RSX-11M/M-PLUS
System Management Guide

Order No. AA-L679B-TC
Update Notice No. 1 (AD-L679B-T1)

RSX-11M Version 4.1
RSX-11M-PLUS Version 2.1

digital equipment corporation · maynard, massachusetts

First Printing, October 1979
Revised, November 1981
Revised, March 1982
Updated, April 1983

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

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

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

Printed in U.S.A.

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

The following are trademarks of Digital Equipment Corporation:

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

ZK2237

HOW TO ORDER ADDITIONAL DOCUMENTATION

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

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

In Canada call 613-234-7726 (Ottawa-Hull)
800-267-6146 (all other Canadian)

DIRECT MAIL ORDERS (USA & PUERTO RICO)*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire 03061

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

DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.
940 Belfast Road
Ottawa, Ontario K1G 4C2
Attn: A&SG Business Manager

DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation
A&SG Business Manager
c/o Digital's local subsidiary or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

CONTENTS

CONTENTS

CONTENTS

CONTENTS

Page

EXAMPLES

FIGURES

TABLES

PREFACE

MANUAL OBJECTIVES

The RSX-11M/M-PLUS System Management Guide documents system programs and procedures that test, monitor, maintain, and customize RSX-11M/M-PLUS operating systems after system generation.

INTENDED AUDIENCE

This manual is intended for system managers, operators, and system programmers who perform system management functions. The manual is not intended for general users or applications programmers.

STRUCTURE OF THE MANUAL

The RSX-11M/M-PLUS System Management Guide consists of 18 chapters and two appendixes as follows:

Chapter 1 summarizes each program and procedure described in Chapters 2 through 18.

Chapter 2 describes how you use the Account File Maintenance Program (ACNT) to create and maintain accounts on multiuser protection systems.

Chapter 3 describes how you use the Console Logger to handle I/O to the console output device (CO): and record system messages at a terminal and/or in a log file.

Chapter 4 describes how you use the I/O Exerciser (IOX) to detect and diagnose I/O problems on the disk and tape units in your system's hardware configuration.

Chapter 5 describes how the Bad Block Replacement Control Task (RCT) performs bad block handling and recovery on Mass Storage Control Protocol (MSCP) devices such as the RA80.

Chapter 6 describes how you use the Resource Monitoring Display (RMD) to display information about the active tasks in your operating system and the availability of system resources.

Chapter 7 describes how to set up the Queue Manager for the convenience of multistream line-printer spooling.

For RSX-11M-PLUS systems only, this chapter describes how you set up batch processors that enable you to direct a command sequence to the operating system at a predetermined time of day.

Chapter 8 describes how you use the SHUTUP Program to shut down the system in an orderly fashion.

Chapter 9 describes how you use the User Environment Test Package (UETP) after system generation to verify that you have generated a working operating system.

Chapter 10 describes how you use the Virtual Monitor Console Routine (VMR) to alter a system image file on disk.

Chapter 11 describes command line interpreter tasks, how to write them, and the use of new Executive directives in them.

Chapter 12 describes how to task build and install the DCL task with various options. It also describes the Macro Meta language, which is a set of macros that describes the mapping from DCL syntax to MCR syntax. With this language, you can alter or add commands to extend DCL.

Chapter 13 describes the Pool Monitoring Support that controls the use of the system's dynamic storage region (pool).

Chapter 14 describes the Shuffler, which is a task that compacts memory space in system-controlled partitions.

Chapter 15 describes how you use the RSX-11M-PLUS reconfiguration services (CON and HRC) to reconfigure system resources to isolate the system from the effects of faulty hardware elements.

Chapter 16 describes RSX-11M-PLUS I/O Queue Optimization, which improves the throughput of the disk subsystems used in your system.

Chapter 17 describes RSX-11M-PLUS Resource Accounting, which creates a transaction file of system usage information.

Chapter 18 describes RSX-11M-PLUS Shadow Recording, which allows your system to back up all information being written to a Files-11 disk.

Appendix A describes the Serial Despooler, which is superseded by the Queue Manager to handle line-printer spooling for most systems.

Appendix B describes the set of User File Directory (UFD) conventions observed by the operating system for files on disk. A section of file-naming conventions is also included.

ASSOCIATED MANUALS

Before you read this manual, you should be familiar with the RSX-11M/M-PLUS MCR Operations Manual, the RSX-11M/M-PLUS Utilities Manual, and, for DIGITAL Command Language users, the RSX-11M/M-PLUS Command Language Manual. Although you do not need to read the RSX-11M System Generation and Installation Guide or the RSX-11M-PLUS System Generation and Installation Guide before reading this manual, system generation logically precedes your use of the programs and procedures documented in this manual.

CONVENTIONS USED IN THIS MANUAL

Convention	Meaning
red ink	Designates user input. Responses from programs are designated in black ink.
blue ink	Designates RSX-11M-PLUS multiprocessor system information.
...	A horizontal ellipsis indicates that the command line element immediately preceding the ellipsis can be repeated one or more times.
. . .	A vertical ellipsis indicates that part of a command sequence or program output continues but is not relevant to the discussion of the example.
lowercase	In a command line, lowercase words and letters indicate that you substitute a value. Usually the lowercase word describes the value to be substituted.
UPPERCASE	Command elements other than those shown in lowercase for which you substitute a value are always uppercase. Within descriptions, these command elements are never truncated. In examples, they may be truncated according to the rules of the particular program.
❶	A reverse number is a white number printed in a black circle. These numbers key a part of an example to the text that explains that part of the example. The reverse number appears both in the example and the explanation.
[optional]	Square brackets indicate that the enclosed item is optional.
shading	Shaded portions of text describe only one operating system. Gray shading indicates that the text describes only RSX-11M-PLUS operating systems. Shading is not used in Chapters that describe programs that are unique to RSX-11M-PLUS. Red shading indicates that the text describes only RSX-11M operating systems. Portions of text that are not shaded describe both operating systems, except in the RSX-11M-PLUS-only chapters.
CTRL/X	The symbol CTRL/x indicates that you press a key while simultaneously pressing and holding down the CTRL key; for example, CTRL/C, CTRL/O.

SUMMARY OF TECHNICAL CHANGES

The RSX-11M/M-PLUS System Management Guide documents these new capabilities of RSX-11M V4.0 and RSX-11M-PLUS V2.0:

1. The Bad Block Replacement Control Task (RCT) performs bad block handling and recovery on Mass Storage Control Protocol (MSCP) devices such as the RA80.

2. DIGITAL Command Language is available on RSX-11M V4.0.

3. You can augment the DIGITAL Command Language with your own commands and write your own command language interpreters.

The other programs and procedures described in this manual have all been enhanced to include new functionality. The following lists descriptions of new software and the enhancements to previously existing software for RSX-11M V4.0 and RSX-11M-PLUS V2.0.

Account File Maintenance Program (ACNT)

1. When adding or modifying an account entry, you can specify a default command language interpreter (CLI).

2. When adding or modifying an account entry, you can specify whether or not the terminal on which a user logs in is to be a slaved terminal.

3. You can specify logical device names as the user's default device (SY:).

Console Logger

1. You interface with the Console Logger with one keyword from MCR or DCL instead of five.

2. Tasks can read from the console terminal.

3. The Console Logger time-stamps messages written to the console output device (CO:).

I/O Exerciser (IOX)

1. Three new commands:

   a. COMPAREDATA allows you to disable data comparisons during testing.

b. RANGE allows you to determine the range of blocks to be tested on nonfile-structured disks. You can also use RANGE to display the current range of blocks to be tested on the specified disk.

c. RECORDS allows you to specify the number of records to test on magnetic tapes and cassettes.

2. The ITERATIONS command has been eliminated.

3. The FILES11, SELECT, and VERIFY commands have been modified to accept commands in their command lines that set exercise parameters.

4. Exercising of magnetic tapes and DECtapes has been improved to allow testing of the entire tape and the testing of one tape while another is rewinding.

5. The CONFIGURE command allows you to specify multiple units per command line. CONFIGURE also accepts any disk or tape unit to which you can assign a logical name with the MCR ASN command.

6. The EXECUTE command sends leading blanks to the task to be executed.

7. You can use CTRL/O with the PARAMETERLIST and PRINTSUMMARY commands while no exercising is in progress.

8. The random data pattern from pure portions of the IOX task is the default pattern.

Resource Monitoring Display (RMD)

1. Two new displays have been added: Active Task Display and Task Header Display.

2. RMD parses an MCR command line so that you can select a display page and enter setup commands from the MCR command line.

3. The Memory Display includes a DECnet node name or system name.

Setting Up and Running the Queue Manager

1. MCR and DCL are supported.

2. Naming conventions for the spooled output device, the device-specific queues (colon optional), the device itself, and the processor that "owns" the device (colon optional) have been changed so that these entities all have the same name; for example, LP0:.

