte in location n and a low-byte in location n+1.

For example:

.UUO form:	<hr style="border-top: 1px dashed black;"/> <table border="0"><tr><td style="width: 15px;"></td><td style="width: 15px; text-align: center;">y</td><td style="width: 15px; text-align: center;">x</td><td style="width: 15px; text-align: right;"> </td><td style="width: 15px; text-align: right;">n</td></tr></table> <hr/>		y	x		n	
	y	x		n			
UUOFQ form:	<hr style="border-top: 1px dashed black;"/> <table border="0"><tr><td style="width: 15px;"></td><td style="width: 15px; text-align: center;">x</td><td style="width: 15px; text-align: center;"> </td><td style="width: 15px; text-align: center;"> </td><td style="width: 15px; text-align: right;"> </td><td style="width: 15px; text-align: right;">n</td></tr></table> <hr/>		x				n
	x				n		
	<hr style="border-top: 1px dashed black;"/> <table border="0"><tr><td style="width: 15px;"></td><td style="width: 15px; text-align: center;"> </td><td style="width: 15px; text-align: center;"> </td><td style="width: 15px; text-align: center;">y</td><td style="width: 15px; text-align: right;"> </td><td style="width: 15px; text-align: right;">n+1</td></tr></table> <hr/>				y		n+1
			y		n+1		

3.3 .CCL -- Check String for CCL Command

Form:

.CCL

Function:

The .CCL directive asks the Monitor to check a string (defined in the XRB) to see if it is a valid Concise Command Language (CCL) command. If the string is a valid CCL command, the Monitor passes control to the appropriate run-time system for that CCL (as defined by the system manager), using the equivalent of a .RUN directive (Section 3.20). If the directive is successful, control does not return in-line.

The run-time system will be entered at the location specified in the P.RUN word in the pseudo-vector region (Section 2.5). Data will be passed to the run-time system in the job's CORCMN, XRB, FIRQB, and KEY areas in the low segment, and the file containing the program to be run as a result of the .CCL command will be open on channel 15. (The contents of these areas are described under the P.RUN description, since they are of interest to the run-time system being entered as a result of a .CCL, not to the caller.)

If the string is not a valid CCL command, the Monitor returns control to the caller (the run-time system or user job image which issued the .CCL directive) with no error. Control resumes with the instruction following the .CCL; since control would not be returned here otherwise, the program would do here whatever processing it deems necessary for an unsuccessful CCL.

As mentioned, the system manager defines acceptable CCL commands (RSTS/E System Manager's Guide). He defines the command, an abbreviation point, the name of the file which is to be executed when the command is given, and an entry point for the program. The CCL command itself may consist of from 1 to 9 characters. The valid character set is alphanumeric (upper/lowercase A-Z, 0-9), plus the @ character. The first character of any CCL command must be alphabetic or the @ character. The abbreviation point defines how many characters must be specified before the command will be accepted as valid. For example, if "DIRECTORY" were defined as a CCL command by the system manager, he could indicate three characters as the abbreviation point for the command. Then "DIR", "DIRE", "DIREC", and so forth, up to the full "DIRECTORY" would all be interpreted by the Monitor as correct CCL commands. (The Monitor will always fill in the full CCL command in CORCMN when it passes control to the appropriate run-time system).

When a .CCL directive is issued, then, the Monitor compares the indicated string with the commands defined by the system manager, as follows.

1. All characters are trimmed to 7-bit ASCII; that is, each byte will lose its high-order bit.
2. All null (ASCII code 000 octal) and delete (ASCII code 177 octal) characters are ignored and are never passed on to the run-time system in CORCMN.
3. Leading spaces (ASCII code 040 octal) and tabs (ASCII code 011 octal) are ignored and are never passed on to the run-time system in CORCMN.
4. Outside of the quote characters " (ASCII code 042 octal) and ' (ASCII code 047 octal):
 1. All tabs (ASCII code 011 octal) are changed to spaces (ASCII code 040 octal).
 2. All control characters (ASCII codes 001 through 037, octal, inclusive) are ignored and never passed in CORCMN.
 3. Adjacent spaces (ASCII code 040 octal) are merged into a single space.
 4. All lower-case alphabetics (ASCII codes 141 through 172, octal, inclusive) are changed into their upper-case equivalents (ASCII codes 101 through 132, octal, inclusive).
5. Inside of the quote characters " (ASCII code 042 octal) and ' (ASCII code 047 octal), all characters are kept as is, and passed on to the run-time system in CORCMN.

The Monitor will also analyze two switches itself, immediately following the CCL command text (for example, DIR/SI:n/DET). These switches will be passed on to the run-time system as status flags set in the XRB, as described in the P.RUN description in Section 2.5. These two switches may appear in either order, but must immediately follow the command. Both are optional switches. The "size" switch has the following format:

[space] /SI[Z[E]]:[+][#]n[.]

where n indicates the size, in K words, of the user job image that the program, when executed, will require. If the + sign is given, n indicates the additional amount of space, in K words, that the file will require over that indicated by the computed size or minimum size (see description of PF.CSZ bit in P.FLAGS word, Section 2.5). If the + sign is omitted, then n simply indicates the size, in K words, that the invoked file should run at. If the # sign is given, n is assumed to be octal. If the period is given, n is assumed to be decimal. If both are given, an error is returned, and if neither is given, n is assumed to be decimal. The value of n must be between 1 and the system-wide maximum for a user job image (see SWAP MAX, RSTS/E System Generation Manual).

The "detach" switch has the following format:

[space]/DET[A[C[H]]]

This switch indicates that the invoked program should be run "detached". In this state, channel 0 (the terminal associated with the job) is marked as detached while the program is running. This can be useful for non-interactive programs; it frees the terminal for other use, and prevents the user from interrupting the job by typing a CTRL/C.

Remember that the Monitor simply examines these switches, and passes the information on to the run-time system. The run-time system is responsible for doing the appropriate processing.

NOTE: This directive should not be used from a user job image running with the RT11 Run-Time System, since the lowest 1000 (octal) bytes are used by RT11 differently from other Run-Time Systems.

Data Passed:

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	length of proposed command string, bytes	0	XRLEN
3	length of proposed command string, bytes	2	XRBC
5	starting address of command string	4	XRLOC
7		6	
11		10	
13	(passed to new run-time system unaltered)	12	
15		14	

- XRB+XRLEN The length of the proposed CCL command string, in bytes.
- XRB+XRBC The length of the proposed CCL command string, in bytes, is also passed in this word.
- XRB+XRLOC The starting address of the proposed CCL command string.
- XRB+10 The remaining three words will be unaltered here if the string is indeed a CCL command and the Monitor takes over and does the equivalent of a .RUN directive. That is, the run-time system which is given control as a result of this command will find the same three words here that the caller left. These three words will also be unchanged if control returns back to the caller for any reason. See the .RUN Monitor call and the P.RUN entry point for details.

Data Returned:

No real arguments are returned to the calling program. (Data passed on to the run-time system if the call is successful is described in the P.RUN description in Section 2.5.) If the call is unsuccessful, the last three words of the XRB are unaltered, but the first four words will be random. Also, an error code will be returned in the first byte of the FIRQB.

Errors:

- (none) No error is returned if the command part of the string passed was not a valid CCL command. The contents of CORCMN have not been altered; the XRB (except for the last three words) has been altered.

BADCNT	The first three words of the XRB, which describe the CCL command string, are illegal.
LINERR	The indicated string is too long to be passed in CORCMN.
BADSWT	An illegal switch was given in the CCL command string.
BDNERR	An illegal number was the argument to one of the switches found in the CCL command string. For example, the "n" value in the /SIZE switch was greater than the system-wide maximum for a user job image (see SWAP MAX, RSTS/E System Generation Manual).
xxxxxx	Any other error returned results from the Monitor's execution of a .RUN directive for the program. (See the .RUN directive, Section 3.20.)

Example:

The following example asks the Monitor to check a 72-byte string beginning at location BUFFER to see if it is a CCL command.

```
BUFFER: .BLKB 72.  
.  
.  
.  
MOV #72.,XRB+XRLEN ;SET LENGTH  
MOV #72.,XRB+XRBC ;SET LENGTH AGAIN  
MOV #BUFFER,XRB+XRLOC ;SET STARTING ADDRESS  
.CCL
```

3.4 .CHAIN -- Execute Under Same RTS

Form:

.CHAIN

Function:

The .CHAIN directive is the same as the .RUN directive, except that it will return an error if the program to be run would cause a new run-time system to be entered. That is, if the call succeeds, the current run-time system is entered at the P.RUN entry point. Also, no change in the user job image size is done.

This call can be used to bypass the special protection afforded by the "compiled file" bit in the protection code. That is, this protection can be supplied by the run-time system without this bit being set in the protection code of the file. It re-enters the run-time system so that a user cannot take control of the file once it is open on channel 15.

Please see the .RUN directive, Section 3.20, for Data Passed, Data Returned, and Errors. For the example, substitute .CHAIN for .RUN.

3.5 .CLEAR -- Clear Keyword Bits

Form:

.CLEAR

Function:

The .CLEAR directive can be used to clear certain bits in the keyword (KEY) location in the user job image (Section 2.4). The bits to be cleared are passed to the Monitor in the XRB.

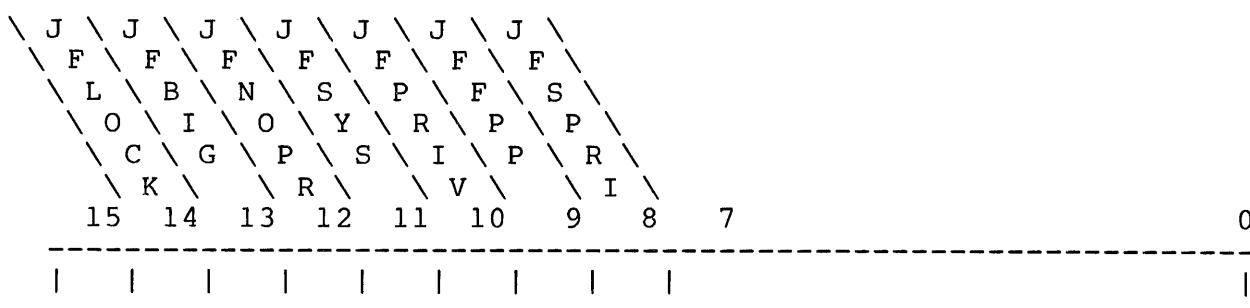
Data Passed:

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	bits to be cleared are set to 1 here	0	
3		2	
5		4	
7		6	
11		10	
13		12	
15		14	

XRB+0

The bits which are to be cleared are set to 1 here.



JFLOCK Clearable by any caller. Clearing JFLOCK indicates that the job wishes to be swapped. When JFLOCK is clear, the Monitor will swap the job (that is, the user job image) to and from disk as necessary.

JFBIG	Clearable by any caller. Clearing JFBIG drops the job's right to exceed its private memory maximum.
JFNOPR	Not clearable by any caller; masked off.
JFSYS	Clearable by any caller. If the job has temporary privileges, and this bit is cleared in a .CLEAR call, the temporary privileges are temporarily lost.
JFPRIV	This bit, when set, indicates permanent privilege. It cannot actually be cleared with the .CLEAR call. However, if a .CLEAR call is issued with this bit indicated for clearing, any temporary privilege that the job has or had is permanently lost.
JFFPP	Clearable by any caller. Clearing JFFPP indicates that this job no longer wishes the hardware floating point unit (if any) to be context switched along with the job's normal context information.
JFSPRI	Clearable by any caller. Clearing JFSPRI lowers the job's run priority by one-half step. (That is, it clears bit 2 of the system-controlled low-order three bits of the run priority. See UTILITY, RSTS/E System Manager's Guide.)

All other bits in the XRB are masked off; that is, the corresponding bits in KEY cannot be cleared by the job with the .CLEAR directive.

Data Returned:

No data is returned with the .CLEAR directive.

Errors:

No errors are possible with the .CLEAR directive.

Example:

```
MOV      #JFLOCK,XRB+0          ;SET JFLOCK FOR CLEAR
.CLEAR
```

3.6 .CORE -- Change Memory Size

Form:

.CORE

Function:

The .CORE directive asks the Monitor to change the amount of memory currently allocated for the user job image (low segment) for this job. The Monitor pre-allocates space for a user job image at the time a .RUN directive is issued, based on the file's size (PF.CSZ = 1 in the P.FLAG word of the pseudo-vector region) or equal to the P.MSIZ word in the pseudo-vector region. This initial size can be changed with the .CORE directive as many times as desired, as long as the requested size (1) falls within a maximum and minimum value, and (2) does not overlap any address windows created by the job for use with resident libraries (see .PLAS, Section 3.15). The Monitor first checks the size requested against maximum and minimum values, as follows.

```
-----      private
1Kword <= P.MSIZ <= |           | <= maximum <= P.SIZE <= system
                      | size requested | for job               maximum
                      | with .CORE   |-----
```

The Monitor determines the maximum allowable amount of space for a user job image as follows:

1. Set <max> (the maximum size that a job image can be) to the system-wide maximum. This maximum is set by the system manager at startup time (see SWAP MAX, RSTS/E System Generation Manual).
2. If the maximum user job image size imposed by the run-time system (P.SIZE in the pseudo-vectors) is less than the current <max>, set <max> to P.SIZE.
3. If the job's private memory maximum is less than <max> and if JFBIG in the job's keyword (KEY) is 0, then set <max> to the job's private memory maximum. The system manager can set a particular job's private memory maximum with the UTILITY system program. A private memory maximum can also be set with the UU.PRI subfunction of the .UUO directive, Section 3.32. A job's private memory maximum is initially defaulted to the system maximum.

Thus, the size requested with .CORE is checked against <max>, as determined by the three steps above. The size requested with .CORE is also checked against a minimum (the P.MSIZ word in the pseudo-vectors, which must be greater than or equal to 1K words). Any size between P.MSIZ and <max>, inclusive, is legal, and the Monitor will then check for overlap with created address windows.

There are two special cases:

1. If the size requested with .CORE is exactly equal to the run-time system's minimum size (P.MSIZ), that request is considered legal even if the requested size is greater than the job's private maximum.
2. If the size requested with .CORE is less than the job image's current size, and within the allowable bounds for the run-time system, but is still larger than the private maximum, that request is considered legal. This could happen if JFBIG was = 1, allowing the current size greater than the private maximum, and then JFBIG was cleared to 0. The Monitor would still allow the size greater than the private maximum.

If the .CORE requests a decrease in job image size, no further checks are made. If the .CORE requests an expansion, and the size is legal according to the tests described above, the Monitor then checks the base APRs of any address windows created by the job (see the CRAFQ subfunction of .PLAS, Section 3.15.2). If the size requested in the .CORE overlaps a created address window, the .CORE fails and returns an error.

If .CORE requests an expansion which is legal, but cannot be made "in place", that is, if there is not enough free memory available for the expansion, the job will be swapped out and swapped back in at the larger size. [This swap will occur even when JFLOCK = 1 in the keyword (KEY).]

When a user job image expands, the content of the newly added memory is random. A run-time system that executes this command should initialize the space with zeroes, as protection against a malicious user reading memory to look for passwords.

NOTE: This directive should not be used from a user job image running under the RT11 Run-Time System. Expanding memory size should be done through the RT-11 Emulator, using the appropriate RT-11 directive.

When running under the RSX Run-Time System or its derivatives, for example RMS11, the EXTK\$ directive should be used to extend the task size, so that subsequent GTSK\$ directives can return the task size correctly.

Data Passed:

XRB

Offset Octal Mnemonic	Offset Octal Mnemonic
1	////////// desired size, K words 0
3	////////// 2
5	////////// 4
7	////////// 6
11	////////// 10
13	////////// 12
15	////////// 14

XRB+0 This byte contains the desired size for the user job image, in K words.

Data Returned:

Other than a possible error in the first byte of the FIRQB, no data is returned with the .CORE directive.

Errors:

EDBMCE The requested user job image size is illegal. It is either too large or too small according to the rules described above, or it overlaps a mapped window.

Example:

```
MOV B #24., XRB+0 ;SET XRB TO INDICATE 24K WORDS
.CORE
```

3.7 .DATE -- Return Current Date and Time

Form:

.DATE

Function:

The .DATE directive returns the current date and time, the current program name (as installed by .NAME, Section 3.13), and the current run-time system name in the XRB.

Data Passed:

No data is passed with the .DATE directive.

Data Returned:

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		current date, in system internal format	0	
3		minutes until midnight	2	
5		ticks until second seconds until minute	4	
7			6	
11		program name (as installed with .NAME), RAD50 format	10	
13			12	
15		run-time system name, RAD50 format	14	

XRB+0 The current date, in system internal format. The Monitor calculates the date as:

$$(year - 1970) * 1000. + (\text{day-within-year})$$

where day-within-year is 1 for January 1, 2 for January 2, and so forth. (Every leap year the day-within-year value for March 1 and following is one higher than in other years.)

XRB+2 The number of minutes until midnight. A value of 1440 (decimal) is midnight; 720 is noon; 0 is never returned.

- XRB+4 This byte contains the number of seconds until the next minute. A value of 60 is xx:xx:00; 1 is xx:xx:59; 0 is never returned.
- XRB+5 This byte contains the number of "ticks" until the next second. A "tick" is either 1/60th or 1/50th of a second, depending on the clock in use and/or the line frequency. (Systems running with the KW11P clock at crystal speeds, rather than at line frequency, have a "tick" of 1/50th of a second. If the system is operating off a 60 Hz power line, a "tick" is 1/60th of a second.)
- XRB+6 The current program name (as installed by the most recent .NAME Monitor call) is returned here as two words in RAD50 format.
- XRB+12 The current run-time system name is returned here as two words in RAD50 format.

Errors:

No errors are possible with the .DATE Monitor call.

Example:

Since no data is passed to the Monitor, the call is simply:

.DATE

3.8 .ERLOG -- Log an Error from RTS

Form:

.ERLOG

Function:

The .ERLOG directive can be issued from the high segment (run-time system) only. It allows the run-time system to log an error into the RSTS/E error log file which can then be printed by the system manager (see the RSTS/E System Manager's Guide). For example, a run-time system might wish to place an entry into the RSTS/E error logging scheme on a hardware floating point unit exception that has an illegal error code -- the Monitor makes no such checks.

This directive can be issued only from the job's current run-time system (high segment). Since this call is not privileged, it is deemed wise not to allow users to fill up the system error log with unimportant errors. If .ERLOG is issued from the user job image (low segment), it is ignored.

Data Passed:

The .ERLOG directive will record in the system error log file the contents of the program counter (PC) and program status word (PS) at the time of the call, as well as the contents of the general registers (R0 through R5). These registers will then be displayed at the system manager's request. Hence, the registers should contain whatever information you wish to record when the .ERLOG is executed.

Data Returned:

No data is returned by .ERLOG.

Errors:

No error is possible with .ERLOG.

Example:

Assuming the general registers contain relevant information, the call is simply:

.ERLOG

3.9 .EXIT -- Exit to System Default RTS

Form:

```
.EXIT
```

Function:

The .EXIT directive returns control to the system default run-time system at the P.NEW entry point (Section 2.5). When a run-time system exits, it would normally pass control to the job's private default run-time system with the .RTS directive (Section 3.19). The .EXIT call can be used as a backup to return control to the system default run-time system should the .RTS fail, or for any other reason that it is desirable to enter the system default run-time system at the entry point specified by P.NEW. The .EXIT directive needs no arguments and never returns in-line to the caller.

Data Passed:

The .EXIT directive needs no arguments; however, the three words beginning at XRB+10 are passed unaltered to the system default run-time system. The Monitor also passes information to the system default run-time system when .EXIT is executed; for details, see the discussion of the P.NEW entry point in Section 2.5.

Data Returned:

No data is returned with .EXIT; control never returns in-line.

Errors:

No errors are possible with .EXIT.

Example:

Since no data is passed or returned with .EXIT, the call is simply:

```
.EXIT
```

NOTE: This (RSTS/E) directive should not be used from a user job image running under the RT11 Run-Time System. the proper way to terminate such a program is to exit to the RT-11 Emulator, which will return control to the job's default Run-Time System.

3.10 .FSS -- Check File Specification String

Form:

.FSS

Function:

The .FSS directive examines a string of characters presumed to be a file specification and, if possible, converts it to the internal RSTS/E file specification format; that is, the FIRQB format. The Monitor returns information to the XRB describing what it found in the string, and returns the converted file specification to the FIRQB. Thus, programs which manipulate files can use .FSS to translate a user-typed string to the FIRQB format.

The Monitor examines the string from left to right, and stops without error when it encounters:

1. The end of the string
2. An equal sign (=, ASCII code 075 octal)
3. A semicolon (;, ASCII code 073 octal)
4. A slash (/, ASCII code 057 octal) that is followed by anything other than the switches described below, which the Monitor translates to the FIRQB format.

1. /CL[USTERSIZE]:[-][#]n[.]

where n is the "cluster size" parameter used in opening files and devices. n may specify a value ranging from -32768. through 32767. (decimal), inclusive.

2. /MO[DE]:[#]n[.]

where n is the "mode" parameter used in opening files and devices. n may specify a value between 0 and 32767., inclusive.

3. /FI[LESIZE]:[#]n[.] or SI[ZE]:[#]n[.]

where n is the "filesize" parameter used in opening files and devices. The value of n defines the file's size in 512-byte blocks, and with the large file capability for disk, can range from 0 through (2**23)-1, or greater than 8 million blocks.

4. /PO[SITION]:n

where n is the "position" parameter word used in creating files (to position block 1 of the file at a device cluster). n may specify a value between 0 and 65,535., inclusive.

The brackets [] in the switches above enclose optional

characters. Where more than one character is enclosed in brackets, any or all of the enclosed characters can be omitted. For example, MO, MOD, and MODE would all be accepted and the following quantity translated to the mode location in the FIRQB. The value n is assumed to be decimal, unless the optional pound sign [#] appears, indicating that n is octal. The optional decimal point also indicates a decimal value.

5. A comma (, ASCII code 054 octal). An exception is the comma separating the project, programmer numbers in a ppm.

The Monitor will translate the following components in a file specification string:

device name A device name can be either a logical device name or a physical device name:

logical A logical device name is a string of alphanumeric characters terminated with a colon (:). Only the first six characters are examined; the remainder are ignored. The Monitor first checks a logical device name against the user's own logical device name assignments in USRLOG (or its equivalent, as defined in the XRB). If a definition is found, it returns the physical device name associated with that logical device name to the FIRQB. If the logical name is not found in the user-logical area, the Monitor then makes a similar search against its own internal table of system-wide user logicals. If the logical name is not there either, the Monitor simply returns the logical device name, and sets a flag in the XRB to indicate that it could make no association. For a logical device name beginning with an underscore, the Monitor does not attempt any translation to a physical device name.

physical A physical device name consists of two alphabetic characters optionally followed by digits and ended with a colon (:). The digits are translated as decimal and must have a value between 0 and 127 (decimal). Leading zeroes are allowed.

account or ppm A project, programmer number can be expressed either as a single special character or as two separate numbers enclosed in square brackets [] or parentheses () and separated by a comma.

The following special characters are translated as described.

- \$ The \$ is translated to the account assigned by the system manager to the system library. It is usually so assigned as [1,2].
 - ! The ! is translated to an account assigned by the system manager. It is usually so assigned as [1,3].
 - % The % is translated to an account assigned by the system manager. It is usually so assigned as [1,4].
 - & The & is translated to an account assigned by the system manager. It is usually so assigned as [1,5].
 - # The # is translated to the caller's "group library". It is always equivalent to [proj,0] where proj is the project number of the user issuing the .FSS directive.
 - @ The @ is translated to the caller's "assignable ppn", the USRPPN value described in Section 2.4. If USRPPN is set, its value is placed in the FIRQB at offset FQPPN. If USRPPN is 0, a string with an @ causes an error.
 - [n,m] This is the explicit construct for a project,programmer number. n specifies the project number, and m, the programmer number. n and m may specify any value from 0 through 254 (decimal) inclusive, except for [0,0]. If a pound sign (#) precedes either n or m, the string is assumed to specify an octal value. Either n or m, or both, can also be the asterisk (*). The * is converted to 255 (decimal) and placed in its corresponding FIRQB location. The * character indicates a "wildcard" specification.
 - (n,m) This is an alternate way to specify an explicit project, programmer number. The name rules for n and m apply as when they are enclosed by brackets.
- filename A filename may consist of alphanumeric characters and the question mark (?). It is the only field in the file specification with no explicit delimiter. Only the first six characters are examined; the rest are ignored. The asterisk character (*) is also an acceptable filename. It is parsed to two words of RAD50, where each RAD50 character is the "unused" code (35 (octal)). Each question mark is also converted to

this "unused" code. This indicates that the filename field is "wild". (The LOKFQ subfunction of CALFIP (Section 3.2.11) and UU.LOK subfunction of .UUO (Section 3.32.25) can be used to lookup wildcard files.)

extension A file extension may consist of alphanumeric characters and the question mark (?), preceded by a period (.). The asterisk (*) is also an acceptable extension. It is translated to one word of RAD50, where each RAD50 code is the "unused" code (035 octal). Each question mark in the extension is also converted to this "unused" code. This indicates that the extension field, or character in the extension field, is "wild".

If given, the extension must always follow the filename.

protection code A file protection code can consist of numeric digits enclosed by angle brackets. The general form of a protection code is <nnn>, where n may be numeric characters indicating a value from 0 through 255 (decimal). If the numeric characters are preceded by a pound sign (#), they are converted as specifying an octal value. If no file protection code is specified in the string, and a default value has been assigned in USRPRT (see Section 2.4), the default value will be placed in the FIRQB.

The components described above can appear in the string in any order, with the exception of the extension and ppn. The extension must follow the filename, if specified. Also, if the device name is a system or user logical device name which has an account (ppn) associated with it, the position of an explicit ppn in the file specification string is significant. If the order is:

device:ppn

then the explicit ppn overrides the ppn associated with the logical device name.

If the order is

[ppn]device:

then an illegal device name error is reported. (This error only occurs when a ppn is associated with the logical name.)

NOTE: This directive should not be used from a user job image running under the RT11 Run-Time System, since the user logical area is not in the standard location.

Data Passed:

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		length of the string, bytes	0	XRLEN
3		length of the string, bytes	2	XRBC
5		starting address of the string	4	XRLOC
7			6	
11		length of nonstandard user defaults	10	
13		starting address of nonstandard defaults	12	
15			14	

- XRB+XRLEN This word contains the length of the character string to be translated, in bytes.
- XRB+XRBC This word also contains the length of the character string to be translated, in bytes.
- XRB+XRLOC This word contains the starting address of the string to be translated.
- XRB+10 If the user logical information (USRPPN, USRPRT, and USRLOG) is in its standard location, this word is passed as 0. If some non-standard set of locations is being used, then the length of that information, in bytes, is specified here.
- XRB+12 If the word at XRB+10 is non-zero, then this word defines the starting location for the user logical information. (The order of the information is assumed to be the same as in its standard location. That is, the user logical ppn, user logical protection code, user logical device name table. The format is also expected to be the same -- see USRPPN, USRPRT, and USRLOG descriptions in Section 2.4 for details.)

Data Returned:

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	no. of untranslated characters in string	2 XRBC
5	address of first untranslated character	4 XRLOC
7		6
11	flag word 2	10
13	flag word 1	12
15	device description	14

XRB+XRBC This word contains a count of the untranslated characters in the string. If all characters were translated, the value of this word is 0.

XRB+XRLOC This word contains the address of the first untranslated byte of the string. (If XRB+XRBC=0, this word identifies the end of the string.)

XRB+10 Bit flags describing the translated string. (NOTE: This word is the same as "flag word 2" for the BASIC-PLUS filename string scan SYS call, as described in the RSTS/E Programming Manual.)

Bit	Octal Value	Meaning
0	1	A filename was found in the source string.
0	0	No filename was found (and bits 1 and 2 of this word are also 0).
1	2	The translated filename consisted of a single * character, and has been translated to two words at FIRQB+FQNAM1 consisting of the RAD50 representation of the string "?????".
	0	The translated filename was not an * character.
2	4	The filename contained at least one ? character.
	0	The filename did not contain any ? characters.

3	10	A dot (.) was found.
	0	No dot was found, implying that no extension was specified (and bits 4, 5, and 6 of this word are also 0).
4	20	An extension was found (that is, the field after the dot was not null).
	0	No extension was found (the field after the dot was null), and bits 5 and 6 of this word are also 0.
5	40	The extension was an * character and is returned in the word at FIRQB+FQEXT as the RAD50 representation of the string "???".
	0	The extension was not an * character.
6	100	The extension contained at least one ? character.
	0	The extension did not contain any ? characters.
7	200	A project,programmer number was found.
	0	No project,programmer number was found (and bits 8 and 9 of this word are also 0).
8	400	The project number was an * character (that is, the project,programmer number was of the form [*],n]. The byte at FIRQB+FQPPN+1 is returned as 255. (377 octal).
	0	The project number was not an * character.
9	1000	The programmer number was an * character (that is, the project,programmer number was of the form [n,*]. The byte at FIRQB+FQPPN is returned as 255. (377 octal)).
10	2000	A protection code was found.
	0	No protection code was found.
11	4000	No file protection code was found in the string, but there was a default output file protection code in location USRPRT. The default has been returned in the FIRQB.
	0	The user-assignable default protection code (at location USRPRT) was not used. Either the system default protection code (60), or the protection code given in the string is returned to the FIRQB.
12	10000	A colon (:), but not necessarily a device name, was found in the source string.
	0	No colon was found (no device was specified); bits 13, 14, and 15 of this word are also 0.

13	20000 0	A device name was found in the source string. No device name was found; bits 14 and 15 of this word are also 0.
14	40000 0	The device name in the string was a logical device name. The device name in the string was an actual device name; bit 15 of this word is also 0.
15	100000 0	The device name in the string was a logical device name, but could not be translated into a physical device name. Or, the device name contains an underscore. The logical device name specified has been returned at FIRQB+FQDEV as two words in RAD50 format. The device name specified, if any, was either an actual device name, or a logical device name to which a physical device has been assigned. The physical device name has been returned to the word at FIRQB+FQDEV as two ASCII characters, and the unit information has been returned appropriately at FIRQB+FQDEVN.
XRB+12		Remaining bit flags describing what was translated. (NOTE: this word is the same as "flag word 1" for the BASIC-PLUS file name string scan SYS call, described in the RSTS/E Programming Manual.)
Bit	Octal Value	Meaning
0	1 0	The /CLUSTERSIZE:n switch was specified. The /CLUSTERSIZE:n switch was not specified.
1	2 0	Either the /MODE:n or /RONLY switch was specified. Neither the /MODE:n nor the /RONLY switch was specified.
2	4 0	The /FILESIZE:n or /SIZE:n switch was specified. Neither the /FILESIZE:n nor the /SIZE:n switch was specified.
3	10 0	The /POSITION:n switch was specified. No /POSITION:n switch was specified.
4-7		(Not currently used.)
8	400	A filename was found in the source string (and is returned as two words in RAD50 format at FIRQB+FQNAM1).

	0	No filename was found in the source string.
9	1000	A dot (.) was found in the source string.
	0	No dot was found in the source string, implying that no extension was specified either.
10	2000	A project,programmer number was found in the source string.
	0	No project,programmer number was found.
11	4000	A left angle bracket (<) was found in the source string, implying that a protection code was found.
	0	No left angle bracket was found (no protection was specified).
12	10000	A colon (but not necessarily a device name) was found in the source string.
	0	No colon was found, implying that no device could have been specified.
13	20000	Device name was specified, and was a logical device name.
	0	Device name (if specified) was an actual device name. (If device name was not specified, this bit will also be 0.)
14		(Not currently used.)
15	100000	Source string contained wild card characters (either ? or * or both) in filename, extension, or project,programmer number fields. In addition, the device name specified, although a valid logical device name, does not correspond to any of the logical device assignments currently in effect or contains an underscore as the first character. Flag word 2 contains more specific information.
	0	None of the above.
XRB+14		This word contains the device description (the same information returned by the BASIC-PLUS STATUS variable, and returned at FIRQB+FQFLAG when a file or device is opened with the OPNFQ or CREFQ subfunctions of CALFIP. The device handler index is in the low byte and descriptive flags in the high byte.

FIRQB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1		0	
3	current job no. * 2	2	FQJOB
5 FQSIZM	MSB of file size	4	
7	project number programmer number	6	FQPPN
11		10	FQNAM1
13	file name (two words in RAD50 format)	12	
15	file extension (in RAD50 format)	14	FQEXT
17	LSB (least significant bits) of file size	16	FQSIZ
21		20	
23	mode parameter	22	FQMODE
25		24	
27 FQPROT	protection code =377, explicit p. cd.	26	FQPROT-1
31	device name	30	FQDEV
33	<>0,unit no.real device unit number	32	FQDEVN
35	cluster size parameter	34	FQCLUS
37	position parameter	36	FQNENT

NOTE: For each of the following field definitions that begin with the word "If", a corresponding statement applies: "If not, the field is left alone." That is, you can insert values in the FIRQB before executing the .FSS to serve as default values for fields when the .FSS returns no result.

FIRQB+FQJOB The current job number times two.

FIRQB+FQSIZM If a /FILESIZE:n or /SIZE:n switch was specified with n greater than 65,535, the most significant bits of the file size are contained in this byte.

FIRQB+FQPPN If a project, programmer number was part of the translated string, or if a logical name was found to be the same as a system or user logical name with an

associated project, programmer number, this word contains the binary value of that ppn. The project number is in the high byte; the programmer number in the low byte. Any value returned here by .FSS will have been verified by the Monitor as syntactically correct; that is, within the range for ppns on a RSTS/E system.

FIRQB+FQNAME1 If a filename was encountered, it will be translated to two words of RAD50, beginning at this location.

FIRQB+FQEXT If an extension was encountered, it will be translated to 1 word of RAD50, beginning at this location.

FIRQB+FQSIZ If a /FILESIZE:n or /SIZE:n switch was encountered, the value of n is translated to binary and the least significant bits of the value are placed in this word. (If the file size indicated was greater than 65,535, the most significant bits are placed in the byte at FIRQB+FQSIZM.)

FIRQB+FQMODE If a /MODE:n switch was encountered, the value specified will be translated to binary and returned in this word. Bit 15 will be set to indicate that a mode switch was translated, and to differentiate between a mode of 0 and no mode at all.

FIRQB+FQPROT-1 If a file protection code was encountered, a word will be returned here. The high byte (FIRQB+FQPROT) will be the binary value of the protection code, and the low byte (FIRQB+FQPROT-1) will be 377. Setting the low byte indicates that a protection code was specified and differentiates a protection code of 0 from no protection code at all.

FIRQB+FQDEV If a device name was specified, it will be returned here as two ASCII characters.

FIRQB+FQDEVN If a device name was specified, but no explicit unit number was specified, this word will be 0. If an explicit unit number was specified, then that unit number will be in the low byte and 377 (octal) will be in the high byte. Setting the high byte indicates an explicit device number and differentiates a device number of 0 from no device number at all.

NOTE: If a syntactically correct logical device name was encountered which could not be translated to a physical device name, then the logical device name is returned as two words of RAD50 starting at offset FQDEV. A status bit in the XRB is set to indicate that this was done.

FIRQB+FQCLUS If a /CLUSTERSIZE:n switch was encountered, the value of n will be returned here, in binary.

FIRQB+FQNENT If a /POSITION:n switch was encountered, the value of n will be returned here, in binary.

Errors:

BADCNT The first three words of the XRB are illegal.

BADNAM Some illegal specification occurred in the string.

BADSWT Some .FSS switch was encountered, but it was in an illegal format.

BDNERR The numeric argument to one of the .FSS switches was illegal.

Example:

The following code causes the Monitor to scan a string beginning at location BUFFER as a possible file name. BUFFER is defined as an 80-byte area, and is filled with zeros, to terminate the string scan if what the user typed did not fill the buffer.

```
BUFFER: .BLKWO #40.  
.  
.  
.  
(read string into BUFFER from terminal)  
.  
.  
.  
MOV    #80.,XRB+XRLEN      ;DEFINE LENGTH  
MOV    #80.,XRB+XRBC      ;DEFINE LENGTH AGAIN  
MOV    #BUFFER:,XRB+XRLOC ;START OF BUFFER  
.FSS  
(test for error; if none, try open)
```

3.11 .LOGS -- Check for Logical Devices

Form:

.LOGS

Function:

The .LOGS directive will (1) translate a system logical device name to a physical device name, (2) verify that a physical device name is valid, or (3) obtain generic information about a particular device.

You specify either a logical device name, a physical device designation (name and, if relevant, unit number), or both, in the XRB and the FIRQB. The Monitor compares the logical device name specified, if any, against its internal table of system-wide logical device names defined by the system manager. (This directive does not check user-specified logical names.) If it finds a match, the Monitor returns the device designation associated with the logical device name to the FIRQB. This physical device designation will consist of a name, unit number, and in some cases, a project, programmer number (ppn).

Next, the Monitor checks the physical device designation (either the one passed or the one returned by the Monitor in the logical-name translation) to be sure it is valid. If so, information describing the device is returned in the FIRQB.

Data Passed:

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	logical device name (two words in RAD50 format)	0	
3		2	
5		4	
7		6	
11		10	
13		12	
15		14	

XRB+0 The logical device name to be checked is passed as two words in RAD50 format beginning at this location. If only a physical device-name check or description is needed, then the first word of the XRB should be passed as 0. The physical device name is passed in the FIRQB.

FIRQB

Offset		Offset	
Octal	Mnemonic	Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	//////////	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	//////////	14	
17	//////////	16	
21	//////////	20	
23	//////////	22	
25	//////////	24	
27	//////////	26	
31	physical device name (ASCII format)	30	FQDEV
33	<> 0, real dev. no. device unit number	32	FQDEVN
35	//////////	34	
37	//////////	36	

FIRQB+FQDEV The physical device name, as two ASCII characters. A value of 0 (and at offset FIRQB+FQDEVN) indicates the public disk structure (SY:). If only a translation from a logical device name to a physical device name is desired, a value of -1 can be passed here. (-1 is guaranteed not to be a valid physical device name.)

FIRQB+FQDEVN The unit number of the physical device name is passed in this byte, in binary. The high byte (at FIRQB+FQDEVN+1) should be set to some non-zero value to indicate an explicit device number. If the physical device name is of the form "XY:" (that is, no unit number is specified), then the entire word at this offset should be set to 0 to indicate no explicit unit number.

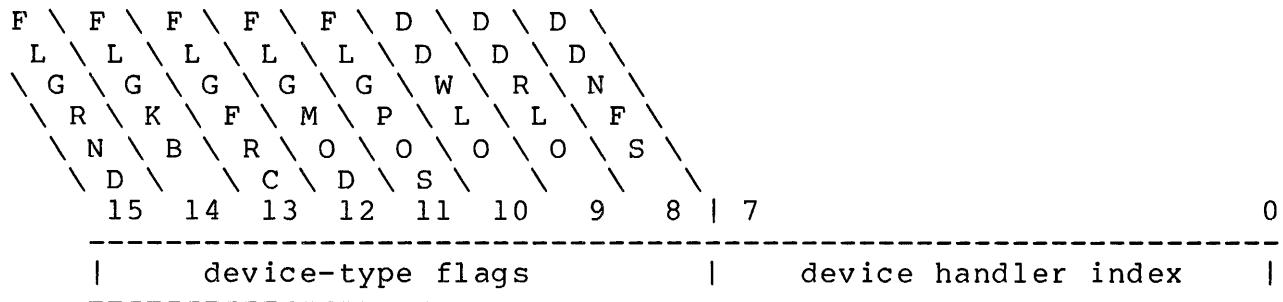
Data Returned:

XRB

Offset Octal Mnemonic	Offset Octal Mnemonic
1	0
3	2
5	flag for log. device:-1=no,-2=yes,0=N.A. 4
7	device description 6
11	reasonable buffer size for device 10
13	12
15	14

XRB+4 If the passed logical device name was translated successfully to a physical device name, this word is returned as -2; if not, as -1. If there was no logical device name specified, this word is returned as 0.