3. You can now initialize print processors as shareable. This means that a line printer is not permanently attached. It is attached only while the Queue Manager is passing a print job to it.

4. For RSX-11M systems, you now have to explicitly create the device-specific queues.

SHUTUP Program

1. Messages include a DECnet node name or system name.
2. The NOBROADCAST bit is honored except during the last five minutes before shutdown.
3. The message "SHUTUP operation complete" is output before the system halts.
4. A task-build option lets you specify the time-out interval for [1,2]SHUTUP.CMD.

User Environment Test Package (UETP)

1. You can specify devices to be excluded from testing.
2. You can determine how many times to run UETP.
3. You can select one or more of the five UETP tests.
4. UETP prints a message identifying the test and the time at the start and end of each test.
5. UETP searches for RSXMC.MAC.

Virtual Monitor Console Routine (VMR)

VMR can now fix and unfix tasks in system-controlled partitions on mapped systems. Both the mapped and unmapped versions of VMR can be used to fix and unfix a task.

The following are new or revised commands for VMR:

FIX — New keywords are:

/DIR	(RSX-11M-PLUS systems only.) Specifies that the task to be fixed is a directive common.
/NSF	(RSX-11M-PLUS systems only.) Specifies that the task to be fixed will be nonshuffleable.

INS — New keywords are:

/CLI	Specifies that the task being installed is a command line interpreter (CLI).
/FIX	(RSX-11S systems only.) Specifies whether memory-resident tasks on RSX-11S will be fixed in memory.
/IOP	Specifies that a privileged task can overmap the I/O page.
/SEC	(RSX-11M-PLUS systems only.) Places in secondary pool the Task Control Block for the task being installed.

	/SYNC	(RSX-11M-PLUS multiprocessor systems only.) Synchronizes task use of a common region.
	/XHR	(RSX-11M-PLUS systems only.) Specifies that the Executive's copy of a task's header will be kept in an area external to the system's dynamic storage area (pool).
SAVE	New keywords are:	
	/BOOT	Allows a system saved on an RX02 to be booted on RX01 and vice versa.
	/DENS	Specifies 1600 bpi density for the following magnetic tapes: TE16, TU16, TU45, and TU77.
SET	New or revised keywords are:	
	/ABAUD	Enables autobaud detection for a terminal's remote dial-up line.
	/BRO	Specifies whether a terminal will receive messages from the MCR BROADCAST command.
	/MAXEXT	Establishes the maximum size to which a task can extend itself by means of the Executive EXTEND TASK directive.
	/NETUIC	Specifies the UIC in which DECnet-related tasks will be stored.
	/OPT	(RSX-11M-PLUS systems only.) Enables disk I/O queue optimization for a device.
	/PLCTL	Displays or sets the pool limit parameters used by the Pool Monitor Task (PMT).
	/PUB	Displays or sets a device public (/PUB) or nonpublic (/NOPUB).
	/REMOTE	Now allows the setting of the initial baud rate of a remote dial-up line for a terminal.
	/RNDC	Defines the length of the Executive round-robin scheduling interval.
	/RNDH	Defines the highest priority that will be considered for Executive round-robin scheduling.
	/RNDL	Defines the lowest priority that will be considered for Executive round-robin scheduling.
	/SECPOL	(RSX-11M-PLUS systems only.) Displays secondary pool use in the system.

/SWPC	Defines the length of a single Executive swapping interval.
/SWPR	Defines a priority range for Executive swapping.
/TERM	New terminal types allowed: LA12, LA34, LA38, LA100, VT101, VT102, VT105, VT125, VT131, and VT132.

I/O Queue Optimization

A new chapter describes how RSX-11M-PLUS users can improve throughput of disk subsystems.

Resource Accounting

The RSX-11M-PLUS Resource Accounting description contains the following new information:

1. You can specify that task accounting be performed.
2. You can convert a transaction file to one that is readable by DATATRIEVE-11. This allows you to write a resource accounting report program in DATATRIEVE-11.
3. You can collect device statistics for disk subsystems.

CHAPTER 1

INTRODUCTION

This chapter summarizes the functions of the programs and procedures documented in the chapters of this manual. Each chapter is summarized in the order in which it is presented in the manual.

Chapter 2 - Account File Maintenance Program (ACNT)

The Account File Maintenance Program (ACNT) allows a privileged user to create and maintain an account file on multiuser protection systems. Each account entry consists of such information as an account number (the user's UIC) and password, default device (SY:), user's first and last name, and default command language interpreter (CLI). ACNT allows a privileged user to create an account file, add and delete accounts, modify accounts, list the accounts, examine an individual account, and sort the account entries.

The chapter gives an example of how to use each ACNT function.

Chapter 3 - Console Logger

The Console Logger consists of a driver (CODRV) and a task (COT...) that handle I/O to the console output device (CO:) and record time-stamped system messages on a terminal and/or in a log file. You use MCR SET commands from a privileged terminal to start and stop console logging, disable the log file or the console terminal, and reassign the log file and console terminal. You select Console Logger support during SYSGEN.

The chapter documents the console logging commands and presents an extensive terminal session with explanations of the commands used in the session.

Chapter 4 - I/O Exerciser (IOX)

The I/O Exerciser (IOX) detects and diagnoses I/O problems on the disk and tape units in your system's hardware configuration. IOX exercises Files-11 disks, non-file-structured disks, magnetic tapes, DECtapes, and cassettes. IOX performs three kinds of exercises. The IOX Command Language enables you to specify and control the exercise for the units in your system. You choose an exercise appropriate to a unit and set exercise parameters that determine how the unit is exercised.

The chapter documents each IOX command in alphabetical order and gives examples that illustrate and explain how to set parameters for the three kinds of IOX exercises.

Chapter 5 - Bad Block Replacement Control Task (RCT)

The Bad Block Replacement Control Task (RCT) handles bad block replacement and recovery on Mass Storage Control Protocol (MSCP) disks such as the RA80. Bad block handling on MSCP disks consists of four stages: detecting, notifying, replacing, and revectoring. The disk controller (UDA50) detects bad blocks and notifies the driver (DUDRV). The driver activates RCT. RCT performs the bad block replacement functions that enable the controller to revector (redirect) I/O from the bad block to the replacement block.

RCT also performs replacement and recovery on MSCP disks that went off line during bad block replacement or before the contents of a write-back cache were copied to the disk.

Chapter 6 - Resource Monitoring Display (RMD)

The Resource Monitoring Display (RMD) provides information about the active tasks in your operating system and the availability of system resources. This information includes the active tasks, their location in memory, the amount of memory they occupy, and the available pool space. RMD generates dynamic displays on CRTs and "snapshot" displays on hardcopy terminals.

There are three displays: memory, active task, and task header. There is also a help page that tells you how to switch display pages. Each display page has a setup page from which you can alter the contents of the associated display page. You can access any RMD display and alter the contents of the display from the MCR command line.

On RSX-11M-PLUS systems, a system statistics display and error log count display are also available.

The chapter contains an example and explanation of the memory display.

Chapter 7 - Setting Up and Running the Queue Manager (QMG)

The Queue Manager (QMG) has the primary function of controlling the system's line printers and the print jobs to be passed to them. Users can specify the number of copies they want printed, the time the job is to be released for printing, whether the file is to be deleted after printing, and which printer they want to use. Users can display all information about the QMG queues and can alter the characteristics of jobs in queues. Some installations may have user-written output processors that pass jobs to devices other than line printers. The nonprivileged functions of QMG are documented in the RSX-11M/M-PLUS Batch and Queue Operations Manual.

This chapter documents the privileged system management functions of QMG. Systems can have as many as 16 output queues (including the default PRINT queue) and 16 output processors. Print processors can be initialized to print a specified number of flag pages before each print job, to accept or reject print jobs specifying that lowercase printing is required, and to handle many kinds of printed forms.

On RSX-11M-PLUS systems, QMG also supports batch processing. RSX-11M-PLUS systems can have as many as 16 batch queues and 16 batch processors.

The chapter includes an illustrated explanation for setting up the Queue Manager.

Chapter 8 - The SHUTUP Program

The SHUTUP program allows a privileged user to shut down the system in an orderly fashion. SHUTUP prompts for the number of minutes to wait before shutdown, the number of minutes between shutdown messages, and the number of minutes to wait before disabling logins. Before halting the system, SHUTUP performs cleanup functions such as logging off all logged-in terminals, submitting the user-written command file LB:[1,2]SHUTUP.CMD to Indirect for execution, stopping the Queue Manager, Console Logger, and Error Logger (if present), deallocating checkpoint space, and dismounting mounted devices.

The chapter provides an example and explanation of SHUTUP output.

Chapter 9 - User Environment Test Package (UETP)

The User Environment Test Package (UETP) is used after system generation to verify that you have generated a working operating system. UETP consists of several command files that verify the presence and operation of devices, test the basic Executive features, and verify the presence of system utilities.

UETP consists of five test modules: Load Test, I/O Exerciser Test, Utilities Test, MCR Command Test, and Interactive Utilities Test. You can select which of these tests to run, indicate how many times to run UETP, select or omit extended comments, and exclude devices from testing.

The chapter contains a sample UETP terminal session.

Chapter 10 - Virtual Monitor Console Routine (VMR)

The Virtual Monitor Console Routine (VMR) is a task that contains a subset of the Monitor Console Routine (MCR) commands. You use VMR to make the same changes to the system image file on disk that you would make to the running system with MCR. Some of these changes are setting the size of pool, creating partitions, loading drivers, and installing tasks. The advantage of using VMR is that you can almost completely configure a system image file before you boot it.

The chapter presents each of the VMR commands in alphabetical order.

Chapter 11 - Command Language Interpreters

A command line interpreter, or CLI, is a task that services unsolicited commands meant for the operating system or an application. These commands are not prompted for by any task nor are they given to a task by a read operation on a terminal. The best known CLI on RSX-11M/M-PLUS is MCR, which controls the operating system.

A CLI is also an efficient means of interfacing an application command processor to any number of terminals, because no read QIO or character AST processing is involved.

This chapter introduces the concepts of CLIs on RSX-11M/M-PLUS systems. The chapter includes enough information to allow the programmer to write application-specific CLIs.

Chapter 12 - The DCL Task

The DIGITAL Command Language (DCL) is a general purpose command language syntax implemented on several DIGITAL operating systems.

This chapter explains the ways in which DCL can be installed on your system and documents the means by which the DCL task parses and translates commands. In addition, this chapter documents the Macro Meta Language, a set of MACRO-11 macros used to define DCL. An experienced programmer can add commands to DCL or remove them, or alter DCL by changing parts of DCL, such as altering defaults or eliminating qualifiers.

Chapter 13 - Pool Monitoring Support

Pool monitoring support controls the use of the system's dynamic storage region (pool). This support monitors pool levels, restricts use of pool, and notifies you when pool is near depletion. This support consists of two parts: the RSX-11M/M-PLUS Executive pool monitor code and the privileged Pool Monitor Task (PMT).

Chapter 14 - Shuffler Support

The Shuffler (SHF) is a privileged task that compacts space in system-controlled partitions when a memory allocation failure occurs. The Shuffler operates only within system-controlled partitions. This chapter describes Shuffler structure, Shuffler interaction with the operating system, ways to monitor the Shuffler, and Shuffler algorithms for the operating system.

Chapter 15 - RSX-11M-PLUS Reconfiguration Services

The reconfiguration services consist of two tasks (CON and HRC) and a driver (RD:). With the services, you can alter or display information about various system resources such as devices and memory. The main purpose of the reconfiguration services is to allow you to isolate faulty hardware so that it does not affect the system adversely.

The chapter describes the functions of CON, HRC, and RD: and describes the reconfiguration commands and gives examples of using each one. The chapter also gives procedures for recovering from various hardware errors.

Chapter 16 - RSX-11M-PLUS I/O QUEUE OPTIMIZATION

I/O Queue Optimization is used to improve the throughput of the disk subsystems used in your RSX-11M-PLUS system. This is done by making the most effective use of the list of I/O requests in the queue for a particular disk device. Three methods of optimization are provided: Nearest Cylinder, Elevator, and Cylinder Scan.

Chapter 17 - RSX-11M-PLUS RESOURCE ACCOUNTING

Resource Accounting provides a transaction file of system usage information. Information is gathered on each user in a system and on total system utilization. This information can be used to bill individual users for the resources used and for measuring overall system performance. By analyzing the data in the transaction file, you can determine such things as who used the system, for how long, how much CPU time they used, and a number of other statistics. For system utilization, you can find out the number of users, number of logons, how much input/output was done, and other system usage information. The information in the transaction file can be displayed or you can write a program to access the file and analyze the data.

The chapter contains an example of a transaction file output.

Chapter 18 - RSX-11M-PLUS SHADOW RECORDING

Shadow Recording allows your system to back up all information as it is being written to a Files-11 disk. It does so by creating two identical disks called a "shadowed pair." More than one pair of disks may be shadowed, but shadowed disk pairs cannot overlap. The first disk of the pair, the primary disk, is the original disk that exists whether or not Shadow Recording is active. You can make any disk on your system, including the system disk, the primary disk of a shadowed pair.

Appendix A - Serial Despooler (PRT)

The Serial Despooler Task (PRT...) provides a means of eliminating contention for the system line printer. Rather than waiting for the line printer to become available, the task directs the output intended for the line printer to a disk file. This appendix briefly describes the Serial Despooler Task and documents Task Builder information for it.

Appendix B - RSX-11M/M-PLUS UFD CONVENTIONS

The RSX-11M and RSX-11M-PLUS operating systems observe a set of User File Directory (UFD) conventions for files on disk. These conventions provide a uniform and consistent method of locating, allocating, and maintaining a file on disk. This appendix describes group and member numbers and their use in identifying the contents of system UFDs. A section on file-naming conventions is also included.

CHAPTER 2

THE ACCOUNT FILE MAINTENANCE PROGRAM (ACNT)

RSX-11M/M-PLUS provides an Account File Maintenance Program (ACNT) for creating and maintaining an account file on a multiuser system. The account file contains entries for all User Identification Codes (UICs) authorized within the multiuser protection system. One UIC can have several users, each user having his or her password. When a user tries to log in, the system checks the HELLO or LOGIN command parameters against the account file to determine whether or not the user should have access to the system.

2.1 CONTENTS OF THE ACCOUNT FILE

Each account in the account file contains the following information:

- UIC (the account number)
- Password
- User's default system device
- First name
- Last name
- User's Command Line Interpreter (CLI) if the system has multi-CLI support
- Date and time of the user's most recent login
- The number of times that the user has logged in to the system
- Whether the user's terminal is to be slaved after login
- On RSX-11M-PLUS systems, a session identifier
- On RSX-11M-PLUS systems, the user account number

2.2 INVOKING ACNT

Invoke ACNT, from a privileged terminal, as follows:

```
>RUN $ACNT RET
```

ACNT identifies itself and displays a list of options after you type the RUN command.

A nonprivileged user can use ACNT only to modify the account entry for his or her password. (See the RSX-11M/M-PLUS MCR Operations Manual).

2.3 ACNT OPTIONS

ACNT is an interactive program that allows privileged users the following options:

- (C) Create an account file.
- (A) Add accounts to the file.
- (E) Examine individual account entries in the file.
- (M) Modify individual account entries in the file.
- (L) List account entries in the file.
- (D) Delete account entries from the file.
- (S) Sort the account entries in the file.
- (CTRL/Z) Exit from ACNT.

ACNT prompts you for an option. Enter one of the letters indicated in the parentheses above. The program responds by requesting further input or by displaying information for the option you selected. The remainder of this chapter discusses the ACNT options and includes examples of each.

2.3.1 Creating the Account File

The Create option lets you create the account file. Create the account file immediately after generating your system to allow normal use of the system. The Create option allocates the account file (LB:[0,0]RSX11.SYS) and sets the file protection so that only privileged users and privileged tasks can access the account file.

You may copy and optionally extend the account file using the Peripheral Interchange Program (PIP) (see the RSX-11M/M-PLUS Utilities Manual). The new copy will have the default file protection. You should, therefore, immediately run ACNT. ACNT will detect the incorrect protection codes and restore proper protection.

The following example shows how to create the account file. After you invoke ACNT, it responds with a list of options and a prompt.

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option:  C (RET)
Enter maximum number of accounts: 35. (RET)
```

Enter a decimal number (n.) to specify the maximum number of account entries to be listed in the file. The system then creates an account file (LB:[0,0]RSX11.SYS) large enough to contain 35 accounts.

When the Create operation is complete, press the ESCAPE key to request a list of options. Then you can add accounts to the account file.

2.3.2 Adding Account Entries

The Add option lets you add accounts to the account file. ACNT prompts you for the following:

- UIC (the account number).
- Password.
- User's default system device.
- First name.
- Last name.
- User's CLI. If your system does not have multi-CLI support, ACNT does not prompt you for the user's CLI (MCR is the default).
- Whether the user's terminal is to be slaved after login.
- On RSX-11M-PLUS systems, a session identifier.
- On RSX-11M-PLUS systems, a user account number.

ACNT maintains the file in ascending order by UIC.

The prompts for the password, first name, and last name specify the maximum number of characters that ACNT accepts. The legal characters for user names and passwords are as follows:

- A through Z, upper- and lowercase
- 0 through 9
- ' (apostrophe)
- . (period)
- - (hyphen)
- $ (dollar sign)
- ! (exclamation point)

The default system device prompt requests a device name and unit number (optional if 0). ACNT accepts logical, physical, and pseudo device names. The acceptance of logical and pseudo device names lets you move the user disk to another drive without having to change the default device in the account file. When specifying the device name, do not include a colon (:).

RSX-11M-PLUS systems include two additional prompts: one is for a session identifier and the other is for the user account number. The session identifier and user account number are optionally used by the Resource Accounting Program (see Chapter 17). The session identifier is also used by the HELLO program (see the RSX-11M/M-PLUS MCR Operations Manual).

The following example shows how to add entries to the account file on RSX-11M-PLUS systems:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: A RET
Enter account or <ESC> for options ( N,N ): 5,15 RET
Password ( <=6 chars. ): A1B2C3 RET
Default system device ( DDU ): DB RET
First name ( <=12 chars. ): JANE RET
Last name ( <=14 chars. ): BARRETT RET
Enter user CLI (default=MCR): DCL RET
Slave terminal? [Y/N]: N RET
Session ident ( 3 chars. ): REC RET
Account number ( <=4 digits ): 117 RET
UFD DB1:[005,015]
```

In the previous example, ACNT creates an account in the account file for Jane Barrett. Jane's password is A1B2C3, her default system device is DB0:, her CLI is DCL, and her terminal is not slaved after login. Jane's session identifier is REC and her account number, 117.