XRB+6 Description of the device. The low byte contains the device's handler index. The high byte contains a set of status flags.



FLGKB = 1 The file/device is a terminal-type file/device (or is generically a terminal).
= 0 The file/device is not a terminal-type file/device.

FLGFRC = 1 The file/device is byte-oriented. That is, .READS and .WRITES handle data in byte units.
= 0 The file/device is block-oriented. .READS and .WRITES handle data in block units.

FLGMOD = 1 The file/device accepts modifiers in .READS and .WRITES (Sections 3.17 and 3.33, XRB+XRMOD).
= 0 The file/device does not accept modifiers in .READS and .WRITES.

FLGPOS = 1 The file/device keeps track of its horizontal position, and translates such characters as TAB to whatever is appropriate for the file/device. (You can determine the current horizontal position of such a device with the .POSTN directive.)
= 0 The file/device does not keep track of its horizontal position.

DDWLO = 1 The file/device has been write-locked (via the protection code in the open), or is generically a write-only device.
= 0 The file/device is not write-locked.

DDRLO = 1 The file/device has been read-locked (via the protection code in the open), or is generically a read-only device.
= 0 The file/device is not read-locked.

DDFNS = 1 The file/device is non-file structured (or is generically not a file-structured device).
= 0 The file/device is file-structured.

Low Byte -- Device Handler Index:

Bits 0-7 of the flags word contain a handler index which indicates the generic kind of device. Current values for this byte are as follows.

Octal Value	Symbol	Meaning
0	DSKHND	All disks
2	TTYHND	All terminals
4	DTAHND	DECtape
6	LPTHND	All line printers
10	PTRHND	Paper tape reader
12	PTPHND	Paper tape punch
14	CDRHND	Card reader
16	MTAHND	Magtape
20	PKBHND	Pseudo-keyboards
22	RXDHND	Floppy disks
24	RJEHND	2780 remote job entry
26	NULHND	The null device
30	DMCHND	The DMC11 DDCMP interface
32	AUDHND	Auto-dial interface
34	PLTHND	All X-Y plotters
36	DT2HND	DECtape II
40	KMCHND	KMC11
42	IBMHND	IBM interconnect

XRB+10

If the physical device name is valid (either the one returned by the Monitor's translation of logical device name, or the one passed), this word contains the Monitor's "best guess" as a reasonable buffer size for this device. (See .READ and .WRITE, Sections 3.17 and 3.33).

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5		4	
7	project number programmer number	6	FQPPN
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no. real device unit number	32	FQDEVN
35		34	
37		36	

FIRQB+FQPPN If a logical device name was passed, and it was translated to a device designation with an associated project,programmer number, the project, programmer number is returned in this word. Otherwise, this word is the same as before the .LOGS call was executed.

FIRQB+FQDEV The physical device name, either the one returned when a successful translation of logical device name is made, or the one passed, if no logical device name was passed. The physical device name is returned as two ASCII characters.

FIRQB+FQDEVN The physical device unit number, either the one returned when a successful translation of logical device name is made, or the one passed, if no logical device name was passed. The low byte contains the unit number, in binary. The high byte (at FIRQB+FQDEVN+1) is either 0, to indicate no explicit device number, or non-zero, to indicate an explicit device number.

Errors:

NODEVC The physical device name (either the one passed or the one corresponding to the logical device name) is invalid.

Example:

The following code asks the Monitor to check the name "SYSDEV" to see if it is a defined system logical device name, and if so, to return the physical device name and characteristics to the XRB and FIRQB.

```
MOV      #^RSYS,XRB+0          ;SET XRB TO TRANSLATE LOGICAL  
MOV      #^RDEV,XRB+2          ;DEVICE NAME "SYSDEV"  
.LOGS
```

3.12 .MESAG -- Message Send/Receive

Form:

```
.  
. .  
(Load FIRQB and/or XRB for appropriate subfunction)  
. .  
. .  
.MESAG
```

Function:

The .MESAG directive provides access from a MACRO program to the RSTS/E local message send/receive services, and, if your system is a DECnet/E system, to DECnet/E network message send/receive services. We do not try to describe here the background information needed to use message send/receive services: see the RSTS/E DECnet/E Network Programming Manuals.

Also, in the descriptions which follow, the parameters passed and returned in the FIRQB and XRB are not described in detail unless they are different in concept from the BASIC-PLUS parameters described in the manual "Network Programming in BASIC-PLUS and BASIC-PLUS-2." In general, the definitions for the parameters in MACRO correspond to the parameters for BASIC-PLUS. The exception is the way that user data to be sent or received is defined. In BASIC-PLUS, you would define an I/O channel on the null device with a RECORDSIZE to define the length of a buffer. In MACRO, you simply define a buffer length, number of bytes to be sent, and starting location for the buffer containing the user data. The errors which can occur are the same as described in BASIC-PLUS. Use Appendix B of this manual to find the corresponding ERR.STB mnemonic or octal value for the decimal ERR values listed in the DECnet/E manual.

3.12.1 Declare Receiver Subfunction

Data Passed:

FIRQB		
Offset Octal	Mnemonic	Offset Octal
1		0
3		2
5	function code = 1	4
7	name	6
11	(receiver logical name in ASCII, zero fill to six bytes)	10
13		12
15	access lobjtyp (object type)	14
17	bmax (buffer maximum)	16
21	lmax (link maximum) mmax (message max.)	20
23		22
25	(must = 0)	24
27	(must = 0)	26
31	(must = 0)	30
33	(must = 0)	32
35	(must = 0)	34
37	(must = 0)	36

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no data is returned by the Declare Receiver subfunction of .MESAG.

3.12.2 Remove Receiver Subfunction

Data Passed:

FIRQB		Offset Octal Mnemonic	Offset Octal Mnemonic
1		0	
3		2	
5	(0 or job no. * 2) function code = 0	4	
7	(all remaining words must = 0)		6
37			36

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no data is returned by the Remove Receiver subfunction of .MESAG.

3.12.3 Send Local Data Message Subfunction

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	0 or job no. * 2 function code = -1	4	
7	name	6	
11	(receiver logical name in ASCII, zero fill to six bytes)	10	
13		12	
15	param	14	
17	optional user parameter string -- up to 20 bytes of additional user data can be specified here.	16	
21		20	
23	Zero fill to 20 bytes.	22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	length of output buffer, in bytes, 0-512	0	XRLEN
3	number of bytes to send, 0-bufferlength	2	XRBC
5	starting address of buffer	4	XRLOC
7	//////////	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	//////////	14	

XRB+XRLEN Length of the output buffer, in bytes. This value may range from zero through 512 (decimal).

XRB+XRBC The number of bytes to be sent. This value may range from zero through the size of the buffer, as specified at XRB+XRLEN.

XRB+XRLOC Starting address of the buffer.

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no data is returned by the Send Local Data Message subfunction of .MESAG.

3.12.4 Send Connect Initiate Subfunction

Data Passed:

	FIRQB	
Offset		Offset
Octal	Mnemonic	Octal
1	/ /	0
3	/ /	2
5	ula (user link addr) function code = -2	4
7	(must = 0)	6
11	(must = 0)	10
13	(must = 0)	12
15	(must = 0)	14
17	(must = 0) llmod (link mod.)	16
21	rmax (max bytes in rcvd. net data msg.)	20
23	(must = 0)	22
25	(must = 0)	24
27	(must = 0)	26
31	(must = 0)	30
33	(must = 0)	32
35	(must = 0)	34
37	(must = 0)	36

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	length of output buffer, bytes (120-136)	0	XRLEN
3	number of bytes to send (120-buffer size)	2	XRBC
5	starting address of buffer	4	XRLOC
7	//////////	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	//////////	14	

XRB+XRLEN Length of the output buffer, in bytes. This value must be greater than or equal to 120 (decimal).

XRB+XRBC The number of bytes to be sent. The value of this word is the sum of the size of the Connect Data Block (120 decimal), which must occupy the first 120 bytes of the buffer, and any optional data (up to 16 bytes) which is to be sent to the target program. Acceptable values range from 120 through 136 (decimal).

XRB+XRLOC Starting address of the buffer.

CONNECT DATA BLOCK

Decimal Position	Octal Offset			Octal Offset	Decimal Position
2	1	remote node name, in		0	1
4	3	ASCII (zero fill to 6 bytes)		2	3
6	5			4	5
8	7	object type format		6	7
10	11	descr. length (must = 0)		10	9
12	13	remote descriptor		12	11
14	15			14	13
16	17			16	15
18	21			20	17
20	23			22	19
22	25			24	21
24	27			26	23
26	31			30	25
28	33	object type format		32	27
30	35	descr. length (must = 0)		34	29
32	37	local descriptor		36	31
34	41			40	33
36	43			42	35
38	45			44	37
40	47			46	39
42	51			50	41
44	53			52	43
46	55			54	45
48	57	ID length (must = 0)		56	47

CONNECT DATA BLOCK (Cont.)

50	61	user identification	60	49
52	63		62	51
54	65		64	53
56	67		66	55
58	71		70	57
60	73		72	59
62	75		74	61
64	77		76	63
66	101	passwd length (must = 0)	100	65
68	103	password	102	67
70	105		104	69
72	107		106	71
74	111		110	73
76	113	acc't. length (must = 0)	112	75
78	115	accounting information	114	77
80	117		116	79
82	121		120	81
84	123		122	83
86	125		124	85
88	127		126	87
90	131		130	89
92	133		132	91

CONNECT DATA BLOCK (Cont.)

94	135		134	93
96	137		136	95
98	141		140	97
100	143		142	99
102	145		144	101
104	147		146	103
106	151	(reserved-- must = 0)	150	105
108	153		152	107
110	155		154	109
112	157		156	111
114	161		160	113
116	163		162	115
118	165		164	117
120	167		166	119

Data Returned:

Except for a possible error message in byte 0 of the FIRQB, no data is returned for the Send Connect Initiate subfunction of .MESAG.

3.12.5 Send Connect Confirm Subfunction

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	ula (user link add.)	4	function code = -3
7	lla (local link address)	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	(must = 0)	14	
17	(must = 0) llmod (link mods)	16	
21	rmax (receive max. for net data msgs)	20	
23	(must = 0)	22	
25	(must = 0)	24	
27	(must = 0)	26	
31	(must = 0)	30	
33	(must = 0)	32	
35	(must = 0)	34	
37	(must = 0)	36	

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	length of buffer, in bytes		0	XRLEN
3	number of bytes to be sent (0 - 16)		2	XRBC
5	starting address of buffer		4	XRLOC
7			6	
11	(must = 0)		10	
13	(must = 0)		12	
15			14	

XRB+XRLEN Length of output buffer, in bytes. This word can be zero if no user data is to accompany the message.

XRB+XRBC Number of bytes of optional data. This word may have any value between 0 and 16 (decimal).

XRB+XRLOC Starting address of the buffer.

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no data is returned by the Send Connect Confirm subfunction of .MESAG.

3.12.6 Send Connect Reject Subfunction

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	function code = -4	4	
7	lla (local link address)	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	(must = 0)	14	
17	reason code	16	
21	(must = 0)	20	
23	(must = 0)	22	
25	(must = 0)	24	
27	(must = 0)	26	
31	(must = 0)	30	
33	(must = 0)	32	
35	(must = 0)	34	
37	(must = 0)	36	

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	length of buffer, in bytes		0	XRLEN
3	number of bytes to be sent (0 - 16)		2	XRBC
5	starting address of buffer		4	XRLOC
7			6	
11	(must = 0)		10	
13	(must = 0)		12	
15			14	

XRB+XRLEN Length of output buffer, in bytes. This word can be zero if no user data is to accompany the message.

XRB+XRBC Number of bytes of optional data. This word may have any value between 0 and 16 (decimal).

XRB+XRLOC Starting address of the buffer.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the Send Connect Reject subfunction of .MESAG.

3.12.7 Send Network Data Message Subfunction

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	ula (user link add.) function code = -5	4	
7	(must = 0)	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	(must = 0) dmflgs (data flags)	14	
17	(must = 0)	16	
21	(must = 0)	20	
23	(must = 0)	22	
25	(must = 0)	24	
27	(must = 0)	26	
31	(must = 0)	30	
33	(must = 0)	32	
35	(must = 0)	34	
37	(must = 0)	36	

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	length of buffer, in bytes		0	XRLEN
3	number of bytes to be sent (0 - 16)		2	XRBC
5	starting address of buffer		4	XRLOC
7			6	
11	(must = 0)		10	
13	(must = 0)		12	
15			14	

- XRB+XRLEN Length of output buffer, in bytes. This word can be zero for a zero-length message.
- XRB+XRBC Number of bytes of user data to be sent to the target program. This value must be less than the transmit maximum for the particular logical link established during the connection sequence. The absolute maximum is 532 bytes.
- XRB+XRLOC Starting address of the buffer.

Data Returned:

The following data is returned if the rls bit (bit 7) is set in the dmflgs byte (FIRQB + 14) in the data passed.

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	ula (user link add.) (octal -7)	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	rstat (remote stat.) lstat (local stat.)	14	
17	lir (lcl int count) ldr(lcl data count)	16	
21	rir (rem int count) rdr(rem data count)	20	
23	dtc(data xmit q cnt dtm(data xmit q max)	22	
25	ltc(ls xmit q count) ltm(ls xmit q max)	24	
27	mcnt(message count) mmax(message max)	26	
31	////////// mula(msgs for link)	30	
33	//////////	32	
35	//////////	34	
37	//////////	36	

3.12.8 Send Interrupt Subfunction

Data Passed:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3		2
5	ula (user link add.) function code = -6	4
7	(must = 0)	6
11	(must = 0)	10
13	(must = 0)	12
15	(must = 0) inflgs	14
17	(must = 0)	16
21	(must = 0)	20
23	(must = 0)	22
25	(must = 0)	24
27	(must = 0)	26
31	(must = 0)	30
33	(must = 0)	32
35	(must = 0)	34
37	(must = 0)	36

XRB			
Offset		Offset	
Octal	Mnemonic	Octal	Mnemonic
1	length of buffer, in bytes	0	XRLEN
3	number of bytes to be sent (0 - 16)	2	XRBC
5	starting address of buffer	4	XRLOC
7		6	
11	(must = 0)	10	
13	(must = 0)	12	
15		14	

XRB+XRLEN Length of output buffer, in bytes. This word can be zero if no user data is to accompany the message.

XRB+XRBC Number of bytes of optional data. This word may have any value between 0 and 16 (decimal).

XRB+XRLOC Starting address of the buffer.

Data Returned:

The following data is returned if the rls bit (bit 7) is set in the inflags byte (FIRQB + 14) in the data passed.

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	lula (user link add.) (octal -7)	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	lrstat (remote stat.) lstat (local stat.)	14	
17	lir (lcl int count) ldr(lcl data count)	16	
21	rir (rem int count) rdr(rem data count)	20	
23	dtc(data xmit q cnt dtm(data xmit q max)	22	
25	ltc(ls xmit q count) ltm(ls xmit q max)	24	
27	mcnt(message count) mmax(message max)	26	
31	////////// mula(msgs for link)	30	
33	//////////	32	
35	//////////	34	
37	//////////	36	

3.12.9 Send Link Service Subfunction

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	ula (user link add.) function code = -7	4	
7	(must = 0)	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	(must = 0) lsflgs	14	
17	ircnt(int req count) drcnt(data req cnt)	16	
21	(must = 0)	20	
23	(must = 0)	22	
25	(must = 0)	24	
27	(must = 0)	26	
31	(must = 0)	30	
33	(must = 0)	32	
35	(must = 0)	34	
37	(must = 0)	36	

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	length of buffer, in bytes		0	XRLEN
3	number of bytes to be sent (0 - 16)		2	XRBC
5	starting address of buffer		4	XRLOC
7			6	
11	(must = 0)		10	
13	(must = 0)		12	
15			14	

- XRB+XRLEN Length of output buffer, in bytes. This word can be zero if no user data is to accompany the message.
- XRB+XRBC Number of bytes of optional data. This word may have any value between 0 and 16 (decimal).
- XRB+XRLOC Starting address of the buffer.

Data Returned:

The following data is returned if either the rls bit (bit 7) or the opr bit (bit 1) is set in the lsflgs byte (FIRQB + 14) in the data passed.

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	ula (user link add.) (octal -7)	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	rstat (remote stat.) lstat (local stat.)	14	
17	lir (lcl int count) ldr(lcl data count)	16	
21	rir (rem int count) rdr(rem data count)	20	
23	dtc(data xmit q cnt ltm(data xmit q max)	22	
25	ltc(ls xmit q count) ltm(ls xmit q max)	24	
27	mcnt(message count) mmax(message max)	26	
31	////////// mula(msgs for link)	30	
33	//////////	32	
35	//////////	34	
37	//////////	36	

3.12.10 Send Disconnect Subfunction

Data Passed:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	/ /	0
3	/ /	2
5	ula(user link add.) func. code = -10(oct)	4
7	/ /	6
11	/ /	10
13	/ /	12
15	/ /	14
17	/ /	16
23	(must = 0)	22
25	(must = 0)	24
27	(must = 0)	26
31	(must = 0)	30
33	(must = 0)	32
35	(must = 0)	34
37	(must = 0)	36

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	length of buffer, in bytes	0	XRLEN
3	number of bytes to be sent (0 - 16)	2	XRBC
5	starting address of buffer	4	XRLOC
7	//////////	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	//////////	14	

XRB+XRLEN Length of output buffer, in bytes. This word can be zero if no user data is to accompany the message.

XRB+XRBC Number of bytes of optional data. This word may have any value between 0 and 16 (decimal).

XRB+XRLOC Starting address of the buffer.

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no data is returned by the Send Disconnect subfunction of .MESAG.

3.12.11 Send Link Abort Subfunction

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	ula (user link add.) func.code= -11 (oct)	4	
7	(must = 0)	6	
11	(must = 0)	10	
13	(must = 0)	12	
15	(must = 0)	14	
17	(must = 0)	16	
21	(must = 0)	20	
23	(must = 0)	22	
25	(must = 0)	24	
27	(must = 0)	26	
31	(must = 0)	30	
33	(must = 0)	32	
35	(must = 0)	34	
37	(must = 0)	36	

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	length of buffer, in bytes		0	XRLEN
3	number of bytes to be sent (0 - 16)		2	XRBC
5	starting address of buffer		4	XRLOC
7	//////////		6	
11	(must = 0)		10	
13	(must = 0)		12	
15	//////////		14	

XRB+XRLEN Length of output buffer, in bytes. This word can be zero if no user data is to accompany the message.

XRB+XRBC Number of bytes of optional data. This word may have any value between 0 and 16 (decimal).

XRB+XRLOC Starting address of the buffer.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the Send Link Abort subfunction of .MESAG.

3.12.12 Receive Subfunction

Data Passed:

	FIRQB	
Offset		Offset
Octal	Mnemonic	Octal
1	//////////	0
3	//////////	2
5	rmod (rcv modifiers) function code = 2	4
7	qual (normally 0) sndr(sender select)	6
11	(must = 0)	10
13	(must = 0)	12
15	//////////	14
17	//////////	16
21	//////////	20
23	slptime (sleep time, in seconds)	22
25	(must = 0)	24
27	(must = 0)	26
31	(must = 0)	30
33	(must = 0)	32
35	(must = 0)	34
37	(must = 0)	36

Offset Octal Mnemonic		Offset Octal Mnemonic
1	0 or size of buffer, in bytes	0 XRLEN
3	(must = 0)	2 XRBC
5	starting address of buffer	4 XRLOC
7	//////////	6
11	(must = 0)	10
13	(must = 0)	12
15	//////////	14

XRB+XRLEN The size of the receive buffer, in bytes. This word may be zero if no user data is desired on the receive. The amount of data transferred from a pending message will never be greater than the buffer size.

XRB+XRBC This word must be passed as zero. The Monitor returns the actual number of bytes of user data transferred in this word location, as shown in the Data Returned sections following.

XRB+XRLOC The starting address of the buffer. The buffer, as defined by XRLOC for its start, and XRLOC+XRLEN-1 for its last byte, must lie wholly within either the job image (low segment) or the run-time system (high segment).

If the buffer is in the low segment, the address defined by the contents of XRB+XRLOC must be greater than 170 (octal) to avoid destroying the job-context information used in swapping the job (Section 2.4).

If the buffer is in the high segment, it must not fall within the pseudo-vector region. That is, it must not fall above the location P.OFF (Section 2.5). Also, the run-time system must currently be mapped read/write (see PF.RW bit description in P.FLAG word, Section 2.5). The run-time system must be read/write in this case, as the Monitor will be writing data to the buffer for the receive.

Data Returned:

The Receive call returns data to the FIRQB and XRB, identifying the type of message received, and user data, if any, to the buffer defined in the data passed.

The FIRQB and XRB formats for the various message types are shown on the following pages.

Data Returned (Local Data Message) :

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	job number * 2 -1 4		
7	project number programmer number 6		
11		10	
13	remainder (no. of bytes not transferred) 12		
15	param	14	
17	(data passed as parameters by the sender of this message. Padded with zeroes to 20 bytes)	16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	/ /	0
3	length (actual no. of bytes transferred)	2 XRBC
5	/ /	4
7	/ /	6
11	/ /	10
13	/ /	12
15	/ /	14

Data Returned (Connect Initiate):

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3			2	
5		-2	4	
7	llla (local link address)		6	
11			10	
13	remainder (no. bytes not transferred)		12	
15			14	
17	rlmod:0=no,1=sg,2=ms		16	
21	rmax(max bytes rcved in 1 net data msg)		20	
23	tmax(max bytes sent in 1 net data msg)		22	
25			24	
27			26	
31			30	
33			32	
35			34	
37			36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	length (no. bytes delivered this Receive)	2	
5	//////////	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	//////////	14	

Data Returned (Connect Confirm):

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	ula(user link addr.)	-3	4
7		6	
11		10	
13	remainder (no. of bytes not transferred)	12	
15		14	
17	rlmod:0=no,1=sg,2=ms	16	
21	rmax(max bytes recv'd in 1 net data msg)	20	
23	tmax(max bytes sent in 1 net data msg)	22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	length (no. bytes transferred this Rcv)	2	
5		4	
7		6	
11		10	
13		12	
15		14	

Data Returned (Connect Reject):

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	/ /	2	
5	ula(user link add.) -4 4		
7	/ /	6	
11	/ /	10	
13	remainder (no. bytes not transferred) 12		
15	/ /	14	
17	reason (for rejection) 16		
21	/ /	20	
23	/ /	22	
25	/ /	24	
27	/ /	26	
31	/ /	30	
33	/ /	32	
35	/ /	34	
37	/ /	36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ / / / / / / / / / / / / / / / / / / /	0	
3	length (actual no. of bytes transferred)	2	XRBC
5	/ / / / / / / / / / / / / / / / / /	4	
7	/ / / / / / / / / / / / / / / / / /	6	
11	/ / / / / / / / / / / / / / / / / /	10	
13	/ / / / / / / / / / / / / / / / / /	12	
15	/ / / / / / / / / / / / / / / / / /	14	

Data Returned (Network Data Message):

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	/ /	2	
5	ula(user link add.) -5 4		
7	/ /	6	
11	/ /	10	
13	remainder (no. bytes not transferred) 12		
15	/ / / / / / / / / / dmflgs 14		
17	/ /	16	
21	/ /	20	
23	/ /	22	
25	/ /	24	
27	/ /	26	
31	/ /	30	
33	/ /	32	
35	/ /	34	
37	/ /	36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	length (actual no. of bytes transferred)	2	XRBC
5	/ /	4	
7	/ /	6	
11	/ /	10	
13	/ /	12	
15	/ /	14	

Data Returned (Interrupt):

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3			2	
5	ula(user link add.) -6 4			
7			6	
11			10	
13	remainder (no. of bytes not transferred) 12			
15			14	
17			16	
21			20	
23			22	
25			24	
27			26	
31			30	
33			32	
35			34	
37			36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	length (actual no. of bytes transferred)	2	XRBC
5	/ / / / / / / / / / / / / / / / / / / /	4	
7	/ / / / / / / / / / / / / / / / / / / /	6	
11	/ / / / / / / / / / / / / / / / / / / /	10	
13	/ / / / / / / / / / / / / / / / / / / /	12	
15	/ / / / / / / / / / / / / / / / / / / /	14	

Data Returned (Link Service):

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	//////////		0	
3	//////////		2	
5	ula (user link add.)	-7	4	
7	//////////		6	
11	//////////		10	
13	//////////		12	
15	rstat (remote stat.) lstat (local stat.)	14		
17	lir (lcl int count) ldr(lcl data count)	16		
21	rir (rem int count) rdr(rem data count)	20		
23	dtc(data xmit q cnt dtm(data xmit q max)	22		
25	ltc(ls xmit q count) ltm(ls xmit q max)	24		
27	mcnt(message count) mmax(message max)	26		
31	////////// mula(msgs for link)	30		
33	//////////		32	
35	//////////		34	
37	//////////		36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	length (actual no. of bytes transferred)	2	XRBC
5	/ /	4	
7	/ /	6	
11	/ /	10	
13	/ /	12	
15	/ /	14	

Data Returned (Disconnect):

FIRQB

Offset			Offset
Octal	Mnemonic	Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	ula (user link add.) -10 (octal)	4	
7	//////////	6	
11	//////////	10	
13	remainder (no. of bytes not transferred)	12	
15	//////////	14	
17	//////////	16	
21	//////////	20	
23	//////////	22	
25	//////////	24	
27	//////////	26	
31	//////////	30	
33	//////////	32	
35	//////////	34	
37	//////////	36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ / / / / / / / / / / / / / / / / / / /	0	
3	length (actual no. of bytes transferred)	2	XRBC
5	/ / / / / / / / / / / / / / / / / /	4	
7	/ / / / / / / / / / / / / / / / / /	6	
11	/ / / / / / / / / / / / / / / / / /	10	
13	/ / / / / / / / / / / / / / / / / /	12	
15	/ / / / / / / / / / / / / / / / / /	14	

Data Returned (Link Abort) :

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	lula (user link add.) -11 (octal)	4	
7	lla (local link address)	6	
11		10	
13	remainder (no of bytes not transferred)	12	
15		14	
17	reason (for link abort)	16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	
37		36	

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ / / / / / / / / / / / / / / / / / / /	0	
3	length (actual no. of bytes transferred)	2	XRBC
5	/ / / / / / / / / / / / / / / / / /	4	
7	/ / / / / / / / / / / / / / / / / /	6	
11	/ / / / / / / / / / / / / / / / / /	10	
13	/ / / / / / / / / / / / / / / / / /	12	
15	/ / / / / / / / / / / / / / / / / /	14	

3.13 .NAME -- Install Program Name with Monitor

Form:

.NAME

Function:

The .NAME directive installs a program "name" with the Monitor. The Monitor enters the name in an internal table; otherwise, it makes no use of the program name. The RSTS/E SYSTAT routine uses the names in listing current information for jobs (under the "What" column) on the system, however. The BASIC-PLUS Run-Time System uses this directive when the user issues an OLD, NEW, or RENAME command, for example.

The program name is passed as two words of RAD50 data in the FIRQB. Note that the data is passed in the same location in the FIRQB where the filename exists at the P.RUN entry point into a run-time system. If you are coding or modifying a run-time system, one of the first things to do on entry at P.RUN is to install the program's name. Thus, the filename's position in the FIRQB at this point is handy.

Data Passed:

FIRQB		Offset	Offset
Octal	Mnemonic	Octal	Mnemonic
1		0	
3		2	
5		4	
7		6	
11	program name, in RAD50 format	10	FQNAM1
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQNAM1 The program name to be installed, as two words in RAD50 format.

Data Returned:

No data is returned with the .NAME directive.

Errors:

No errors are possible with the .NAME directive.

Example:

The following code installs the name "PROGRM" with the Monitor.

```
MOV      #^RPRO,FIRQB+FQNAM1    ;SET FIRQB TO DECLARE
MOV      #^RGRM,FIRQB+FQNAM+2  ;NAME OF "PROGRM"
.NAME
```

3.14 .PEEK -- Look at Monitor Memory

Form:

.PEEK

Function:

The .PEEK directive returns the contents of one word of the Monitor's memory (that is, the memory mapped by the kernel-mode APRs, as discussed in Section 2.1). .PEEK can only be executed by a privileged job.

Data Passed:

XRB

Offset Octal Mnemonic	XRB	Offset Octal Mnemonic
1	virtual address of desired monitor word	0
3	/ /	2
5	/ /	4
7	/ /	6
11	/ /	10
13	/ /	12
15	/ /	14

XRB+0

This word contains the (virtual) address of the data word in Monitor memory whose contents are to be returned. The value must be even, since word addresses on the PDP-11 are always even. Peeking at data in the I/O page (kernel APR 7, or 111 (binary) in bits 15, 14, and 13) can cause unpredictable system results and is not recommended. Also, a .PEEK to obtain data in APRs 5 or 6 returns random data.

Data Returned:

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	contents of the Monitor memory word	0
3	//////////	2
5	//////////	4
7	//////////	6
11	//////////	10
13	//////////	12
15	//////////	14

XRB+0 This word contains the contents of the requested Monitor memory location.

Errors:

- B.4 The address specified caused a trap to the kernel mode vector at 4 (UNIBUS timeout, odd address, and so forth).
- B.250 The address specified caused a memory management unit violation (trap to the kernel mode vector at 250).
- PRVIOL The job is not privileged; .PEEK can only be issued from a privileged job.

Example:

The following code obtains the contents of Monitor memory location 518. (the low byte of which is, incidentally, the current job number times 2).

```
MOV      #518.,XRB+0          ;SET ADDRESS TO 518
.PEEK
```

3.15 .PLAS -- Access Resident Library

The .PLAS (Program Logical Address Space) directive has six subfunctions that allow a MACRO program to access a resident library. Resident libraries must be so defined by the system manager with the ADD LIBRARY command (see RSTS/E System Manager's Guide). As noted in Section 2.3, the easiest way to do this is to link the resident library to your program using TKB -- the Task Builder which links modules assembled or compiled under the RSX Run-Time System or its derivatives. However, you can use .PLAS subfunctions to access resident libraries using octal addresses.

A summary of the .PLAS subfunctions is given below; they are described in alphabetical order in following subsections.

FQFUN Value (Octal)	Mnemonic	Action Performed
0	ATRFQ	Attach Resident Library. Attaches the job to a resident library; necessary before the job can map a window to the library. The Monitor will load the library from disk if necessary.
2	DTRFQ	Detach Resident Library. Detach the job from a resident library. If the library is not permanently resident, and no other jobs are attached to the library, the Monitor will remove the library from memory.
4	CRAFQ	Create Address Window. Define a range of virtual addresses to be a "window" for looking at all or some portion of a resident library. Optionally, map the window to all or some portion of a resident library. (The mapping can be done separately with MAPFQ). The CRAFQ subfunction reserves one or more APRs, so CRAFQ "takes space" in the job area even though the window may not actually be mapped.
6	ELAFQ	Eliminate Address Window. Releases the APRs used by a particular window.
10	MAPFQ	Map Window. Map an already-created address window of virtual addresses to actual memory locations in an attached resident library.
12	UMPFQ	Unmap Address Window. Releases a window of virtual addresses from a mapping to actual memory locations.

When a job exits, or a user logs out, the Monitor automatically detaches all libraries, and unmaps and eliminates all windows for the job.

3.15.1 ATRFQ (Attach Resident Library)

Form:

```
MOVB #ATRFQ,FIRQB+FQFUN
.
.
.
(set appropriate parameters)
.
.
.
.PLAS
```

Function:

The ATRFQ (attach resident library) subfunction of .PLAS declares intent to access a resident library. The type of access is specified in the call. If the calling job can access the library in that fashion*, the Monitor loads the library from disk (if necessary) and sets up its own internal tables which lay the groundwork for the job to map windows to the library. Note, however, that the resident library does not take up space in the job area (virtual memory) with an attach. APRs are assigned (virtual memory in the job area is taken) when a window is created (CRAFQ).

Up to five resident libraries can be attached to a job at any given time.

*The job's ability to access the resident library depends upon the protection assigned to the library by the system manager, when the library was installed. The default protection grants read access to all users, and denies write access to all users.

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3			2	
5		ATRFQ (= 0)	4	
7			6	
11			10	
13		resident library name	12	FQNAM1
15		(2 words in RAD50 format)	14	
17			16	
21			20	
23		access mode	22	FQMODE
25			24	
27			26	
31			30	
33			32	
35			34	
37			36	

FIRQB+4 The function code ATRFQ (octal value = 0).

FIRQB+FQNAM1 The name of the resident library to which the job is to be attached, as two words of RAD50 data. (Resident libraries are made known to the Monitor by the system manager with the ADD LIBRARY command of UTILITY (RSTS/E System Manager's Guide). With this command, the system manager defines a file (filnam.LIB) as a resident library. The Monitor regards "filnam" as the resident library's name.)

FIRQB+FQMODE The low-order two bits of this word define the way the job wishes to access the library:

Bit 0 = 1 Read-only access is desired.

Bit 1 = 1 Read/write access is desired.

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	current job no. * 2	2 FQJOB
5		4
7	resident library identification	6
11	size, in 32K-word blocks, of the library	10
13		12
15		14
17		16
21		20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

FIRQB+FQFUN The current job number times two.

FIRQB+6 This word is an identifier which must be used in subsequent calls to identify a resident library, rather than the resident library name. Thus, you will use this identifier to detach the job from the library (DTRFQ) and map and unmap windows to the library (MAPFQ and UMPFQ). (Keep it around!)

FIRQB+10 The size of the resident library, in 32-word blocks.

Possible Errors:

NOROOM The job has tried to attach to more than five resident libraries. At least one must be detached from the job (DTRFQ) before another can be attached.

NOSUCH The resident library specified in the data passed is not known to the Monitor. A resident library must be installed by the system manager before it can be used.

PRVIOL The attach did not succeed because the caller's privilege did not allow the access specified in the data passed. This could happen either because the access code specified in the data passed is not compatible with the possible access defined when the library was installed by the system manager, or because the protection code associated with the resident library file excludes access by the user.

Example:

The following code attaches the job to a resident library called DATBAS. The access desired is defined as read/write.

```
MOVB        #ATRFQ,FIRQB+4                         ;DEFINE FUNCTION CODE
MOV        #^RDAT,FIRQB+FQNAM1                     ;LIBRARY NAME IS
MOV        #^RBAS,FIRQB+FQNAM1+2                  ;DEFINED AS "DATBAS"
MOV        #2,FIRQB+FQMODE                          ;ACCESS=READ/WRITE
.PLAS
```

3.15.2 CRAFQ (Create Address Window)

Form:

```
MOVB    #CRAFQ,FIRQB+4
.
.
.
(set up parameters)
.
.
.
.PLAS
```

Function:

The CRAFQ subfunction of .PLAS can be used either to create a window (a range of virtual addresses) or to create a window of virtual addresses and map it to a range of actual addresses in an attached library. You define the range of addresses by (1) naming a base APR (which defines the starting address of the window), and (2) specifying the size of the window in 32-word blocks. Thus, a window always begins on a 4K-word boundary in virtual memory, and always takes at least 4K words. It may take more than 4K words, depending on the size of the window.

If the address range overlaps the user job image, the call fails with an error. The address range cannot overlap the run-time system (high segment). If the RSX Run-Time System has been installed as "disappearing" (see .RSX, Section 3.18), this is not a consideration, however. APR 7, normally used to map the RSX Run-Time System, can be used instead to map a window to a resident library. If the address range overlaps an existing window, the previously created window is eliminated.

The difference between (1) creating a window and (2) creating and mapping a window is best illustrated by example. By using create without map, you can define one window which can be mapped to a library or portion of a library, and then remapped to another portion of the same library or another library, as many times as desired, using the MAPFQ subfunction of .PLAS. For example, suppose your program takes up 24K words, and you want to access a 24K-word resident library of data values. You can use create without map to set up a 4K-word window in APR 6. You can then map the window (using MAPFQ) to the first 4K words of the library, process the data, map to the next 4K words of the library, and so forth.

If, on the other hand, you had a 4K program, and still wished to access a 24K-word library, you could use CRAFQ to create a 24K-word window and map it to the entire library in APRs 1 - 6.

A job can create a maximum of seven windows. A window takes at least one APR (it may take more, depending on the size you specify for the window). Thus, the maximum of seven assumes seven windows in APRs 1 through 7. APR 0 can never be used to create a window, since the user program takes at least this much space. As mentioned above, a window cannot overlap the user job image; thus, the size of the user job image determines the lowest base APR that can be used. If the program (user job image) is less than 4K words, APRs 1 and up (to the limit imposed by the run-time system boundary) can be used to create windows. If the user job image is between 4K words and 8K words, APRs 2 and up can be used to create windows, and so forth.

If a window is created which overlaps an already-existing window, the old window is eliminated. For example, if you created a 6K-word window using a base APR of 5, the window would use APRs 5 and 6. If you then created a 4K-word window using a base APR of 6, the entire old window would be eliminated. APR 5 would be free for other use; APR 6 is used for the new window.

Data Passed:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3		2
5	CRAFQ (=octal 4)	4
7	base APR (1-7)	6
11		10
13	size of window, in 32-word blocks	12
15	library identification (for map only)	14
17	offset, in 32-word blocks (for map only)	16
21	length, in 32-word blocks (for map only)	20
23	access flags	22
25		24
27		26
31		30
33		32
35		34
37		36

FIRQB+4 The function code CRAFQ (octal value = 4).

FIRQB+7 The base APR of the window, 1 - 7. Implicitly defines the starting virtual address of the window. This byte cannot be zero; nor can it name an APR already being used to map the user job image (see discussion above).

FIRQB+12 The desired size of the window, in 32-word blocks. For example, a value of 128. = 4K words.

FIRQB+14 The identifier of the resident library to which the window is to be mapped. (This is the value returned by the ATRFQ function of .PLAS at FIRQB+6.) This word is ignored for calls requesting a create without mapping (bit 7 at FIRQB+22 = 0).

FIRQB+16 The offset, in 32-word blocks, from the start of the library where the mapping is to begin. This word is ignored if no mapping is requested (bit 7 at FIRQB+22 = 0). A value of zero for this word indicates the window is to be mapped beginning at the first byte of the library. A value of 1 indicates the window is to be mapped beginning at the 33rd word of the library (starting address + 64), and so forth.

FIRQB+20 The length, in 32-word blocks, of the area to be mapped (ignored if bit 7 at FIRQB+22 = 0). This value cannot be greater than the size of the window specified at FIRQB+12. Also, this value, combined with the offset value at FIRQB+16, cannot indicate an address beyond the end of the library.

A value of 0 for this word defaults to either the size of the window (specified at FIRQB+14), or the space remaining in the library, whichever is smaller.

FIRQB+22 Two bits in this word define whether the window is to be mapped, and whether write access to the window is desired.

bit 1 = 1 Write access to the window is desired.
 = 0 No write access to the window is desired.

bit 7 = 1 The window is to be mapped.
 = 0 The window is not to be mapped.

(The decimal value to set bit 7 is 128.; the value to set bit 1 is 2. Thus, a value of 130. for this word requests mapping and write access.) A separate setting for write access in CRAFQ and in ATRFQ allows you to attach to a library read/write, and map a portion of the library read-only.

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	current job no. * 2	2 FQJOB
5		4
7	window ID	6
11	starting virtual address of new window	10
13		12
15		14
17		16
21	mapped length, in 32-word blocks	20
23	status flags	22
25		24
27		26
31		30
33		32
35		34
37		36

FIRQB+FQJOB The current job number times two.