On an RSX-11M system, this option adds accounts to the account file in the same manner.

ACNT remains in Add mode until you press the ESCAPE key to request a list of options.

2.3.3 Examining Account Entries

The Examine option displays two lines of detailed information for each account in the account file. The first line contains, from left to right, the following information:

- UIC
- Password
- First name
- Last name
- User's default system device
- On RSX-11M-PLUS systems, a session identifier

The second line contains the following information:

- On RSX-11M-PLUS systems, a user account number.
- Date of the last login.
- Time of the last login.
- Number of times the user has logged in to the system.
- User's CLI. If your system does not have multi-CLI support, ACNT displays the default CLI (MCR).
- Whether the user's terminal is to be slaved after login.

The following example shows how to display information for an account on an RSX-11M system:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: E RET
Enter account or <ESC> for options ( N,N ): 5,14 RET
[005,014]    007       SEAN          CONNOIT          DB0:
    03/05/81  15:05:55   00001  DCL     N
```

In this example, Sean Connoit's account is examined. The first line lists his UIC ([005,014]), password (007), first name, last name, and default system device (DB0:). The second line displays the date of his last login (03/05/81), the time of his last login (15:05:55), the number of times he has logged in (00001), his CLI (DCL), and the slave/no-slave status of his terminal (N).

The following example shows how to display information for an account on an RSX-11M-PLUS system:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: E<RET>
Enter account or <ESC> for options ( N,N ): 5,5 RET
[005,005]    SKIP      JOHN          MORAN            DB1:
     MGR
     0111    03/06/81  08:36:57   00002  DCL     N
```

In this example, John Moran's account is examined. His account includes two additional information fields, a session identifier (MGR) and a user account number (0111).

ACNT remains in Examine mode until you press the ESCAPE key to request a list of options.

2.3.4 Modifying Account Entries

The Modify option allows you to change the password, default system device, first name, and last name for an account, and the slave/no-slave status of a terminal. You can change the CLI of an account if your system includes the multi-CLI option.

In an RSX-11M-PLUS system, the Modify option also allows you to change the session identifier and the account number.

After you specify the account to be modified, the program displays an account entry and asks for verification. Type either Y (Yes) or N (No) to indicate that the correct account was or was not specified. If you specify N, ACNT searches the file for another entry with the same UIC. When you enter a Y, ACNT prompts for the entry details. Press the ESCAPE key if you do not want to change an item.

The following example shows how to modify account information on an RSX-11M system:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: M RET
Enter account or <ESC> for options ( N,N ): 5,12 RET
[005,012]    BARB       BARBARA      SPENCER              DB0:
    03/05/81  15:00:11   00006  MCR     N
This entry ? [Y/N]: Y RET
Type <ESC> to leave entry unchanged
Password ( <=6 chars. ): BES RET
Default system device ( DDU ): ESC
First name ( <=12 chars. ): ESC
Last name ( <=14 chars. ): ESC
Enter user CLI (default=MCR): DCL RET
Slave terminal? [Y/N]: ESC
```

In this example, Barbara Spencer's account is modified. Her password (BARB) is changed to BES, and her CLI (MCR) is changed to DCL.

The following example shows how to modify account information on an RSX-11M-PLUS system:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: M RET
Enter account or <ESC> for options ( N,N ): 5,6 RET
[005,006]    LIZ        ELIZABETH    FORD                 DB1:
     SEC
    0112    03/06/81  08:37:47   00001  MCR     N
This entry ? [Y/N]: N RET
[005,006]    MLH        MARY         HAZNAR               DB0:
     SEC
    0113    03/06/81  08:38:45   00001  DCL     N
This entry ? [Y/N]: Y RET
Type <ESC> to leave entry unchanged
Password ( <=6 chars. ): ESC
Default system device ( DDU ): ESC
First name ( <=12 chars. ): ESC
Last name ( <=14 chars. ): ESC
Enter user CLI (default=MCR): ESC
Slave terminal? [Y/N]: Y RET
Session ident ( 3 chars. ): REC RET
Account number ( <=4 digits ): 116 RET
```

In this example, Mary Haznar's account is modified. Her terminal is changed to a slaved terminal, her session identifier (SEC) is changed to REC, and her account number (0113) is changed to 116.

ACNT remains in Modify mode until you press the ESCAPE key to request a list of options.

2.3.5 Listing Account Entries

The List option either displays on the user's terminal or spools to the line printer all account entries in the account file or a specified group of entries. If you specify the line printer, ACNT creates the file ACCNT.DMP. This file contains the account information you specified. ACNT prints the file if the spooler is installed in your system. If the spooler is not installed, you may print ACCNT.DMP after you exit from ACNT.

The following example shows how to list account entries on an RSX-11M-PLUS system:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: L RET
Print passwords? [Y/N]: Y RET
All accounts? [Y/N]: N RET
Enter group number (N): 5 RET
Enter T - TERMINAL or L - SPOOLED LINEPRINTER: T RET
[005,005]    SKIP       JOHN          MORAN              DB1:     MGR
    0111    03/06/81  08:36:57   00002  DCL     N
[005,006]    LIZ        ELIZABETH     FORD               DB1:     SEC
    0112    03/06/81  08:37:47   00001  MCR     N
[005,006]    MLH        MARY          HAZNAR             DB0:     REC
    0116    03/06/81  08:38:45   00001  DCL     Y
[005,015]    A1B2C3     JANE          BARRETT            DB0:     REC
    0117    03/09/81  10:32:53   00001  DCL     N
```

In the previous example, ACNT lists all the account entries, for group 5, at the terminal.

On RSX-11M systems, this option lists accounts in the same manner.

When the list operation is complete, ACNT displays its options and prompts for a choice.

2.3.6 Deleting Account Entries

The Delete option eliminates individual accounts from the file. After you specify the account to be deleted, ACNT displays the account entry and requests a Y (Yes) or N (No) response for verification. If the response is N, ACNT searches for another account with the same UIC. If the response is Y, ACNT asks if you want to delete the account's UFD and other files. Another Y response deletes these files. If you enter N in response to the deletion query, the account's files and UFD are left intact.

The following example shows how to delete accounts from the account file on an RSX-11M system:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: D RET
Enter account or <ESC> for options ( N,N ): 5,12 RET
[005,012]    MLK        MAUREEN       KACZKA             DB1:
    03/06/81  08:46:51   00004  DCL     N
This entry ? [Y/N]: Y RET
Delete UFD and files? [Y/N]: Y RET
PIP DB1:[005,012]*.*;*/DE
>PIP DB1:[0,0]005012.DIR;*/DE
```

In the previous example, ACNT deletes Maureen Kaczka's UFD and files.

On an RSX-11M-PLUS system, this option deletes accounts in the same manner.

ACNT remains in Delete mode until you press the ESCAPE key to request a list of options.

2.3.7 Sorting the Account Entries

The Sort option sorts the account entries in ascending order by UIC. The following example shows how to sort the account file on RSX-11M/M-PLUS systems:

```
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: S (RET)
```

When the sort operation is complete, ACNT displays all its options and prompts for a choice.

2.3.8 Exiting from ACNT

The Exit option allows you to exit from the ACNT program. The following example shows how to exit from ACNT on RSX-11M/M-PLUS systems:

```
OPTIONS ARE: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
ENTER OPTION: (CTRL/Z) (RET)
```

When you exit from ACNT, ACNT displays the following message on your terminal:

```
OPERATION COMPLETE
```

2.4 ERROR MESSAGES

The ACNT program returns the following error messages:

ACNT -- Account already exists

Explanation: You attempted to add an account with a UIC and password that already exists in the account file.

ACNT -- Account file already exists

Explanation: You attempted to create an account file that already exists.

ACNT -- Account file error

Explanation: The program detected an I/O error while processing the account file.

ACNT -- Account file full

Explanation: The account file is full; you cannot add further accounts. See the description of the Create option for details on how to enlarge the account file.

ACNT -- Cannot find account

Explanation: You attempted to delete, examine, or modify an account that does not exist.

ACNT -- Invalid default device name

Explanation: The default system device specified is not a directory device.

ACNT -- Warning ** Device does not exist on current system. OK? [Y/N]:

Explanation: The specified default device is not in the system. If you enter Y, ACNT retains the device name as a logical device name. ACNT prompts again for the device name if you enter N.

ACNT -- Invalid option

Explanation: In response to the prompt Enter option:, you did not specify A, C, D, E, L, M, S, or CTRL/Z, which represent valid ACNT options.

ACNT -- Listing file error

Explanation: When you use the List option, ACNT opens the file SY:[current uic]ACCNT.DMP. This message indicates that ACNT cannot open the file; for example, the device is full or it does not contain a UFD for the current UIC. If the device is full, delete some files. Create a new UFD if there is no UFD for the current UIC.

ACNT -- Syntax or command input error

Explanation: You typed an invalid character. Enter the correct data.

ACNT -- Workfile - dyn. mem. exhausted

Explanation: There is insufficient dynamic memory. Install the task in a larger partition, or install it with a larger increment.

ACNT -- Workfile - virtual storage exceeded

Explanation: The workfile exceeds virtual memory.

CHAPTER 3

THE CONSOLE LOGGER

The Console Logger consists of a driver (CODRV) and a task (COT...) that handle I/O to the console output device (CO:) and record time-stamped system messages on a terminal or in a log file or both. You select support for console logging in SYSGEN.

For RSX-11M systems, console logging is available only on mapped systems.

3.1 THE CONSOLE OUTPUT DEVICE (CO:)

Your system has a console output device (CO:) and a console terminal. When the Console Logger is active, I/O to CO: is controlled by the console output task (COT...) and the console driver (CODRV). When the Console Logger is not active, CO: is a pseudo device, and I/O to CO: is controlled by the terminal driver. In both cases, output to CO: can be forwarded to a terminal that you assign as the console terminal.

3.1.1 CO: as a Pseudo Device

When the Console Logger is not active, CO: is a pseudo device. You can redirect CO: to any terminal on your system using the MCR REDIRECT command. The terminal driver controls all I/O to CO: that is redirected to the console terminal.

3.1.2 CO: with the Console Logger Active

When the Console Logger is active, the console driver (CODRV) and the console output task (COT...) control all I/O to CO:. You use a console logging command to assign the console terminal (see Section 3.4.5) rather than redirecting CO: with the MCR REDIRECT command.

There are two advantages to using the console logger to control I/O to CO:. These are as follows:

1. Console logging allows you to record messages sent to CO: in a log file. You can record messages on the console terminal and in the log file, only on the console terminal, or only in the log file. This flexibility enables you to inspect the log file at any time and to free a terminal when you do not need an on-line display of console logging.

2. The console logger time-stamps messages to CO: that have not been time-stamped by the task issuing the message in the form:

   hh:mm:ss hours, minutes, and seconds.

   Also, prior to outputting the first message after a change of date, the Console Logger outputs the current time and date.

3.2 INSTALLING COT... AND LOADING CODRV

If you select console logging in SYSGEN, SYSGEN installs the console output task (COT...) and loads the console driver (CODRV) by including the appropriate commands in SYSVMR.CMD. You only need to install COT... and load CODRV if:

1. You have removed one or both of them and want to restore console logging.

2. You have altered SYSVMR.CMD to eliminate them.

3.3 COMMUNICATING WITH THE CONSOLE LOGGER

You can use the COT... task to communicate with the Console Logger. Tasks use the CODRV driver to communicate with the Console Logger.

3.3.1 User Communication with the Console Logger

COT... is a privileged task. From a privileged terminal, you issue console logging commands to start and stop console logging, assign the console terminal, and specify the console log file. Table 3-1 summarizes the console logging commands. There are no DCL equivalents for the console logging commands. Section 3.4 discusses each console logging command in detail.

3.3.2 Task Communication with the Console Logger

Tasks communicate with the console driver (CODRV) by issuing QIOs. CODRV handles QIOs to CO: for reading, writing, attaching, and detaching. All other I/O functions to CO: are passed to the terminal driver as if they were issued directly to the console terminal.

3.3.2.1 **Writing to CO:** - Tasks can issue QIOs to write system messages to the console terminal and/or console log file. For example, HELLO issues a QIO to write a message to the console terminal each time a user logs in to the system.

Messages to CO: may begin with a CTRL/G to ensure that COT... sends the message to the most recently assigned console terminal even if you have disabled the console terminal using the NOCOTERM option (discussed in Section 3.4.4).

The buffer in COT... accepts messages not longer than 256(10) bytes. (Setting the buffersize of device CO: with the MCR SET /BUFFERSIZE command has no affect on this restriction.)

Table 3-1
Summary of Console Logging Commands

MCR Command	Meaning
SET /COLOG	Displays the current console terminal and logfile assignments.
SET /COLOG=ON	Starts console logging.
SET /COLOG=OFF	Stops console logging.
SET /COLOG/NOC[OTERM]	Disables the console terminal.
SET /COLOG/COT[ERM][=TTnn:]	Changes the console terminal assignment.
SET /COLOG/NOLOGFILE	Disables the logfile.
SET /COLOG/LOG[FILE][=[filespec]]	Changes the logfile assignment.

3.3.2.2 **Reading from CO:** - Tasks can issue QIOs to read input from the console terminal. For example, an operator can use the console terminal to respond to prompts by a system task that require user input. COT... does not record messages in the log file that are read from the console terminal.

3.3.2.3 **Attaching and Detaching CO:** - Tasks may attach to CO:; however, the attach is a NOP. Though tasks may attach or detach CO: they are not allowed exclusive access to CO:.

3.4 COMMAND DESCRIPTIONS

You use console logging commands to perform the following functions:

- Display the current console terminal and log file assignments
- Start and stop console logging
- Change the console terminal and log file assignments
- Disable the console terminal or the log file

The general format of console logging commands in MCR is:

```
>SET /COLOG[MCRoption]...
```

SET

The MCR SET command.

COLOG

Console logging keyword.

MCRoption

One of the console logging functions. You can truncate these options to three letters.

3.4.1 Displaying the Current Console Terminal and Log File Assignments

To display the current console terminal and log file assignments, type:

```
>SET /COLOG
```

Use this option to determine if the Console Logger is active. When the Console Logger is active, COT... displays the current console terminal and/or log file assignments. For example:

```
>SET /COLOG
COT --
Console = TT000:
Logfile = DB000:[1,4]CONSOLE.LOG;1
```

If the Console Logger is not active, the display option indicates the following:

```
COT --
Console = None
Logfile = None
```

3.4.2 Starting Console Logging

The following command line starts console logging:

```
>SET /COLOG=ON
```

When you start console logging, COT... retains the current console terminal assignment. If you have just booted the system, COT... assigns CO: to TT0:. The default location and name of the log file is LB:[1,4]CONSOLE.LOG. COT... always creates a new version of the default log file when you start console logging.

3.4.3 Stopping Console Logging

The following command line stops console logging:

```
>SET /COLOG=OFF
```

When you stop console logging, the current console terminal assignment is retained. Use the MCR DEV command to display the current console terminal. Section 3.5 discusses how you can use the MCR DEV command to monitor how CO: is redirected.

NOTE

Do not abort COT.... Aborting COT... causes severe pool fragmentation.

3.4.4 Disabling the Console Terminal

To disable the console terminal, type:

```
>SET /COLOG/NOC[OTERM]
```

This command line enables you to maintain a log file of system messages without using a terminal as the console terminal. This is especially useful if your system does not have many terminals or if you cannot spare a terminal to dedicate to receiving system messages.

You can disable the console terminal by assigning the console terminal to the null device (NL:).

Once you have disabled the console terminal with the NOCOTERM option, the command

```
>SET /COLOG/COTERM
```

restores the most recent console terminal assignment.

3.4.5 Reassigning the Console Terminal

The following command line reassigns the console terminal:

```
>SET /COLOG/COT[ERM][=TTnn:]
```

where TTnn: is the newly assigned console terminal. When you boot your system, the default console terminal assignment is TT0:. When you start console logging, the default console terminal assignment is the current console terminal.

3.4.6 Disabling the Log File

To disable the log file, type:

```
>SET /COLOG/NOLOGFILE
```

This command line enables you to continue recording system messages time-stamped by COT... at the console terminal without recording the messages in a log file.

3.4.7 Reassigning the Log File

The following command line reassigns the log file:

```
>SET /COLOG/LOG[FILE][=[filespec]]
```

where filespec is any legal file specification as described in the RSX-11M/M-PLUS Utilities Manual. The default filespec is LB:[1,4]CONSOLE.LOG.