FIRQB+6 This byte contains the window ID; it can be used in later MAPFQ calls to map the newly created window, and must be used in any ELAFQ calls to eliminate the newly created windows. (Save it if you'll need to use it.) The value returned may range from 1 - 7.

FIRQB+10 The starting virtual address of the new window.

FIRQB+20 Length, in 32-word blocks, actually mapped by the window.

FIRQB+22

Status flags.

bit 15 = 1 (Decimal value = 32768.) Window was created successfully.
= 0 Window was not created successfully.

bit 13 = 1 (Decimal value = 8192.) An existing window was eliminated because it overlapped the newly created window.
= 0 No existing windows were eliminated by this create.

bit 14 = 1 (Decimal value = 16384.) An existing window was unmapped because it overlapped the newly created mapping.
= 0 No existing windows were unmapped by this mapping.

Possible Errors:

BADFUO	Either the base APR - window length specified was invalid, or the offset - mapping length values specified were invalid. (For example, an offset indicating a starting address for the mapping which was beyond the end of the library.)
NOBUFS	Creating a window requires a small buffer; a small buffer is not currently available.
NOROOM	An attempt was made to create more than seven address windows.
NOSUCH	The library ID specified for mapping is not a library currently attached to the job.
PRVIOL	The create was unsuccessful because the user privileges do not allow the access desired. At this point, since the library has been attached successfully with some access defined, this error means that the access requested in the CRAFQ is not allowed by the access requested in the ATRFQ.

Example:

The following code creates a 4K-word address window and maps it to the beginning of a library whose ID (returned from a previous ATRFQ) has been stored at location LIBID.

```
MOVB    #CRAFQ,FIRQB+4          ;DEFINE FUNCTION CODE
MOV     #6,FIRQB+7              ;BASE APR = 6
MOV     #128.,FIRQB+12          ;WINDOW = 4K WORDS
MOV     LIBID,FIRQB+14          ;SET LIBRARY ID
CLR     FIRQB+16                ;OFFSET = 0
CLR     FIRQB+20                ;MAP 4K WORDS OR TO
                                ;END OF LIBRARY
MOV     #128.,FIRQB+22          ;MAP WINDOW, READ-ONLY
.PLAS
```

3.15.3 DTRFQ (Detach Resident Library)

Form:

```
MOVB      #DTRFQ,FIRQB+4
.
.
.
(define library to be detached)
.
.
.
.PLAS
```

Function:

The DTRFQ function of .PLAS detaches a previously attached resident library. Any windows mapped to the library by the calling job are unmapped. If no other jobs are currently attached to the library, and it is not a permanently resident library, the Monitor will remove the library from memory.

Data Passed:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3		2
5	DTRFQ (= octal 2)	4
7	library identification (ret. by ATRFQ)	6
11		10
13		12
15		14
17		16
21		20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

FIRQB+4 The function code DTRFQ (octal value = 2).

FIRQB+6 This word is the library identification returned at FIRQB+6 by the ATRFQ which attached the job to the library.

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	current job no. * 2	2	FQJOB
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23	status flag	22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQJOB The current job number times two.

FIRQB+22 Bit 14 of this word is set to 1 (decimal value = 16384.) if any windows were unmapped as a result of this detach.

Possible Errors:

NOSUCH The library ID specified at FIRQB+6 in the data passed does not identify any library currently attached to the job.

Example:

The following code detaches the library whose ID is stored at LIBID.

```
MOVB    #DTRFQ,FIRQB+4      ;SET FUNCTION CODE
MOV     LIBID,FIRQB+6        ;SET LIBRARY ID
.PLAS
```

3.15.4 ELAFQ (Eliminate Address Window)

Form:

```
MOVB    #ELAFQ,FIRQB+$  
.  
.  
.  
(set up parameters in FIRQB)  
.  
.  
.  
.PLAS
```

Function:

The ELAFQ subfunction of .PLAS eliminates an address window that was created by the job, unmapping the window if necessary. ELAFQ frees the APRs used by the window and makes them available for creating another window or for expanding the user job image size.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	ELAFQ (= octal 6)	4	
7	window ID	6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+4 The function code ELAFQ (octal value = 6).

FIRQB+6 The ID of the window to be eliminated (returned at FIRQB+6 with the CRAFQ which created the window).

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	/ / / / / / / / /	0
3	/ / / / current job no. * 2	2 FQJOB
5	/ / / / / / / / /	4
7	/ / / / / / / / /	6
11	/ / / / / / / / /	10
13	/ / / / / / / / /	12
15	/ / / / / / / / /	14
17	/ / / / / / / / /	16
21	/ / / / / / / / /	20
23	status flags	22
25	/ / / / / / / / /	24
27	/ / / / / / / / /	26
31	/ / / / / / / / /	30
33	/ / / / / / / / /	32
35	/ / / / / / / / /	34
37	/ / / / / / / / /	36

FIRQB+FQJOB The current job number times two.

FIRQB+22 Two bits in this word indicate the status of the window.

Bit 13 = 1 (Decimal value = 8192.) The window was successfully eliminated.
 = 0 The window was not eliminated.

Bit 14 = 1 (Decimal value = 16384.) The address window was mapped to a resident library and has been unmapped.
 = 0 The address window was not mapped; no unmapping was done.

Possible Errors:

- BADFUO An invalid window ID was given in the data passed at FIRQB+6 (outside the range 1-7).
- NOSUCH The window ID specified at FIRQB+6 is in the range 1-7, but matches no currently created window for this job.

Example:

The following code eliminates an address window whose ID is stored in location WINID.

```
MOVB    #ELAFQ,FIRQB+4      ;SET FUNCTION CODE  
MOVB    WINID,FIRQB+6       ;SET WINDOW ID  
.PLAS
```

3.15.5 MAPFQ (Map Address Window)

Form:

```
MOV B    #MAPFQ, FIRQB+4  
.  
.  
.  
(set up parameters)  
.  
.  
.  
.PLAS
```

Function:

The MAPFQ subfunction of .PLAS maps a previously created address window to an attached resident library. In other words, the MAPFQ relates the virtual address range defined by a CRAFQ to actual locations in memory within a resident library which has been attached to the job by an ATRFQ.

If the window specified is already mapped, it is unmapped from its previous actual memory locations, and remapped to the new area. A job may map a maximum of seven address windows at any given time.

Data Passed:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3		2
5	MAPFQ (= octal 10)	4
7	window ID	6
11		10
13		12
15	resident library ID	14
17	offset, in 32-word blocks	16
21	length, in 32-word blocks, to be mapped	20
23	desired access mode	22
25		24
27		26
31		30
33		32
35		34
37		36

FIRQB+4 The function code MAPFQ (octal value = 10).

FIRQB+6 The ID of the window to be mapped (returned at FIRQB+6 by the CRAFQ subfunction call that created the window).

FIRQB+14 The ID of the resident library to which the window is to be mapped (returned as a full word at FIRQB+6 by the ATRFQ subfunction call that attached the job to the resident library).

FIRQB+16 The offset, in 32-word blocks, from the start of the library where the mapping is to begin. A value of zero for this word indicates the window is to be mapped beginning at the first byte of the library. A value of 1 indicates that the window is to be mapped beginning at the 33rd word of the library (starting address + 64), and so forth.

FIRQB+20

The length, in 32-word blocks, of the area to be mapped. This value cannot be greater than the size of the window (specified at FIRQB+12 in the CRAFQ which created the window). Also, this value, combined with the offset value at FIRQB+16, cannot indicate an address beyond the end of the library.

A value of 0 for this word defaults to either the size of the window or the space remaining in the library, whichever is smaller.

FIRQB+22

Bit 1 of this word specifies whether the window is to be mapped read/write or read-only.

bit 1 = 1 (Decimal value = 2.) Read/write access.
= 0 Read-only access.

A separate setting for access in MAPFQ and in ATRFQ allows you to attach to a library read/write, and map a portion of the library read-only. You cannot, however, attach to a library read-only and then map to the library read/write.

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3	////////// current job no. * 2	2 FQJOB
5	//////////	4
7	//////////	6
11	//////////	10
13	//////////	12
15	//////////	14
17	//////////	16
21	length, in 32-word blocks, actually mapped	20
23	status flag	22
25	//////////	24
27	//////////	26
31	//////////	30
33	//////////	32
35	//////////	34
37	//////////	36

FIRQB+FQJOB The current job number times two.

FIRQB+20 Length, in 32-word blocks, actually mapped by the call.

FIRQB+22 One bit of this word is set as a status flag.

Bit 14 = 1 (Decimal value = 16384.) The window specified was already mapped; the window was unmapped before this mapping was done.
 = 0 The window specified had no previous mapping; no unmapping was done for this call.

Possible Errors:

BADFUO	The offset and length specified are inconsistent; either the mapping attempted to go beyond the end of the library, or the length is greater than the created window.
NOSUCH	Either the resident library ID or the address window ID is incorrect. (The job is not currently attached to the specified resident library, or no address window has been created with the specified window ID.)
PRVIOL	The mapping was unsuccessful because user privileges did not allow the access desired.

Example:

The following code maps a window whose ID is stored at WINID to the attached library whose ID is stored at LIBID. The offset of 256. indicates that the mapping is to begin 8K words from the start of the library. Access to the library is read-only.

MOVB	#MAPFQ,FIRQB+4	;SET FUNCTION CODE
MOVB	WINID,FIRQB+6	;SET WINDOW ID
MOV	LIBID,FIRQB+14	;SET LIBRARY ID
MOV	#256.,FIRQB+16	;OFFSET 8K WORDS
CLR	FIRQB+20	;MAP WINDOW SIZE OR ;TO END OF LIBRARY
CLR	FIRQB+22	;READ-ONLY ACCESS
.PLAS		

3.15.6 UMPFQ (Unmap Address Window)

Form:

```
MOV B    #UMPFQ, FIRQB+4
.
.
.
(set up parameters)
.
.
.
.PLAS
```

Function:

The UMPFQ subfunction of .PLAS unmaps a specified address window from a resident library. (Note that a MAPFQ on an already-mapped window will unmap the existing windows.)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5	UMPFQ (= octal 12)	4	
7	window ID	6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+4 The function code UMPFQ (octal value = 12).

FIRQB+6 The ID of the window to be unmapped (returned at FIRQB+6 by the CRAFQ that created the window).

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	/ / / / / / / / / / / current job no. * 2	2	FQJOB
5	/ /	4	
7	/ /	6	
11	/ /	10	
13	/ /	12	
15	/ /	14	
17	/ /	16	
21	/ /	20	
23	status flag	22	
25	/ /	24	
27	/ /	26	
31	/ /	30	
33	/ /	32	
35	/ /	34	
37	/ /	36	

FIRQB+FQJOB The current job number times two.

FIRQB+22 Bit 13 of this word is set (decimal value = 8192.) if the unmapping was successful.

Possible Errors:

BADFUO The window ID specified is invalid (not in the range 1 - 7).

NOSUCH The window ID specified is in the range 1 - 7, but no such window is currently created for the job.

Example:

The following code unmaps the window whose ID is stored at WINID.

```
MOV B    #UMPFQ,FIRQB+4          ;SET FUNCTION CODE  
MOV B    WINID,FIRQB+6  
.PLAS
```

3.16 .POSTN -- Return Current Horizontal Position

Form:

. POSTN

Function:

The .POSTN directive returns the maximum line width and the current horizontal position of devices for which this information is relevant (line printers and terminals). Data is passed in the XRB defining the channel where the device is currently opened. The information is returned in the XRB.

XRB

Data Passed:

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ / / / / / / / / / / / / / / / /	0	
3	/ / / / / / / / / / / / / / / / /	2	
5	/ / / / / / / / / / / / / / / / /	4	
7	/ / / /channel no. * 2	6	XRCI
11	/ / / / / / / / / / / / / / / /	10	
13	/ / / / / / / / / / / / / / / /	12	
15	/ / / / / / / / / / / / / / /	14	

XRB+XRCI

Channel number times two; defines the channel on which the file or device is open.

Data Returned:

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	maximum line length	0	
3	current position	2	
5		4	
7		6	
11		10	
13		12	
15		14	

XRB+0 The file/device's maximum line length plus one. A value of 81 decimal would indicate a maximum line length of 80 bytes (characters), for example.

XRB+2 The file/device's current horizontal position is returned here. The value may range from 0 (leftmost character) to the value returned at XRB+0 minus one (rightmost character). If the device does not keep track of its own horizontal position, then this value will be 0. The FLGPOS status bit returned in the FIRQB when the file/device was opened indicates whether the value returned here is meaningful.

Errors:

NOTOPN No file or device is open on the specified channel.

Example:

The following code requests the current horizontal position of the device open on channel 4.

```
MOV B #4*2,XRB+XRCT ;SET CHANNEL TO 4
.POSTN
```

3.17 .READ -- Read Data from File or Device

Form:

.READ

Function:

The .READ directive reads data from a file or device previously opened on a channel. The amount of data read depends on the device and the size of the buffer area, as defined in the XRB. The number of bytes transferred is always less than or equal to the buffer size. The actual number of bytes read is returned in the XRB when the directive is complete. Specific details for each device are given in Table 3-2. The "best guess" buffer sizes (returned by the Monitor at FIRQB+FQBUFL When the device was opened) are also shown in Table 3-2, for comparison.

Table 3-2
Data Input With .READ

Device	Block size, bytes	"Best guess*", bytes	.READ's Intent to deliver	What Happens When	
				buffer size >= Intent	buffer size < Intent
Byte-Oriented Devices (FLGFRC=1, FLGRND=0)					
Keyboard (Terminal)	N/A	128.	1 line**	1 line**	Fill buffer; next .READ reads next part of line
Pseudo-Keyboard	N/A	128.	Full buffer	N/A	N/A
Paper Tape Reader	N/A	128.	Full buffer	N/A	N/A
Card Reader	N/A	160.	1 card	1 card	Fill buffer; next .READ reads next part of card
Block-Sequential Devices (FLGFRC=0, FLGRND=1)					
Magtape	18. to 30,000.	512.	1 block	1 block	Fill buffer; next .READ starts with new block (Error returned)
DECtape (file-structured)	510.	510.	1 block	1 block	Fill buffer; next .READ starts with new block (No error returned)
Block-Random Devices (FLGFRC=0, FLGRND=0)					
Disk	512.	512.	Full buffer	Fills buffer; next .READ starts with new block.***	
Floppy Disk	512. (block mode) or 128. (RX01 or RX02 single-density sector mode) or 256. (RX02 double-density sector mode)	512.	Full buffer	Fills buffer; next .READ starts with new block or sector.***	
DECtape (non-file-structured)	512.	512.	1 block	1 block	Fill buffer; next .READ starts with new block (No error returned)

* Returned at FIRQB+FQBUFL when file or device was opened.

** A line is any number of characters terminated by RETURN, LINE FEED, ESCAPE, FORM FEED, CTRL/Z, CTRL/D, CTRL/C, or a user-set private delimiter.

*** If the buffer size is not a multiple of 512. bytes, the remainder of the last block will not be input. The next .READ will start with a new block; either the next sequential block (XRB+XRBLK=0), or the nth block (XRB+XRBLK=n). No error is returned.

Data Passed:

XRB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1	size of the buffer in bytes (must not =0)	0	XRLEN
3	(must = 0)	2	XRBC
5	starting address of buffer	4	XRLOC
7	XRBLKM MSB of block number channel no. * 2	6	XRCI
11	LSB of block number to begin (0 = next)	10	XRBLK
13	wait time for terminal input	12	XRTIME
15	device-dependent modifier	14	XRMOD

XRB+XRLEN The length of the input buffer, in bytes. This word must be non-zero. The amount of data read depends on both the device and the buffer size. The amount of data will never be more than the buffer size, however. Details on .READS for specific devices are given in Table 3-2.

XRB+XRBC This word must be passed as zero. The Monitor returns the actual number of bytes transferred in this word location, as described in the "Data Returned" section, below.

XRB+XRLOC The starting address of the buffer. The buffer, as defined by XRLOC for its start, and XRLOC+XRLEN-1 for its last byte, must lie wholly within either the job image (low segment), or the run-time system's address space (high segment).

If the buffer is in the low segment, the address defined by the contents of XRB +XRLOC must be greater than 170 (octal) to avoid destroying job-context information used in swapping the job (Section 2.4).

If the buffer is in the high segment, it must not fall within the pseudo-vector region. That is, it must not fall above the location P.OFF (Section 2.5). Also, the run-time system must currently be mapped read/write (see PF.RW bit description in P.FLAG word, Section 2.5). The run-time system must be read/write in this case, as the Monitor will be writing data to the buffer for the .READ.

XRB+XRCI	Channel number times two ; defines the channel for the read, as previously defined in an open (OPNFQ, CRTFQ, CREFQ, or CRBFQ functions of CALFIP, Section 3.2).
XRB+XRBLKM	For large files on disk (>65,535 blocks), this byte contains the most significant bits of the block number to begin the read. This byte is combined with the word at XRBLK to form a 24-bit field defining the block number. This byte is ignored for non-disk devices.
XRB+XRBLK	For channels opened as file-structured, this word defines the starting block number for this read. (For large files, this word forms the least significant bits of the block number to start the read.) The value performs the same action as the BLOCK option for disk, and the RECORD option for floppy disk and non-file-structured DECTape, as described in the RSTS/E Programming Manual. This word is ignored if the device is not a random-access device. If the device is random-access, and this field is non-zero, it is interpreted as the block number where the read is to start (1 to n, where n is the length of the file, in 512.-byte blocks). If zero, the next sequential block is read. For example, if a disk file is being read with a 1024.-byte buffer size, a .READ with this parameter equal to 4 would cause the fourth and fifth blocks of the file (512.-bytes each) to be read into the buffer.
	For channels opened non-file structured, this word defines the device cluster number where the read is to begin. In this case, each .READ begins with a new device cluster, so the buffer size at XRB+XRLEN should be a multiple of the device cluster size.
XRB+XRTIME	If positive, the maximum time to wait for terminal input data, in seconds. Zero indicates an infinite wait. A negative value (bit 15=1) indicates an infinite "keyboard monitor wait". This wait is used by run-time systems that act as keyboard monitors for their command input. This type of wait time acts as a flag to the Monitor and to the batch subsystem that the job is in a command input wait state.
XRB+XRMOD	Input operation modifier; significant only for card reader, terminal, or paper tape devices. (The Monitor informs you with the FLGMOD bit of the flag word returned at FIRQB+FQFLAG on the open whether or not the device will accept modifiers.) This parameter performs the same action as the RECORD modifier in BASIC-PLUS for these devices, as described in the <u>RSTS/E Programming Manual</u> .

Data Returned:

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	actual number of bytes read	2	XRBC
5	//////////	4	
7	XRBLKM	MSB of block no. //////////	6
11		LSB of block number where .READ began	10 XRBLK
13		//////////	12
15		//////////	14

XRB+XRBC Actual number of bytes just read. The value will be between 0 and the value of XRB+XRLEN passed in the XRB. If an error is returned on the read (as indicated by byte 0 of the FIRQB), this word may or may not be zero. That is, data may be input even if an error occurs. For example, if the card reader detects an illegal card column punch combination, it will place the decoded card data in the input buffer, substituting a special character for the bad column. The XRB+XRBC field will be correctly set for the number of characters input, and an error is returned.

XRB+XRBLKM For large disk files (>65,535 blocks), this byte contains the most significant bits of the block number just read. (See XRB+XRBLK, below).

XRB+XRBLK For random-access devices (see Table 3-2), this word contains the block number of the block just input with this .READ. Block numbers range from 1 to n (where n is the length of the file, in 512.-byte blocks); they define the order in which the file was written. (For disk files opened non-file structured, this is the device cluster number.)

Errors:

BADCNT The first three words describing the input buffer are illegal. (Illegal byte count for I/O.)

BSERR The specified channel number is illegal.

NOTOPN No file or device is open on the specified channel number.

PRVIOL The file/device open on the specified channel is write-only, or the caller did not obtain read access to the file/device when it was opened.

All other errors are device-dependent. Some common errors are:

DATERR Some data error occurred. There may or may not be any valid data in the input buffer. This error is issued for parity errors, bad card columns, etc.

HNGDEV Some hard device I/O error occurred. There is usually no data in the input buffer when this error occurs. (A .READ for an off-line device would cause this error.)

Example:

In this example, we assume that a disk file has been opened on channel 2, and that the XRB has already been cleared to zero. Space for the buffer is allocated with the .BLKB directive, and the buffer size and location are defined. The next sequential block in the file is to be read. (Remember that the XRB has been filled with zeroes, so XRB+XRBLK is already 0.)

```
BUFFER: .BLKB 512. ;ALLOCATE SPACE FOR BUFFER
.
.
.
MOV #512.,XRB+XRLEN ;SET BUFFER SIZE TO 512. BYTES
MOV #BUFFER,XRB+XRLOC ;STARTING ADDRESS OF BUFFER
MOV B #2*2,XRB+XRCI ;CHANNEL 2 FOR INPUT
.READ
```

3.18 .RSX -- Execute Job and Disappear (RSX only)

Form:

.RSX

Function:

The .RSX directive is used only by the RSX Run-Time System, to transfer control to the Monitor to start a user job image which has been loaded by the RSX Run-Time System.

If RSX directive emulation has been installed as part of the Monitor, the Monitor will pass control to the user job image, and the RSX Run-Time System will "disappear" from the high segment of the job space. If RSX directive emulation has not been installed as part of the Monitor, processing continues as "normal", that is, with the Run-Time System as the high segment.

The .RSX call can only be issued from the high segment. The Monitor checks the PC register, to ensure that the call originated in the high segment, and the SP register, to ensure that the stack is less than or equal to 1000. Further, this call is only relevant to the RSX Run-Time System. It is documented here for completeness, and to provide an understanding of how the RSX Run-Time System "disappears".

The RSX Run-Time System passes relevant parameters to the Monitor on the stack, including any requests from the program or from the user to make use of the extra space in APR 7. (Such information is built in to the task header of the executable file by the Task Builder, or can be requested by the user with the /SIZE switch in the CCL execute line.) There are three possibilities for requests: no request to use the extra space, a request to map a window with the extra space, or a request to extend the user job image into the extra space.

3.18.1 No Requests For Extra Space

In this case, the RSX Run-Time System places on the stack only the information needed to start the user job image:

SP--> New SP for program
New PC for program
New PSW for program

The RSX Run-Time System then issues the .RSX directive to pass control to the Monitor. The Monitor checks the value of the first word on the stack to see if it is even or odd. In this case, it is an address (of the program's stack), and so an even value. If RSX directive emulation has been installed as part of the Monitor, the Monitor releases APR 7 from mapping to the RSX Run-Time System, and passes control to the user job image, loading the appropriate registers from the information provided in the stack. If RSX directive emulation has not been installed as part of the Monitor, the Monitor does not

release APR 7 from mapping to the RSX Run-Time System. It simply passes control to the user program according to the information provided on the stack.

3.18.2 Request For Mapping A Window In APR 7

In this case, the RSX Run-Time System places on the stack the information needed to create and map the window, and to run the user job image. The first two words in the stack are in the same format as a "Create Address Window" request (see CRAW\$, Section 5.6).

```
SP--> High byte = 2, Low byte = 117.  
Address of 8-word Window Definition Block  
New SP for program  
New PC for program  
New PSW for program
```

The RSX Run-Time System then passes control to the Monitor with the .RSX directive. The Monitor checks the first word in the stack to see if it is even or odd. In this case, it is odd. If RSX directive emulation has been installed as a part of the Monitor, the Monitor releases APR 7 from its mapping to the RSX Run-Time System and loads it to map the requested window. Then it passes control to the user job image according to the information on the stack. If RSX directive emulation has not been installed as a part of the Monitor, the Monitor returns control to the Run-Time System with an error. (The requested mapping cannot be done.)

3.18.3 Request To Extend The User Job Image

In this case, the RSX Run-Time System places on the stack the information needed to extend the user job image, and to start the user job image. The first three words in the stack are in the same format as an "Extend Task" directive to the RSX Emulator (see EXTK\$, Section 5.11).

```
SP--> High byte = 3, Low byte = 89.  
Number of 32-word blocks to extend  
Reserved word  
New SP for program  
New PC for program  
New PSW for program
```

The RSX Run-Time System then passes control to the Monitor with the .RSX directive. The Monitor checks the first word in the stack to see if it is even or odd. In this case, it is odd. If the RSX Emulator has been installed as a part of the Monitor, the Monitor releases APR 7 from its mapping to the RSX Run-Time System, remaps part of the APR to the user, and passes control to the user job image according to the information passed in the stack. If RSX directive emulation has not been installed as a part of the Monitor, the Monitor returns control to the RSX Run-Time System with an error.

Form:

.RTS

Function:

In general, the .RTS directive passes control to a run-time system at the P.NEW entry point, where the intent is not to run an executable file (see P.NEW, Section 2.4). There are four general cases:

1. Passing control to the job's private default run-time system,
2. Passing control to a named run-time system,
3. Passing control to a named run-time system and establishing it as the job's private default run-time system, and
4. Passing control to a named run-time system without changing job-context information.

Once established, a job's private default run-time system replaces the system default run-time system as the one to which control passes in default situations. This concept is best explained by example.

The SWITCH utility (RSTS/E System User's Guide) uses the .RTS directive to establish the run-time system it passes control to as the job's private default run-time system. Consider the following sequence:

Ready

```
SWITCH RT11
/.MAC
MAC>^C
.^C
.SWITCH
```

Ready

The first line shows the BASIC-PLUS "Ready" prompt, indicating that the user is in the normal BASIC-PLUS entry situation. He then runs SWITCH to pass control to the RT11 Run-Time System. SWITCH establishes RT11 as the job's private default run-time system. The RT11 Run-Time System displays the dot prompt, and the user runs "MAC", the RSX Run-Time System's assembler. Control passes to the RSX Run-Time System, which loads the assembler, and runs it. MAC displays the MAC> prompt. The user, realizing his mistake in typing MAC instead of MACRO, types Control/C to exit. Since RT11 has been established as the job's private default run-time system, control

passes back to RT11, and it displays the dot prompt. The user, feeling queasy and wishing to get back to BASIC-PLUS, types another Control/C. Since RT11 is the job's private default run-time system, however, he gets another dot prompt. Gathering his wits, he runs SWITCH from the RT11 Run-Time System, leaving out the run-time system name. SWITCH transfers control back to the system default run-time system, establishing it once again as the job's private default run-time system.

In any case, the four ways that the .RTS directive can be used are:

1. The .RTS directive is used to switch control back to the already-established private default run-time system. In this case, the name of the run-time system isn't known. The word in the FIRQB that would contain the first part of the run-time system name is 0. If the run-time system which issues the .RTS isn't itself the job's private default run-time system, control is passed to the private default run-time system at entry point P.NEW. If the run-time system which issues the .RTS is the job's private default run-time system, control returns in-line (to the instruction following the .RTS) with no error indicated.
2. The .RTS directive is used to switch control to the run-time system named in the FIRQB. (Run-time systems are made known to the Monitor by the system manager with the ADD command of the UTILITY routine. With this command, the system manager defines a file (filnam.RTS) as an auxiliary run-time system. The Monitor regards "filnam" as the run-time system's name.) If the run-time system which issues the .RTS directive isn't itself the named run-time system, control passes to the named run-time system at the P.NEW entry point. If the run-time system which issues the .RTS directive is the named run-time system, control returns to the instruction following the .RTS with no error indication. If the named run-time system does not exist or is not available for some reason, control returns to the instruction following the .RTS with an error in byte 0 of the FIRQB.
3. The .RTS directive is used to switch control to a named run-time system and establish the named run-time system as the job's private default run-time system. If the run-time system which issues the .RTS isn't itself the named run-time system, control passes to the named run-time system at the P.NEW entry point, and the named run-time system is established as the job's private default run-time system. If the run-time system which issues the .RTS names itself as the run-time system, control returns to the instruction following the .RTS with no error, and the run-time system is established as the job's private default run-time system. If the named run-time system does not exist or is unavailable, control returns to the instruction following the .RTS with an error in byte 0 of the FIRQB.

4. The .RTS directive is used to switch control to a named run-time system preserving the job's current job-context information. Control passes to the named run-time system at the P.NEW entry point, but the Monitor does not refresh the keyword, reset the stack pointer, or perform any of the initialization operations described in the discussion of P.NEW in Section 2.5.

Note: This directive should not be used when running under the RT11 Run-Time System, because the lowest 1000 (octal) bytes are used by RT11 differently from other Run-Time Systems. The proper way to terminate a program running under RT11 is to exit through the RT-11 Emulator.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5		4	
7		6	
11		10	FQNAM1
13	run-time system name (2 words in RAD50 format)	12	
15	-1,priv. default;-2, don't change context	14	FQEXT
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

- FIRQB+FQNAM1 If control is being passed to a named run-time system (cases 2, 3, and 4 described previously), the name of the run-time system is stored as two words in RAD50 format beginning here. If control is being passed to the private default run-time system (case 1 described previously), the word beginning here must contain a value of 0.
- FIRQB+FQEXT To establish a named run-time system as the job's private default run-time system (case 3), set this word to -1. To switch control to a named run-time system without altering job-context information (case 4), set this word to -2. When the word at FIRQB+FQNAM1 is zero, this word is ignored.

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3		2
5		4
7		6
11		10
13	unaltered whether or not control passes to new run-time system	12
15		14

XRB+10 The three words starting at this location will pass unaltered to the new run-time system. They will also be unaltered if control returns in-line.

Data Returned:

Since control is usually passed to some other run-time system at its P.NEW entry point, no arguments as such are returned by .RTS. The three words starting at XRB+10 remain unaltered if control returns in-line.

Errors:

NORTS No run-time system exists with the specified name.

NRRTS The named run-time system does exist, but cannot be made resident right now.

PRVIOL The named run-time system does exist, but cannot be switched to for some reason. (For example, if the switch is made to a named run-time system with the intent of establishing it as a private default run-time system and its PF.KBM bit in the P.FLAG word is not set to 1.)

Example:

The following example establishes "NEWRTS" (which the system manager must have ADDED with UTILITY) as the job's private default run-time system.

```
MOV        #^RNEW,XRB+0                      ;SET RUN-TIME SYSTEM  
MOV        #^RRTS,XRB+2                      ;NAME TO "NEWRTS"  
MOV        #-1,XRB+4                          ;ESTABLISH AS PRIVATE DEFAULT  
.RTS
```

3.20 .RUN -- New Program to Run

Form:

.RUN

Function:

The .RUN directive searches for a binary (executable) file, (defined in the FIRQB), opens it on channel 15., and passes control to the P.RUN entry point of the run-time system associated with the file. [The associated run-time system is identified in the file's directory information. This directory information is initially set to indicate the run-time system under which the file was created. The system manager can change the run-time system associated with the file with the NAME command of UTILITY (see RSTS/E System Manager's Guide).]

The run-time system is responsible for loading and executing the file (see the P.RUN description in Section 2.5). Control is not returned in-line unless an error occurs.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	//////////	2	
5	//////////	4	
7	project number programmer number	6	FQPPN
11	name of binary file (two words in RAD50 format)	10	FQNAM1
13		12	
15	file extension (one word in RAD50 format)	14	FQEEXT
17	//////////	16	
21	//////////	20	
23	//////////	22	
25	//////////	24	
27	//////////	26	
31	device name (two ASCII characters)	30	FQDEV
33	<>0,unit no. real device unit number	32	FQDEVN
35	//////////	34	
37	entry parameter	36	FQENT

FIRQB+FQPPN The project, programmer number of the file to be opened. The project number is specified in the high byte, the programmer number in the low byte. If this word is passed as 0, the ppn defaults to the project, programmer number of the job which issues the .RUN directive.

FIRQB+FQNAM1 The filename of the file to be opened, as two words in RAD50 format, begins at this location.

FIRQB+FQEEXT The extension of the file to be opened, as one word in RAD50 format. If this word is set to -1, the Monitor will search for the filename supplied, substituting the default extensions for the currently installed set of run-time systems until a file with the given name and one of the default extensions is found.

FIRQB+FQDEV	The device name of the file to be opened, as two ASCII characters. If the device name is not a disk, the .RUN directive will return in-line with one of the "soft" errors described in the "Errors" section following. If a full word of 0 is passed here, and in the FIRQB+FQDEVN, the public disk structure (SY:) is searched for the named file.
FIRQB+FQENT	A 16-bit parameter word can be passed to the run-time system here. If the job issuing the .RUN is privileged, the word will be passed unaltered. If the job is non-privileged, bit 15 (the sign bit) will be cleared unconditionally. The Monitor takes no other action as a result of the the contents of this word; any processing is up to the run-time system. (See the P.RUN entry point, Section 2.5.)

Errors:

(Hard Errors):

BADNAME	The specified filename was 0. This is an illegal filename for disk files.
DETKEY	The specified device is "KB:", but the job is detached.
DEVNFS	The specified disk device is currently being used non-file-structured.
NOTCLS	Channel 15. is currently open. It must be closed before any .RUN call.
NODEVC	The specified device doesn't exist or is in an illegal format.
NORTS	The run-time system named in the file's directory information has not been installed.
NOTMNT	The specified disk device is not now mounted.
NRRTS	The run-time system named in the file's directory information is currently non-resident and cannot be made resident for some reason.
PAKLCK	The disk pack upon which the file exists is locked against further file opens.
PRVIOL	Some real protection violation occurred, such as attempting to run a file which is protected against the caller.

(Soft Errors):

PRVIOL The device is not disk, but is still a legal device on the system. Or the file was found and is on disk, but does not have the "compiled program" bit set in its file protection code (bit 6).

NOSUCH The file was not found. Note that if FIRQB+FQEXT was -l in the data passed, the Monitor has looked for the given filename with all default runnable file extensions for the currently installed run-time systems. In this case, a source version of the file may yet exist.

Example:

The following example uses the .FSS directive to translate a user-typed string to the FIRQB format. If no errors in the .FSS occur, a test is made to see if a file extension was specified. If not, a -l is supplied in FIRQB+FQEXT so that the Monitor will search for the given filename with all possible default runnable file extensions.

(read user-typed line, set up FIRQB for .FSS)

```
.  
. . .  
.FSS  
TSTB        FIRQB               ;ERROR ON .FSS?  
BNE          ERRTN              ;BRANCH TO PROCESS ERROR  
TST          FIRQB+FQEXT      ;EXTENSION GIVEN?  
BEQ          SKIP1             ;YES, SKIP NEXT  
MOV          #-1,FIRQB+FQEXT ;NO, SEARCH ALL  
SKIP1:       .RUN
```

3.21 .SET -- Set Keyword Bits

Form:

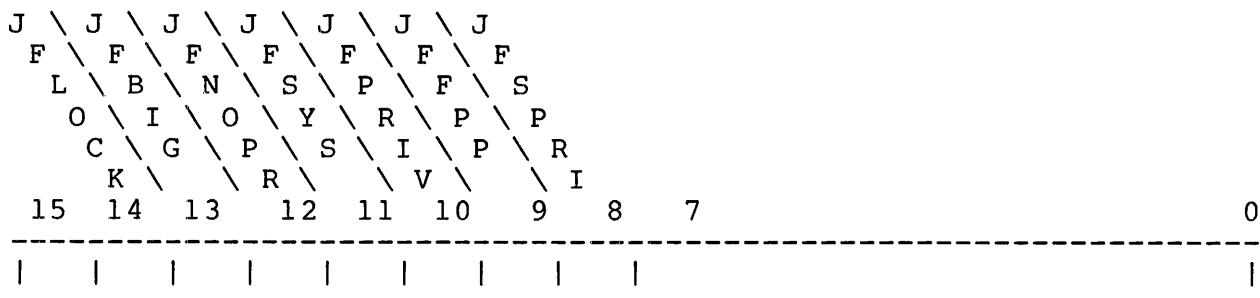
.SET

Function:

The .SET directive sets certain bits in the keyword (KEY) location in the user job image (Section 2.4). The bits to be set are passed to the Monitor in the XRB.

Data Passed:

XRB		Offset	Octal Mnemonic
1	bits to be set are set to 1 here	0	
3		2	
5		4	
7		6	
11		10	
13		12	
15		14	



JFLOCK Can be set only by privileged jobs. When this bit is set, the Monitor will swap the user job image out of memory only when

1. The job issues a .CORE directive to expand the memory allocated for the user job image, and there is not sufficient room in memory where the user job image now is to do the expansion. In this case, the job will be swapped out to disk, and back in at the indicated size.

2. A fatal error such as a memory parity failure occurs.

JFBIG	Can be set only by privileged jobs. When this bit is set, the job is allowed to exceed its private memory maximum (see .CORE, Section 3.6).
JFNOPR	Cannot be set by any job; masked off.
JFSYS	Can be set by a non-privileged job only if JFSYS was set at one time, and the temporary privileges gained were only temporarily dropped. (See description of KEY, Section 2.4.)
JFPRI	Cannot be set by any job; masked off.
JFFPP	Can be set by any caller if the PDP-11/45 compatible hardware floating point unit exists; masked off if it doesn't exist. When this bit is set, the Monitor will save information in the floating point unit as part of the job-context information kept when jobs are swapped in and out of memory.
JFSPRI	Can be set only by a privileged caller. Setting JFSPRI raises the job's run priority by one-half step. (That is, it sets bit 2 of the system-controlled low-order three bits of the run priority. Priorities are normally set by the system manager with UTILITY.)

All other bits in the XRB word are masked off; that is, the corresponding bits in KEY cannot be set by the job with the .SET directive.

Data Returned:

No data is returned with the .SET directive.

Errors:

No errors are possible with the .SET directive.

Example:

The following code sets JFBIG, allowing the job to exceed its private memory maximum.

```
MOV      #20000,XRB+0          ;SET JFBIG
.SET
```

3.22 .SLEEP -- Suspend Job

Form:

.SLEEP

Function:

The .SLEEP directive will cause the Monitor to suspend the job for some specified time interval, or until an event occurs that the job should be aware of. Optionally, the Monitor will check, before the job is suspended, to see if some event has already occurred which would cause it to awaken. If so, the job is not suspended. Control returns in-line in either case.

When a .SLEEP is executed, execution of the job is suspended until one of the following happens:

1. The sleep time (specified in the XRB) expires.
2. A local or network message is queued for the job (assuming that the job is using local or network send/receive services -- see .MESAG, Section 3.12).
3. A delimiter is typed on a terminal assigned to this job.
4. Log-ins are disabled by the system manager. (This could occur if the system is being shut down.)
5. A state change occurs on a pseudo keyboard assigned to the job. (The job has PRINTed output for the controlling job to read, or has entered an input wait state.)
6. The DMC11 driver (XM:) is open and a message is pending for the job.

If you request it, the Monitor will check before suspending the job for the following conditions.

1. A delimiter has been typed on any terminal opened by the job, or any terminal assigned to the job if the job also has a keyboard open on a non-zero channel.
2. A message has been queued for the job.
3. A state change has occurred on a pseudo keyboard opened by the job.
4. The job has a DMC11 device driver open and a message has been received by that device driver.

If the Monitor determines that any of these conditions are true, the .SLEEP is not executed.

Data Passed:

XRB

Offset Octal	Mnemonic	sleep time, in seconds	Offset Octal	Mnemonic
1		sleep time, in seconds	1	0
3			2	
5			4	
7			6	
11			10	
13			12	
15			14	

XRB+0

This word defines the sleep interval, in seconds. If the value is 0, then .SLEEP returns immediately. If the sign bit (bit 15.) is set, the Monitor checks to see if any condition which would cause the sleep to terminate has already occurred. If so, the .SLEEP is not executed.

Data Returned:

No data is returned with the .SLEEP directive.

Errors:

No errors are possible with the .SLEEP directive.