To change only part of the log file filespec, type only that part of the filespec after the equal sign that you want to change. For example, if LB: is DR0: and you want to make the log file DB0:[301,55]CONSOLE.LOG, type:

```
>SET /COLOG/LOGFILE=DB0:[301,55]
```

If you want to make the log file TESTLOG.LOG in the default UIC on LB:, type

```
>SET /COLOG/LOGFILE=TESTLOG
```

In all cases, COT fills in the missing parts of the filespec with parts from the default filespec, LB:[1,4]CONSOLE.LOG, and creates a new version of the log file.

To create a new version of the current log file, type:

```
>SET /COLOG/LOGFILE=
```

If you have disabled the log file with the NOLOGFILE option, this command line restores the most recent log file assignment. Although a new version of the log file is created, the rest of the filespec does not change.

To change the log file back to the default, type:

```
>SET /COLOG/LOGFILE
```

This command line creates and opens a new version of the default log file.

3.4.8 Using Multiple Console Logging Commands

You can use multiple console logging commands in a command line. The following command line starts console logging, assigns the console terminal to TT17:, and opens a log file named TESTLOG.TST in the default UFD on LB:

```
>SET /COLOG=ON/COTERM=TT17:/LOGFILE=TESTLOG.TST
```

The following command line disables the console terminal and restores the log file to the default:

```
>SET /COLOG/NOCOTERM/LOGFILE
```

3.5 MONITORING THE STATUS OF CO:

To use the MCR DEV command to determine the status of CO:, while the console logger is active, enter:

```
>DEV CO:
```

The system responds:

```
CO0:  LOADED
```

This response means that the console driver (CODRV) is loaded and that COT... handles system messages to CO:.

Use the console logging display option to display the console terminal assignment:

```
>SET /COLOG
```

If you have disabled console logging, use the MCR DEV command to determine the terminal assigned as CO:

```
>DEV CO:
CO0:  TTnn:
```

where nn is the terminal to which the CO: pseudo device is redirected.

Before COT... exits, it redirects CO: to the current console terminal. If there is no console terminal (because you disabled it with the NOCOTERM option) when you stopped console logging, then CO: is redirected to the last terminal COT... assigned as the console terminal. In this case, the terminal driver handles messages directed to CO: and redirects them to TTnn:.

If you have disabled the console terminal with the NOCOTERM option but continue to record system messages to CO: in a log file, then use the MCR DEV command to determine the status of CO:

```
>DEV CO:
CO0:  LOADED
```

This response means that the console driver is loaded and that COT... handles all messages directed to CO:; however, system messages are not directed to a terminal. Use the display option to verify that CO: is not assigned to a terminal.

3.6 REDIRECTING CO: WITH THE MCR REDIRECT COMMAND

Use the MCR REDIRECT command to redirect CO: to a terminal only when the console logger is not active. If you try to redirect CO: while the console logger is active, you will receive the following error message:

```
>RED -- Device not redirectable
```

For example, if the Console Logger is active, the console terminal is assigned to TT2:, and you stop console logging, COT... redirects CO: to TT2: before exiting. Use the MCR DEV command to verify that CO: is redirected to TT2:. To assign the console terminal to TT0: use the MCR REDIRECT command:

```
>RED TT0:=CO:
```

If you then start console logging again, COT... will assign TT0: as the console terminal.

NOTE

Do not redirect CO: to CO:.

3.7 READING THE LOG FILE

Because the current log file is open for writing if LOGFILE is enabled, you can read it only by using PIP with the Shared Read (/SR) switch. For example, if the log file is DR0:[1,4]CONSOLE.LOG, and you type

```
>PIP TI:=DR0:[1,4]CONSOLE.LOG
```

PIP responds:

```
PIP -- Open failure on input file
DR0:[1,4]CONSOLE.LOG;1 -- Accessed for write
```

To read the log file, type:

```
>PIP TI:=DR0:[1,4]CONSOLE.LOG/SR
```

A sample log file follows:

```
16:43:19 Logout user [1,4] TT53:
16:43:35 Login  user ALPHA          [7,334] TT53:
16:45:07 Login  user BRAVO          [7,42] TT13:
16:46:21 *** DM0: -- DISMOUNT COMPLETE
16:46:32 Login  user DECNET         [240,240] TT30:
16:47:22 Login  user CHARLIE        [7,373] TT22:
16:47:38 *** LP0: -- NOT READY
16:47:49 Logout user [1,4] TT53:
16:47:58 Logout user [7,373] TT22:
16:48:01 Login  user ALPHA          [7,334] TT53:
16:49:40 Login  user DECNET         [240,240] HT0:
```

3.8 SAMPLE CONSOLE LOGGING TERMINAL SESSION

Example 3-1 shows a console logging terminal session illustrating how you use all the commands described in the previous Sections 3.4 through 3.7.

Example 3-1 Sample Console Logging Terminal Session

(This example assumes that LB: is assigned to DB0:.)

```
>SET /COLOG                                                    ❶
COT --                                                         ❶
Console = None                                                 ❶
Losfile = None                                                 ❶
DEV CO:                                                        ❷
COO: TTO:                                                      ❷
>SET /COLOG=ON                                                 ❸
>DEV CO:                                                       ❹
COO: LOADED                                                    ❹
>SET /COLOG                                                    ❹
COT --                                                         ❹
Console = TT000:                                               ❹
Losfile = DB000:[1,4]CONSOLE.LOG;1                             ❹
>SET /COLOG/COTERM=TT51:                                       ❺
>SET /COLOG/LOGFILE=DR:[301,55]TEST                            ❻
>SET /COLOG                                                    ❼
COT --                                                         ❼
Console = TT051:                                               ❼
Losfile = DR000:[301,55]TEST.LOG;1                             ❼
>SET /COLOG/NOCOT                                              ❽
>SET /COLOG                                                    ❽
COT --                                                         ❽
Console = None                                                 ❽
Losfile = DR000:[301,55]TEST.LOG;1                             ❽
>DEV CO:                                                       ❾
COO: LOADED                                                    ❾
>SET /COLOG/COTER/NOLOG                                        ❿
>SET /COLOG                                                    ❿
COT --                                                         ❿
Console = TT051                                                ❿
Losfile = None:                                                ❿
>SET /COLOG/COTERM=TT17:/LOG=                                  ⓫
>SET /COLOG                                                    ⓫
COT --                                                         ⓫
Console = TT017:                                               ⓫
Losfile = DR000:[301,55]TEST.LOG;2                             ⓫
>SET /COLOG=OFF                                                ⓬
>SET /COLOG                                                    ⓬
COT --                                                         ⓬
Console = None                                                 ⓬
Losfile = None                                                 ⓬
>DEV CO:                                                       ⓭
COO: TT17:                                                     ⓭
>RED TT51:=CO:                                                 ⓭
>DEV CO:                                                       ⓭
COO: TT51:                                                     ⓭
>SET /COLOG=ON/LOG=DB1:[7,301]                                 ⓮
>SET /COLOG                                                    ⓮
COT --                                                         ⓮
Console = TT051:                                               ⓮
Losfile = DB001:[7,301]CONSOLE.LOG;1                           ⓮
>SET /COLOG/COT=TT0:/LOG                                       ⓯
>SET /COLOG                                                    ⓯
COT --                                                         ⓯
Console = TT000:                                               ⓯
Losfile = DB000:[1,4]CONSOLE.LOG;3                             ⓯
```

(continued on next page)

Example 3-1 (Cont.) Sample Console Logging Terminal Session

```
>PIP TI:=LB:[1,4]CONSOLE.LOG                                          ⓰
PIP -- Open failure on input file
DB0:[1,4]CONSOLE.LOG;3 -- Accessed for write
>PIP TI:=LB:[1,4]CONSOLE.LOG/SR
16:43:19  Logout user [1,4] TT53:
16:43:35  Login  user ALPHA          [7,334] TT53:
16:45:07  Login  user BRAVO          [7,42] TT13:
16:46:21  *** DM0: -- DISMOUNT COMPLETE
16:46:32  Login  user DECNET         [240,240] TT30:                  ⓱
16:47:22  Login  user CHARLIE        [7,373] TT22:
16:47:38  *** LP0: -- NOT READY
16:47:49  Logout user [1,4] TT53:
16:47:58  Logout user [7,373] TT22:
16:48:01  Login  user ALPHA          [7,334] TT53:
16:49:40  Login user DECNET          [240,240] HT0:
>SET /COLOG/NOCOTERM/NOLOG                                            ⓲
COT -- Not logging on a terminal or to a file, COT exiting
>DEV CO:
CO0: TT0:
```

❶ Displays the current console terminal (none) and log file (none) assignments indicating that the Console Logger is not active.

❷ Illustrates the MCR DEV command that shows how CO: is redirected. CO: is redirected to TT0:, which means that the Console Logger is not active and that the terminal driver redirects CO: to TT0:.

❸ Starts console logging.