Example:

```
MOV      #5,XRB+0          ;SET TIMER TO 5 SECONDS
.SLEEP
```

3.23 .SPEC -- Special Functions for I/O

Form:

```
.  
. .  
(set XRB for special function)  
. .  
. .  
.SPEC
```

Function:

The .SPEC directive performs special functions for disk, terminal, magtape, and floppy disk devices.

For disk, the .SPEC directive allows you to explicitly lock up to seven disk blocks on a file open for update (mode parameter). A locked block cannot be accessed by another user (or from another channel). This extends the "implicit lock" feature, by which the last block or blocks read on a file open for update cannot be accessed by anyone else. The disk special functions also allow you to release explicit and implicit locks. (All locks, both explicit and implicit, are released when the file is closed.)

Data Passed (Disk):

XRB			
Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	special function code	0	
3	least signif. bits of block no. (release)	2	
5	//////////////// MSB of block no.(rl)	4	
7	DSKHND (=octal 0) channel no. * 2	6	
11	////////////////	10	
13	////////////////	12	
15	////////////////	14	
XRB+0	Defines function to be performed.		

- 0 Release any implicit lock and all explicit locks.
The Monitor de-allocates the extended internal table space it needed to do the explicit locks.
(See code 3.)
- 1 Release implicit lock.
- 2 Make implicit lock into explicit lock.
- 3 Release the explicit lock on the block specified by the word at XRB+2 (least significant bits) and the byte at XRB+4 (most significant bits). If all three bytes are zero, all explicit locks are released, but the Monitor does not de-allocate the extra space needed to do explicit locks. (This may be useful if you intend to use the explicit lock feature again during this run. An error occurs if no space is available for this purpose.)
- 4 Make implicit lock into explicit lock and release the implicit lock.

XRB+2, XRB+5 Both these bytes specify the starting block number for releasing an explicit lock. If these bytes are zero, all explicit locks will be released, but the Monitor will retain the extended table area it needs to maintain these locks. (This may be useful if you wish to use this capability again during a run.)

XRB+6 Channel number times two; defines the channel for the lock/unlock operation.

XRB+7 Handler index for disk: DSKHND (octal value = 0).

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no data is returned by the disk subfunctions of .SPEC.

Errors:

For code 2:

INTLCK Occurs if the implicit lock overlaps any current explicit lock. For example, if you read blocks 1 and 2 into a 1024-byte buffer in update mode, an implicit lock exists on blocks 1 and 2. If you explicitly locked these blocks, and then read blocks 2 and 3, and tried to explicitly lock blocks 2 and 3, you would get this error. An exact match is legal (for example, if the second read also read blocks 1 and 2), and results in a no-op.

NOBUFS Occurs if the Monitor needs to expand its internal
 table space, but memory is not available.

 NOROOM There are already seven explicit locks on this channel.

 PRVIOL There is no current implicit lock; that is, no blocks
 have been read.

For code 3:

NOSUCH The block number specified at XRB+2, XRB+4, and XRB+5
 does not correspond to the first block number of an
 explicit lock.

Data Passed (Terminal):

XRB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1	special function code		0
3	KB no. or no. of bytes to send or force	2	
5	starting address of bytes to send/force	4	
7	TTYHND (=octal 2) channel no. * 2	6	
11	KB no. (XRB+0 = 5 or 6)		10
13		12	
15		14	

XRB+0 Defines special function:

- 0 = cancel CTRL-O
- 1 = set tape mode
- 2 = enable echo and clear tape mode
- 3 = disable echo
- 4 = set ODT mode
- 5 = force to keyboard
- 6 = broadcast to keyboard
- 7 = cancel all type-ahead

XRB+2 For XRB+0 = 0,1,2,3,4, or 7:

When XRB+2 = 0, these functions take place on the terminal currently open for this job. When XRB+2 > 0, these functions take place on the keyboard number specified in XRB+2. This keyboard must be "owned" (assigned to but not open) by the calling job.

For XRB+0 = 5 or 6:

XRB+2 is the number of bytes to send or force

XRB+4 For XRB+0 = 5 or 6, this word contains the starting address of the bytes to be sent or forced.

XRB+6 Channel number times two; defines the channel for the terminal specified at XRB+2.

XRB+7 Handler index for terminals; TTYHND (octal value = 2).

XRB+10 For XRB+0 = 5 or 6, this word contains the keyboard number to which the data is to be sent or forced.

Data Returned (Terminal):

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	number of bytes not sent or forced	2
5		4
7		6
11		10
13		12
15		14

XRB+2 Number of bytes that could not be sent or forced. (Returned only when XRB+0 in the data passed was 5 or 6.)

Data Passed (Magtape):

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	special function code		0	
3	parameter		2	
5			4	
7	MTAHND (=octal 16) channel no. * 2		6	
11			10	
13			12	
15			14	

- XRB+0 This word defines the special magtape function to be performed. These codes are summarized in Table 3-2. For a detailed discussion of these functions, see the discussion of the MAGTAPE function in the RSTS/E Programming Manual.
- XRB+2 The meaning of this word varies according to the special function code specified at XRB+0. Table 3-2 summarizes these values; for a detailed discussion, see the MAGTAPE function description in the RSTS/E Programming Manual.
- XRB+6 Channel number times two; defines the channel on which the tape is currently open.
- XRB+7 Handler index for magtape: MTAHND (octal value = 16).

Data Returned (Magtape) :

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3	value (see Table 3-3)	2
5	//////////	4
7	//////////	6
11	//////////	10
13	//////////	12
15	//////////	14

XRB+2

The meaning of this word according to the value at XRB+0 in the data passed. Table 3-3 summarizes these values. For a detailed discussion, see the MAGTAPE description in the RSTS/E Programming Manual.

TABLE 3-3
SPECIAL FUNCTIONS FOR MAGTAPE

Action	Function Code (octal)	Parameter	Value Returned
Rewind and off-line	1	unused	0
Write end-of-file	2	unused	0
Rewind	3	unused	0
Skip record	4	# records to skip	# records not skipped
Backspace over record	5	# records to backspace	# records not backspaced
Set density and parity	6	E+D*4+P+S (see below)	0
Tape status function	7	unused	status (see below)
File characteristics	10	unused	file characteristics (see below)
Rewind on close	11	unused	0

NOTE: Values below are given in octal.

Parameter word for function code 6:

E = Extended Density

400 = see values of D where E = 400
0 = see values of D where E = 0

D = Density

If E = 0, then D values indicate:

0 = 200. BPI (7-track only)
1 = 556. BPI (7-track only)
2 = 800. BPI (7-track only)
3 = 800. BPI, dump mode (7-track)
 800. BPI (9-track)

If E = 400, then D values indicate:

0 = 1600. BPI, phase-encoded (9-track only)
1 = reserved for future use
2 = reserved for future use
3 = reserved for future use

P = Parity (0= odd, 1 = even)

S = Stay

0 = mode value specified in open does not stay on close
20000 = mode value specified in open is retained after close

Value Returned at XRB+2 for Function Code 7 (Magtape Status Word):

Bit	Octal Value	Meaning
15	100000	Last command caused an error
14-13	If bit 3 = 0, these bits indicate density:	
	00000	200 BPI
	20000	556 BPI
	40000	800 BPI
	60000	800 BPI, dump mode
	If bit 3 = 1, these bits indicate density:	
	00000	1600. BPI
	20000	reserved
	40000	reserved
	60000	reserved
12	00000	9-track tape
	10000	7-track tape
11	0000	Odd parity
	4000	Even parity
10	2000	Magtape is physically write-locked
9	1000	Tape is beyond end-of-tape marker
8	400	Tape is at beginning-of-tape (Load Point)
7	200	Last command detected an EOF
6	100	The last command was .READ and the record read was longer than the I/O buffer size (that is, part of the record was lost).
5	40	Unit is non-selectable (off-line).
4	00	Unit does not accept 1600. BPI
	20	Unit accepts 1600. BPI
3	00	See values for bits 14-13.
	10	See values for bits 14-13.
2-0	Indicates last command issued:	
	0	= off-line
	1	= read
	2	= write
	3	= write eof
	4	= rewind
	5	= skip record
	6	= backspace record

Value Returned at XRB+2 for Function Code 10 (File Characteristics Word):

word = 0, DOS format or ANSI U (undefined) format
<>0, ANSI format, with bit meanings as defined below:

Bit	Octal	Value	Meaning
13-12	00000	carriage control embedded 'M'	
	10000	FORTRAN carriage control 'A'	
	20000	implied LF/CR before record ''	
15-14	40000	F (fixed-length)	
	100000	D (variable-length)	
	140000	S (spanned)*	
11-0	--	For Format F, this value is the record length, in bytes For Format D, this value is the maximum record length, in bytes	

*ANSI format S is not supported by RSTS/E systems.

For floppy disk devices, the .SPEC directive allows you to obtain the density (single or double) of the current floppy disk, to mount a new floppy disk and recompute the density, and to reformat an RX02 floppy disk for a desired density. Because the RX02 floppy disk drive supports single and double density floppy disks, the .SPEC function is especially useful for programmed floppy disk operations. For example, .SPEC allows you to mount a series of single and double density floppy disks without having to close and reopen the device for each mount. That is, the driver computes density once; during the initial open. If you insert a second floppy that is incompatible with the initially computed density, a read or write operation will fail. .SPEC permits you to include an instruction in your program that causes the driver to recompute the density. Also, for RX02 floppy disk drives, .SPEC permits you to specify a density reformat operation.

.SPEC can require as much as 20 seconds to reformat the density of the RX02 floppy disk and cannot be interrupted with CTRL/C. Note that if the operation is interrupted (by power failure or catastrophic error), the floppy disk is rendered unusable. That is, the floppy disk will contain both single and double density. To recover, you must reformat the floppy disk.

Data Passed (Floppy Disks):

XRB

	Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		special function code		0	
3		parameter word		2	
5				4	
7		RXDHND (=octal 22) channel no. * 2		6	
11				10	
13				12	
15				14	

XRB+0 Function code specifying the desired operation:

- = 0 Return density of currently mounted floppy disk.
(The parameter word at XRB+2 must be 0.)
- = 1 Recompute density, and return density. (The parameter word at XRB+2 must also be 0.) This code must be issued prior to any I/O operation on the floppy disk.
- = 2 Reformat the current floppy disk to the density specified in the parameter word at XRB+2. Only allowed on RX02 drives.

XRB+2 Parameter word must = 0 when function code = 0 or 1.
Otherwise, when function code = 2:
 = 1 Reformat as single-density (one sector = 128 bytes)
 = 2 Reformat as double-density (one sector = 256 bytes)

XRB+6 Channel number * 2; defines the channel on which the floppy disk is currently open.

XRB+7 Handler index for floppy disk: RXDHND (octal value = 22).

Data Returned:

XRB			
Offset Octal Mnemonic	-----	Offset Octal Mnemonic	-----
1	/ /	0	
3	/ / / / / / / / / / / / density	2	
5	/ /	4	
7	/ /	6	
11	/ /	10	
13	/ /	12	
15	/ /	14	

XRB+2 Density, returned when XRB+0 in data passed is 0 or 1.
Equal to 1 for single-density (sector = 128 bytes) or 2 for double-density (sector = 256 bytes).

Errors:

HNGDEV A hardware error occurred. This can often be a transient condition. Retry the operation.

ERRERR An attempt was made to reformat on an RX01 floppy disk drive. You can use .SPEC to reformat floppy disk density only on RX02 drives.

3.24 .STAT -- Return Job Statistics

Form:

.STAT

Function:

The .STAT directive returns current statistics on the job to the XRB.

Data Passed:

No data is passed with this call.

Data Returned:

XRB

Offset Octal	Mnemonic	XRB	Offset Octal	Mnemonic
1	current job image size, in K words	0		
3	current run-time system size, in K words	2		
5	current private memory max, in K words	4		
7	maximum job image size, in K words	6		
11	current project, programmer number	10		
13	current run priority	12		
15	current run burst	14		

XRB+0	The current size of the user job image for this job, in K words.
XRB+2	The size of the current run-time system for this job, in K words.
XRB+4	The current private memory maximum for the user job image, in K words. If the job has an unlimited memory maximum, or if its private memory maximum is larger than the possible maximum size allowed by its current run-time system, then the value returned here is the maximum size possible for the current run-time system, in K words. If the job's private maximum is less than the run-time system minimum, then the value returned here is the run-time system's minimum size. (See .CORE, Section 3.6, for a discussion of these values.)

In all cases, this value represents the maximum size that the user job image can be under its current run-time system.

XRB+6 The maximum job image size possible (under the current run-time system), in K words.

XRB+10 The job's current project,programmer number is returned here. The programmer number is returned as a binary value in the low byte (XRB+10); the project number, as a binary value in the high byte (XRB+11). If the job is not logged in, a value of 0 is returned here.

XRB+12 The job's current run priority. Run priority may range from -128., indicating a suspended job, to +127., the highest priority. The Monitor schedules jobs for time-shared execution according to this priority. When a user first logs in to RSTS/E, LOGIN is run with priority 0. LOGIN sets the user's job run priority to -8. Only in unusual cases should the run priority be changed. It can be changed by a privileged user for any job with the UTILITY system program. It can also be changed with the UU.PRI subfunction of the .UUO directive (Section 3.32.32). It can be modified by the job by 1/2 step with the .SET and .CLEAR directives.

The special-case value of -128. indicates that the job is never scheduled to run; it is suspended.

XRB+14 The job's current run burst. The run burst is the amount of time that the job will be allowed to execute compute-bound before the next job in the schedule is given control. The units of run burst are 1/60ths or 1/50ths of a second, depending on the clock in use and/or the line frequency. (Systems running with the KW11 clock at crystal speeds, rather than at line frequency, have a run burst unit of 1/50th of a second. If the system is operating off a 60 Hz power line, one run burst unit equals 1/60th of a second.) The range of run burst is from 1 to 127 (decimal) inclusive. When a job is created, the Monitor sets the run burst to a value of 6. This value can be modified by the system manager for a particular job with the UTILITY system program. It can also be modified with the UU.PRI subfunction of the .UUO directive (Section 3.32.32).

Errors:

No errors are possible on this directive.

Example:

No data is passed to the Monitor with the .STAT directive. Therefore, the call is simply:

.STAT

3.25 .TIME -- Return Timing Information

Form:

```
.TIME
```

Function:

The .TIME directive returns job timing information: elapsed CPU time, elapsed time connected to a user terminal (channel 0), elapsed device time, and memory utilization.

Data Passed:

No data is passed with the .TIME directive.

Data Returned:

XRB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	low 16 bits of elapsed CPU time, .1sec	0	
3	elapsed connect time to channel 0, min	2	
5	low 16 bits of memory utilization, KCTs	4	
7	elapsed device time, min	6	
11	high 16 bits of elapsed CPU time, .1sec	10	
13	(Reserved for future use)	12	
15	high 16 bits of memory utilization, KCTs	14	

XRB+0 This word contains the low-order 16 bits of the job's elapsed CPU time, in tenths of a second of CPU utilization.

XRB+2 This word contains the elapsed time that the job has been connected to a channel 0 terminal, in minutes.

XRB+4 This word contains the low-order 16 bits of the job's memory utilization, in kilo-core-ticks (KCTs). A kilo-core-tick is the use of 1K of memory for one-tenth of a second.

XRB+6 This word contains the job's elapsed device time. Device time is the use of an assignable device for one minute. If two devices are owned for one elapsed minute, then two units of device time are accrued.

- XRB+10 This word contains the high-order 16 bits of the job's elapsed CPU time (see XRB+0).
- XRB+12 This word is reserved for future use.
- XRB+14 This word contains the high-order 16 bits of the job's memory utilization (see XRB+4).

Errors:

No errors are possible with the .TIME directive.

Example:

Since no data is passed with this directive, the call is simply:

.TIME

3.26 .TTAPE -- Enter Tape Mode

Form:

.TTAPE

Function:

The .TTAPE directive enters "tape mode" on the job's terminal (channel 0). This directive is useful when it is necessary to read data from the low-speed paper tape reader available on some terminals. The terminal would have to be open on channel 0, and this call would be given before the .READ. Three things happen:

1. No incoming characters are echoed, preventing needless output at the terminal while the tape is being read.
2. The DELETE character (ASCII code 177 octal) is ignored, rather than deleting the previous character.
3. A LINE FEED character (ASCII code 012 octal) is not automatically appended to an incoming RETURN (ASCII code 015 octal).

The .TTECH directive (Section 3.28) returns character processing to normal on channel 0.

Data Passed:

No data is passed with the .TTAPE directive.

Data Returned:

No data is returned with the .TTAPE directive.

Errors:

DETKEY Channel 0 is not currently open for this job; it is running detached.

Example:

Since no data is passed (or returned), the call is simply:

.TTAPE

3.27 .TTDDT -- Disable Full-Line Buffering

Form:

.TTDDT

Function:

The .TTDDT directive disables the Monitor's usual practice of buffering a full line of data from the user's terminal (channel 0) before passing it on to the job on a .READ. The Monitor accepts data typed at a user terminal and stores it in a buffer until the job associated with the terminal reads the data. Normally, a .READ causes the Monitor to pass a line to the job's buffer. (A line is any number of characters ended with a RETURN, LINE FEED, ESCAPE, FORM FEED, or CTRL/D combination.) If a full line is not in the Monitor's buffer, the Monitor will stall the job until it gets a delimiter, then awaken the job and pass the line on to the job's buffer.

The .TTDDT directive tells the Monitor that, when the next .READ on channel 0 is issued, it is to pass on whatever is currently in the Monitor's buffer, whether or not a delimiter has been typed. If no characters are in the Monitor's buffer, the job is stalled until at least one character has been typed.

.TTDDT is a "one-shot" directive -- it affects only the next .READ on channel 0. If you want to do this type of input consistently, you must execute a .TTDDT before each .READ.

This type of input is useful when you want to respond to each character that a user types. (Note that more than one character may be in the Monitor's buffer. If you really want only one character, use .TTDDT before each .READ, and define a 1-character input buffer for the .READ.) For example, the Octal Debugging Tool (ODT) uses this capability to make changes in memory based on octal values typed by the user. This type of input puts a high load on the system, and is not recommended except in unusual circumstances.

Data Passed:

No data is passed with the .TTDDT directive.

Data Returned:

No data is returned with the .TTDDT directive.

Errors:

DETKEY Channel 0 is not currently open for this job; the job is running detached.

Example:

Since no data is passed, the call is simply:

.TTDDT

3.28 .TTECH -- Undo .TTAPE or .TTNCH

Form:

.TTECH

Function:

The .TTECH directive causes the Monitor to resume normal character input processing on channel 0, when this has been disabled with either a .TTAPE directive (Section 3.26) or a .TTNCH directive (Section 3.29).

Data Passed:

No data is passed with the .TTECH directive.

Data Returned:

No data is returned with the .TTECH directive.

Errors:

DETKEY Channel 0 is not open for this job; the job is running detached.

Example:

Since no data is passed, this call is simply:

.TTECH

3.29 .TTNCH -- Stop Echo

Form:

.TTNCH

Function:

The .TTNCH directive disables terminal echo on the job's terminal (channel 0). That is, whatever the user types is accepted, but it is not echoed back for display on the terminal. Otherwise, all normal character processing occurs. (The .TTECH directive, Section 3.28, returns character processing to normal on channel 0.)

Data Passed:

No data is passed with the .TTNCH directive.

Data Returned:

No data is returned with the .TTNCH directive.

Errors:

DETKEY Channel 0 is not open for this job; the job is running detached.

Example:

Since no data is passed or returned, the call is simply:

.TTNCH

3.30 .TTRST -- Restart Output

Form:

.TTRST

Function:

The .TTRST directive restarts any programmed output to the user's terminal when such output has been stopped by the user's typing a CTRL/O or CTRL/C.

Terminal service on a RSTS/E system (a driver within the Monitor itself) maintains a "junk all programmed output" indicator for each terminal. When this indicator is set, the driver will ignore a .WRITE directive to channel 0 rather than display the data at the user's terminal. When the indicator is clear, a .WRITE to channel 0 is processed normally. The .TTRST directive clears this indicator.

The driver sets the indicator when the user types a CTRL/C combination. It reverses the indicator when the user types a CTRL/O combination. Also, a .READ directive for channel 0 clears the indicator.

The .TTRST is used effectively in run-time systems with keyboard monitors. By issuing .TTRST before displaying any prompt, you can be sure that the prompt is actually displayed at the user's terminal.

Data Passed:

No data is passed with the .TTRST directive.

Data Returned:

No data is returned with the .TTRST directive.

Errors:

DETKEY Channel 0 is not currently open for this job; the job is running detached.

Example:

Since no data is passed, the call is simply:

.TTRST

3.31 .ULOG -- Assign/Reassign/Deassign Device or Enter/Deassign User Logical

Form:

.ULOG

Function:

The .ULOG directive has three subfunctions. The particular action desired is selected by setting a function field in the FIRQB (at offset FQFUN). The subfunctions are described in the following subsections; a summary is given below.

FQFUN Value (Octal)	Mnemonic	Action Performed
12	UU.ASS	Assign/Reassign a device, or enter user logical
13	UU.DEA	Deassign a device or user logical
14	UU.DAL	Deassign all devices and user logicals

3.31.1 UU.ASS (Assign/Reassign a Device, or Enter User Logical)

Form:

```
MOV B    #UU.ASS,FIRQB+FQFUN  
.  
.  
.  
(set up FIRQB and XRB)  
.  
.  
.  
.ULOG
```

Function:

The UU.ASS subfunction of .ULOG allows you to do one of three things:

1. Assign a device to a job.
2. Reassign a device to another job, or
3. Enter a user logical. This feature allows you to do one of the following:
 1. Assign a logical name to a device. The Monitor will then use this assignment for logical-to-physical device translation by the .FSS directive (Section 3.11).
 1. You can also associate a project,programmer number with a particular logical name. This "user logical ppn" will be used if the associated logical name is found in a string parsed by .FSS, but no ppn was found in the string. This feature is useful if you wish to override a system-wide logical name with an associated ppn. The logical name LB, for example, is commonly associated with a disk on the public structure, and project,programmer number [1,1]. With this feature, you can set up a different device and ppn for the logical name LB.
 2. Assign a project, programmer number to be substituted for a commercial at sign character (@) encountered in a file specification string parsed by an .FSS directive.
 3. Assign a protection code to be used as a default if no protection code is specified in a file specification string translated by an .FSS directive.

NOTE: Assigning user logicals assigns values to the USRPPN, USRPRT, and USRLOG areas in the low 1000 bytes of virtual memory (Section 2.4). If these values are in a non-standard location, you must specify where they are by setting the XRB. For all other uses of .ULOG, the XRB should be cleared to zeroes.

Data Passed -- Assign/Reassign Device:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		-----	0	
3	FQFUN	UU.ASS (= octal 12) -----	2	
5		-----	4	
7		(must = 0)	6	FQPPN
11		(must = 0) 0=>assign; <>0=>job 10 FQNAM1		
13		-----	12	
15		DOS or ANS (1 word in RAD50 format)	14	
17		-----	16	
21		-----	20	
23		100001 for snagging assign/reassign;else0 22		
25		-----	24	
27		(must = 0)	26	
31		device name (2 ASCII characters)	30	FQDEV
33		<>0, unit no. real device unit number 32 FQDEVN		
35		-----	34	
37		-----	36	

- FIRQB+FQFUN The function code UU.ASS (octal value = 12).
- FIRQB+FQNAM1 For assigning a device, the two bytes beginning here must = 0.
- For reassigning a device, this byte is the job number to which the device is to be reassigned. The byte at FIRQB+FQNAM+1 must = 0.
- FIRQB+14 For assigning or reassigning a magtape device, this word is either DOS or ANS (in RAD50 format) to indicate DOS or ANSI label format for the magtape drive.
- FIRQB+22 For assigning or reassigning a device, setting this field to 100001 octal indicates a "snagging" assign or reassign. That is, it will assign or reassign the device even if it is currently assigned to another job.

This feature can only be issued from a privileged account. Also, the target device must not be open, and the current owner cannot be performing a directory on that device. ("Performing a directory" is the UU.DIR subfunction of the .UUO directive, Section 3.32.) If a snagging assign or reassign is not desired, set this word to 0.

FIRQB+FQDEV Device name to be assigned or reassigned, specified as two ASCII characters. If this word is zero, the public disk structure is assumed.

FIRQB+FQDEVN Device unit number, passed as a binary value in byte FIRQB+FQDEVN. A non-zero value in byte FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in byte FIRQB+FQDEVN+1 indicates no device unit number.

NOTE: The XRB should be cleared to zeroes for assign/reassign device.

Data Passed -- Enter User Logical:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3	FQFUN	UU.ASS (= octal 12)	2	
5			4	
7		project number programmer number	6	FQPPN
11			10	FQNAM1
13		user logical device name (2 words in RAD50 format)	12	
15			14	
17			16	
21			20	
23			22	
25			24	
27	FQPROT	protection code 377=assign prt. code	26	
31		device name (2 ASCII characters)	30	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35			34	
37			36	

FIRQB+FQFUN The function code UU.ASS (octal value = 12).

FIRQB+FQPPN This word is the project, programmer number for features which use one [see items (3ai) and (3b), above.] For a user-assignable ppn, this is the project,programmer number to be used to replace an @ sign in a string parsed by an .FSS directive. In this case, the two bytes at FIRQB+FQNAM1 and FIRQB+FQNAM1+1 must be 0 [case (3b), above].

If bytes FIRQB+FQNAM1 through FIRQB+FQNAM1+3 contain a logical device name, and a ppn is to be associated with that name [case (3ai), above], the ppn is specified here. If no ppn is to be associated with the logical device name, this word should be set to 0.

FIRQB+FQNAM1 For assigning a user logical name to a device, the two words beginning here are the logical name, in RAD50

	format. Otherwise, these bytes should be set to zero.
FIRQB+26	For assigning a user-assignable default protection code, set this byte non-zero. (See FIRQB+FQPROT.)
FIRQB+FQPROT	This byte is the protection code to be used as a default if no protection code is specified in a string scanned by an .FSS directive [case (3c), above]. The byte at FIRQB+26 must be non-zero, and the two words at FIRQB+FQPPN and FIRQB+FQNAM1 must be zero.
FIRQB+FQDEV	Device name for assigning a logical name to be associated with a device. [case (3a), above]. Specified as two ASCII characters. If this word is zero, the public disk structure is assumed.
FIRQB+FQDEVN	Device unit number for assigning a logical name to be associated with a device [case (3a), above]. The device unit number is passed as a binary value in byte FIRQB+FQDEVN. A non-zero value in byte FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in byte FIRQB+FQDEVN+1 indicates no device unit number.

Offset Octal Mnemonic		Offset Octal Mnemonic
1	length of user logical information	0 XRLEN
3	length of user logical information	2 XRBC
5	starting address for user logical info.	4 XRLOC
7		6
11		10
13		12
15		14

- XRB+XRLEN If the user logical information is in its standard location (USRPPN, USRPRT, and USRLOG), this word is passed as 0. If some non-standard set of locations is being used, then the length of that information, in bytes, is specified here. Thus, when the user logical information is in a non-standard location, this word must be at least 4 (for user-assignable ppn and default protection code). The space for user logical names is optional. If no space is reserved for logical names in the non-standard location, no such assignments will be allowed.
- XRB+XRBC This word also contains the length of the information. (same as the word at XRB+XRLEN).
- XRB+XRLOC If the word at XRB+XRLEN is non-zero, then this word defines the starting location for the user logical information (the default ppn). The order and format of the information created by .ULOG is described in Section 2.4; see USRPPN, USRPRT, and USRLOG.

Data Returned:

Except for a possible error code in byte 0 of the FIRQB, no meaningful data is returned by the UU.ASS subfunction of .ULOG.

Errors:

INUSE	For assign/reassign device, the specified device is currently open or has an open file. For assign user logical, no space is currently available for storing the information.
NODEVC	The device name specified at FIRQB+FQDEVN is a logical device name for which a physical device is currently not assigned.
NOTAVL	The device specified exists on the system, but the operation failed for one of the following reasons: <ol style="list-style-type: none">1. The device is currently reserved by another job (see description of FIRQB+22).2. Ownership of the device requires privilege that the user does not have.3. The device or its controller is disabled.4. The device is a keyboard line for a pseudo keyboard only.
BDNERR	An attempt was made to reassign a device to a non-existent job. This error can only occur for a reassign call.

Example:

The following code assigns the user logical name "OUT" to the public disk structure. (Assume that the FIRQB and XRB have been previously cleared to zero.)

```
MOV B #UU.ASS,FIRQB+FQFUN ;SET FUNCTION CODE
MOV #^ROUT,FIRQB+FQNAM1 ;SET LOGICAL NAME AS
MOV #^R ,FIRQB+FQNAM1+2 ;TWO WORDS RAD50
.ULOG
```

3.31.2 UU.DEA -- Deassign A Device or User Logical

Form:

```
MOVB #UU.DEA,FIRQB+FQFUN
.
.
(set up FIRQB)
.
.
.
.ULOG
```

Function:

The UU.DEA subfunction of .ULOG deassigns a device from the current job (releases it for use by other job), or deassigns a user-logical assignment.

Data Passed for Deassign Device:

FIRQB

Offset			Offset
Octal	Mnemonic	Octal	Mnemonic
1		0	
3	FQFUN	UU.DEA (=octal 13)	2
5			4
7		(must = 0)	6 FQPPN
11		(must = 0)	10 FQNAM1
13			12
15			14
17			16
21			20
23			22
25			24
27		(must = 0)	26
31		device name (2 ASCII characters)	30 FQDEV
33		<>0, unit no. real device unit number	32 FQDEVN
35			34
37			36

FIRQB+FQFUN The UU.DEA function code (octal value = 13).

FIRQB+FQDEV The name of the device to be deassigned, as two ASCII characters.

FIRQB+FQDEVN The device unit number is passed in this byte, in binary. A non-zero value in FIRQB+FQDEVN+1 indicates an explicit device unit number. A zero value in FIRQB+FQDEVN+1 indicates no device unit number.

Note: The XRB should be cleared to zeroes for deassigning a device.

Data Passed for Deassign User Logical:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		/ /	0	
3	FQFUN	UU.DEA (=octal 13) / / / / / / / / / / /	2	
5		/ /	4	
7		project number programmer number	6	FQPPN
11			10	FQNAM1
13		user logical device name (2 words in RAD50 format)	12	
15		/ /	14	
17		/ /	16	
21		/ /	20	
23		/ /	22	
25		/ /	24	
27	FQPROT	protection code <>0 => assgn prt cd	26	
31		/ /	30	
33		/ /	32	
35		/ /	34	
37		/ /	36	

FIRQB+FQFUN The UU.DEA function code (octal value = 13).

FIRQB+FQPPN The default project,programmer number to be deassigned. In this case, the word at FIRQB+FQNAM1 must be zero. If deassigning default protection code, this word must be set to zero.

FIRQB+FQNAM1 The logical device name to be deassigned, as two words in RAD50 format. If deassigning default ppn or protection code, the word at FIRQB+FQNAM1 must be zero.

FIRQB+26 This byte is non-zero if deassigning a protection code.

FIRQB+FQPROT The default protection code to be deassigned. In this case, the two words at FIRQB+FQPPN and FIRQB+FQNAM1 must be set to zero.

XRB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1	length of user logical information		0	XRLEN
3	length of user logical information		2	XRBC
5	starting address for user logical info.		4	XRLOC
7			6	
11			10	
13			12	
15			14	

XRB+XRLEN If the user logical information is in its standard location (USRPPN, USRPRT, and USRLOG), this word is passed as 0. If some non-standard set of locations is being used, then the length of that information, in bytes, is specified here.

XRB+XRBC This word also contains the length of the information, (same as the word at XRB+XRLEN).

XRB+XRLOC If the word at XRB+XRLEN is non-zero, then this word defines the starting location for the user logical information (the default ppn). The order and format of the information created by .ULOG is described in Section 2.4; see USRPPN, USRPRT, and USRLOG.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.DEA subfunction of .ULOG.

Errors:

NODEVC The device or its type specified at FIRQB+FQDEV and FIRQB+FQDEVN is not part of your system configuration.

Example:

The following code deassigns MT0: from the current job.

MOVB #UU.DEA, FIRQB+FQFUN	;SET FUNCTION CODE
CLR FIRQB+10	;CLEAR WORD 10
MOV #"MT,FIRQB+FQDEV	;SET DEVICE
MOVB #0,FIRQB+FQDEVN	;SET UNIT NUMBER
MOVB #377,FIRQB+FQDEVN+1	;UNIT NUMBER REAL

3.31.3 UU.DAL -- Deassign All Devices And User Logicals

Form:

```
MOVB    #UU.DAL,FIRQB+FQFUN
.
.
.
(set up FIRQB and XRB)
.
.
.
.ULOG
```

Function:

The UU.DAL function of .ULOG deassigns all devices and user logicals for the calling program.

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	UU.DAL (=octal 14)
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

FIRQB+FQFUN The function code UU.DAL (octal value = 14).

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	length of user logical information	0 XRLEN
3	length of user logical information	2 XRBC
5	starting address for user logical info.	4 XRLOC
7		6
11		10
13		12
15		14

- XRB+XRLEN If the user logical information is in its standard location (USRPPN, USRPRT, and USRLOG), this word is passed as 0. If some non-standard set of locations is being used, then the length of that information, in bytes, is specified here.
- XRB+XRBC This word also contains the length of the information. (same as the word at XRB+XRLEN).
- XRB+XRLOC If the word at XRB+XRLEN is non-zero, then this word defines the starting location for the user logical information (the default ppn). The order and format of the information created by .ULOG is described in Section 2.4; see USRPPN, USRPRT, and USRLOG.

Data Returned:

No data is returned by the UU.DAL subfunction of .ULOG.

Errors:

No errors are possible with DALFQ.

Example:

The following code deassigns all devices and user logicals for the current job. The user logical information is in its standard location.

```

MOV B    #UU.DAL,FIRQB+FQFUN
CLR     XRB+XRLEN
CLR     XRB+XRBC
CLR     XRB+XRLOC
.ULOG

```

3.32 .UUO -- Execute BASIC-PLUS SYS Call

Form:

.UUO

Function:

The .UUO directive allows a MACRO program to execute the BASIC-PLUS SYS calls to the File Processor (FIP). The SYS calls are described in the RSTS/E Programming Manual. For the MACRO programmer's convenience, the FIRQB formats for the calls are shown here; for detailed descriptions, however, refer to the RSTS/E Programming Manual.

Table 3-4 summarizes the FIP SYS calls; note that some of the calls are handled by directives other than .UUO. (.FSS, for example, handles the file name string scan in MACRO.) The mnemonic subfunction names are provided by the COMMON.MAC prefix file; the decimal values are the BASIC-PLUS FIP call subfunction codes, and the .UUO subfunction codes which COMMON.MAC relates to the mnemonics listed.

The rest of this section gives the FIRQB formats for data passed and returned for the .UUO subfunctions; the subfunctions are organized in alphabetical order of the mnemonics provided by COMMON.MAC.

TABLE 3-4
.UUO SUBFUNCTIONS -- CALLS TO THE FILE PROCESSOR (FIP)

Mnemonic	BASIC-PLUS SYS Call Code (Decimal)	Privileged Status	Function	Section
UU.FIL	-26	Yes	File placement and modification	3.32.20
UU.ATR	-25	No	Read or write attributes	3.32.2
UU.CCL	-24	Yes	Add/delete CCL command	3.32.7
(.FSS)	-23	No	Terminating file name string scan	3.10
(.SET)	-22	Yes	Set special run priority	3.21
(.SET/ .CLEAR)	-21	Yes	Drop/regain temporary privilege	3.21, 3.5
(.SET/ .CLEAR)	-20	Yes	Lock/unlock job in memory	3.21, 3.5
UU.LOG	-19	Yes	Set number of logins	3.32.24
UU.RTS	-18	Yes	Add run-time system	3.32.34
		Yes	Remove run-time system	
		Yes	Load run-time system	
		Yes	Unload run-time system	
		Yes	Add resident library	
		Yes	Remove resident library	
		Yes	Load resident library	
		Yes	Unload resident library	
UU.NAM	-17	No	Name run-time system	3.32.27
UU.DIE	-16	Yes	System shutdown	3.32.15
UU.ACT	-15	Yes	Accounting dump	3.32.1
UU.DAT	-14	Yes	Change system date/time	3.32.12
UU.PRI	-13	Yes	Change priority/run burst/job size	3.32.31
UU.TB2	-12	No	Get Monitor tables -- Part II	3.32.39
UU.BCK	-11	Yes	Change file backup statistics	3.32.5
(.FSS)	-10	No	Filename string scan	3.10
UU.HNG	-9	Yes	Hangup a dataset	3.32.21
UU.FCB	-8	No	FCB/DDB information	3.32.19
(internal to BASIC-PLUS)	-7	No	CTRL/C trap enable	--
UU.POK	-6	Yes*	Poke memory	3.32.30
(.SPEC)	-5	Yes	Broadcast to terminal	3.23
(.SPEC)	-4	Yes	Force input to terminal	3.23
UU.TB1	-3	No	Get Monitor tables -- Part I	3.32.38
UU.NLG	-2	Yes	Disable logins	3.32.28
UU.YLG	-1	Yes	Enable logins	3.32.41
UU.PAS	0	Yes	Create user account	3.32.29
UU.DLU	1	Yes	Delete user account	3.32.17
UU.CLN	2	Yes	Clean up a disk pack	3.32.10
UU.MNT	3	Yes	Disk packs	3.32.26
		Yes	Set terminal characteristics	
UU.LIN	4	Yes	Login	3.32.23
UU.BYE	5	Yes	Logout	3.32.6
UU.ATT	6	Yes	Attach	3.32.4
		Yes	Reattach	

*Poke memory can be executed only from account [1,1].

TABLE 3-4 (Cont.)
.UUO SUBFUNCTIONS -- CALLS TO THE FILE PROCESSOR (FIP)

Mnemonic	BASIC-PLUS			Section
	SYS Call Code	Status	Privileged Function	
UU.DET	7	Yes	Detach	3.32.14
UU.CHU	8	Yes	Change password/quota	3.32.9
		Yes	Kill job	
		Yes	Disable terminal	
UU.ERR	9	No	Return error message	3.32.18
UU.ASS	10	No	Assign/reassign device	3.32.2
(.ULOG)		Both	Assign user logical	3.31
UU.DEA	11	No	Deassign device	3.32.13
UU.DAL	12	No	Deassign all devices	3.32.11
UU.ZER	13	Both	Zero a device	3.32.42
UU.RAD	14	Both	Read or read-and-reset accounting data	3.32.33
UU.DIR	15	No	Directory lookup on index	3.32.16
		No	Special Magtape directory lookup	
UU.TRM	16	Both	Set terminal characteristics	3.32.40
UU.LOK	17	No	Disk directory lookup on filename	3.32.25
		No	Disk wildcard directory lookup	
(.MESAG)	18	Both	Message send/receive	3.12
UU.CHE	19	Yes	Enable/disable disk caching	3.32.8
UU.CNV	20	No	Reserved	
UU.SLN	21	Yes	System logical names	3.32.35
(.MESAG)	22	Both	Message send/receive	3.12
UU.SWP	23	Yes	Add/remove system files	3.32.36
UU.JOB	24	Yes	Job creation	3.32.22
UU.PPN	25	No	Wild card PPN lookup	3.32.31
UU.SYS	26	No	Return System Status information	3.32.37

3.32.1 UU.ACT (Accounting Information Dump)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.ACT (= -15.)	2	
5		4	
7	project number programmer number	6 FQPPN	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no meaningful data is returned with the UU.ACT subfunction of .UUO.

3.32.2 UU.ASS (Assign/reassign Device)

Data Passed:

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3 FQFUN	UU.ASS (= 10.) //////////	2
5	//////////	4
7	//////////	6
11	(must = 0) 0,assign;<>0, job	10
13	//////////	12
15	DOS or ANS (1 word in RAD50 format)	14
17	//////////	16
21	//////////	20
23	100001 for snagging assign/reassign;else0	22
25	//////////	24
27	//////////	26
31	device name (2 ASCII characters)	30 FQDEV
33	<>0,unit no. real device unit number	32 FQDEVN
35	//////////	34
37	//////////	36

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no meaningful data is returned with the UU.ASS subfunction of .UUO.

3.32.3 UU.ATR (Read/Write File Attributes)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3 FQFUN		UU.ATTR (= -25.)	2	
5		=0,read; 1-11,write channel number	4 FQFIL	
7		attribute data (used only for write)	6	
11			10	
13			12	
15			14	
17			16	
21			20	
23		number of words written = 1-11, as specified in byte 5.	22	
25		1 attribute per word	24	
27			26	
31			30	
33			32	
35			34	
37			36	

Data Returned:

Other than a possible error in byte 0 of the FIRQB, data is returned on a "read" only (byte 5 of data passed = 0).

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// current job no. * 2	2	FQJOB
5	//////////	4	
7	attribute data	6	
11	(If file has no attributes, FIRQB+6 and FIRQB+7 = 0)	10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35	name of run-time system	34	
37	(two words in RAD50 format)	36	

3.32.4 UU.ATT (Attach/Reattach Job)

Data Passed -- Attach:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.ATT (= 6.)	2	
5	(must = 0)	4	job no. to attach
7	project number	6	FQPPN
11		10	
13	password (2 words in RAD50 format)	12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Passed -- Reattach:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.ATT (= 6.)	2	
5	KB no. to attach to caller's job no.	4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Other than a possible error in byte 0 of the FIRQB, no data is returned by either the attach or reattach functions.

3.32.5 UU.BCK (Change File Statistics)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.BCK (= -11.)	2	
5	////////////// channel no.(1.-12.)	4	
7	desired date of last access*	6	
11	desired date of creation*	10	
13	desired time of creation*	12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

*The system internal format for dates is of the form:

(year - 1970 * 1000.) + day-within-year

Time is specified in minutes until midnight, 1440 = midnight. See the .DATE directive for a discussion of these formats.

Data Returned:

Other than a possible error in byte 0 of the FIRQB, no data is returned by the UU.BCK subfunction.

3.32.6 UU.BYE (Logout)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.BYE (= 5.)	2	
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

No data is returned with the UU.BYE subfunction; no errors are possible.

3.32.7 UU.CCL (CCL Command Add/Delete)

Data Passed -- Add:

	FIRQB	
Offset		Offset
Octal Mnemonic		Octal Mnemonic
1		0
3 FQFUN	UU.CCL (= -24.)	2
5	abbrev. point, chars must = 0 for add	4
7	project number programmer number	6 FQPPN
11		10 FQNAM1
13	file name of program to run (2 words in RAD50 format)	12
15	file extension (1 word in RAD50 format)	14 FQEXT
17	CCL command -- one to nine characters	16
21	padded with NULS (ASCII code 000)	20
23	to nine characters	22
25		24
27	(must = 0)	26
31	device name (2 ASCII characters)	30 FQDEV
33	<>0, unit no. real device unit number	32 FQDEVN
35	* line number to start execution	34
37		36

*Privilege indication in bit 15.

Data Passed -- Delete:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.CCL (= -24.)	2	
5	//////////////// = -2 for delete	4	
7	////////////////	6	
11	////////////////	10	
13	////////////////	12	
15	////////////////	14	
17	CCL command to be deleted	16	
21	(1 to 9 ASCII characters, padded with	20	
23	NULs (ASCII code 000 octal)	22	
25	to 9 characters	24	
27	////////////////	26	
31	////////////////	30	
33	////////////////	32	
35	////////////////	34	
37	////////////////	36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.CCL.

3.32.8 UU.CHE (Enable/Disable Disk Caching)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3 FQFUN		UU.CHE (= 19.)	2	
5		cache clustersize	4	subfunction code*
7		limit on total no. of cache clusters	6	
11		limit on clusters for directory caching	10	
13		limit on clusters for user data caching	12	
15		ctrls. sm. buffers	14	modifier for enabl/dil
17		//////////	16	
21		//////////	20	
23		//////////	22	
25		//////////	24	
27		//////////	26	
31		//////////	30	
33		//////////	32	
35		//////////	34	
37		//////////	36	

* 0 = enable directory and data caching

128 = return current caching parameters

1 = disable all caching

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	(internal coding)	2
5	cache clustersize currnt.cache setting	4
7	limit on total no. of cache clusters	6
11	limit on clusters for directory caching	10
13	limit on clusters for user data caching	12
15	ctrls. sm. buffers mod. for enable/dis.	14
17		16
21		20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

3.32.9 UU.CHU (Change Password/Quota, Disable Terminal, Kill Job)

Data Passed (Change Password/Quota):

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3 FQFUN		UU.CHU (= 8.)	2	
5			4	
7			6	
11		project number	10	programmer number
13			12	
15		new password	14	(2 words in RAD50 format)
17			16	number of blocks for quota; 0=unlimited
21			20	
23			22	
25			24	
27		=377 to change quota	26	
31		device name	30	FQDEV
33		(2 ASCII characters)	32	FQDEVN
35		<>0, unit no. real	34	device unit number
37		(must = 0)	36	

Data Passed (Disable terminal) :

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	keyboard no.
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35	(must = 377)	34	(must = 377)
37		36	

Data Passed (Kill job):

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.CHU (= 8.)	2	
5	job number to kill	4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35	(must = 377)	34	(must = 0)
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.CHU subfunction.

3.32.10 UU.CLN (Clean Up a Disk Pack)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.CLN (= 2.)	2
5			4
7			6
11			10
13			12
15			14
17			16
21			20
23			22
25			24
27			26
31	device name (2 ASCII characters)	30	FQDEV
33	<>0,unit no. real	32	FQDEVN
35	device unit number	34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned with the UU.CLN subfunction.

3.32.11 UU.DAL (Deassign All Devices)

Data Passed:

FIRQB

Offset Octal Mnemonic	FIRQB	Offset Octal Mnemonic
1		0
3 FQFUN	UU.DAL (= 12.)	2
5		4
7		6
11		10
13		12
15		14
17		16
21		20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

Data Returned:

No data is returned by the UU.DAL subfunction; no errors are possible.

3.32.12 UU.DAT (Change System Data/Time)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5	new current date*	4	
7	new current time*	6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

*The system internal format for date is:

$$((\text{year} - 1970) * 1000.) + \text{day-within-year}$$

Time is expressed as minutes until midnight, with 1440 = midnight. A value of 0 in either of these fields means no change is to be made. See .DATE directive for a discussion of these formats.

Data Returned:

No data is returned by the UU.DAT subfunction; no errors are possible.

3.32.13 UU.DEA (Deassign Device)

Data Passed:

FIRQB for Deassign Device

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.DEA (= 11.)	2	
5		4	
7		6	
11	(must = 0)	10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31	device name (2 ASCII characters)	30 FQDEV	
33	<>0, unit no. real device unit number	32 FQDEVN	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.DEA subfunction.

3.32.14 UU.DET (Detach)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	close flag & job no*
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

*Bit 7 = "close flag"

= 0, no close on terminal channels

= 1, close all channels on which terminal is open

Bits 0-6 = job number to detach. If 0, detach calling job.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.DET subfunction.

3.32.15 UU.DIE (Special Shutup Logout)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.DIE (= -16.)	2
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.DIE.

3.32.16 UU.DIR (Directory Lookup)

Data Passed -- Directory Lookup on Index:

FIRQB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1			0
3 FQFUN	UU.DIR (= 15.)	1	2
5	index of file to read	2	4
7	project number	3	FQPPN
11	programmer number	4	10
13		5	12
15		6	14
17		7	16
21		8	20
23		9	22
25		10	24
27		11	26
31	device name (2 ASCII characters)	12	FQDEV
33	<>0, unit no. real	13	device unit number
35		14	FQDEVN
37		15	34
		16	36

Data Returned -- Directory Lookup on Index:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		-----	0	
3		----- current job no. * 2	2	FQJOB
5		----- (same as data passed)	4	
7		----- project number programmer number	6	FQPPN
11		-----	10	FQNAM1
13		----- file name (2 words in RAD50 format)	12	
15		----- file extension (1 word in RAD50 format)	14	FQEXT
17		----- LSB (least significant bits) of file size	16	FQSIZ
21		----- MSB of file size protection code	20	
23		----- date of last access*	22	
25		----- date of creation*	24	
27		----- time of creation*	26	
31		-----	30	FQDEV
33		----- (same as data passed)	32	FQDEVN
35		----- file cluster size	34	FQCLUS
37		----- USTAT byte no. entries ret.	36	

*System internal format for date is:

((year -1970) * 1000.) + day-within-year

Time is returned as minutes before midnight, where 1440 = midnight.
See .DATE directive for a discussion of these formats.

Data Passed -- Special Magtape Directory Lookup:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3 FQFUN	UU.DIR (= 15.) index of file to read	2 4
7	(must = 177777 octal for magtape lookup)	6
11	//////////	10
13	//////////	12
15	//////////	14
17	//////////	16
21	//////////	20
23	//////////	22
25	//////////	24
27	//////////	26
31	MT or MM (2 ASCII characters)	30 FQDEV
33	<>0, unit no. real device unit number	32 FQDEVN
35	//////////	34
37	//////////	36

Data Returned -- Special Magtape Directory Lookup

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3		current job no. * 2	2	FQJOB
5		(same as data passed)	4	
7		(same as data passed)	6	FQPPN
11		file name (2 words in RAD50 format)	10	FQNAM1
13			12	
15		file extension (1 word in RAD50 format)	14	FQEEXT
17			16	
21		protection code	20	
23		date of creation*	22	
25			24	
27		project number programmer number	26	
31		(same as data passed)	30	FQDEV
33			32	FQDEVN
35			34	
37		USTAT byte no. entries ret.	36	

*System internal format for date is:

$$((\text{year} - 1970) * 1000.) + \text{day-within-year}$$

3.32.17 UU.DLU (Delete User Account)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1			0
3 FQFUN	UU.DLU (= 1.)		2
5			4
7			6
11	project number	programmer number	10
13			12
15			14
17			16
21			20
23			22
25			24
27			26
31	device name (2 ASCII characters)		30 FQDEV
33	<>0, unit no. real	device unit number	32 FQDEVN
35			34
37			36

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.DLU subfunction.

3.32.18 UU.ERR (Return Error Messages)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1			0
3	FQFUN	UU.ERR (= 9.)	2
5		/ error number 4	FQERNO
7		/ 6	
11		/ 10	
13		/ 12	
15		/ 14	
17		/ 16	
21		/ 20	
23		/ 22	
25		/ 24	
27		/ 26	
31		/ 30	
33		/ 32	
35		/ 34	
37		/ 36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	KB*2;l's comp=dtch.) current job no. * 2	2	FQJOB
5		4	
	error message -- padded with NULs to 28 characters (ASCII format)		
35		34	
37		36	

(No errors are possible with the UU.ERR subfunction).

3.32.19 UU.FCB (Get Open Channel Statistics (FCB/DDB))

Data Passed:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3 FQFUN	UU.FCB (= -8.)	2
5	chnl no. of FCB/DDB	4
7		6
11		10
13		12
15		14
17		16
21		20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	current job no. * 2	2	FQJOB
5	word 1 of either the FCB or DDB	4	
7	word 2 of either the FCB or DDB	6	
11	word 3 of either the FCB or DDB	10	
13	word 4 of either the FCB or DDB	12	
15	word 5 of either the FCB or DDB	14	
17	word 6 of either the FCB or DDB	16	
21	word 7 of either the FCB or DDB	20	
23	word 8 of either the FCB or DDB	22	
25	word 9 of either the FCB or DDB	24	
27	word 10 of either the FCB or DDB	26	
31	word 11 of either the FCB or DDB	30	
33	word 12 of either the FCB or DDB	32	
35	word 13 of either the FCB or DDB	34	
37	word 14 of either the FCB or DDB	36	

3.32.20 UU.FIL (File Placement and Modification)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3	UU.FIL (=-26.)		2	
5	function code	channel number	4	FQFIL
7	project number	programmer number	6	FQPPN
11			10	FQNAM1
13		file name (2 words in RAD50 format)	12	
15	file extension	(1 word RAD50)	14	FQEXT
17		least significant bits of block number	16	FQSIZ
21	MSB of block no.	placed/cache/seq.flags	20	
23		new date of last access	22	
25		new date of creation	24	
27		new time of creation	26	
31	device name	(2 ASCII characters)	30	FQDEV
33	<>0, unit number real	device unit number	32	FQDEVN
35			34	
37		new run-time system name (2 words in RAD50 format)	36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// current job no. * 2	2	FQJOB
5	device cluster number	4	
7		6	
11		10	
13		12	
15	file attribute data	14	
17	(filled with 0 words)	16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35	run-time system name	34	
37	(2 words in RAD50 format)	36	

3.32.21 UU.HNG (Hang Up a Dataset)

Data Passed:

FIRQB		
Offset Octal	Mnemonic	Offset Octal
1		0
3	FQFUN	UU.HNG (= -9.)
5		seconds to wait
7		KB no. of line
11		4
13		6
15		10
17		12
21		14
23		16
25		20
27		22
31		24
33		26
35		30
37		32
		34
		36

Data Returned:

No data is returned with a UU.HNG subfunction; no errors are possible.

3.32.22 UU.JOB (Job Creation)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3	FQFUN	UU.JOB (= 24.)	2	
5		must = 0 0 or 128.*	4	
7		project number programmer number	6	FQPPN
11			10	FQNAM1
13		file name (2 words in RAD50 format)	12	
15		file extension	14	FQEXT
17			16	
21		10 bytes of any information	20	
23		placed in created job's core common area	22	
25			24	
27			26	
31		device name (2 ASCII characters)	30	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35		parameter word (same as for .CCL)	34	
37			36	

* FIRQB+4=0 => create job only if logins are enabled.

FIRQB+4=200 (128.decimal) => create job even if logins are disabled.
Bits 0-6 reserved; must = 0.

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	/ / / / / / / / / / / / current job no. * 2	2	FQJOB
5	/ /	4	
7	/ /	6	
11	/ /	10	
13	/ /	12	
15	/ /	14	
17	/ /	16	
21	/ /	20	
23	/ /	22	
25	/ /	24	
27	/ /	26	
31	/ /	30	
33	/ /	32	
35	/ /	34	
37	/ /	36	

3.32.23 UU.LIN (Login)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	UU.LIN (= 4.)	2
5		4	
7		project number	programmer number
6		FQPPN	
11		10	
13		password (2 words in RAD50 format)	12
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	current job no. * 2	2	FQJOB
5	total number of jobs under this acc't	4	
7	det. job no. (2) det. job no. (1)	6	
11	(detached job numbers stop with 0 byte)	10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

3.32.24 UU.LOG (Set Number Of Logins)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	number allowed jobs
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	current job no. * 2	2	
5	actual logins set	4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

(No errors are possible with the UU.LOG subfunction.)

3.32.25 UU.LOK (Disk Directory Lookup by Filename/Wildcard Lookup)

Data Passed -- Lookup by File Name:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.LOK (= 17.)	1
5		(must = 177777 octal)	4
7		project number programmer number	6 FQPPN
11			10 FQNAM1
13		file name (2 words in RAD50 format)	12
15		file extension (1 word in RAD50 format)	14 QEEXT
17		1	16
21		1	20
23		1	22
25		1	24
27		1	26
31		device name (disk) (2 ASCII characters)	30 FQDEV
33		<>0, unit no. real device unit number	32 FQDEVN
35		1	34
37		1	36

Data Returned -- Lookup by File Name:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	current job no. * 2	2	FQJOB
5	(= 177777 octal)	4	
7	project number programmer number	6	FQPPN
11	file name (2 words in RAD50 format)	10	FQNAM1
13		12	
15	file extension (1 word in RAD50 format)	14	FQEXT
17	file length in 512-byte blocks	16	FQSIZ
21	protection code	20	
23	date of last access*	22	
25	date of creation*	24	
27	time of creation*	26	
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no. real device unit number	32	FQDEVN
35	file cluster size	34	FQCLUS
37	file identification index	36	

*System internal format for dates is:

((year -1970) * 1000.) + day-within-year

Time is in minutes until midnight, with 1440 = midnight. See the .DATE directive for a discussion of these formats.

Data Passed -- Disk Wildcard Directory Lookup:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3	FQFUN	LOKFQ (= 17.)	2	
5		index: n means search for n+1 occurrence	4	
7		project number	6	FQPPN
11		programmer number	10	FQNAM1
13		wild card file name specification	12	
15		(2 words in RAD50 format)	14	FQEEXT
17		//////////	16	
21		//////////	20	
23		//////////	22	
25		//////////	24	
27		//////////	26	
31		device name (disk) (2 ASCII characters)	30	FQDEV
33		<>0,device no. real	32	FQDEVN
35		device unit number	34	
37		//////////	36	

Data Returned -- Disk Wildcard Directory Lookup:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	current job no. * 2	2 FQJOB
5	(same as data passed)	4
7	project number programmer number	6 FQPPN
11	file name (2 words in RAD50 format)	10 FQNAML
13		12
15	file extension (1 word in RAD50 format)	14 FQEEXT
17	LSB of file length	16 FQSIZ
21	MSB of file length protection code	20
23	date of last access (disk only)	22
25	date of creation	24
27	time of creation	26
31	(same as data passed)	30 FQDEV
33		32 FQDEVN
35	file cluster size (disk only)	34 FQCLUS
37	USTAT byte	36

3.32.26 UU.MNT (Disk Pack and Terminal Status)

Data Passed:

FIRQB

	Offset Octal Mnemonic		Offset Octal Mnemonic
1			0
3 FQFUN	UU.MNT (=3.)		2
5	action flag*		4
7			6
11			10
13	pack identification label (2 words in RAD50 format)		12
15	=177777, use logical; =0, use pack ID		14
17			16
21	logical name for disk (2 words in RAD50 format)		20
23	mode word		22
25			24
27			26
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no. real device unit number	32	FQDEVN
35			34
37			36

*byte 4 = any odd Set terminal status (see UU.TRM for remaining bytes.)
 = 0 Mount a disk pack or cartridge
 = 2 Dismount a disk pack or cartridge
 = 4 Lock out a disk pack or cartridge
 = 6 Unlock a disk pack or cartridge

Data Returned (Mount Only):

Data is returned only for the disk mount operation (byte 4 = 0 in the data passed).

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	current job no. * 2	2 FQJOB
5		4
7		6
11		10
13		12
15		14
17		16
21	logical name for disk (2 words in RAD50 format)	20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

3.32.27 UU.NAM (Associate a Run-time System with a File)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5	UU.NAM (= -17.)	4	run-time channel number
7		6	system name (2 words in RAD50)
11		10	format
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Other than a possible error in byte 0 of the FIRQB, no data is returned by the UU.NAM subfunction.

3.32.28 UU.NLG (Disable Further Logins)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// current job no. * 2	2	FQJOB
5	////////// no. jobs on system	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	//////////	14	
17	//////////	16	
21	//////////	20	
23	//////////	22	
25	//////////	24	
27	//////////	26	
31	//////////	30	
33	//////////	32	
35	//////////	34	
37	//////////	36	

(No errors are possible with the UU.NLG subfunction.)

3.32.29 UU.PAS (Create User Account)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11	project number programmer number	10	
13	password (2 words in RAD50 format)	12	
15		14	
17	disk quota (max. number of blocks)	16	
21		20	
23		22	
25		24	
27		26	
31	device name (2 ASCII characters)	30	FQDEV
33	<>0, unit no. real device unit number	32	FQDEVN
35	UFD cluster size	34	FQCLUS
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.PAS subfunction.

3.32.30 UU.POK (Poke Memory)

The UU.POK subfunction can be executed only by a job running on account [1,1].

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5	address of word to be changed	4	
7	new contents of word	6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.POK subfunction.

3.32.31 UU.PPN (Wildcard PPN Lookup)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		/ /	0	
3	FQFUN	UU.PPN (=25.) / / / / / / / / / / / /	2	
5		index: n means search for n+1 occurrence	4	
7		255. or project no. 255. or progr. no.	6	
11		/ /	10	
13		/ /	12	
15		/ /	14	
17		/ /	16	
21		/ /	20	
23		/ /	22	
25		/ /	24	
27		/ /	26	
31		device name (2 ASCII characters)	30	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35		/ /	34	
37		/ /	36	

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3	current job no. * 2	2 FQJOB
5	(internal code)	4
7	project number programmer number	6 FQPPN
11		10
13		12
15		14
17		16
21		20
23		22
25		24
27		26
31	(same as data passed)	30
33	(same as data passed)	32
35		34
37		36

3.32.32 UU.PRI (Change Priority/Run Burst/Size)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3	FQFUN	UU.PRI (= -13.)	2	
5		<>0,change run pri. job no.;377=caller	4	
7		<>0,change run brst. new run priority	6	
11		<>0,change size new run burst	10	
13		maximum size	12	
15			14	
17			16	
21			20	
23			22	
25			24	
27			26	
31			30	
33			32	
35			34	
37			36	

Data Returned:

No data is returned with the UU.PRI subfunction; no errors are possible.

3.32.33 UU.RAD (Read or Read-and-Reset Accounting Data)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.RAD (= 14.)	2
5		index number of account; if 0, use ppn	4
7		0 => read; <>0 => read and reset	6
11		project number programmer number	10
13			12
15			14
17			16
21			20
23			22
25			24
27			26
31		device name (disk) (2 ASCII characters)	30
33		<>0, unit no. real device unit number	32
35			34
37			36

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	/ /	0	
3	/ / / / / / / / / / / current job no. * 2	2	FQJOB
5	(same as data passed)	4	
7	number of blocks owned by account read	6	
11	ppn of account read	10	
13		12	
15	password of account read (2 words in RAD50 format) if caller privileged	14	
17	low-order 16 bits of CPU time (.1 secs)	16	
21	connect time (minutes)	20	
23	low-order 16 bits of KCTs	22	
25	device time (minutes)	24	
27	MSB of CPU time MSB of KCTs	26	
31	(same as data passed)	30	
33	(same as data passed)	32	
35	disk quota in blocks; 0 = unlimited	34	
37	user file directory cluster size	36	

3.32.34 UU.RTS (Add/Remove/Load/Unload Run-Time System or Resident Library)

Data Passed -- Add a Run-Time System:

FIRQB -- Add a Run-Time System

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.RTS (= -18.)	2
5		///////////////// = 0 for add RTS	4
7		project number programmer number	6 FQPPN
11			10 FQNAM1
13		run-time system name (2 words of RAD50)	12
15		=n, n*1K starts load; =0, Monitor decides	14
17		max user job image size, K words (P.SIZE)	16
21		min user job image size, K words (P.MSIZ)	20
23		stay flag linked-list position	22
25		flag word (P.FLAG)	24
27		dflt. exec. file ext. (P.DEXT), RAD50	26
31		device name (disk) where stored	30
33		<>0, unit no. real device unit number	32
35		/////////////////	34
37		/////////////////	36

NOTE: The word at FIRQB+14 must be some number n for a read/write run-time system.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Remove a Run-Time System:

FIRQB -- Remove a Run-Time System

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.RTS (-18.)	2
5		1 ////////////// = 4 for remove RTS	4
7		1 ///////////////////////////////	6
11		1 run-time system name (2 words in RAD50 format)	10 FQNAM1
13			12
15		1 ///////////////////////////////	14
17		1 ///////////////////////////////	16
21		1 ///////////////////////////////	20
23		1 ///////////////////////////////	22
25		1 ///////////////////////////////	24
27		1 ///////////////////////////////	26
31		1 ///////////////////////////////	30
33		1 ///////////////////////////////	32
35		1 ///////////////////////////////	34
37		1 ///////////////////////////////	36

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Load a Run-Time System:

FIRQB -- Load a Run-Time System

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	1 UU.RTS (= -18.)	2
5		1 ////////////// = 2 for load RTS	4
7		1 //////////////	6
11		1	10 FQNAM1
13		run-time system name (2 words RAD50)	12
15		1 =n, n*1K load starts; =0, use add's	14
17		1 //////////////	16
21		1 //////////////	20
23		1 stay flag*	22
25		1 //////////////	24
27		1 //////////////	26
31		1 //////////////	30
33		1 //////////////	32
35		1 //////////////	34
37		1 //////////////	36

* = 128. => permanently resident

= 0 => memory occupied by this run-time system can be released when usage count goes to 0.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Unload a Run-Time System:

FIRQB -- Unload a Run-Time System

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	UU.RTS (= -18.)	2
5		=6 for unload RTS	4
7		6	
11		10	FQNAM1
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Add a Resident Library:

FIRQB -- Add a Resident Library

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3 FQFUN	UU.RTS (= -18.)	2
5	=16. for add R. Lib.	4
7	project number programmer number	6 FQPPN
11		10 FQNAM1
13	resident library name (2 words RAD50)	12
15	=n,where n*1Kword = address to begin load	14
17		16
21		20
23	stay flag*	22
25	flag word**	24
27		26
31	device name (2 ASCII characters)	30 FQDEV
33	<>0, unit no. real device unit number	32
35		34
37		36

*stay flag = 128. => permanently resident
 = 0 => remove when usage count goes to zero

**flag word bit settings indicate resident library characteristics:

Bit 9 = 1 (Decimal value = 512.) The resident library is to be accessed by only one user. That is, the library is not to be shared.
 = 0 The resident library can be shared by more than one user.

Bit 10 = 1 (Decimal value = 1024.) Read/write access is allowed.
 = 0 Read-only access.

Bit 12 = 1 (Decimal value = 2048.) Errors occurring within code in the resident library are not to be recorded in the system error log.
 = 0 Errors may be recorded in the system error log.

- Bit 13 = 1 (Decimal value = 4096.) The resident library should be immediately removed from memory when its usage count goes to zero.
= 0 When the usage count for this library goes to zero, the Monitor will remove it from memory when the space it takes is needed for something else.

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Remove a Resident Library:

FIRQB -- Remove a Resident Library

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11		10	FQNAM1
13	resident library name (2 words RAD50)	12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Load a Resident Library:

FIRQB -- Load a Resident Library:

Offset	Octal Mnemonic	Offset	Octal Mnemonic
1		0	
3 FQFUN	UU.RTS (= -18.)	2	
5	=18. for load lib.	4	
7		6	
11	resident library name (2 words RAD50)	10	FQNAM1
13		12	
15	n, where n*1Kword = address load begins	14	
17		16	
21		20	
23	stay flag*	22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

*stay flag = 128. => permanently resident

= 0 => released when usage count goes to zero

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

Data Passed -- Unload a Resident Library:

FIRQB -- Unload a Resident Library

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11		10	FQNAM1
13	resident library name (2 words RAD50)	12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.RTS.

3.32.35 UU.SLN (System Logical Names)

Data Passed -- Add New Names:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		/ /	0	
3	FQFUN	UU.SLN (= 21.) / / / / / / / / / / / / / / / / / /	2	
5		/ = 1 for add	4	
7		/ ppn for this name (if 0, none associated)	6	FQPPN
11		/ logical name (2 words in RAD50 format)	10	FQNAM1
13		/ /	12	
15		/ /	14	
17		/ /	16	
21		/ /	20	
23		/ /	22	
25		/ /	24	
27		/ /	26	
31		/ device name (2 ASCII characters)	30	FQDEV
33		/ must <> 0 device unit number	32	FQDEVN
35		/ /	34	
37		/ /	36	

Data Passed -- Remove Logical Names:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1		0
3 FQFUN	UU.SLN (= 21.)	2
5	= 0 for remove	4
7		6
11	logical name (2 words in RAD50 format)	10 FQNAM1
13		12
15		14
17		16
21		20
23		22
25		24
27		26
31		30
33		32
35		34
37		36

Data Passed -- Change Disk Logical Name:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1			0	
3 FQFUN	UU.SLN (= 21.)		2	
5		= 377 for change	4	
7			6	
11			10	FQNAM1
		new logical name (2 words in RAD50 fmt.)	12	
13				
15			14	
17			16	
21			20	
23			22	
25			24	
27			26	
31		device name (2 ASCII characters)	30	FQDEV
33		must <> 0 device unit number	32	FQDEVN
35			34	
37			36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no meaningful data is returned by the UU.SLN subfunction.

3.32.36 UU.SWP (Add/Remove System Files)

Data Passed -- Add System File:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3	FQFUN	UU.SWP (= 23.)	2	
5		= 1 for add	4	
7		file to add*	6	
11		for [0,1] file with .SYS extension,	10	FQNAM1
13		filename as 2 words RAD50. If both		
		words = 0, non-file structured disk.	12	
15		//////////	14	
17		//////////	16	
21		//////////	20	
23		//////////	22	
25		//////////	24	
27		//////////	26	
31		device name (disk) (2 ASCII characters)	30	FQDEV
33		<>0, unit no. real device unit number	32	FQDEVN
35		//////////	34	
37		//////////	36	

* byte 4 = 0 swapping file slot 0
 1 swapping file slot 1
 2 illegal -- generates error, as slot 2 is always added
 3 swapping file slot 3
 4 overlay file
 5 error message file

Data Passed -- Remove System File:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.SWP (= 23.)	2	
5	= 0 for remove	4	file to remove*
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

*byte 4 = 0 swapping file slot 0
= 1 swapping file slot 1
= 2 illegal -- generates error, as slot 2 is always added
= 3 swapping file slot 3
= 4 overlay file
= 5 error message file

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by the UU.SWP subfunction.

3.32.37 UU.SYS (Return Job Status Information)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3 FQFUN	UU.SYS (= 26.)	2	
5	subcode = 0 or 1	4	job number or 0
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned (for subcode = 0):

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// calling job no. * 2	2	
5	swap slot location controlling job no*2	4	
7	least significant word of CPU time,.1 sec	10	
11	job's current connect time, in mins.	12	
13	least significant word of KCTs	14	
15	job's accumulated device time, mins.	16	
17	MSB of CPU time MSB of KCTs	20	
21	job name (2 words RAD50)	22	
23		24	
25	project number programmer number	26	
27	default run-time system (2 words	30	
31	RAD50)	32	
33	current run-time system (2 words	34	
35	RAD50)	36	

Data Returned (for subcode = 1):

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// calling job no * 2	2	
5	KB number job no * 2	4	
7	current flag word	6	
11	curr.info. posting current IOSTS	10	
13	current JBSTAT word	12	
15	current JBWAIT word	14	
17	mem. cntrl sublock size, in K words	16	
21	current physical addr., 32-word increm.	20	
23	run burst priority	22	
25	value at offset 6 max. memory size	24	
27	(depends on JBWAIT and JBSTAT)	26	
31	read or write state	30	
33	pointer to Job Data Block	32	
35	pointer to second Job Data Block	34	
37	pointer to Receiver ID Block	36	

3.32.38 UU.TB1 (Get Monitor Tables -- Part I)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

FIRQB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3	////////// current job no. * 2	2 FQJOB
5	MAXCNT(max job no.) CNT.KB-1(max KB no.)	4
7	(DEVCNT) address of max. unit no. table	6
11	(DEVPTR) address of pointers to dev. DDBs	10
13	(MEMLST) rt lnk wrd in lsrt mem ctrl sb1k	12
15	(JOBTBL) address of the job table	14
17	(JBSTAT) address of job status table	16
21	(JBWAIT) address of job wait flag table	20
23	(UNTCLU) add. of unit cluster/err cnt tb1	22
25	(UNTCNT) address of disk status table	24
27	(SATCTL) address of free-block table	26
31	(JSBTBL) add. of job stat. ordered by drv	30
33	(SATCTM) add. of free block count table	32
35	current date in system internal format*	34
37	(UNTOWN) add. of disk owner/option table	36

* System internal format for date is:
 $((year - 1970) * 1000.) + day-within-year$

(No errors are possible with UU.TB1.)

3.32.39 UU.TB2 (Get Monitor Tables, Part II)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// current job no. * 2	2	FQJOB
5	(FREES) add. of free buffer info table	4	
7	(DEVNAM) add. of device name table	6	
11	(CSRTBL) add. of CSR table	10	
13	(DEVOKB) no. disks in DEVNAM times 2	12	
15	(TTYHCT) no. hung term. errs since strtup	14	
17	(JOBCNT) no. jobs now/no. logins allowed	16	
21	(RTSLST) rt. lnk.word in RTS dsc.blk.list	20	
23	(ERLCTL) error logging control data	22	
25	(SNDLST) list of eligible msg. rec. jobs	24	
27	(LOGNAM) table of system logical names	26	
31	(DEVSYN) start of synonym names in DEVNAM	30	
33	(MEMSIZ) physical memory size,Kwrds*32	32	
35	(CCLLST) root link word of CCL desc. blks.	34	
37	(FCBLST) add. of table of roots of FCBs	36	

(No errors are possible with UU.TB2.)

3.32.40 UU.TRM (Set Terminal Characteristics)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3	FQFUN	UU.TRM (= 16.)	2	
5		377(current) or KB no	4	
7		TAB/NO TAB	6	WIDTH: 0 or width+1
11		LC OUTPT/NO LC OUTPT	10	NO FORM/FORM
13		FULL DPLX/LOCAL ECHO	12	NO XON/XON
15		NO LC INPT/LC INPUT	14	NO SCOPE/SCOPE
17		xmt baud rt;377=2741	16	NO FILL/FILL
21		rcv baud rt;377=2741	20	NO PARITY/EVEN/ODD
23		UP ARROW/NO UP ARROW	22	NO STALL/STALL
25		2741 byte	24	
27		NO ESC SEQ/ESC SEQ	26	377= /RING
31		NO ESC/ESC	30	DELIMITER/DELIM x
33		RESUME ANY/CTRL-C	32	CTRL/R,CTRL/T flag
35		//////////	34	
37		//////////	36	

Data Returned (Current Keyboard Characteristics):

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		0		
3	current job no. * 2	2	FQJOB	
5	377(current) or KB no	4		
7	TAB/NO TAB WIDTH: 0 or width+1	6		
11	LC OUTPT/NO LC OUTPT NO FORM/FORM	10		
13	FULL DPLX/LOCAL ECHO NO XON/XON	12		
15	NO LC INPT/LC INPUT NO SCOPE/SCOPE	14		
17	xmt baud rt;377=2741 NO FILL/FILL	16		
21	rcv baud rt;377=2741 NO PARITY/EVEN/ODD	20		
23	UP ARROW/NO UP ARROW NO STALL/STALL	22		
25	(always 0) interface for line	24		
27	NO ESC SEQ/ESC SEQ 377= /RING	26		
31	NO ESC/ESC DELIMITER/DELIM x	30		
33	RESUME ANY/CTRL-C CTRL/R,CTRL/T flag	32		
35		34		
37		36		

3.32.41 UU.YLG (Enable Logins)

Data Passed:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3	FQFUN	2	UU.YLG (= -1.)
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	
21		20	
23		22	
25		24	
27		26	
31		30	
33		32	
35		34	
37		36	

Data Returned:

FIRQB

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	//////////	0	
3	////////// current job no. * 2	2	FQJOB
5	////////// no. logins allowed	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	//////////	14	
17	//////////	16	
21	//////////	20	
23	//////////	22	
25	//////////	24	
27	//////////	26	
31	//////////	30	
33	//////////	32	
35	//////////	34	
37	//////////	36	

(No errors are possible with UU.YLG.)

3.32.42 UU.ZER (Zero Device)

Data Passed:

FIRQB

Offset Octal	Mnemonic		Offset Octal	Mnemonic
1		//////////	0	
3	FQFUN	UU.ZER (= 13.)	2	
5		//////////	4	
7		project number	6	FQPPN
11		volume ID for label (ANSI fmt MT only) (2 words in RAD50 format)	10	
13			12	
15		//////////	14	
17		//////////	16	
21		//////////	20	
23		//////////	22	
25		//////////	24	
27		//////////	26	
31		device name (disk, magtape, or DECtape)	30	FQDEV
33		<>0, unit no. real	32	FQDEVN
35		//////////	34	
37		//////////	36	

Data Returned:

Except for a possible error in byte 0 of the FIRQB, no data is returned by UU.ZER.

3.33 .WRITE -- Write Data to File or Device

Form:

.WRITE

Function:

The .WRITE directive writes a specified number of bytes of data from a user buffer (defined in the XRB) to a file or device currently open on a channel. Unless an error occurs, the .WRITE always transfers the number of bytes specified. The number of bytes specified must be less than or equal to the size of the buffer. Specific details for each device are given in Table 3-5. The "best guess" buffer sizes (returned by the Monitor at FIRQB+FQBUFL when the device was opened) are also shown in Table 3-5 for comparison.

TABLE 3-5
DATA OUTPUT WITH .WRITE

Device	Block size, bytes	"Best guess*", bytes	Restrictions on Number of Bytes Written
Byte-Oriented Devices (FLGFRC=1, FLGRND=0)			
Keyboard (Terminal)	N/A	128.	None
Pseudo-Keyboard	N/A	128.	None
Paper Tape Punch	N/A	128.	None
Line Printer	N/A	128.	None
Block-Sequential Devices (FLGFRC=0, FLGRND=1)			
Magtape	18. to 30,000.	512.	14. - 32767. bytes
DECtape (file-structured)	510.	510.	Must be <= 510. bytes; if <510., automatically filled with NULs to 510. bytes.
Block-Random Devices (FLGFRC=0, FLGRND=0)			
Disk	512.	512.	Must be multiple of 512. bytes.
Floppy Disk	512. (block mode) 128. (RX01 or RX02 single-density sector mode) or 256. (RX02 double-density sector mode)	512.	If not a multiple of block/sector size, 0 fill to end of block/sector.
DECtape (non-file structured)	512.	512.	Must be <= 512. bytes; if <512., automatically filled with NULs to 512. bytes.

* Returned at FIRQB+FQBUFL when file or device was opened.

Data Passed:

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	length of output buffer, in bytes	0 XRLEN
3	number of bytes to be written	2 XRBC
5	starting address of buffer	4 XRLOC
7 XRBLSK	MSB of block number channel no. * 2	6 XRCI
11	LSB of block number	10 XRBLSK
13	//////////	12
15	optional modifier	14 XRMOD

XRB+XRLEN Length of the output buffer, in bytes. The value of this word must be non-zero.

XRB+XRBC The number of bytes to be written. The value of this word must be non-zero and less than or equal to the buffer size, as specified at offset XRB+XRLEN. For disk, this must be some multiple of 512 (decimal).

XRB+XRLOC Starting address of the output buffer. The buffer, as defined by XRLOC for its start, and XRLOC+XRLEN-1 for its last byte, must lie wholly within either the user job image (low segment), or the run-time-system's address space (high segment).

If the buffer is in the low segment, the address defined by the contents of XRB+XRLOC must be greater than 170 (octal) to avoid destroying job-context information used in swapping the job (Section 2.4).

If the buffer is in the high segment, it must not fall within the pseudo-vector region. That is, it must not fall above the location P.OFF (Section 2.5). Also, the run-time system must currently be mapped read/write (see PF.RW bit description in P.FLAG word, Section 2.5).

XRB+XRCI Channel number times two; defines the channel on which the data is to be written, as previously defined in an open (OPNFO, CREFQ, CRTFQ, CRBFQ functions of CALFIP, Section 3.2).

XRB+XRBLSK For random files greater than 65,535 bytes in length, this byte contains the most significant bits of the starting block number where the write is to begin. This byte is combined with the word at XRB+XRBLSK (below) to form a 24-bit field.

XRB+XRBLK Starting block number where the write is to begin. Performs the same action as the BLOCK option for disk, or the RECORD option for floppy disk and non-file-structured DECTape, as described in the RSTS/E Programming Manual. This parameter is ignored if the device is not a random-access device. If the device is random-access, and this field is non-zero, it is interpreted as the block number where the write is to start (1 to n, where n is the length of the file in 512-byte blocks). If zero, the next sequential block is written.