❹ Shows that the Console Logger is active because CODRV is loaded. The display option shows that COT... redirects CO: to TT0: and that DB0:[1,4]CONSOLE.LOG;1 is the log file. These are both the defaults when you boot your system.

❺ Shows how you assign the console terminal to TT51:.

❻ Shows how you assign the log file to DR:[301,55]TEST.LOG. COT... fills in the missing parts of the filespec with parts from the default filespec. In this case, COT... supplies the file type.

❼ Displays the current console terminal (TT51:) and log file (DR:[301,55]TEST.LOG;1) assignments resulting from command lines 5 and 6.

❽ Disables the console terminal and retains the current log file (DR:[301,55]TEST.LOG;1). The display option confirms this.

❾ Shows that when you disable the console terminal with the NOCOTERM option, the console logger is still active (CODRV is loaded), but CO: is not redirected to any terminal.

❿ Illustrates the multiple commands that restore the most recent console terminal assignment (TT51:) and disable the log file. The display option confirms this.

⓫ Illustrates the multiple commands that assign the console terminal to TT17: and restore the most recent log file. The display option confirms these assignments. The log file version number increments.

⓬ Stops console logging. The display option confirms that the console logger is not active. CO: is redirected to TT17:, which was the last console terminal assignment before COT... exited.

⓭ Redirects CO: to TT51: with the MCR REDIRECT command and confirms this with the MCR DEV command.

⓮ Illustrates multiple commands on a command line. Starts console logging, retaining the current console terminal assignment (TT51:) redirected with the MCR REDIRECT command in step 13, and opening a log file DB001:[7,301]CONSOLE.LOG;1. The display option confirms these assignments.

⓯ Illustrates multiple commands on a command line. Assigns the console terminal to TT0: and restores the default log file. The display option confirms these assignments. The version number of the log file increments.

⓰ Illustrates what happens when you try to read an open log file without the Shared Read switch (/SR).

⓱ Shows how to read an open log file using the Shared Read switch. Each message in the log file begins with a time stamp.

⓲ Disables the console terminal and the log file. COT... exits because no console logging is in progress. COT... displays a message informing you that it is exiting. The MCR DEV command shows that CO: is redirected to TT0:, the console terminal assignment before COT... exited.

3.9 ERROR MESSAGES

COT -- LUN assignment to console terminal failed

Explanation: You specified an incorrect terminal number as the console terminal.

User Action: Use the COTERM option to assign the console terminal to a terminal in your hardware configuration.

COT -- Illegal console terminal

Explanation: You tried to assign a device that is off line or is not a terminal to CO:.

On RSX-11M-PLUS systems, the device may be illegal because it is a spooled terminal.

User Action: Use the COTERM option to assign a terminal in your hardware configuration to CO:

COT -- Console driver not loaded, COT exiting

Explanation: The console driver, CODRV, is not loaded.

User Action: Load the console driver and start console logging.

COT -- Maximum consecutive write errors, disabling logfile

Explanation: An error occurred on five consecutive write operations to the log file. COT... closes the log file and disables logging to the log file.

User Action: Determine the problem that caused the write errors and correct it. Then restart COT...

COT -- Not logging on a terminal or to a file, COT exiting

Explanation: You have disabled the console terminal with the NOCOTERM option and the log file with the NOLOGFILE option.

User Action: If you did not intend to stop console logging, start it again specifying either a log file, console terminal, or both.

COT -- Unable to redirect CO: to a terminal - CO: is offline, COT exiting

Explanation: CO: is off line. There is a problem with the most recent console terminal assignment. It may be off line or not a terminal.

On RSX-11M-PLUS systems, the problem may be that a spooled terminal was assigned.

User Action: Start COT... and specify a legal console terminal with the COTERM option. Then stop console logging.

COT -- Syntax error

Explanation: You made an error in entering a console logging command.

User Action: Check the command description in Section 3.4, and enter the command correctly.

COT -- Will attempt console LUN assignment to TTnn:

Explanation: This error message may occur when you stop console logging. If COT... cannot redirect the console terminal to the terminal currently assigned as the console terminal, it tries to redirect CO: to the previous console terminal assignment. If that fails, it tries TT0:. If that fails, it generates the error message:

$COT -- LUN assignment to console terminal failed

User Action: Start console logging again, specifying a legal console terminal. Then stop console logging.

COT -- Illegal command - Console logging has not been turned on

Explanation: You issued a console logging command before starting console logging.

User Action: Start console logging and issue the command.

COT -- Logfile x error

filespec -- QIO Error Code -nn.

Explanation: An I/O error occurred while COT... was performing an operation on the log file. The function being performed, x, is WRITE, OPEN, or CLOSE. The filespec is any legal filespec designated for the log file. The error code, -nn., is an I/O Error Code described in the IAS/RSX-11 I/O Operations Reference Manual.

User Action: Look up the I/O Error Code in the IAS/RSX-11 I/O Operations Reference Manual and take the appropriate action.

CHAPTER 4

THE I/O EXERCISER (IOX)

The I/O Exerciser (IOX) detects and diagnoses I/O problems on the disk and tape units in your hardware configuration. Using the IOX Command Language, you can execute, control, and monitor I/O exercises on device units that you specify for exercising.

This chapter has a narrative part and a reference part. The narrative part presents the information you need to run the I/O Exerciser in the order you need that information. The reference part describes each command in the IOX Command Language.

4.1 INTRODUCTION

This section serves the following functions:

- Provides an overview of the steps involved in running an I/O exercise
- Describes how the I/O Exerciser exercises different types of devices
- Distinguishes between file-structured and nonfile-structured volumes
- Categorizes the IOX commands by function

4.1.1 Overview of the Steps Involved in Running an I/O Exercise

Before you run an I/O exercise, you need to consider the following:

- Kinds of devices to exercise (magnetic tapes, cassettes, disks, and DECtapes)
- Kind of exercise to run for each device (IOX has three kinds of exercises)
- Exercise parameters to set for each exercise (such as whether or not to test random blocks on disks, how many records to test on magnetic tapes, and how long to run the exercise)

The steps involved in running an I/O exercise include your making decisions about these three aspects of testing. Using the IOX Command Language, you implement these decisions and run an exercise.

Running an I/O exercise consists of the following basic steps:

1. Invoke IOX.
2. Determine which devices you want to test and include them in your "test configuration."
3. Determine the kind of exercise to run on each unit in your test configuration.
4. Set exercise parameters to affect how IOX tests the units you want to exercise.
5. Begin the exercise and possibly interrupt the exercise to change steps 2, 3, and 4.
6. Monitor the activity and error reports that IOX generates during an exercise.
7. Exit IOX or repeat this process beginning at step 2, 3, 4, or 5 as appropriate.

4.1.2 How IOX Exercises Different Types of Devices

IOX performs IOX exercising on disks, magnetic tapes, cassettes, and DECtapes. You can use IOX to exercise any disk or tape unit in your hardware configuration that has a mounted volume and is not allocated to another user. By default, IOX can exercise 28 units simultaneously. You can edit the task build command file (see Section 4.11) to enable IOX to test as many as 250 units simultaneously.

In general, I/O exercises consist of four steps:

1. Writing 1024. bytes of data made up of a specific data pattern to a mounted volume
2. Reading the data just written
3. Checking that the data read matches the data written
4. Reporting mismatches of data

IOX repeats this process until the exercise has completed.

4.1.2.1 **How IOX Exercises Disks** - IOX writes a buffer of data to a disk and then reads that data from the disk. You can set exercise parameters that determine: the data pattern to be written, the range of blocks to be tested, and whether to test blocks sequentially or randomly.

4.1.2.2 **How IOX Exercises Magnetic Tapes and Cassettes** - IOX uses the same algorithm to exercise magnetic tapes and cassettes. IOX rewinds the tape, writes one record, space reverses one record, and then reads the record that was just written. This repeats until the number of records that you specified for testing are read. At the end of the exercise, IOX writes two tape marks and rewinds the tape. If you are testing more than one unit, IOX continues testing on those units while the tape is rewinding. You can set exercise parameters that determine: the data pattern to be written, the size of the data buffer, and the number of records to be exercised.

4.1.2.3 **How IOX Exercises DECtapes** - IOX exercises DECtapes by writing forward/reading forward, then writing reverse/reading reverse until the end of the tape is reached. IOX exercises DECtapes using a data buffer of two blocks (1024. bytes). You can set a test parameter only to determine the data pattern that IOX writes to the DECtape.

4.1.2.4 **Nonfile-Structured and File-Structured Volumes** - IOX exercises devices with two kinds of volumes: nonfile-structured (NFS) and file-structured (Files-11). The meanings of these terms may differ slightly from other manuals within the RSX-11M/M-PLUS set. In the context of the I/O Exerciser, they are defined as follows. All tapes are NFS volumes. (Throughout this chapter, the term "NFS" describes devices and the testing of devices that contain nonfile-structured volumes.) Disks can be either NFS or Files-11 volumes. Files-11 volumes are only those disks that have been initialized with the MCR INITIALIZE command and therefore have a home block and a Files-11 structure.