For disk devices opened as non-file-structured, this value is the starting device cluster number where the write is to begin.

XRB+XRMOD Output operation modifier; significant only for line printer or terminal devices. (The Monitor informs you with the FLGMOD bit of the flag word whether or not the device will accept modifiers.) This parameter performs the same action as the RECORD modifier in BASIC-PLUS for these devices, as described in the RSTS/E Programming Manual.

Data Returned:

XRB

Offset Octal Mnemonic		Offset Octal Mnemonic
1	//////////	0
3	//////////	2
5	//////////	4
7 XRBLSK	MSB of block number	6
11	block number where write began (random)	10 XRB+XRBLK
13	//////////	12
15	//////////	14

XRB+XRBLSK For large disk files (greater than 65,535 bytes), the most significant bits of the block number of the block just written. (Combined with XRB+XRBLK to form a 24-bit field.)

XRB+XRBLK For random-access devices, this word contains the block number of the block just written. Block number range from 1 to n, where n is the length of the file, in 512-byte blocks. They define the order in which the file is written.

For disk devices opened as non-file-structured, this is the device cluster number of the cluster just written.

Errors:

BADCNT The first three words of the XRB that describe the output buffer are illegal. (An illegal byte count.)

BSERR The specified channel number is illegal; that is, not in the range 0 - 30 (decimal). (Since channel is specified as channel * 2.)

NOTOPN No device is currently open on the specified channel.

PRVIOL The file or device currently open on the specified channel is read-only, or the open for this file or device did not specify write access.

All other errors are device-dependent. Common errors are:

DATERR Some data error occurred. This error is issued when a parity error occurs, etc.

HNGDEV Some hard device I/O error occurred. For example, the

specified device is off-line, is physically write-locked, etc.

NOROOM No more room is available on the device. For example, the end-of-tape marker has been encountered, no more available disk space, etc.

Example:

The following code writes the contents of a buffer to the file open on channel 5.

```
MOV      #1024.,XRB+XRLEN      ;SET BUFFER LENGTH
MOV      #1024.,XRB+XRBC      ;SET BYTES TO WRITE
MOV      #BUFFER,XRB+XRLOC    ;SET BUFFER ADDRESS
MOVB    #5*2,XRB+XRCI        ;SET CHANNEL = 5
.WRITER
```



CHAPTER 4

RSX RUN-TIME SYSTEM ENVIRONMENT

4.1 INTRODUCTION

In Chapter 1, the RSX Run-Time System was described as emulating the RSX operating system environment on RSTS/E. This is only partially true; the RSX Run-Time System does not provide a real-time, multiprocessing environment to users of a RSTS/E time-sharing system. The RSX Run-Time System, and its derivatives -- BASIC2, BP2COM, RMS11 -- do, however, process directives that are identical in form, and similar in objective, to a subset of the RSX-11M Monitor directives described in the RSX-11M Executive Reference Manual. The part of the RSX Run-Time System that handles these directives is called the RSX Emulator.

If you use the RSX directives described in Chapter 5, you must assemble your program with the MAC assembler, and link the modules with the task builder (TKB). The program can then be run as a user job image under the control of the RSX run-time system or its derivatives.

4.1.1 Advantage: Transportable Code

The RSX Emulator directives are useful if you are coding a program to be run under both the RSX-11M and RSTS/E operating systems. Not all RSX-11M Executive directives are emulated, however, and the meaning of those that are emulated are fitted to the RSTS/E environment.

So, to design a program to be run under both operating systems, you need to know how the directives work under RSX-11M and under RSTS/E. This document describes what the directives do under RSTS/E. Some comparison is made here to RSX-11M, but you must refer to the RSX-11M Executive Reference Manual for a complete description of how these directives work in the RSX-11M environment.

Another benefit of using compatible directives is that programs may be executed on VAX/VMS under the RSX-11M Application Migration Executive (AME), which is similar in function to the RSX Emulator. AME emulates most RSX-11M Executive directives.

4.1.2 General Services

Besides transportable code, the RSX Emulator directives provide:

1. Non-File-Structured Input/Output. The RSX Emulator directives handle only non-file-structured I/O (for example, terminal I/O). For this type of I/O, using the directives will take less memory than using RMS*. RMS routines are available to MACRO programmers through resident libraries, or they can be linked as part of the user job image. In either of these cases, the RMS routines take space in the job area. The RSX Emulator directives that do I/O take no extra space, since the code to handle the processing is a part of the run-time system.
2. Trap Handling. The RSX Emulator includes processing to handle synchronous and asynchronous system traps. In most cases, the Emulator aborts the program when such traps occur, and prints an error message at the job's terminal. With certain of the directives, you can request that this processing be bypassed, to handle such traps in your program. For example, you can specify an address to which control is to be passed when a Floating Point Processor (FPP) exception trap occurs.

4.1.3 RSX Directive Emulation within RSTS/E Monitor

Prior to RSTS/E V7.0, the RSX Emulator has been an integral part of the RSX Run-Time System or its derivatives. In Version 7, there is an option at System Generation time to put RSX Emulation capability into the RSTS/E Monitor. This allows task images to execute without a Run-Time System, because the Monitor handles the RSX Emulation directly. As a result, task images may expand to be as large as 31K words, or else a Resident Library (for example, RMS) may be used which occupies the address space otherwise taken by the Run-Time System.

Task images which are to execute using the RSX Emulator in the Monitor must be built to run with the RSX Run-Time System. This Run-Time System will perform the initial task image loading, set up the low-core parameters needed for RSX Emulation, and then invoke the RSTS/E Monitor via the .RSX directive. Since RSTS/E jobs are never without a Run-Time System, the Monitor associates the job with a zero-length Run-Time System called "...RSX" for the duration of the task's execution. Upon termination of the task, the job exits to its default Run-Time System.

If the SYSGEN option to put the RSX Emulator in the Monitor is not used, or if a task runs with a Run-Time System other than RSX, (one of the RSX derivatives, such as BP2COM, for example) the RSX Emulation is done by the Run-Time System, as with previous releases.

*See the RMS-11 MACRO-11 Reference Manual for a description of RMS (Record Management Services) capabilities for MACRO programmers.

4.2 SYSTEM MACRO LIBRARY

The RSX Emulator directives that you code into your program are macro calls; they must be expanded into executable code at assembly time. The macro expansions for the RSX Emulator directives are contained in the system macro library -- LB:RSXMAC.SML. This library file can be named in the input-file list when the assembly is done. For example:

```
MAC OBJ,OBJ=SRC1,SRC2,LB:RSXMAC.SML/ML
```

However, the MAC assembler searches the library automatically to resolve undefined symbols, so this is not really necessary.

In your code, though, you must use the MACRO-11 .MCALL directive to define the directives you use as external macros needed to assemble the source program. The .MCALL must appear before the first directive is called. For example:

```
.MCALL ALUN$,QIO$  
.  
.  
.  
ALUN$ 2,TT,5  
.  
.  
.
```

4.3 DIRECTIVE PROCESSING

The RSX directives are expanded at assembly time to code which defines the action to be done and the parameters to be used in a Directive Parameter Block (DPB). Figure 4-1 shows the general form of a DPB.

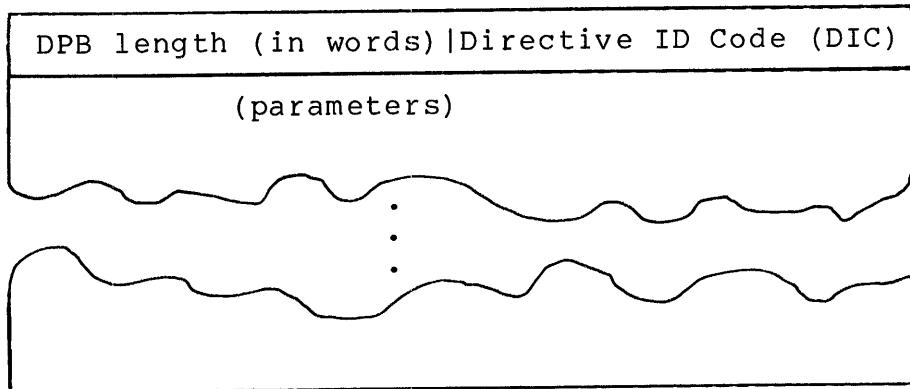


Figure 4-1 General Form of the Directive Parameter Block (DPB)

The Directive Identification Code (DIC) in the low byte of the first word of the DPB defines the particular function to be performed. The RSX Emulator examines the DIC to determine what it is supposed to do. The high byte of the first word of the DPB gives the length of the DPB in words (including the DIC and length byte). The remaining words contain parameters; usually these are values specified by the user in the directive call, expanded via MACRO-11 data definition macros such as .BYTE, .WORD, .RAD50, and so forth.

The directive expansions also include executable code to tell the Emulator where the DPB is, an EMT 377 instruction to pass control to the Emulator, and, optionally, transfer control to an error processing routine specified by the user.

At execution time, either the DPB address or the DPB itself is pushed on the stack. The choice depends on the form of the directive, as described in Section 4.4. The EMT 377 instruction passes control to the RSTS/E Monitor. The Monitor recognizes the EMT 377 and passes control to the RSX Run-Time System*. The RSX Run-Time System examines the DPB and processes the directive. It then pops the DPB address or the entire DPB off the stack, and either returns control to the program which executed the directive, or, if the directive was an exit of some kind, to whatever location is appropriate to the exit.

When control is returned in-line, the Carry condition code in the program status word (PSW) indicates that the directive executed successfully (carry code = 0) or failed (carry code = 1). Further information on the success or failure of the directive is given in the Directive Status Word. You can refer to the Directive Status Word in your program with the mnemonic \$DSW; this variable is defined and its location is resolved automatically at link time by the task builder (TKB). Other mnemonics are similarly related to the possible octal values of \$DSW, so that the MACRO programmer can test for error conditions using symbols such as IS.SUC, for successful completion. The symbols appropriate to each directive are listed under the heading "DSW Return Codes" in Chapter 5.

*Unless the system was generated with RSX Emulation in the Monitor.

4.4 DIRECTIVE FORMS -- \$, \$C, \$S -- AND THEIR EXPANSIONS

The library file RSXMAC.SML contains three different expansions for each of the RSX-11M directives listed in Chapter 5. The form of the directive name you use determines which of the three expansions will be inserted in your code. Directive names consist of up to four letters, followed by a dollar sign (\$) and, optionally, a C or an S. Figure 4-2 shows an example of the three forms for the ALUN\$ directive (assign logical unit).

	YOUR CODE	EXPANSION	DESCRIPTION
\$ FORM	<pre>A: ALUN\$ 5,TT,0</pre> <pre>.</pre>	<pre>A: .BYTE 7,4 Create DPB at assembly .WORD 5 time, in data section .ASCII /TT/ .WORD 0</pre> <pre>.</pre>	<pre>Create DPB at assembly time, in data section of program</pre>
\$C FORM	<pre>DIR\$ #A,ERR</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre>	<pre>MOV #A,-(SP) Push DPB address on stack EMT 377 Pass control to RSX Eml. BCC .+6 Branch if no error JSR PC,ERR Jump to ERR on error</pre>	<pre>Push DPB address on stack Pass control to RSX Eml. Branch if no error Jump to ERR on error</pre>
\$S FORM	<pre>ALUN\$C 5,TT,0,PSEC1,ERR</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre>	<pre> .PSECT \$DPB\$\$ Create DPB at assembly \$ \$\$=. time, automatically put in separate PSECT called .BYTE 7,4 \$DPB\$\$.WORD 5 .ASCII /TT/ .WORD 0 .PSECT PSEC1 Re-enter specified PSECT MOV \$\$\$,-(SP) Push DPB address on stack EMT 377 Pass control to RSX Eml. BCC .+6 Branch if no error JSR PC,ERR Jump to ERR on error</pre>	<pre>Re-enter specified PSECT Push DPB address on stack Pass control to RSX Eml. Branch if no error Jump to ERR on error</pre>
	<pre>ALUN\$S #5,#"TT,#0,ERR</pre> <pre>.</pre> <pre>.</pre> <pre>.</pre>	<pre>MOV #0,-(SP) Create DPB at execution MOV #"TT,-(SP) time, push entire DPB on MOV #5,-(SP) stack MOV (PC)+,-(SP) .BYTE 7,4 EMT 377 BCC .+6 JSR PC,ERR</pre>	<pre>Create DPB at execution time, push entire DPB on stack</pre>
	<pre>.</pre>		<pre>Pass control to RSX Eml.</pre>
	<pre>.</pre>		<pre>Branch if no error</pre>
	<pre>.</pre>		<pre>Jump to error routine</pre>

Figure 4-2 Example of RSX Directive Forms

4.4.1 \$ Form (and DIR\$ Directive)

As shown in Figure 4-2, the \$ form of an RSX Emulator directive simply generates a DPB in-line. The DIR\$ directive expands to code which pushes the address of the DPB on the stack, and passes control to the RSX Emulator. The \$ form/DIR\$ combination is useful when the same directive is to be executed repeatedly. One DPB is defined, and then invoked by as many DIR\$ directives as needed. This saves memory. Also, the individual parameters in the DPB can be modified, when the same directive is used many times with varying parameters.

\$ Form

The \$ form of an RSX Emulator directive expands to code which defines the directive parameter block (DPB). The expansions consist of MACRO-11 data storage directives (for example, .BYTE, .WORD, .RAD50); they are not executable. Therefore, use the \$ form of an RSX Emulator directive only in a data section of your program.

Finally, use parameters which fit the MACRO-11 data storage directive in which they will appear in the expanded code. For example,

```
ALUN$ 5,TT,0
```

expands to:

```
.BYTE 7,4  
.WORD 5  
.ASCII /TT/  
.WORD 0
```

Although the two-character second parameter (TT) is used in a MACRO-11 .ASCII directive, you would not need to specify the enclosing slashes; the expansion does that for you.

Note that the first word of the expansion (the first word of the DPB) consists of two bytes. The octal 7 in the low byte is the directive identification code (DIC) byte for the ALUN\$ directive. The octal 4 in the high byte indicates that the DPB is 4 words long.

DIR\$ Directive

The general form of the DIR\$ directive is

```
DIR$ [adr] [,err]
```

adr is the address of the first word of the DPB. As shown in the expansion below, this address must be a valid source address for a MOV instruction. (The adr parameter can be omitted. In this case, the entire DPB must be on the stack when the DIR\$ is executed.)

err is an optional error address. Control passes to this address if the directive does not execute successfully. If this parameter is omitted, control returns in-line with the C-bit set in the program status word (PSW).

For example,

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

expands to:

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

The address of the DPB is placed on the stack, and the EMT transfers control to the RSX Emulator. The emulator processes the directive according to the parameters specified in the DPB, pops the DPB address off the stack, and (for ALUN\$) returns control in-line. The BCC instruction branches around the JSR to ERROR if the C-bit is not set (no error occurred).

Other Features of \$ Form/DIR\$ Combination

The \$ form of the RSX Emulator directives generate local offsets which can be used to refer to parameters in the DPB.* For example, in the ALUN\$ directive, you specify a logical unit number, physical device name, and device unit number:

```
LN5DEF: ALUN$ 5,TT,0
```

When this directive is expanded, it includes code that allows you to refer to the parameters as:

```
LN5DEF+A.LULU      (logical unit number)
LN5DEF+A.LUNA      (device name)
LN5DEF+A.LUNU      (unit number)
```

In Chapter 5, these offset names are listed under "Local Symbol Definitions" for each directive. The number of bytes defined for each offset is given in parentheses following a description of the area referenced by the offset. For example:

A.LUNU -- Logical unit number (2)

(Offset names can also be used with the \$C form of directives, but cannot be used with the \$S form, since \$S generates values which are pushed on the stack.)

The \$ form expansions are also designed so that you can use them to simply set up the offset symbols to reference a DPB in another module. If you define the symbol \$\$\$GLB in your program (\$\$\$GLB = 0 will do),

*The expansions use the MACRO-11 .NLIST directive so that the code which generates these offsets (not of interest to your program) does not show up in your assembly listings.

the \$ form of any directive will generate the symbolic offsets as global symbols and will not generate the DPB itself. Thus, for example, if another module to be assembled includes the code:

```
MYMY: ALUN$ 5,TT,0
```

In the current module, define MYMY as a global symbol, define \$\$\$GLB, specify an ALUN\$, and you will be able to reference MYMY+A.LULU, and so forth.

4.4.2 \$C Form

If you know the DPB parameters at assembly time, and need to issue a directive only once, the \$C form saves you the trouble of coding two directives: the \$ form and DIR\$. The \$C form also saves execution time over the \$S form, which uses MOV instructions to place the entire DPB on the stack (Section 4.4.3).

The \$C form of the directives expand to code which:

1. Defines the DPB in a separate PSECT (called \$DPB\$\$).
2. Returns to a user-specified PSECT. If no PSECT name is given, returns to the blank (unnamed) PSECT.
3. Pushes the DPB address on the stack.
4. Passes control to the RSX Emulator.
5. On return, passes control to a user-specified error routine if the directive did not execute successfully. If no error routine is specified, no such code is generated.

The directive descriptions in Chapter 5 show, for the most part, the form and expansion for the \$ form. To use the \$C form, you must add the C to the directive name, and, optionally, add a PSECT name and error address to the parameter list. For example:

```
ALUN$C 5,TT,0,PART1,ERROR
```

expands to:

```
.PSECT  $DPB$$
$ $$=.
    .BYTE   7,4
    .WORD   5
    .ASCII   /TT/
    .WORD   0
    .PSECT  PART1
    MOV    #$$,$-(SP)
    EMT    377
    BCC    .+6
    JSR    PC,ERROR
```

The DPB is placed in a separate PSECT called \$DPB\$. The \$\$\$ symbol is equated to the address of the DPB, which is then used in the MOV instruction that pushes the DPB address on the stack before control is passed to the emulator via the EMT 377. The last two instructions set up the test for error and transfer control to the specified error routine. If no error address had been specified in the call, these two instructions would not have been generated.

Other Features of \$C Form

The offset mnemonics generated for the \$ form are also generated for the \$C form. You can save and use the \$\$\$ address to form the base address for the offsets if you want to. The \$\$\$GLB feature discussed for the \$ form in Section 4.4.2 also works for the \$C form of the directives.

4.4.3 \$S Form

The \$S form is useful if your code is re-entrant and the DPB parameters can vary. The DPB is generated when the directive is executed: in other words, at run-time. The \$S form of the directives expand to code which:

1. Generates the DPB at execution time, and pushes it on the stack.
2. Transfers control to the RSX Emulator*.
3. On return, passes control to a user-specified error routine if the directive did not execute successfully. If no error routine is specified, no such code is generated.

To use the \$S form of the directives, add S to the directive names shown in Chapter 5. You can also specify an error address as the last parameter in the line. For example:

```
ALUN$S #5,DEV,UN,ERR
```

expands to:

```
MOV UN,-(SP)
MOV DEV,-(SP)
MOV #5,-(SP)
MOV (PC)+,-(SP)
.BYTE 7,4
EMT 377
BCC +.6
JSR PC,ERR
```

*The RSX Emulator checks the low byte of the first word on the stack to determine if it is a DPB address (even low byte) or a DIC (odd low byte). If the low byte is odd, then, it knows that the rest of the DPB is on the stack.

The DPB is generated at execution time and pushed on the stack. Control passes to the RSX emulator, and if control returns with an error (the C-bit is set on return), control transfers to the error routine. If no error address had been specified in the call, the last two instructions would not have been generated.

4.5 FIRST 1000 BYTES OF LOW SEGMENT FOR RSX

As noted in Section 2.4, the first 1000 (octal) bytes of virtual address space have special meaning to the Monitor. The Task Builder (TKB), which links programs assembled or compiled under the RSX Run-Time System and its derivatives, automatically allocates the first 1000 bytes of virtual address space and assigns mnemonics as shown in Figure 4-3.

Task Name in RAD50	0 R.TSKN	(Reserved for Monitor PPP Context Use)	112 FPPTXT
Partition Name in RAD50	2		
	4 R.PRTN		
	6	LUN Table	200 R.LUNS
Task Size (without extension), 32-word blocks	10 R.CDSSZ		
ODT SST Vector Address	12 R.ODTV		
	14 R.ODTL		
Task SST Vector Address	16 R.SSTV	keyword	400 KEY USRSP
	18 R.SSTL	FIRQB	402 FIRQB
Task SST Vector Length	20 R.FAST		
ODT SST Vector Length	22 R.FAST		
FPP AST Service Address	24 R.CAST		
CTRL/C AST Service Address	26 R.TSKF	XRB	442 XFB
Task Flags	28 R.PARM	core common area	460 CORCMN
Run Parameter on Entry (from FIRQB+FQENT)	30 R.CCLF		
CCL Entry Flags, Run Size/Extension (XRB+0)	32 R.CCLF		
Load Size of Task in 32-word blocks	34 R.LDSZ	SP saved here on a post-mortem dump	660 PMDSP
Current Size of Task in 32-word blocks	36 R.TKSZ	XRB saved here on a post-mortem dump	662 PMDXRB
Number of LUNS	40 R.LUN	FIRQB also saved on PM dump starting at PMDXRB	
Monitor RSX Emulation Reserved Word 1	42 R.MON1		
Monitor RSX Emulation Reserved Word 2	44 R.MON2	SP stack space for RSX Emulator	722
Directive Status Word	46 \$DSW	User's assignable project, programmer number	734 USRPPN
FCS Impure Area Pointer	50 .FSRPT	User's assignable protection code	736 USRPRT
OTS Impure Area Pointer	52 \$OTSV	User's logical device table	740 USRLOG
Auto-Load Impure Area Pointer	54 NOVPT		
Extended Impure Area Pointer	56 \$VEXT		
(Reserved for Monitor Context Use)	60 CONTEXT	Stack guard word (must = 0)	1000 NSTORG
		Stack guard word	

Figure 4-3 First 1000 Bytes of Low Segment for RSX.



CHAPTER 5

RSX EMULATOR DIRECTIVES

5.1 INTRODUCTION

The RSX Emulator Directives:

1. Perform non-file-structured input/output.
 - ALUN\$ assigns a logical unit number to a device and unit. The logical unit number then serves to define the device in I/O requests.
 - GLUN\$ returns information about a logical unit.
 - QIO\$ and QIOW\$ open, read from, write to, and close logical units. Input/output in RSTS/E is "synchronous". That is, it is not "asynchronous" as in some systems such as RSX-11M where a user program can request I/O, do other processing, and get an interrupt when the I/O is complete. Thus, under RSTS/E, QIO\$ (queue input/output) and QIOW\$ (queue input/output and wait) do the same thing. Control returns to the calling program when the I/O is done.
 - WTSE\$ is provided for compatibility with RSX-11M, where QIO\$ and WTSE\$ can be used together to cause the program to wait until I/O is completed. Since I/O in RSTS/E is synchronous, the WTSE\$ is implemented here as a "no-operation".
 - GMCR\$ returns the command line typed by the user to invoke the program currently running, if it was invoked by a RSTS/E CCL command.
 - WSIG\$ is provided for compatibility with RSX-11M, where some recoverable error conditions require waiting for some "significant event" to occur before retrying the operation that caused the error. In RSTS/E, such events are transparent to the user program. WSIG\$ simply causes the program to sleep for one second.

2. Let you specify your own trap routines.

- SFPA\$ lets you specify an address to which control is to pass when an FPP exception trap occurs. (The FPP is the floating point processor used on all PDP-11 processors for which RSTS/E is available, except the PDP-11/40.)
- ASTX\$ exits from an FPP trap-handling routine, restoring the directive status word (DSW), program counter (PC), and program status word (PSW) to the values they had before the trap occurred.
- SCCA\$ defines an address to which control passes when a user at the job's terminal types a CTRL/C combination. The SCCA\$ directive is unique to RSTS/E; it does not exist in the RSX-11M operating system.
- SVTK\$ lets you specify a table of addresses to which control is to pass when synchronous system traps occur. Eight possible traps can be handled by the job, including PDP-11/40 floating point instruction set (FIS) errors, TRAP instructions, breakpoint (BPT) instructions, IOT instructions, and T-bit traps.
- SVDB\$ lets you specify a table of synchronous system trap addresses which are to override any specified with an SVTK\$ directive. This is a useful capability for a debugging routine, to divert breakpoint traps, for example, to the debugging routine regardless of any SVTK\$ directive in other modules of the job.

3. Return control to the run-time system when the program has finished executing; change the size of the user job image.

- EXIT\$ returns control to the job's private default run-time system. (A normal exit.)
- ABRT\$ prints a "severe error" message at the job's terminal, and returns control to the job's private default run-time system.
- EXST\$ prints a status message at the job's terminal, and returns control to the job's private default run-time system.
- EXTK\$ increases or decreases the amount of memory allocated for execution of the user job image.

4. Return system information to the job.

- GTSK\$ returns "task" parameters: run priority, project, programmer number, number of logical units assigned, address of SST vector table (set up the SVTK\$ or SVDB\$), and the system in which the job/task is running.
- GTIM\$ returns the current day and time.
- GPRT\$ returns "partition" parameters: general information about the job area and user job image area.

5. Provide access to resident libraries.

- ATRG\$ attaches the job to a resident library, laying the groundwork for access to the library. If necessary, the resident library will be loaded from disk.
- DTRG\$ detaches the job from a resident library.
- CRAW\$ creates a "window" of virtual addresses for reference to a library. Optionally, the window can be mapped to actual locations in a resident library.
- ELAW\$ eliminates an address window; it releases the virtual addresses for other use (expanding the job image, creating another window).
- MAP\$ relates a created address window to actual memory locations in an attached resident library. The user program can then refer to locations in the library using the virtual addresses defined in the window.
- UMAP\$ unmaps a window from a library; the window can then be mapped to another portion of the library or to a different library.

The RSX Emulator directives are described in alphabetical order in the remainder of this chapter.

5.2 ABRT\$ -- Abort

The ABRT\$ directive terminates execution of the calling job. Any open files or devices are reset (that is, closed without the usual cleanup operations). On a RSTS/E system, ABRT\$ acts the same as the EXST\$ directive with a "severe error". Or, if EXST\$ is not implemented, ABRT\$ acts as a simple EXIT\$.

Control is passed to the job's private default run-time system at the keyboard monitor entry point (P.NEW, Section 2.4).*

Macro Call:

```
ABRT$ name
```

The name parameter is ignored in RSTS/E; the calling program can only abort itself.

Macro Expansion:

```
ABRT$ ALPHA
.BYTE 83.,3 ;DIC=83., DPB SIZE = 3 WORDS
.RAD50 /ALPHA/ ;IGNORED IN RSTS/E
```

Local Symbol Definitions:

```
A.BTTN -- name (4)
```

DSW Return Codes:

IE.SDP -- DIC or DPB size is invalid. This could not occur unless your program changes the value of the first word of the DPB during execution. Control would otherwise not return in-line for the ABRT\$ directive.

*The private default run-time system will be the system default run-time system unless the job has run the SWITCH utility or executed the .RTS directive (Section 3.19) to establish a private default run-time system.

5.3 ALUN\$ -- Assign Logical Unit Number

The ALUN\$ directive assigns a logical unit number (LUN) to a physical device unit. The logical unit number can then be used in QIO\$ or QIOW\$ directives to do I/O (open, read, write, or close the associated device).

The device name specified can be:

1. A (two-character) user logical device name [assigned by the user at a terminal with ASSIGN, as described in the RSTS/E System User's Guide, or by the job with the CALFIP ASSFQ subfunction (Section 3.2.1)],
2. A system-wide logical device name (such as SY or TI), or
3. A standard RSTS/E device name (such as TT, PP, and so forth).

The RSX Emulator checks the name specified in the call against the user logical names; if a match is found, the logical unit number is assigned to the corresponding physical device. If the device name specified does not match any user logical device names, a similar search is performed for a match with a system-wide logical device name. If there is no match, the emulator checks the Monitor's physical device table, and, if a match is found, assigns the logical unit number specified in the call to the actual physical device unit specified. Otherwise, the directive returns an error.

Macro Call:

```
ALUN$ lun,dev,unt  
  
lun = Logical unit number; 1 - 14.  
  
dev = Device name; two ASCII characters.  
  
unt = Device unit number. -1 indicates no unit number, as  
opposed to a unit number of 0. RSTS/E makes a  
distinction between SY (meaning the public disk  
structure), and SY0 (meaning the disk from which the  
system was booted.) Likewise, TI, TT, and KB all refer  
to the job's terminal, whereas KB0 and TT0 both refer  
to the system console terminal.
```

Macro Expansion:

```
ALUN$      5,TT,0  
.BYTE    7,4      ;ALUN$ MACRO DIC=7, DPB SIZE=4 WORDS  
.WORD     5      ;LOGICAL UNIT NUMBER 5  
.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.IDU -- Invalid device and/or unit
IE.ILU -- Invalid logical unit number
IE.SDP -- DIC or DPB size is invalid

5.4 ASTX\$ -- AST Service Exit

In the RSTS/E environment, the ASTX\$ directive can be used to terminate a routine which handles an asynchronous floating-point-processor trap, if the program has indicated its intention to handle such traps with an SFPAS\$ directive (Section 5.20). No other condition in RSTS/E will cause an asynchronous trap from which an ASTX\$ exit would be useful. [The CTRL/C trap (see SCCA\$, Section 5.19) only requires an RTI instruction to exit properly.]

When an ASTX\$ is executed, the stack must be in the state:

```
SP --> (DSW) at the time trap occurred  
          (PC) at the time trap occurred  
          (PS) at the time trap occurred  
          word SP pointed to before the trap
```

So, your routine must pop the Floating Point Exception Code (FEC) and Floating Point Address (FEA) pushed on the stack when the trap occurred, as described for the SFPAS\$ directive (Section 5.20).

When the ASTX\$ directive is executed, the RSX emulator restores the DSW value to the DSW location, and loads the PC and PS registers with their appropriate values. Thus, these values are popped from the stack, and control returns to the user program at the point where it left off when the trap occurred.

You can change the contents of the PC address in the stack before the ASTX\$ is executed, so that control is passed to some other point than where it left off. Be sure you know what you are doing, though, because it is hard to debug errors in an asynchronous trap service routine.

Macro Call:

```
ASTX$ [err]  
  
err = Error routine address
```

Macro Expansion:

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

NOTE: The \$S form is recommended for this directive since (1) only a one-word DPB is generated, and (2) no BCC instruction is generated for the \$S form (or the \$C form). Thus, the \$S form takes less space than the \$form/DIR\$ combination. And, since the one-word DPB requires only one MOV instruction, the \$S form takes less space and no more execution time than the \$C form.

Local Symbol Definitions:

None

DSW Return Codes:

IE.SDP -- DIC or DPB size invalid. This can only occur if your program changes the first word of the DPB at execution time.

5.5 ATRG\$ -- Attach Resident Library

The ATRG\$ directive attaches the job to a resident library. The type of access (read-only or read/write) is specified in the call. If the calling job can access the library in that fashion*, the Monitor loads the library from disk (if necessary) and sets up its own internal tables which lay the groundwork for the job to map windows to the library. Note, however, that the resident library does not take up space in the job area (virtual memory) with an attach. APRs are assigned -- virtual memory in the job area is taken -- when a window is created (CRAW\$).

If the attach is successful, a resident library ID is returned to the job which is used in other directives to detach the job from the library (DTRG\$), or to map and unmap windows to and from the library (MAP\$ and UMAP\$). Once the job is attached to a library, the ATRG\$ directive can be used simply to determine the library ID.

Up to five resident libraries can be attached to a job at any given time.

Macro Call:

```
ATRG$ adr
```

adr = Address of an 8-word area defining the library to be attached and the type of access desired. Information is also returned to this area by the ATRG\$ directive. Two supplementary directives are available to define (RDBDF\$) or define and fill (RDBBK\$) such an 8-word area, called the resident library definition block, or RDB. The RDBDF\$ and RDBBK\$ directives are described in Appendix C. The descriptions below show the offsets automatically defined by these directives.

Macro Expansion:

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

*The job's ability to access the resident library depends upon protection assigned by the system manager when the resident library was installed. The default protection grants read access to all users, and denies write access to all users.

Data Passed in RDB:

	Offset Octal Mnemonic	Offset Octal Mnemonic
1	////////// 0	
3	////////// 2	
5		4 R.GNAM
7	resident library name (2 words in RAD50 format)	6
11	////////// 10	
13	////////// 12	
15	access flags	14 R.GSTS
17	////////// 16	

adr+R.GNAM The name of the resident library to which the job is to be attached, as two words of RAD50 data. (Resident libraries are made known to the RSTS/E Monitor by the system manager with the ADD LIBRARY command of UTILITY (RSTS/E System Manager's Guide). With this command, the system manager defines a file (filnam.LIB) as a resident library. The Monitor regards "filnam" as the resident library's name.)

adr+R.GSTS The low-order two bits in this word define the desired access to the library.

Mnemonic	Bit	Meaning
RS.RED	0 = 1	Read-only access is desired.
RS.WRT	1 = 1	Read/write access is desired.

Once a job is attached to a resident library, the access cannot be changed without detaching (DTRG\$) and re-attaching. Also, once the job is attached to a library, an ATRG\$ with the same resident library name specified at adr+R.GNAM will simply return the assigned resident library ID.

Data Returned to RDB:

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	resident library ID	1 0	R.GID
3	size of library, in 32-word blocks	1 2	R.GSIZ
5		1 4	
7		1 6	
11		1 10	
13		1 12	
15		1 14	
17		1 16	

adr+R.GID This word is an identifier which must be used in subsequent calls to identify a resident library, rather than the resident library name. Thus, you will use this identifier to detach (DTRG\$) and to map and unmap windows (MAP\$ and UMAP\$) to the library.

adr+R.GSIZ The size of the resident library, in 32-word blocks.

Local Symbol Definition:

A.TRBA -- Resident library definition block address (2)

DSW Return Codes:

IS.SUC -- Successful completion.

IE.UPN -- Attaching a job to a resident library requires a small buffer and no small buffers are currently available.

IE.PRI -- The attach did not succeed because the caller's privileges do not allow the access requested. This could happen either because the access code specified is not compatible with the possible access defined when the library was installed by the system manager, or because the protection code associated with the resident library file excludes the access requested by the user.

IE.PNS -- The resident library name specified is not known to the Monitor. A resident library must be installed by the system manager before it can be used.

5.6 CRAW\$ -- Create Address Window

The CRAW\$ directive can be used either to create a window (a range of virtual addresses) or to create a window and map it to a range of actual addresses in an attached resident library. You define the range of addresses by (1) naming a base APR, which defines the starting virtual address for the window, and (2) specifying the size of the window in 32-word blocks. Thus a window always begins on a 4K-word boundary in virtual memory. It may take more than 4K words, depending on the size of the window.

If the virtual address range overlaps the user job image, the call fails with an error. The address range cannot overlap the run-time system (high segment) unless it is the RSX Run-Time System on a system which has been generated with the RSX Emulator as a part of the Monitor. If the address range overlaps an existing window, the previously created window is eliminated.

The difference between (1) creating a window and (2) creating and mapping a window is best illustrated by example. By using create without map, you can define one window which can be mapped to a library or portion of a library, and then remapped to another portion of the same library or another library, as many times as desired, using MAP\$. For example, suppose your program takes up 24K words, and you want to access a 24K-word resident library of data values. You can use create without map to set up a 4K-word window in APR 6. You can then map the window to the first 4K words of the library, process the data, map to the next 4K words of the library, and so forth.

If, on the other hand, you had a 4K-word program, and still wished to access a 24K-word library, you could use CRAW\$ to create a 24K-word window and map it to the entire library in APRs 1 - 6.

A job can create a maximum of seven windows. A window takes at least one APR. It may take more, depending on the size you specify for the window. Thus, the maximum of seven assumes seven 4K windows in APRs 1 through 7. APR 0 can never be used to create a window, since the user program takes at least this much space. As mentioned above, a window cannot overlap the user job image; thus, the size of the user job image determines the lowest base APR that can be used. If the program (user job image) is less than 4K words, APRs 1 and up (to the limit imposed by the run-time system boundary) can be used to create windows. If the user job image is between 4K words and 8K words, APRs 2 and up can be used to create windows, and so forth.

If a window is created which overlaps an already-existing window, the old window is eliminated. For example, if you created a 6K-word window using a base APR of 5, the window would use APRs 5 and 6. If you then created a 4K-word window using a base APR of 6, the entire old window would be eliminated. APR 5 would be free for other use; APR 6 is used for the new window.

Macro Call:

```
CRAW$    adr
```

adr = Address of an 8-word area defining the window and the resident library to which the window is to be mapped, if mapping is requested. Information is also returned to this area by the CRAW\$ directive. Two supplementary directives are available to define (WDBDF\$) or define and fill (WDBBK\$) such an area called the window definition block, or WDB. The WDBDF\$ and WDBBK\$ directives are described in Appendix C. The descriptions below show the offsets automatically defined by these directives.

Macro Expansion:

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

Data Passed in WDB:

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	base APR (1-7) / / / / / / / / / / / / 0	W.NAPR	
3	/ 2		
5	size, in 32-word blocks, of window 4	W.NSIZ	
7	resident library ID (for map) 6	W.NRID	
11	offset, in 32-word blocks (for map) 10	W.NOFF	
13	map length, in 32-word blocks 12	W.NLEN	
15	access flags 14	W.NSTS	
17	/ 16		

- adr+W.NAPR The base APR of the window, 1-7. Implicitly defines the starting address of the window. This byte cannot be zero; nor can it name an APR already being used to map the user job image (see discussion above).
- adr+W.NSIZ The desired size of the window, in 32-word blocks. For example, a value of 128. = 4K words.

- adr+W.NRID The identifier of the resident library to which the window is to be mapped. (This is the value returned at adr+R.GID in the RDB for an ATRG\$ directive.) This word is ignored for calls requesting a create without mapping (bit 7 at adr+W.NSTS, below, is zero).
- adr+W.NOFF The offset, in 32-word blocks, from the start of the library where the mapping is to begin. This word is ignored if no mapping is requested (bit 7 at adr+W.NSTS, below, is zero.) A value of zero for this word indicates the window is to be mapped beginning at the first byte of the library. A value of 1 indicates the window is to be mapped beginning at the 33rd word of the library (starting address + 64), and so forth.
- adr+W.NLEN The length, in 32-word blocks, of the area to be mapped (ignored if bit 7 at adr+W.NSTS, below, is zero). This value cannot be greater than the length of the window specified at adr+W.NSIZ. Also, this value, combined with the offset specified at adr+W.NOFF, cannot indicate an address beyond the end of the library.
- A value of zero defaults to either the size of the window or the space remaining in the library, whichever is smaller.
- adr+W.NSTS Two bits in this word define whether the window is to be mapped, and whether read-only access or read/write access to the window is desired.

WDBDF\$, WDBBK\$		Mnemonic	Bit	Meaning
WS.MAP	7 = 1 = 0			The window is to be mapped. The window is not to be mapped
WS.WRT	1 = 1 = 0			Read/write access is desired. Read-only access is desired.

Data Returned in WDB:

Offset Octal Mnemonic		Offset Octal Mnemonic
1	////////// window ID	0 W.NID
3	starting virtual address of window	2 W.NBAS
5	//////////	4
7	//////////	6
11	//////////	10
13	length, in 32-word blocks, mapped	12 W.NLEN
15	status flags	14 W.NSTS
17	//////////	16

- adr+W.NID Window ID, a value from 1 - 7. This ID can be used in later MAP\$ calls to map the newly created window, and must be used in any ELAW\$ calls to eliminate the window.
- adr+W.NBAS The starting virtual address of the new window.
- adr+W.NLEN The length, in 32-word blocks, actually mapped by the window.
- adr+W.NSTS Status flags.

Mnemonic	Bit	Meaning
WS.CRW	15 = 1	The window was created successfully.
	= 0	The window was not created.
WS.ELW	13 = 1	An existing window was eliminated because it overlapped the newly created window.
	= 0	No existing windows were eliminated by this create.
WS.UNM	14 = 1	An existing window was unmapped because it overlapped the newly created mapping.
	= 0	No existing windows were unmapped by this mapping.

Local Symbol Definitions:

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

DSW Return Codes:

IS.SUC -- Successful completion.

IE.ALG -- Either the base APR - window length specified was invalid, or the offset - mapping length values specified were invalid. (For example, an offset indicating a starting address for the mapping which was beyond the end of the library.)

IE.NVR -- The library ID specified for mapping does not specify any library currently attached to the job.

IE.PRI -- The create was unsuccessful because the user privileges do not allow the access desired. At this point, since the library has been attached successfully with some access defined, this error means that the access requested in the CRAW\$ is not allowed by the access requested in the ATRG\$.

IE.WOV -- An attempt was made to create more than seven address windows.

IE.UPN -- Creating a window requires a small buffer; a small buffer is not currently available.

5.7 DTRG\$ -- Detach Resident Library

The DTRG\$ directive detaches the job from a previously attached resident library. Any windows mapped to the library by the calling job are unmapped and eliminated. If no other jobs are currently attached to the library, and it is not a permanently resident library, the Monitor will remove the library from memory.

Macro Call:

```
DTRG$ adr
```

adr = Address of an 8-word area defining the library to be detached. Information is also returned to this area by the DTRG\$ directive. Two supplementary directives are available to define (RDBDF\$) or define and fill (RDBBK\$) such an 8-word area, called the resident library definition block, or RDB. The RDBDF\$ and RDBBK\$ directives are described in Appendix C. The descriptions below show the offsets automatically defined by these directives.

Macro Expansion:

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

Data Passed in RDB:

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	resident library ID	0	R.GID
3		2	
5		4	
7		6	
11		10	
13		12	
15		14	
17		16	

adr+R.GID This word is the library identification returned (at the same position in the RDB) by the ATRG\$ directive.

Data Returned in RDB:

Offset Octal Mnemonic	Offset Octal Mnemonic
1	/ / / / / / / / 0
3	/ / / / / / / / 2
5	/ / / / / / / / 4
7	/ / / / / / / / 6
11	/ / / / / / / / 10
13	/ / / / / / / / 12
15	status flags 14 R.GSTS
17	/ / / / / / / / 16

adr+R.GSTS Bit 14 = 1 (RS.UNM) if any windows were unmapped as a result of this detach.

Local Symbol Definitions:

D.TRBA -- Resident library definition block address (2)

DSW Return Codes:

IS.SUC -- Successful completion.

IE.NVR -- The library ID specified does not identify any library currently attached to the job.

5.8 ELAW\$ -- Eliminate Address Window

The ELAW\$ directive eliminates an address window that was created by the job, unmapping the window if necessary. ELAW\$ frees the APRs used by the window and makes them available for creating another window or for expanding the user job image size.

Macro Call:

```
ELAW$    adr
```

adr = Address of an 8-word area defining the window to be eliminated. Information is also returned to this area by the ELAW\$ directive. Two supplementary directives are available to define (WDBDF\$) or define and fill (WDBBK\$) such an area, called the window definition block, or WDB. The WDBDF\$ and WDBBK\$ directives are described in Appendix C. The descriptions below show the offsets automatically defined by these directives.

Macro Expansion:

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

Data Passed in WDB:

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1	////////// window ID	0	W.NID
3	//////////	2	
5	//////////	4	
7	//////////	6	
11	//////////	10	
13	//////////	12	
15	//////////	14	
17	//////////	16	

adr+W.NID The ID of the window to be eliminated (returned at the same location in the WDB by the CRAW\$ directive).

Data Returned in WDB:

Offset	Octal Mnemonic	Offset	Octal Mnemonic
1	/ / / / / / / / / / / / / / / /	0	
3	/ / / / / / / / / / / / / / / /	2	
5	/ / / / / / / / / / / / / / / /	4	
7	/ / / / / / / / / / / / / / / /	6	
11	/ / / / / / / / / / / / / / / /	10	
13	/ / / / / / / / / / / / / / / /	12	
15	status flags	14	R.GSTS
17	/ / / / / / / / / / / / / / / /	16	

adr+W.NSTS Two bits in this word give status information:

	WDBDF\$, WDBBK\$	Mnemonic	Bit	Meaning
WS.ELW			13 = 1	The window was successfully eliminated.
			= 0	The window was not eliminated.
WS.UNM			14 = 1	The window eliminated was mapped to a resident library, and has been unmapped.
			= 0	The window eliminated was not mapped; no unmapping was done.

Local Symbol Definitions:

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

DSW Return Codes:

IS.SUC -- Successful completion.

IE.NVW -- Bad window ID. Either outside the range 1 - 7, or matches no currently created window for this job.

5.9 EXIT\$ -- Task Exit

The EXIT\$ directive terminates execution of the calling job. Control passes to the job's private default run-time system* at the keyboard monitor entry point (P.NEW, Section 2.5). Any devices still open will be reset; that is, closed without performing the usual cleanup operations.

Macro Call:

```
EXIT$S [err]  
err = Error routine address.
```

Macro Expansion:

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

NOTE: The \$S form is recommended for this directive since (1) only a one-word DPB is generated, and (2) no BCC instruction is generated for the \$S form (or the \$C form), since control never returns in-line unless an error occurs. Thus, the \$S form takes less space than the \$form/DIR\$ combination. And, since the one-word DPB requires only one MOV instruction, the \$S form takes less space and no more execution time than the \$C form.

Local Symbol Definitions:

None

DSW Return Codes:

```
IE.SDP -- DIC or DPB size is invalid. This can only occur if  
your program changes the first word of the DPB at  
execution time.
```

*The job's private default run-time system will be the system default run-time system, unless the job has run the SWITCH utility or has issued an .RTS directive (Section 3.20) to create a private default run-time system.

5.10 EXST\$ -- Exit with Status

Like EXIT\$, the EXST\$ directive terminates execution of the calling job. In addition, EXST\$ may print a message at the job's terminal. The message varies according to a status value passed as a parameter in the call:

Status Value	Message printed
0	?TASK EXIT STATUS: Warning (no message printed)
1	?TASK EXIT STATUS: Error
2	?TASK EXIT STATUS: Severe error

NOTE: If the job is detached when the EXST\$ directive is executed, the job will hibernate because of the attempt to print on the job's terminal.

Control passes to the job's private default run-time system* at the keyboard monitor entry point (P.NEW, Section 2.4). Any devices still open will be reset; that is, closed without performing the usual cleanup operations. Any attached resident libraries will be detached, and any address windows created will be eliminated.

Some older Run-Time Systems based on RSX (for example, RMS11.RTS for V6C) do not eliminate this directive and will return IE.SDP error code. To guard against this, code an EXIT\$ directive following the EXST\$ directive.

Macro Call:

```
EXST$    sts
```

sts = Status value defining the message to be printed on the job's terminal, as described above.

Macro Expansion:

```
EXST$    STATS
.BYTE   29.,2      ;DIC=29., DPB SIZE = 2 WORDS
.WORD   STATS      ;EXIT STATUS WORD
```

Local Symbol Definitions:

```
E.XSTS -- Exit status word (2)
```

DSW Return Codes:

No errors are possible with EXST\$.

*The job's private default run-time system will be the system default run-time system, unless the job has run the SWITCH utility or has issued an .RTS directive (Section 3.20) to create a private default run-time system.

5.11 EXTK\$ -- Extend Task

The EXTK\$ directive lets you increase the amount of memory allocated to the user job image. For compatibility with RSX-11M, the argument specified in the call indicates a positive or negative increment of 32-word blocks. The RSTS/E Monitor, however, actually allocates space for the user job image in 1K-word units. So the RSX Emulator keeps track of the user job image size as a number of 32-word blocks. When an EXTK\$ directive is executed, the emulator checks to see if the extension or reduction requires or frees one or more 1K-word blocks of memory. If so, the emulator directs the Monitor to make the new allocation. If not, the current allocation remains.

NOTE: You should not use the .CORE directive (Section 3.7) before an EXTK\$ directive in the same program. The .CORE directive is executed directly by the Monitor, without the intervention (or knowledge) of the RSX Emulator. If a .CORE were executed before an EXTK\$, the Emulator would have incorrect information on the current size of the user job image when it executed the EXTK\$, and base the new allocation on the size before the .CORE was executed.

You cannot extend beyond the smallest of:

1. The job's private maximum size. (A private job maximum size can be set by the system manager.)
2. Maximum size allowed by the RSX run-time system. If RSX Emulation was not installed as part of the Monitor, this limit is 28K words. If RSX Emulation was installed as part of the Monitor, this limit is 31K words.
3. The system-wide maximum user job image size. (Set by the system manager; see SWAP MAX, RSTS/E System Generation Manual.)

You also cannot extend into virtual address space used by an address window created for use with resident libraries (see CRAW\$ directive, Section 5.6).

If there is not enough contiguous memory for the extension to be made where the job currently is in memory, the user job image will be swapped out to disk, then returned to memory at the new size.

If no increment is given in the call, the Emulator allocates the amount of space the job had initially, based on information built in at link time (with TKB); that is, before any EXTK\$ was executed.

Macro Call:

EXTK\$ [inc]

inc = The number of 32-word blocks by which the task size is to be extended (positive value for inc) or reduced (negative value for inc).

Macro Expansion:

```
EXTK$      40
.BYTE     89.,3      ;DIC = 89., DPB SIZE = 3 WORDS
.WORD     40          ;EXTEND 40(8) BLOCKS (1K WORDS)
.WORD     0           ;RESERVED WORD
```

Local Symbol Definitions:

E.XTIN -- Extend increment (2)

DSW Return Codes:

IE.SUC -- Successful completion.

IE.UPN -- The requested increment would have caused the user job image size to exceed that allowed (see discussion, above). The user job image size stays at the current size.

5.12 GLUN\$ -- Get LUN Information

The GLUN\$ directive fills a 6-word buffer with information about a physical device unit to which a logical unit (LUN) is assigned.

Macro Call:

```
GLUN$ lun,buf
```

lun = Logical unit number

buf = Address of 6-word buffer to receive the information.

Buffer Format:

Offset Octal		Offset Octal	Mnemonic
1	device name (2 ASCII characters)	0	G.LUNA
3	device unit number	2	G.LUNU
5	first device characteristics word	4	G.LUCW
7	second device characteristics word	6	
11		10	
13	standard device buffer size	12	

buf+0 Name of the assigned device; two ASCII characters. This must be a valid physical device name. Exceptions are the logical device names OV and LB, which default to SY if they are not defined, and CL, which defaults to TT.

buf+2 Unit number of the assigned device. If this word is -1, it indicates there is no unit number. If this word is 0 or a positive number, it is a real unit number.

buf+4 First device characteristics word:

Bit 0 = 1, record-oriented device
Bit 1 = 1, carriage-control device
Bit 2 = 1, terminal device
Bit 3 = 1, directory device
Bit 4 = 1, single-directory device
Bit 5 = 1, sequential device
Bits 6-15, reserved.

Settings for bits 0-5 are (in octal):

Terminal = 03
Disk = 10
Card Reader = 01
Line printer = 02
Magtape = 40
Paper tape reader/punch = 01

buf+6 If this word is 0, it indicates the job is non-privileged. If this word is 10 (octal), it indicates the job is privileged.

buf+12 The standard device buffer size to use for input or output on this device.

Macro Expansion:

```
GLUN$    7,LUNBUF
.BYTE    5,3      ;DIC = 5, 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 (not used in RSTS/E) (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 (must be in range 1-14.)
IE.SDP -- DIC or DPB size is invalid
```

5.13 GMCR\$ -- Get MCR (CCL) Command Line

The GMCR\$ directive transfers an 80-byte CCL command line to the job. When a program file is installed in RSTS/E to be invoked with a CCL command, it must contain code to analyze what the user typed to invoke the file. The GMCR\$ directive returns the command line typed by the user in an 80-byte buffer in the DPB.

The \$S form of the macro is not supplied. Since the DPB receives the CCL command line, the space must be allocated in read/write memory.

Macro Call:

```
GMCR$
```

Macro Expansion:

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

Local Symbol Definitions:

```
G.MCRB -- CCL command buffer (80)
```

DSW Return Codes:

- +n -- Successful completion. n is the number of data bytes transferred (excluding the termination character).
- IE.AST -- Task was not run as a CCL command, or the GMCR\$ directive has already been executed.
- IE.SDP -- DIC or DPB size is invalid.

5.14 GPRT\$ -- Get Partition (Job) Parameters

The GPRT\$ directive fills a 3-word buffer with "partition" parameters. For RSTS/E, a partition is considered to correspond to the job area.

Macro Call:

```
GPRT$ [name],buf
```

name Partition name. Omit to get actual job image size in buf+1.
buf Address of 3-word buffer where information is to be returned.

Buffer Format:

Offset Octal	Offset Octal Mnemonic
1	virtual base address (always 0 in RSTS/E) 0 G.PRPB
3	user job image size in 32-word blocks 2 G.PRPS
5	flags word (always 0 in RSTS/E) 4 G.PRFW

buf+0 Partition virtual base address -- always returned as 0 in RSTS/E.
buf+1 "Partition size" (user job image size in RSTS/E), expressed as a multiple of 32 words. If a partition name is given, a value of 1600 is returned here, to indicate the maximum job image size of 28K words.
buf+2 "Partition flags word". Always returned as 0, since all job's are system-controlled in RSTS/E. (RSX-11M returns 1 for a user-controlled partition).

Macro Expansion:

```
GPRT$ ,DATBUF  
.BYTE 65.,4 ;DIC=65., DPB SIZE = 4  
.WORD 0,0 ;GENERATE 0 WORDS IF NO NAME  
.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 buffer:

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

DSW Return Codes:

Successful completion is indicated by carry clear, and the starting address of the partition is returned in the DSW. (always 0 in RSTS/E). Unsuccessful completion is indicated by carry set, and the following code in the DSW:

ID.SDP -- DIC or DPB size is invalid.

5.15 GTIM\$ -- Get Time Parameters

The GTIM\$ directive fills an 8-word buffer with the current day and time of day. All time values are stored as one-word binary integers. The value ranges (in decimal) are shown below.

Macro Call:

```
GTIM$    buf  
buf = Address of 8-word buffer
```

Buffer Format:

Offset Octal		Offset Octal	Mnemonic
1	year (since 1900); e.g., 78.=1978	0	G.TIYR
3	month (1 - 12.); e.g., 1=January	2	G.TIMO
5	day of month (1 - 31.)	4	G.TIDA
7	hour (0 -23.); e.g., 0 = midnight	6	G.TIHR
11	minute (0 - 60.)	10	G.TIMI
13	second (0 - 60.)	12	G.TISC
15	tick of second*	14	G.TICT
17	ticks per second*	16	G.TICP

*The tick of second can range from 0 - 59. for a 60-Hz line frequency clock (KW11-L), and from 0 - 49. for a 50-Hz line frequency clock or a programmable clock (KW11-P). The ticks per second will always be 60, unless an optional patch is installed by the system manager to set the ticks per second to 50.

Macro Expansion:

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

Local Symbol Definitions:

G.TIBA -- Buffer address (2)

The following offsets are assigned relative to the start of the 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.SDP -- DIC or DPB size is invalid.
```

5.16 GTSK\$ -- Get Task (Job) Parameters

The GTSK\$ directive fills a 16-word buffer with parameters for the "task"; as with "partition" in GPRT\$, "task" in the RSTS/E environment is considered to be equivalent to the job.

Macro Call:

```
GTSK$    buf
```

buf = Address of the 16-word buffer to receive the parameters.

Buffer Format:

Offset Octal		Offset Octal Mnemonic
1		0 G.TSTN
3	task name (2 words in RAD50 format)	2
5		4 G.TSPN
7	partition name (2 words in RAD50 format)	6
11		10
13		12
15	run priority (0 - 255.)	14 G.TSPR
17	G.TSPC project number programmer number	16 G.TSGC
21	number of LUNs assigned	20 G.TSNL
23		22
25		24
27	address of SST vector table	26 G.TSVA
31	length of SST vector table in words	30 G.TSVL
33	size (in bytes) of memory for job image	32 G.TSTS
35	system where job is running (4=RSTS/E)	34 G.TSSY

- buf+0 Task name. The name supplied with TASK = name at the time the task was built (linked) with TKB; otherwise, the filename of the file being run, in RAD50 format.
- buf+4 Partition name specified with PAR = name at task-build time. If no such name was specified, the characters "GEN " (GEN and three blanks), to indicate the general partition, the default system-controlled partition in RSX-11M.

buf+14 Run priority; may range from 0 to 255 (decimal). This is the RSTS/E job priority plus 128 (decimal).

buf+16 Programmer number of the user currently running the program.

buf+17 Project number of the user currently running the program.

buf+20 Number of logical unit numbers (LUNs) currently assigned.

buf+26 Address of SST vector table. If buf+30 = 0, the contents of this word are meaningless.

buf+30 Length of SST vector table, in words. If no SST vector table, this word is set to 0.

buf+32 Job size, in bytes.

buf+34 System in which the program/job/task is running:

0 = RSX-11D
1 = RSX-11M
2 = RSX-11S
3 = IAS
4 = RSTS/E
5 = TRAX
6 = VMS (VAX processor)

Macro Expansion:

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

Local Symbol Definitions:

G.TSBA -- Buffer Address

The following offsets are assigned relative to the buffer:

G.TSTN -- Task name (4)
G.TSPN -- Partition name (4)
G.TSPR -- Priority (2)
G.TSGC -- Programmer number (1)
G.TSPC -- Project number (1)
G.TSNL -- Number of logical units (2)
G.TSVA -- SST vector address (2)
G.TSVL -- Length of SST vector table (2)
G.TSTS -- Size of space for user job image (2)
G.TSSY -- System being run under (2)

DSW Return Codes:

IS.SUC -- Successful completion
IE.SDP -- DIC or DPB size is invalid

5.17 MAP\$ -- Map Address Window

The MAP\$ directive maps a previously created address window to an attached resident library. In other words, MAP\$ relates the virtual address range defined by a CRAWS directive (Section 5.6) to actual locations in memory within a resident library that has been attached to the job by an ATRG\$ directive (Section 5.8).

If the window specified is already mapped, it is unmapped from its previous actual memory locations, and remapped to the new area. A job may map a maximum of seven address windows at any given time.

Macro Call:

```
MAP$    adr
```

adr = Address of an 8-word area defining the window to be mapped. Information is also returned to this area by the MAP\$ directive. Two supplementary directives are available to define (WDBDF\$) or define and fill (WDBBK\$) such an area, called the window definition block, or WDB. The WDBDF\$ and WDBBK\$ directives are described in Appendix C. The descriptions below show the offsets automatically defined by these directives.

Macro Expansion:

```
MAP$      WDBADR  
.BYTE    121.,2           ;DIC=121., DPB SIZE=2 WORDS
```

Data Passed in WDB:

Offset Octal Mnemonic		Offset Octal Mnemonic
1	window ID	0 W.NID
3		2
5		4
7	resident library ID	6 W.NRID
11	offset, in 32-word blocks	10 W.NOFF
13	length, in 32-word blocks	12 W.NLEN
15	access flags	14 W.NSTS
17		16

adr+W.NID The ID of the window to be mapped (returned at the same location in the WDB by the CRAW\$ directive).

adr+W.NRID The ID of the resident library to which the window is to be mapped (returned at offset R.GID in the RDB by the ATRG\$ directive that attached the job to the resident library).

adr+W.NOFF The offset, in 32-word blocks, from the start of the library where the mapping is to begin. A value of zero for this word indicates that the window is to be mapped beginning at the first byte of the library. A value of 1 indicates that the window is to be mapped beginning at the 33rd word of the library (starting address +64), and so forth. The offset cannot indicate a starting address past the end of a library.

adr+W.NLEN The length, in 32-word blocks, to be mapped to the library. This value cannot be greater than the size of the window defined in the CRAW\$ directive with which the window was created. Also, this value, combined with the offset value at adr+W.NOFF, cannot indicate an address beyond the end of the library.

A value of 0 for this word defaults to the size of the window or the space remaining in the library, whichever is smaller.

adr+W.NSTS Set bit 1 = 1 (WS.WRT) for read/write access. A separate setting for access in MAP\$ and in ATRG\$ allows you to attach to a library read/write and map a portion of the library read-only. You cannot, however, attach to a library read-only and then map to the library read/write.

Data Returned in WDB:

Offset Octal Mnemonic	Offset Octal Mnemonic
1	/ 0
3	/ 2
5	/ 4
7	/ 6
11	/ 10
13	length actually mapped 12 W.NLEN
15	status flags 14 W.NSTS
17	/ 16

adr+W.NLEN Length, in 32-word blocks, actually mapped by the call.

adr+W.NSTS Bit 14 = 1 (WS.UNM) if the window specified was unmapped before the new map.

Local Symbol Definitions:

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

DSW Return Codes:

IS.SUC -- Successful completion.

IE.ALG -- The offset and length specified are inconsistent; either the mapping attempted to go beyond the end of the library, or the length is greater than the created window.

IE.PRI -- The mapping did not succeed because user privileges did not allow the access desired.

IE.NVW -- Either the resident library ID or the address window ID is incorrect. (The job is not currently attached to the specified resident library, or no address window has been created with the specified window ID.)

5.18 QIO\$ and QIOW\$ -- Queue I/O Request (and Wait)

The QIO\$ and QIOW\$ directives do non-file-structured input/output. In the RSTS/E environment, QIO\$ and QIOW\$ do the same thing. Input/output in RSTS/E is "synchronous". That is, you cannot request I/O, do other processing, and get an interrupt when the I/O completes. In RSTS/E, control only returns to the program when the I/O is complete.

As "non-file-structured" implies, the QIO\$ and QIOW\$ directives are primarily useful for input/output on a terminal, card reader, paper tape reader/punch, or line printer.

Macro Call:

```
{ QIO$ } fnc,lun,[efn],[pri],[isb],[ast][,par]  
 { QIOW$ }
```

fnc = I/O function code:

IO.ATT Attach. In RSTS/E, opens the device specified by the logical unit number (lun).

IO.DET Detach. In RSTS/E, closes the device specified by the logical unit number (lun).

IO.RLB } Read. IO.RLB and IO.RVB do the same thing in RSTS/E. The actions performed depend on the device, as described below.
 { IO.RVB

IO.WLB } Write. IO.WLB and IO.WVB do the same thing in RSTS/E. The actions performed depend on the device, as described below.
 { IO.WVB

lun = Logical unit number; defines the device for which the I/O operation is to be performed. Must have been previously defined with an ALUN\$ directive.

efn = Event flag. (Ignored in RSTS/E.)

pri = Priority. (Ignored in RSTS/E.)

isb = I/O status block. Address of a 2-word buffer to which information is returned about the outcome of the I/O operation. The contents vary according to the function code and device, as described below.

ast = Asynchronous trap address. (Ignored in RSTS/E).

par = Parameter list. Of the form <p1,p2[,,,p3]> for RSTS/E users. Form and meaning varies according to function code and device, as described below.

Format and Description by Function Code:

{ QIO\$ } IO.ATT,lun[,,isb]
{ QIOW\$ }

Opens the device specified by the logical unit number (lun). Status information is returned to the 2-word buffer at address isb.

{ QIO\$ } IO.DET,lun[,,isb]
{ QIOW\$ }

Closes the device specified by the logical unit number (lun). Status information is returned to the 2-word buffer at address isb.

{ QIO\$ } { IO.RLB } ,lun,,,[isb],<stadd,size[,,,block]>
{ QIOW\$ } { IO.RVB }

For devices termed "record-oriented" in the RSX-11M environment (see GLUN\$, Section 5.12), IO.RLB and IO.RVB return a record (a line on a terminal, card on a card reader, record on a paper tape reader). The data is read into the buffer whose starting address is defined by stadd, and whose length in bytes is defined by size. The amount of data read never exceeds the buffer size. If the buffer size is smaller than the "record", the next IO.RLB or IO.RVB for that lun will read another buffer-full. The block parameter is omitted for these devices.

For disk, IO.RLB and IO.RVB fill the buffer whose starting address is defined by stadd, and whose length in bytes is defined by size. The read begins with the device cluster number specified by the block parameter. If block is omitted or 0, sequential reads are performed. In this case, the next read will begin with the next device cluster. The buffer size should be a multiple of the device cluster size for the disk (see Appendix A). NOTE: Only privileged jobs can do non-file-structured reads from disk while the disk is being used by other jobs.

For all devices, the isb parameter is the address of a 2-word buffer to which status information on the read is returned. If isb is omitted, no such status is returned.

{ QIO\$ } { IO.WLB } ,lun,,,[isb],<stadd,size[,vfc]>
{ QIO\$ } { IO.WVB }

For devices termed "record-oriented" in the RSX environment (see GLUN\$, Section 5.12), IO.WLB and IO.WVB write the contents of the buffer to the device specified by the logical unit number (lun). The starting address for the buffer is defined by stadd, and the length in bytes is defined by size. For devices which are "record-oriented" and "carriage-control" devices, the vfc parameter specifies an action to be performed after the buffer contents are written. (The vfc parameter is a vertical format control character for terminals and

line printers.) Values for vfc and actions taken are listed in Table 5-1.* The vfc parameter is omitted for devices which are not carriage-controlled.

For disk, IO.WLB and IO.WVB write the contents of the buffer to disk, beginning at the device cluster number specified by the block parameter. If block is omitted, sequential writes are performed. The next write (for sequential writes) will begin with the next device cluster. The buffer size for disk writes must be a multiple of the device cluster size. NOTE: No job (privileged or non-privileged) can write to disk in non-file-structured mode while any other user is accessing the disk.

For all devices, the isb parameter defines the starting address of a 2-word buffer to which status information on the write is to be returned. If omitted, no such status information is returned.

RSX MODE TERMINAL I/O

In releases prior to RSTS/E V7.0, the emulation of QIO\$ and QIOW\$ directives was not identical to the behavior of the same directives on RSX-11M systems, due to inherent differences between the RSTS/E terminal handler and the RSX-11M terminal handler.

The major difference between the two systems is how the handlers echo a "RETURN" key typed on the terminal. In RSTS/E, the handler echoes a carriage return - line feed combination. In RSX-11M, the handler echoes only a carriage return.

This difference was a problem for programmers using QIO\$ or QIOW\$ in programs doing terminal I/O to be run on both systems. The problem arose for programs doing a read operation followed by a write operation using the null vfc control character. On RSX-11M, such write operations must include a leading line feed character, or else the rest of the data would overwrite the characters typed by the user. Programs on RSTS/E in the same situation would not have included a leading line feed character, since the terminal handler had already echoed one to the terminal.

This problem is resolved in RSTS/E Version 7.0 by code in the RSTS/E terminal handler. When a QIO\$ or QIOW\$ is executed, the RSX Emulator tells the terminal handler to enter "RSX Mode" on that particular terminal. In this mode, the terminal handler will echo typed RETURNS as they are on RSX-11M; that is, with a carriage return only. However, if the user types more characters before any programmed read request, a line feed is printed before the typed characters, so that they will not overprint the previously typed line. Furthermore, when the terminal handler is in RSX mode, it will print a line feed whenever a .READ or .WRITE directive (Chapter 3) is executed for that terminal. The terminal handler will remain in RSX mode for that terminal until the job exits to its private default run-time system.

*See also the special section on RSX Mode Terminal I/O above.

Table 5-1
Vertical Format Control Characters

Octal Value	Character	Meaning
040	blank	SINGLE SPACE -- Output a line feed, print the contents of the buffer, and output a carriage return. Normally, printing immediately follows the previously printed line.
060	0	DOUBLE SPACE -- Output two line feeds, print the contents of the buffer, and output a carriage return. Normally, the buffer are contents are printed two lines below the previously printed line.
061	1	PAGE EJECT -- Output eight line feeds (or, for a line printer or an LA180, output a form feed), print the contents of the buffer, and output a carriage return.
053	+	OVERPRINT -- Print the contents of the buffer and output a carriage return. Normally overprints the previous line.
044	\$	PROMPTING OUTPUT -- Output a line feed and print the contents of the buffer. This mode of output is intended for use with a terminal on which a prompt message is output, and input is then read on the same line.
000	null	INTERNAL VERTICAL FORMAT -- Print the buffer contents without addition of vertical format control characters. In this mode, more than one line of contiguous output can be printed for each I/O request.

Contents of I/O Status Buffer:

The general format of the information returned to the 2-word status buffer defined by isb is:

(terminal I/O code)	general I/O code
number of bytes read or written	

The first word contains information on the success or failure of the I/O operation requested. As with \$DSW, you can use mnemonic codes to test this word. The mnemonics are resolved automatically by the task builder (TKB).

For all devices and operations except terminal reads, you test the low-order byte of the first word to determine the I/O status code. Successful terminal reads provide additional information on what character terminated the line typed at the terminal. For terminal reads, you test the low-order byte first to see if the read was successful (IS.SUC). If it was, you can then test the entire first word to find out if the line was terminated with a carriage return (IS.CR) or an ESC or ALT character (IS.ESC).

The second word of the block contains the number of bytes read or written, if the read or write was successful. No information is returned in this word for IO.ATT and IO.DET operations.

I/O codes are listed below. Remember that IS.CR and IS.ESC are full-word values; the others are tested by the low-order byte only.

NOTE: The RSX Emulator does the requested I/O operation using the non-file-structured OPNFQ subfunction of CALFIP, the CLSFQ subfunction of CALFIP, .READ, and .WRITE, and does not clear the FIRQB or XRB when it returns control to the user program. The codes given in square brackets below [] are the RSTS/E errors that were returned to the emulator. You can do further checking in the case of IE.BAD, for example, by checking byte 0 of the FIRQB.

IS.SUC Successful completion of I/O operation. For terminal reads, you can determine the terminating character of the line by testing the full first word of the I/O status buffer for:

IS.CR The terminal line was terminated with a carriage return/ line feed combination. (The terminating characters are not included in the character count in the second word. They do appear in the input buffer.)

IS.ESC The terminal line was terminated with an ESC (called ALTMODE on some terminals). (The terminating character is not included in the character count in the second word. It does appear in the input buffer.)

- IE.BAD This code is returned by the RSX Emulator when an error occurs on the I/O operation which did not fit in to any of the error categories diagnosed below. To determine the RSTS/E error, examine byte 0 of the FIRQB.
- IE.DAA An open (IO.ATT) was attempted for a device which this job has already opened. The device must be closed (IODET) before it can be opened again. [NOTCLS]
- IE.DNR Some hard device I/O error occurred (a read for an off-line device, a write for a physically write-locked device, and so forth). [HNGDEV]
- IE.EOF An end-of-file mark, record, or control character (CTRL-Z for terminals) was recognized on a read operation. [EOF]
- IE.EOT No more room is available on the device for a write operation (end of paper tape, end of magtape, not enough disk space available, etc.) [NOROOM]
- IE.PRI Privilege violation. A non-privileged job attempted to read or write to disk while the disk was in use. [PRVIOL]
- IE.NLN A close (IODET) was attempted for a device which was not open. [NOTOPN]
- IE.RSU Shareable resource in use. The device is not available; it is assigned to another user. [NOTAVL]
- IE.VER Unrecoverable error, such as parity error, bad card column, etc. [DATERR]

Macro Expansion:

```

QIO$    IO.RVB,7,,,IOSTAT,,<IOBUFR,80.>
.BYTE  1,12.      ;DIC=1, DPB SIZE = 12. WORDS
.WORD   IO.RVB    ;FUNCTION CODE FOR READ
.WORD   7          ;LOGICAL UNIT NUMBER 7
.BYTE   0,0        ;EFN,PRI IGNORED IN RSTS/E
.WORD   IOSTAT    ;I/O STATUS BUFFER
.WORD   0          ;AST IGNORED IN RSTS/E
.WORD   IOBUFR    ;BUFFER ADDRESS
.WORD   80.        ;BYTE COUNT = 80.
.WORD   0          ;NO VFC NEEDED FOR READ
.WORD   0          ;THIS PARAM. IGNORED IN RSTS/E
.WORD   0          ;THIS PARAM. IGNORED IN RSTS/E
.WORD   0          ;NO BLOCK FOR THIS INVOCATION

```

The expansion for QIOW\$ is the same except that the DIC = 3.

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)
Q.IOSB -- Address of I/O status block (2)
Q.IOAE -- AST address (2)
Q.IOPL -- Parameter list (6 words) (12)

DSW Return Codes:

IS.SUC -- Successful completion.
IE.LUN -- Unassigned logical unit number (lun). Use ALUN\$.
IE.ILU -- Invalid logical unit number (lun). Must be in the
range 0 -14.
IE.SDP -- DIC or DPB size is invalid.

5.19 SCCA\$S -- Specify Control/C AST

The SCCA\$S directive (actually, a macro) lets you specify an address to which control is to be transferred if the user types a CTRL/C while the program is executing. Only the \$S form is allowed, since no DPB is ever generated. The SCCA\$S directive is unique to the RSTS/E environment; no similar directive exists for the RSX-11M operating system.

The SCCA\$S expands to code which takes the address specified and stores it in location 24 (part of the first 1000 bytes of memory used by the run-time system, as described in Section 4.2). The RSX Run-Time System checks this location when a CTRL/C asynchronous trap occurs, and if it is non-zero, clears the word and transfers control to the location that was specified. If this word is zero, the RSX Run-Time System falls back to its own processing of CTRL/C traps. Thus, SCCA\$S is a "one-time" operation; you must reset the trap as part of the trap-processing routine, if desired.

Your routine can handle the CTRL/C in any way it sees fit. An RTI instruction will return control to the point where it left off at the time of the trap. The stack upon entry to such a routine is:

```
SP --> (PC) at the time trap occurred  
          (PS) at the time trap occurred  
          word to which SP pointed before the trap
```

Macro Call:

```
SCCA$S [arg]
```

arg = Contains the address (usually specified in the form "#addr") to which control passes should the user at the job's terminal type a (CTRL/C) combination. If this address is omitted, word 24 is cleared, indicating that the RSX Run-Time System is to handle such traps.

Macro Expansion:

```
SCCA$S #CTRAP  
MOV     #CTRAP,@#24
```

or, if the address is omitted,

```
SCCA$S  
CLR     @#24
```

Local Symbol Definitions:

None

DSW Return Codes:

None. (SCCA\$S does not execute an EMT; control does not pass to the RSX Emulator when SCCA\$S is executed.)

5.20 SFPA\$ -- Specify Floating Point Processor Exception Address

The SFPA\$ directive tells the RSX Emulator one of two things:

1. The job wants to handle floating point processor exception traps itself. The RSX Emulator passes control to the address specified in the directive when such an asynchronous trap occurs.*
2. The job does not want to handle floating point processor exception traps. The address is omitted in the directive. In this case, if an FPP exception trap occurs, the RSX Emulator will abort the job, displaying the message "?Reserved instruction trap" on the job's terminal (TI:).

If an address is specified in the SFPA\$ directive, control will be passed to the address when an FPP trap occurs. The stack will have the following information:

```
SP --> Floating exception address (FEA)
      Floating exception code (FEC)
      (DSW) at the time the trap occurred
      (PC) at the time the trap occurred
      (PS) at the time the trap occurred
      word pointed to by SP before the trap
```

Your routine can process the exception in any way it sees fit. To exit from the trap-handling routine, pop the FEA and FEC from the stack, and issue an ASTX\$ directive (Section 5.4).

Macro Call:

```
SFPA$ [add]
```

add = Address to which control passes when an FPP trap occurs.

Macro Expansion:

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

Local Symbol Definitions:

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

DSW Return Codes:

```
IS.SUC -- Successful completion
IE.SDP -- DIC or DPB size is invalid
```

*The FPP is the floating point processor used on all PDP-11 processors for which RSTS/E is available, except the PDP-11/40. The FPP executes concurrently with the PDP-11 central processor. Thus, an error in the FPP unit occurs "asynchronously" with the execution of the PDP-11 CPU.

5.21 SVDB\$ -- Specify SST Vector Table for Debugging Aid

The SVDB\$ directive, like the SVTK\$ directive, defines a table of addresses to which control is to pass when certain synchronous system traps (SSTs) occur. When an SST occurs, the RSX Emulator will first check the table defined with SVDB\$ to see if an address has been specified for that trap. If so, control is passed to that address. If not, the Emulator then checks the table defined with SVTK\$ (if any).

In other words, the SVDB\$ sets up an SST vector table whose entries, if non-zero, override (but do not destroy) any SST vector table entries defined with SVTK\$. The SVDB\$ directive is handy within a debugging routine which uses BPT instructions, for example, to set breakpoints. With SVDB\$, the debugging routine can divert breakpoint traps to itself, regardless of SVTK\$ directives in other modules of the job.

For example, the Octal Debugging Tool (ODT) object module uses the SVDB\$ directive to divert traps to itself, regardless of what may have been requested with SVTK\$ in other modules.* This makes it possible to debug without changing your source code; you can link with ODT, use ODT to insert breakpoints, test and debug, and when finished, re-link without ODT. It is not necessary to change source code to debug, only re-link. The SVDB\$ makes this possible.

Macro Call:

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

adr = Address of SST vector table. If adr and len are both omitted, any SST vector table previously defined with an SVDB\$ directive will be deassigned; that is, the Emulator will not check for SVDB\$ addresses. (A previously executed SVTK\$ will remain in effect.)

len = Length of SST vector table, in words. As shown below, up to eight addresses can be specified. If you only wished to use the third, you could save space by using a len of 3, rather than a len of eight and filling the last 5 words of the buffer with zeroes. Specifying a length greater than 8 has the same effect as a length of 8.

*The ODT object module (the file ODT.OBJ) can be linked to user modules with the RSTS/E Task Builder (TKB) using the /DA switch, as described in the RSTS/E Task Builder Reference Manual. ODT.OBJ is different from ODT.BAC, which is a BASIC-PLUS utility in RSTS/E. The commands and syntax for ODT.OBJ are described in the RSX/IAS ODT Reference Manual.

The vector table format is shown below. A non-zero value is interpreted as an address to which control is to pass when that particular trap occurs. A zero value for a word indicates no interest in handling that trap. Should such a trap occur, the emulator will check the vector table assigned by SVTK\$, if any. If there is no SVTK\$ vector table, the emulator will abort the job and display an error message at the job's terminal.

Buffer Format:

Octal Offset		Octal Offset
1	odd address or nonexistent memory error	0
3	memory protect violation	2
5	T-bit trap or execution of a BPT instr.	4
7	execution of an IOT instruction	6
11	execution of a reserved instruction	10
13	execution of a non-RSX EMT instr.	12
15	execution of a TRAP instruction	14
17	PDP-11/40 flt. pt. exception	16

Macro Expansion:

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

Local Symbol Definitions:

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

DSW Return Codes:

```
IS.SUC -- Successful completion
IE.SDP -- DIC or DPB size invalid
```

5.22 SVTK\$ -- Specify SST Vector Table for Task

The SVTK\$ directive defines a table of addresses to which control is to pass when certain synchronous system traps (SSTs) occur. When an SST occurs, the RSX Emulator will first check to see if a similar table has been defined for this job with an SVDB\$ directive (Section 5.21). If so, any non-zero entries in that table will override those defined with an SVTK\$. If not, the Emulator uses the trap addresses defined with SVTK\$.

Macro Call:

SVTK\$ [adr,len]

adr = Address of SST vector table. If adr and len are both omitted, any SST vector table previously defined with an SVTK\$ directive will be deassigned. (A previously executed SVDB\$ will remain in effect.)

len = Length of SST vector table, in words. As shown below, up to eight addresses can be specified. If you only wished to use the first five, you could save space by using a len of 5; rather than a len of 8 and filling the last 3 words of the buffer with zeroes. Specifying a length greater than 8 has the same effect as a length of 8.

Buffer Format:

Octal Offset		Octal Offset
1	odd address or nonexistent memory error	0
3	memory protect violation	2
5	T-bit trap or execution of a BPT instr.	4
7	execution of an IOT instruction	6
11	execution of a reserved instruction	10
13	execution of a non-RSX EMT instr.	12
15	execution of a TRAP instruction	14
17	PDP-11/40 flt. pt. exception	16

Macro Expansion:

```
SVTK$    SSTTBL,4
.BYTE    105.,3      ;DIC = 105., DPB LENGTH = 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.SDP -- DIC or DPB size is invalid
```

5.23 UMAP\$ -- Unmap an Address Window

The UMAP\$ unmaps a specified address window from a resident library. The unmapping does not eliminate the window (created with a CRAW\$ directive), nor does it release the APRs used by the window. The virtual address window can now be remapped to new actual memory locations, with the MAP\$ directive, if desired.

Macro Call:

```
UMAP$ adr
```

adr = Address of an 8-word area defining the window to be unmapped. Information is also returned to this area by the UMAP\$ directive. Two supplementary directives are available to define (WDBDF\$) or define and fill (WDBBK\$) such an area, called the window definition block, or WDB. The WDBDF\$ and WDBBK\$ directives are described in Appendix C. The descriptions below show the offsets automatically defined by these directives.

Macro Expansion:

```
UMAP$ WDBADR  
.BYTE 123.,2 ;DIC=123., DPB SIZE=2 WORDS  
.WORD WDBADR
```

Data Passed in WDB:

Offset Octal Mnemonic	Offset Octal Mnemonic
1 / / / / / / / /	window ID 0 W.NID
3 / / / / / / / /	2
5 / / / / / / / /	4
7 / / / / / / / /	6
11 / / / / / / / /	10
13 / / / / / / / /	12
15 / / / / / / / /	14
17 / / / / / / / /	16

adr+W.NID The ID of the window to be unmapped (returned at the same location in the WDB by the CRAW\$ directive).

Data Returned in WDB:

Offset Octal	Mnemonic	Offset Octal	Mnemonic
1		0	
3		2	
5		4	
7		6	
11		10	
13		12	
15	status flag	14	W.NSTS
17		16	

adr+W.NSTS Bit 14 = 1 (WS.UNM) if the window specified was successfully unmapped.

Local Symbol Definitions:

U.MABA -- Window definition block address (2)

DSW Return Codes:

IS.SUC -- Successful completion.

IE.ITS -- The window ID specified is either invalid (outside the range 1-7), or is not currently mapped.

5.24 WSIG\$ -- Wait for Significant Event Flag

The WSIG\$ directive is included for compatibility with RSX-11M, in which some recoverable error conditions require waiting for a "significant event" to happen, at which time a retry of the operation that failed may succeed. On a RSTS/E system, any events which may be termed significant within the RSTS/E Monitor are transparent to user jobs. Hence, the WSIG\$ directive in RSTS/E merely causes the calling job to sleep for one second. The job can then retry whatever operation failed previously.

Macro Call:

```
WSIG$
```

Macro Expansion:

```
WSIG$  
.BYTE    49.,1          ;DIC=49., DPB SIZE=1 WORD
```

Local Symbol Definitions:

None

DSW Return Codes:

IS.SUC -- Successful completion
IE.SDP -- DIC or DPB size is invalid

5.25 WTSE\$ -- Wait for Single Event Flag

The WTSE\$ directive is included for compatibility with RSX-11M, in which some programs are written with a QIO\$ followed by a WTSE\$, to cause the program to wait until the I/O is complete. This has the same effect as a QIOW\$, and since all I/O in RSTS/E is synchronous QIO\$ functions the same as QIOW\$. Thus, the WTSE\$ directive simply returns immediately; it is a "no-op" in RSTS/E.

Macro Call:

```
WTSE$    efn
        efn      =      Event flag number (ignored in RSTS/E).
```

Macro Expansion:

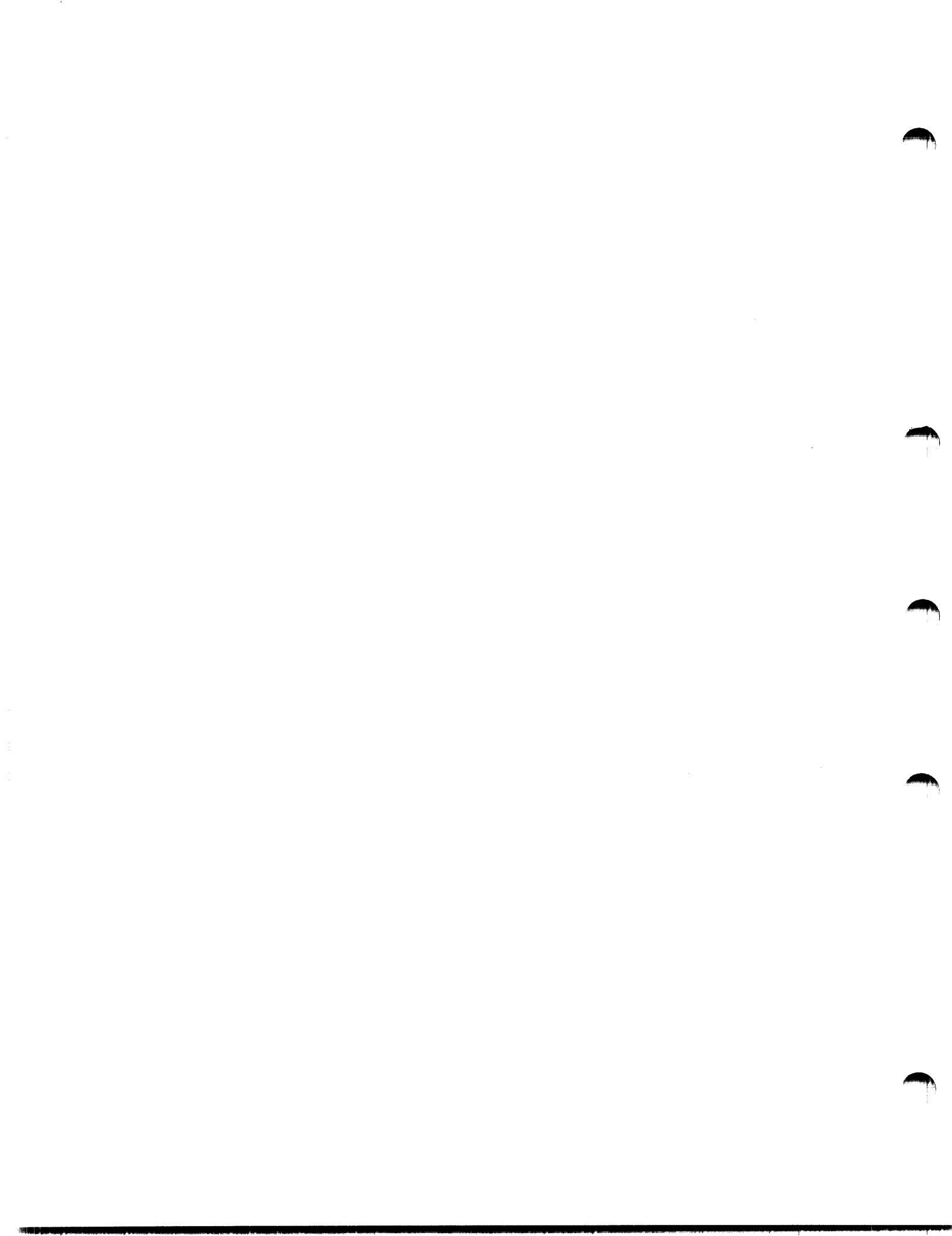
```
WTSE$    0
        .BYTE   41.,2      ;DIC=41., DPB SIZE = 2 WORDS
        .WORD   0          ;FLAG IGNORED IN RSTS/E
```

Local Symbol Definition:

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

DSW Return Codes:

```
IS.SUC -- Successful completion
IE.SDP -- DIC or DPB size is invalid
```

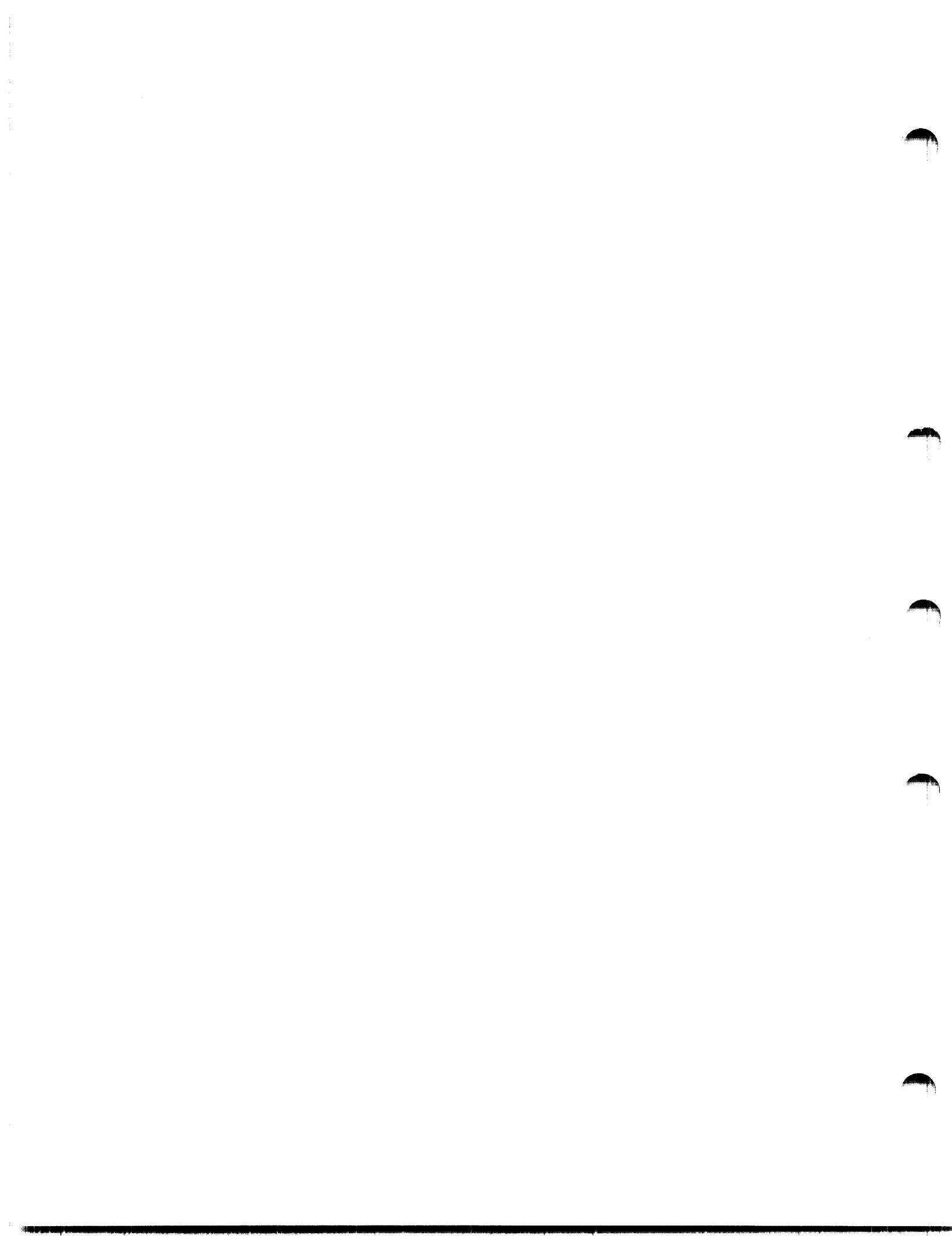


APPENDIX A
FULL LIST OF ERRORS

ERR.STB Mnemonic	Decimal Value	Full Error text
BADDIR	1	?Bad directory for device
BADNAME	2	?Illegal file name
INUSE	3	?Account or device in use
NOROOM	4	?No room for user on device
NOSUCH	5	?Can't find file or account
NODEVC	6	?Not a valid device
NOTCLS	7	?I/O channel already open
NOTAVL	8	?Device not available
NOTOPN	9	?I/O channel not open
PRVIOL	10	?Protection violation
EOF	11	?End of file on device
ABORT	12	?Fatal system I/O failure
DATERR	13	?User data error on device
HNGDEV	14	?Device hung or write locked
HNGTTY	15	?Keyboard wait exhausted
FIEXST	16	?Name or account now exists
DTOOOF	17	?Too many open files on unit
BADFUO	18	?Illegal SYS() usage
INTLCK	19	?Disk block is interlocked
WRGPAK	20	?Pack ids don't match
NOTMNT	21	?Disk pack is not mounted
PAKLCK	22	?Disk pack is locked out
BADCLU	23	?Illegal cluster size
PRIVAT	24	?Disk pack is private
INTPAK	25	?Disk pack needs 'CLEANing'
BADPAK	26	?Fatal disk pack mount error
DETKEY	27	?I/O to detached keyboard
CTRLCE	28	?Programmable ^C trap
SATTBD	29	?Corrupted file structure
DEVNFS	30	?Device not file-structured
BADCNT	31	?Illegal byte count for I/O
NOBUFS	32	?No buffer space available
B.4	33	?Odd address trap
B.10	34	?Reserved instruction trap
B.250	35	?Memory management violation
B.STAK	36	?SP stack overflow
B.SWAP	37	?Disk error during swap
B.PRTY	38	?Memory parity failure
MAGSEL	39	?Magtape select error

MAGRLE	40	?Mag tape record length error
NRRTS	41	?Non-res run-time system
VCSERR	42	?Virtual buffer too large
VCAERR	43	?Virtual array not on disk
SIZERR	44	?Matrix or array too big
VCOERR	45	?Virtual array not yet open
BSERR	46	?Illegal I/O channel
LINERR	47	?Line too long
FLTERR	48	%Floating point error
EXPERR	49	%Argument too large in exp
FMTERR	50	%Data format error
FIXERR	51	%Integer error
BDNERR	52	%Illegal number
LOGERR	53	%Illegal argument in log
SQRERR	54	%Imaginary square roots
SUBERR	55	?Subscript out of range
MINVER	56	?Can't invert matrix
ODD	57	?Out of data
ONBAD	58	?ON statement out of range
NEDERR	59	?Not enough data in record
IOLERR	60	?Integer overflow, for loop
DIVBY0	61	?Division by 0
NORTS	62	?Do run-time system
FIELDE	63	?Field overflows buffer
NORACS	64	?Not a random-access device
NOTMTA	65	?Illegal MAGTAP() usage
ERRERR	66	?Missing special feature
BADSWT	67	?Illegal Switch Usage
	68	Unused error message
	69	Unused error message
	70	Unused error message
STMERR	71	?Statement not found
EXITTM	72	?RETURN without GOSUB
EXITNR	73	?FNEND without function call
UNDFNI	74	?Undefined function called
COSERR	75	?Illegal symbol
TLOPNV	76	?Illegal verb
TLNZSP	77	?Illegal expression
TLNOIT	78	?Illegal mode mixing
TLIFFE	79	?Illegal IF statement
TLCONI	80	?Illegal conditional clause
TLNOTF	81	?Illegal function name
TLQDUM	82	?Illegal dummy variable
TLMFND	83	?Illegal FN redefinition
TLRNNM	84	?Illegal line number(s)
MODERR	85	?Modifier error
	86	Unused error message
OUTCAS	87	?Expression too complicated
FUNERR	88	?Arguments don't match
TLTMAF	89	?Too many arguments
TLINCD	90	%Inconsistent function usage
CPNSDF	91	?Illegal DEF nesting
CPUPFR	92	?FOR without NEXT
CPUFNX	93	?NEXT without FOR
CPUPDF	94	?DEF without FNEND
CPUPED	95	?FNEND without DEF
TLJNKY	96	?Literal string needed
TLNOFN	97	?Too few arguments
SASYNE	98	?Syntax error

SAFNOS	99	?String is needed
SASNOI	100	?Number is needed
TLURTP	101	?Data type error
TLXDIM	102	?1 or 2 dimensions only
FUCORE	103	?Program lost-sorry
RESERR	104	?RESUME and no error
DIMED2	105	?Redimensioned array
TLIDIM	106	?Inconsistent subscript use
NOGOTO	107	?ON statement needs GOTO
EOSERR	108	?End of statement not seen
TLCNTD	109	?What?
TLPRNM	110	?Bad line number pair
EDBMCE	111	?Not enough available memory
EDEXON	112	?Execute only file
NRNERR	113	?Please use the RUN command
EDCONE	114	?Can't continue
EDARSV	115	?File exists-RENAME/REPLACE
PRERRS	116	?PRINT-USING format error
BADSWT	117	?Illegal switch usage
PRNER1	118	?Bad number in PRINT-USING
NONOIM	119	?Illegal in immediate mode
PRNER2	120	?PRINT-USING buffer overflow
BADERR	121	?Illegal statement
DISERR	122	?Illegal field variable
STPERR	123	Stop
DIMERR	124	?Matrix dimension error
NOMATH	125	?Wrong math package
XCDCOR	126	?Maximum memory exceeded
SCAERR	127	%SCALE factor interlock



APPENDIX B

DISK DEVICE SIZES

The following table lists the device cluster size and device size (in 512-byte blocks) for each of the disk devices supported by RSTS/E.

Disk Device	Device Cluster Size	Device Size
RF11	1	1024 times number of platters
RS03	1	1024
RS04	1	2048
RK05	1	4800
RK05F	1	4800 for each unit; 2 units for each drive
RL01	1	10220
RK06	1	27104
RK07	1	53768
RP02	2	40000
RP03	2	80000
RM02	4	131648
RM03	4	131648
RP04	4	167200
RP05	4	167200
RP06	8	334400



APPENDIX C

SUPPLEMENTARY RSX DIRECTIVES FOR RESIDENT LIBRARIES

The RSX Emulator directives which deal with resident libraries (ATRG\$, DTRG\$, CRAWS\$, ELAWS\$, MAP\$, and UMAP\$) use 8-word areas to pass and receive data to and from the Emulator. These areas, called the resident library definition block (RDB) and window definition block (WDB) can be defined, or defined and filled, using supplementary directives included in the RSXMAC.SML file.

NOTE: The expansions for these directives are the same as in the RSX-11M environment, where their use is more extensive. Only the arguments relevant to RSTS/E are described here.

C.1 RDB Directives

Two directives are available for use with resident library definition blocks (RDBs): RDBDF\$ and RDDBK\$.

The RDBDF\$ directive simply assigns literal values to the offsets and status bit mnemonics shown in Chapter 5 for the RDB areas for the ATRG\$ and DTRG\$ directives. You can use these mnemonics to reference offsets and bit values in an RDB you have allocated space for in your program.

The RDDBK\$ directive defines these offsets, and in addition, generates code to allocate space for the RDB and fills it with values you specify in the call.

The form for the relevant arguments in the ATRG\$ call is as follows:

RDDBK\$, ,libnam,,<RS.WRT> (for read/write access)

or

RDDBK\$, ,libnam,,<RS.RED> (for read-only access)

where libnam is the name of the resident library to be attached.

For example, the call:

```
RDBBK$ ,,DATLIB,,<RS.WRT>
```

is expanded as follows:

```
.WORD 0
.WORD 0
.RAD50 /DATLIB/
.WORD 0
.WORD 0
.WORD RS.WRT
.WORD 0
```

(RS.WRT is assigned to a literal value of 2, so bit 1 is set in the seventh word of the RDB, requesting read/write access.)

In addition, all the offsets necessary to reference the data returned in the RDB by ATRG\$ or DTRG\$ are generated. For example, you can use the mnemonic RS.UNM to test bit 14 of the word at offset R.GSTS in the RDB after execution of a DTRG\$ directive. RS.UNM is assigned to a literal value of 40000 (octal) by the RDBBK\$ directive.

C.2 WDB Directives

Two directives are available for use with window definition blocks (WDBs): WDBDF\$ and WDBBK\$.

The WDBDF\$ directive simply assigns literal values to the offsets and status bit mnemonics shown in Chapter 5 for the WDB areas for the CRAW\$, ELAW\$, MAP\$, and UMAP\$ directives. You can use these mnemonics to reference offsets and bit values in a WDB you have allocated space for in your program.

The WDBBK\$ directive defines these offsets, and in addition, generates code to allocate space for the WDB and fill it with values you specify in the call.

The form for the relevant arguments in a CRAW\$ call is as follows:

```
WDBBK$ apr,siz,rid,off,len,<bit1![bit2!bit3]>
```

apr = The base apr.

siz = The size of the window.

rid = The resident library ID.

off = The offset into the library, in 32-word blocks.

len = The length to be mapped.

bit.. = Mnemonic values for bit settings, separated by exclamation points. Relevant mnemonics for CRAW\$ are:

WS.MAP = Window is to be mapped.
WS.WRT = Map with read/write access.
WS.RED = Map with read-only access.

For example, the call:

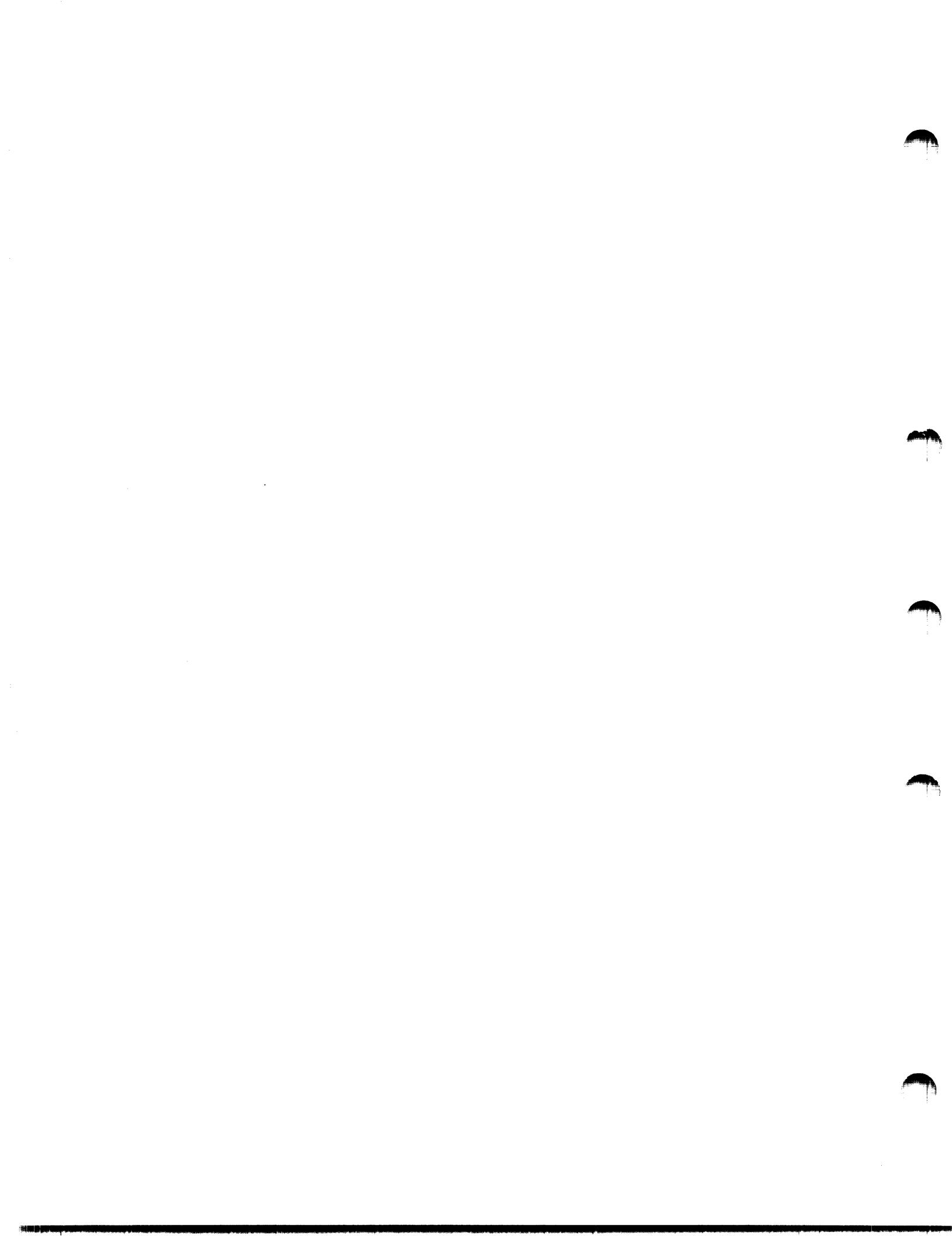
WDBBK\$ #7,128.,RDBK+R.GID,0,0,<WS.MAP!WS.RED>

expands to the following code:

```
.BYTE 0,7  
.WORD 0  
.WORD 128.  
.WORD RDBK+R.GID  
.WORD 0  
.WORD 0  
.WORD WS.MAP!WS.RED
```

This WDB, when used in a CRAW\$ directive, would create a window 4K words long (128. 32-word blocks) in APR 7. The resident library ID is used from the RDB returned by an ATRG\$ call. An offset and map length of zero are specified, and mapping read-only is requested.

The WDBBK\$ directive also defines the offsets and bit settings referred to in the discussion of ELAW\$, MAP\$, and UMAP\$. You would probably not wish to use WDBBK\$ to fill in values for a MAP\$, since it will zero out the first byte where the window ID is stored from a CRAW\$.



ABRT\$, 5-4
Account, user,
 create, 3-301
 delete, 3-278
Accounting information dump,
 3-253
Actual addressing, 2-1
Address window,
 create,
 with CRAFQ, 3-168
 with CRAW\$, 5-12
 eliminate,
 with ELAFQ, 3-178
 with ELAW\$, 5-19
 map,
 with MAPFQ, 3-182
 with MAP\$, 5-34
 unmap,
 with UMPFQ, 3-187
 with UMAP\$, 5-50
Addressing,
 actual, 2-1
 virtual, 2-1
ALUN\$, 5-5
AME, 4-1
APR, 2-1
ASSFQ, 3-10
Assign,
 device,
 with CALFIP, 3-10
 with .ULOG, 3-236
 with .UUO, 3-254
 user logical, 3-236
ASTX\$, 5-7
Asynchronous trap
 in pseudo-vector region, 2-17,
 2-26
 RSX Run-Time System, 5-7, 5-44
ATRFQ, 3-164
ATRG\$, 5-9
Attach resident library,
 with ATRFQ, 3-164
 with ATRG\$, 5-9

.BLKW0, 3-6
.BSECT, 3-5

CALFIP, 3-9
CALL, 3-6
CALLR, 3-6
CALLRX, 3-6
CALLX, 3-6

CCL command,
 add/delete, 3-261
 check string for, 3-78
 get command line, 5-27
 use of core common, 2-13
.CCL, 3-78
.CHAIN, 3-83
Change,
 password, 3-265
 system date/time, 3-270
Channel, 3-8
Cleanup diskpack, 3-268
.CLEAR, 3-84
Close channel, 3-13
 no cleanup, 3-74
CLSFQ, 3-13
COMMON.MAC, 3-3
Concise Command Language, 3-78
Control/C AST, 5-44
CORCMN, 2-13
.CORE, 3-86

DALFQ, 3-35
Dataset, hang up, 3-285
.DATE, 3-89
DEAFQ, 3-37
Deassign,

INDEX (Cont.)

resident library,
 with DTRFQ, 3-174
 with DTRG\$, 5-17

Device,
 assign,
 with CALFIP, 3-10
 with .ULOG, 3-236
 with .UUO, 3-254

close, 3-13

deassign,
 with CALFIP, 3-37
 with .ULOG, 3-244
 with .UUO, 3-271

deassign all,
 with CALFIP, 3-35
 with .ULOG, 3-247
 with .UUO, 3-269

horizontal position, 3-191

open, 3-64

zero, 3-333

DIC, 4-3

Directive parameter block, 4-3

Directive status word, 4-4

Directory,
 disk lookup, 3-292
 lookup, 3-274
 get information, 3-40

DIRFQ, 3-40

DIR\$, 4-6

Disable terminal, 3-266

Disappearing RSX Run-Time System, 2-6, 3-199, 4-2

Disk caching, 3-263

Disk clustersize, B-1

Disk device size, B-1

Disk pack
 cleanup, 3-268
 mount/dismount, 3-296

DLNFQ, 3-48

DPB, 4-3

\$DPB\$\$, 4-8

DSECT, 3-5

\$DSW, 4-4

DTRFQ, 3-174

DTRG\$, 5-17

ELAFQ, 3-178

ELAW\$, 5-19

Eliminate address window,
 with ELAFQ, 3-178
 with ELAW\$, 5-19

.EQUATE, 3-5

.ERLOG, 3-91

ERRFQ, 3-51

Error logging, 3-91

Error message text return, 3-51,
 3-279

Error mnemonics, 3-7, A-1

ERR.STB, 3-7, A-1
.EXIT, 3-92

EXIT\$, 5-21

EXST\$, 5-22

Extend user job image
 with EXTK\$, 5-23
 with .CORE, 3-86

EXTK\$, 5-23

FEA, 2-17, 2-26

FEC, 2-17, 2-26

File,
 attributes, 3-255
 close, 3-13
 create and open, 3-22
 create and open binary, 3-16
 create and open temporary,
 3-30

delete, 3-48

lookup, 3-54

open, 3-64

placement, 3-283

processor, 3-9

rename, 3-71

system, add/remove, 3-320

FIP, 3-9

FIRQB,
 data returned to, 3-7
 general definition, 2-12
 presetting to zero, 3-7

.FSS, 3-93

General Monitor directives, 3-1

GLOBAL, 3-6

GLUN\$, 5-25

GMCR\$, 5-27

GPRTS\$, 5-28

GTIM\$, 5-30

GTSK\$, 5-32

High segment, 2-4, 2-16

Horizontal position, 3-191

INCLUDE, 3-4

INDEX (Cont.)

JFBIG, 2-10
JFFPP, 2-11
JFLOCK, 2-10
JFNOPR, 2-10
JFPRI, 2-11
JFSPRI, 2-11
JFSYS, 2-10
JMPX, 3-6
Job,
 area, 2-4
 creation, 3-286
 definition of, 1-3
 kill, 3-267
 status, 3-322

KEY,
 clearing bits, 3-84
 definition, 2-10
 refresh, 2-10, 2-30, 2-31
 setting bits, 3-211

Keyword,
 clearing bits, 3-84
 definition, 2-10
 refresh, 2-10, 2-30, 2-31
 setting bits, 3-211

Kill job, 3-267

Libraries, resident,
 add, 3-312
 attach,
 with ATRFQ, 3-164
 with ATRG\$, 5-9
 detach,
 with DTRFQ, 3-174
 with DTRG\$, 5-17
 general definition, 2-7
 load, 3-15
 remove, 3-14
 unload, 3-16

LINK, 1-3, 3-7

Logical device names,
 system, 3-105, 3-317
 user, 2-15, 3-236, 3-245,
 3-247

Logical unit,
 assign (ALUN\$), 5-5
 get information (GLUN\$), 5-25

Login, 3-288
 disable further, 3-299
 enable, 3-331
 set number of, 3-290

Logout, 3-260

.LOGS, 3-105

Lookup file, 3-54
Low segment, 2-4, 2-8, 4-10

MAKSIL, 1-3
Map address window,
 with MAP\$, 5-34
 with MAPFQ, 3-182

MAP\$, 5-34
MAPFQ, 3-182

Maximum memory size,
 change, 3-305
 for user job image, 2-23,
 3-86,
 5-23

Memory,
 actual, 2-1
 change maximum, 3-305
 extend, 3-86, 5-23
 poke, 3-302
 maximum size, 2-23, 3-86
 minimum size, 2-23
 virtual, 2-1

.MESAG, 3-112

Minimum memory size,
 for user job image, 2-23

Monitor,
 directives, general, 3-1
 tables, 3-325, 3-327

Mount/dismount disk pack, 3-296

.NAME, 3-159

Name, run-time system, 3-298

OPNFQ, 3-64

ORG, 3-4

Page address register, 2-1

PAR, 2-1

Password change, 3-265

P.BAD, 2-24, 2-28

P.BPT, 2-25

P.CC, 2-27

P.2CC, 2-27

P.CRAS, 2-28

P.DEXT, 2-22

P.EMT, 2-25

PF.1US, 2-21

PF.CSZ, 2-21

PF.EMT, 2-19

INDEX (Cont.)

P.FIS, 2-24
PF.KBM, 2-22
P.FLAG, 2-19
PF.NER
P.FPP, 2-26
PF.REM, 2-21
PF.RW, 2-21
P.IOT, 2-25
P.ISIZ, 2-18
.PLAS, 3-163
P.MSIZ, 2-23
P.NEW, 2-28
P.OFF, 2-19
Poke memory, 3-302
.POSTN, 3-191
PPN,
 user-assignable, 2-14
 wildcard lookup, 3-303
Priority, change, 3-305
Private default run-time system,
 2-26, 3-201, 5-4, 5-21, 5-22
Privilege, temporary, 2-34
Program name, 3-159
Protection code,
 user-assignable default, 2-14,
 3-290
P.RUN, 2-31
P.SIZ, 2-23
P.START, 2-28
Pseudo-vectors, 2-16
P.TRAP, 2-26

detach
 with DTRFQ, 3-174
 with DTRG\$, 5-17
load, 3-15
remove, 3-14
unload, 3-16
RETURN, 3-6
RSTFQ, 3-74
RSX directive forms,
 \$ form/DIR\$ combination, 4-6
 \$C form, 4-8
 \$S form, 4-9
RSX Emulator, 4-1
RSX Run-Time System, 1-2
 directives, 5-1
 disappearing, 2-6, 3-199, 4-2
 environment, 4-1
.RSX, 3-199
RT11 Run-Time System, 1-2
.RTS, 3-201
.RUN, 3-207
Run burst, change, 3-305
Run-time system,
 add/remove/load/unload, 3-308
 general discussion, 1-1
 naming with UU.NAM, 3-298
 private default, 2-26, 3-201,
 5-4, 5-21, 5-22
 switch control to, 3-201
 system default, 2-26, 3-92,
 3-202, 5-21, 5-22

QIOS\$, 5-37
QIOW\$, 5-37
Quota change, 3-265

RDB, 5-9
RDBBK\$, C-1
RDBDF\$, C-1
.READ, 3-193
Read/write file attributes,
 3-255
Receive, 3-139
Rename file, 3-71
RENFQ, 3-71
Remove receiver, 3-114
Resident library,
 add, 3-312
 attach,
 with ATRFQ, 3-164
 with ATRG\$, 5-9
 definition of, 2-7

SCCA\$S, 5-44
Send Connect Confirm, 3-122
Send Connect Initiate, 3-117
Send Connect Reject, 3-124
Send Disconnect, 3-135
Send Interrupt, 3-129
Send Link Abort, 3-137
Send Link Service, 3-132
Send Local Data Message, 3-115
Send Network Data Message, 3-126
.SET, 3-211
SFPAS, 5-45
Shutdown, system, 3-273
SILUS, 1-3
.SLEEP, 3-212
SP, 2-11
.SPEC, 3-215
SST vector table, 5-46
Stack pointer, 2-11
.STAT, 3-226
SVDB\$, 5-46
SVTK\$, 5-48

SWITCH utility,
 use of .RTS, 3-201

Synchronous system traps,
 in pseudo-vector region, 2-16,
 2-24

RSX Run-Time System, 5-46,
 5-48

SYSTAT utility,
 use of program name, 3-159

System
 date/time change, 3-270
 files, add/remove, 3-320
 logical names, 3-317
 shutdown, 3-273

Temporary privilege, 2-34

Terminal,
 characteristics, set, 3-329
 echo, disable, 3-233
 output, restore, 3-234
 status, 3-296

.TIME, 3-228

TITLE, 3-3

TKB, 1-3, 2-7, 3-7

TMPORG, 3-4

Transportable code, 4-1

.TTAPE, 3-230

.TTDDT, 3-231

.TTNCH, 3-233

.TTRST, 3-234

.ULOG,
 call format, 3-235
 setting up USRLOG, 2-15

UMAP\$, 5-50

UMPFQ, 3-187

Unmap address window,
 with UMAP\$, 5-50
 with UMPFQ, 3-187

User account,
 create, 3-301
 delete, 3-278

User job image,
 changing size of,
 with .CORE, 3-86
 with EXTK\$, 5-23
 definition of, 2-4
 maximum size of, 2-23, 3-86

User logical,
 deassign, 3-246
 deassign all, 3-247
 enter, 3-236

User stack area, 2-11

USRLOG, 2-14, 2-15

USRPPN, 2-14

USRPRP, 2-14

USRSP, 2-11

UTILITY,
 install resident libraries,
 2-7
 install run-time systems, 2-22
 override pseudo-vectors, 2-17

UU.ACT, 3-253

UU.ASS,
 of .ULOG, 3-236
 of .UUO, 3-254

UU.ATR, 3-255

UU.ATT, 3-257

UU.BCK, 3-259

UU.BYE, 3-260

UU.CCL, 3-261

UU.CHE, 3-263

UU.CHU, 3-265

UU.CLN, 3-268

UU.DAL,
 of .ULOG, 3-247
 of .UUO, 3-269

UU.DAT, 3-270

UU.DEA,
 of .ULOG, 3-244
 of .UUO, 3-271

UU.DET, 3-272

UU.DIE, 3-273

UU.DIR, 3-274

UU.DLU, 3-278

UU.ERR, 3-279

UU.FCB, 3-281

UU.FIL, 3-283

UU.HNG, 3-285

UU.JOB, 3-286

UU.LIN, 3-288

UU.LOG, 3-290

UU.LOK, 3-292

UU.MNT, 3-296

UU.NAM, 3-298

UU.NLG, 3-299
 .UUO, 3-250

UU.OFQ, 3-77

UU.PAS, 3-301

UU.POK, 3-302

UU.PPN, 3-303

UU.PRI, 3-305

UU.RAD, 3-306

UU.RTS, 3-308

UU.SLN, 3-317

UU.SWP, 3-320

UU.SYS, 3-322

UU.TB1, 3-325

UU.TB2, 3-327

UU.TRM, 3-329

INDEX (Cont.)

UU.YLG, 3-331
UU.ZER, 3-333

WDB, 5-13
WDBBK\$, C-1
WDBDF\$, C-1
Wildcard lookup,
 file, 3-54
 PPN, 3-303
Window,
 create,
 with CRAWS, 5-12
 with CRAFQ, 3-168
eliminate,
 with ELAWS, 5-19
 with ELAFQ, 3-178

map,
 with MAP\$, 5-34
 with MAPFQ, 3-182
unmap,
 with UMAP\$, 5-50
 with UMPFQ, 3-187
.WRITE, 3-334
WSIG\$, 5-52
WTSE\$, 5-53

XRB,
 data returned to, 3-7
 general definition, 2-13
 presetting to zero, 3-7

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

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

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

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

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

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

Name _____ Date _____

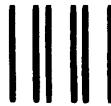
Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country _____

— — — — — Do Not Tear - Fold Here and Tape — — — — —

digital



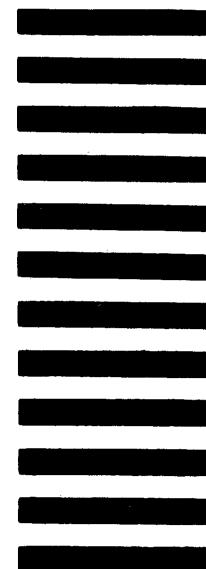
No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN: Commercial Engineering Publications MK1-2/H3
DIGITAL EQUIPMENT CORPORATION
CONTINENTAL BOULEVARD
MERRIMACK N.H. 03054



— — — Do Not Tear - Fold Here and Tape — — — — —

Cut Along Dotted Line