4.1.3 The Four Categories of IOX Commands

The IOX Command Language has four categories of commands:

Function Commands

> Select devices for testing, deselect devices from testing, and initiate tasks to run in parallel with IOX.

Control Commands

> Start and stop IOX processing, switch IOX modes of operation, and exit from IOX to the operating system monitor.

Display Commands

> Display current default parameters, get help text for IOX commands, and display status and activity reports during an I/O exercise.

Parameter Commands

> Set default parameters that affect how IOX tests the devices you select for testing.

Table 4-1 lists the IOX commands under their appropriate categories.

Table 4-1
Four Categories of IOX Commands

Function	Control	Display	Parameter
DESELECT EXECUTE FILES11 SELECT VERIFY	ABORT CTRL/C EXIT PROCEED RESTART START	BADBLOCKS[1] CONFIGURE[1] DENSITY[1] HELP PARAMETERLIST PATTERN[1] PRINTSUMMARY RANGE[1] SPY	BADBLOCKS[1] BUFFERSIZE COMPAREDATA CONFIGURE[1] CONTROL_C DENSITY[1] ERRORLIMIT INTERLEAVE LOGFILE PATTERN[1] RANDOM RANGE[1] RECORDS REPORTERRORS RETRIES RUNTIME SUMMARYTIME TEMPORARYFILE VOLUMECHECK WAIT WRITECHECK

1. These commands have two forms: one that displays the current default for the parameter and one that sets the current default for the parameter.

4.2 INSTALLING AND INVOKING IOX

IOX is task built as a nonprivileged, checkpointable task that runs at a priority of 50 in the GEN partition. You can run IOX by using the MCR or DCL RUN command or by installing and invoking IOX.

4.2.1 Installing IOX

Install IOX from a privileged terminal with the MCR or DCL INSTALL command.

For RSX-11M systems that do not support the Extend Task Directive (see the RSX-11M/M-PLUS Executive Reference Manual), install IOX with an increment large enough to accommodate all buffers. Assuming all initial IOX defaults, allocate 600. words per unit to be tested simultaneously. For example, to test three devices simultaneously, install IOX as follows:

```
>INS $IOX/INC=1800.
```

For RSX-11M systems that support the Extend Task Directive, install IOX without an increment.

For RSX-11M-PLUS systems, install IOX by typing:

```
>INS $IOX
```

4.2.2 Invoking IOX

You can invoke IOX from any terminal.

On RSX-11M single user systems, mount all disks that you want to exercise as file-structured disks. If you have a multiuser system, first allocate all units that you want to exercise. Then mount only those disks that you want to exercise as file-structured disks.

On RSX-11M-PLUS systems, allocate and mount all units that you want to exercise. Mount nonfile-structured volumes using the /FOREIGN switch for the MOUNT command.

4.2.2.1 Invoking an Installed Version of IOX - If IOX is installed, invoke it with the command:

```
>IOX
IOX>
```

IOX responds by prompting you with the name of the task that you assigned to IOX when you installed it. If you did not specify a task name when you installed IOX, its task name is ...IOX. After receiving the prompt, you can enter IOX commands.

4.2.2.2 Invoking IOX When It Is Not Installed - If IOX is not installed, invoke it with the MCR or DCL RUN command. In this case, IOX uses your terminal number to name the task and prompts you using this task name. For example, if from TT10: you type

```
>RUN $IOX
```

you will be prompted:

```
TT10>
```

When you receive this prompt, you can enter IOX commands.

For RSX-11M systems, if your system does not support the Extend Task Directive, then run the I/O Exerciser with an increment large enough to accommodate all buffers. Assuming all initial IOX defaults, allocate 600. words per device to be tested simultaneously. For example, to test four devices simultaneously, invoke IOX as follows:

```
>RUN $IOX/INC=2400.
```

When you invoke an uninstalled version of IOX on RSX-11M systems that support the Extend Task Directive, you do not need to run the task with an increment.

When you invoke an uninstalled version of IOX on RSX-11M-PLUS systems, you do not need to run the task with an increment.

4.3 USING INDIRECT COMMAND FILES WITH IOX

You can use indirect command files as input to IOX. IOX accepts one level of indirect command files. You can use indirect command files to set up your test configuration, choose devices for testing, set exercise parameters, and start and exit the exercise. You cannot interrupt an I/O exercise to enter IOX commands from an indirect command file. The default file type is "CMD."

For example, use an editor to create a file named TEST.CMD that contains the following IOX command sequence:

```
CONFIGURE DB2:      ; Adds DB2: to the test configuration.

FILES11 DB2:        ; Chooses DB2: for an exercise that preserves
                    ; the contents and structure of the disk.

RUNTIME 10          ; Sets an exercise parameter that runs
                    ; for 10 minutes.

START               ; Starts the exercise.

EXIT                ; Exits from IOX to the operating system
                    ; monitor when the exercise completes.
```

After you invoke IOX, type:

```
IOX>@TEST
```

IOX exercises DB2: (preserving its contents and structure) for 10 minutes and then exits to the operating system monitor.

4.4 SETTING UP YOUR TEST CONFIGURATION

Before you can exercise a unit, you must first include it in your test configuration. When you include a unit in your test configuration, IOX adds the unit to its list of units available for exercising and creates an empty data structure for the unit. IOX includes in the default test configuration one of each device type supported by DIGITAL as unit "0".

To add units to the test configuration, you use the CONFIGURE command. The syntax for this command is

```
CONFIGURE [ddnn:[=type]] ....
```

where

dd

Device mnemonic

nn

Device number

type

Device type. Either DISK or TAPE.

You can add any unit to the test configuration and assign a logical device name to it by using the MCR ASN (ASSIGN) command.

The CONFIGURE command with no arguments displays the current test configuration. For example:

```
IOX>CONFIGURE

Disks:
DB0  DD0  DF0  DK0  DL0  DM0  DP0  DR0  DS0  DU0
DX0  DY0  EM0  SY0

Tapes:
CT0  DT0  MF0  MM0  MS0  MT0
```

To add DK1: and MM1: to your configuration, type:

```
IOX>CONFIGURE DK1:  MM1:
```

This command line adds DK1: to the configuration as a disk unit (because DK0: is already configured as a disk) and MM1: to the configuration as a tape unit (because MM0: is already configured as a tape). Section 4.12 describes the CONFIGURE command in detail.

4.5 SELECTING DEVICES FOR EXERCISING

After having configured all the units you want to exercise, you select units for exercising. When you select a unit, IOX fills in the data structure created when you configured the unit. This data structure contains all the information IOX needs to exercise the specified unit. You select units for exercising with the FILES11, SELECT, and VERIFY commands. These commands require device and unit specifications within their command lines.

4.5.1 The FILES11 Command

The FILES11 command selects a disk unit with a mounted Files-11 volume for an I/O exercise. Exercising with FILES11 does not affect the contents of the disk being tested.

4.5.2 The SELECT Command

The SELECT command selects a unit with an NFS volume for an I/O exercise. Exercising with SELECT destroys the contents of the volume being exercised.

4.5.3 The VERIFY Command

The VERIFY command selects a disk unit with a mounted NFS volume for an I/O exercise. VERIFY does not write on the volume being exercised. Exercising with VERIFY does not affect the contents of the volume being exercised because VERIFY only reads from the volume.

4.6 EXERCISE PARAMETERS

You set parameters that determine how IOX exercises units. These exercise parameters determine aspects of testing such as how long an exercise is to run, how many records are tested on magnetic tapes, what range of blocks are tested on disks, and whether output is directed to your terminal or to a log file. There are two kinds of exercise parameters: general and device dependent.

4.6.1 General Exercise Parameters

The following commands set general exercise parameters and therefore affect all IOX testing:

```
CONFIGURE
CONTROL_C
LOGFILE
PATTERN
REPORTERRORS
RUNTIME
SUMMARYTIME
```

4.6.2 Device-Dependent Exercise Parameters

You use the commands listed in this section to set default parameters for the units that you select for exercising. You can include these commands within the FILES11, SELECT, and VERIFY command lines to override the default parameter for the specified unit.

4.6.2.1 **Parameter Commands for Files-11 Disks with FILES11** - The following commands affect the exercising of disks that you select with the FILES11 command:

```
COMPAREDATA
ERRORLIMIT
INTERLEAVE
RANDOM
TEMPORARYFILE
```

You can use these commands to set the default parameters before you use the FILES11 command to select a unit. You can also use these commands within the FILES11 command line to override the default parameters for the unit being selected.

4.6.2.2 **Parameter Commands for NFS Volumes with SELECT** - The following commands affect the exercising of NFS volumes that you select with the SELECT command:

```
BADBLOCKS       RANDOM
BUFFERSIZE      RANGE
COMPAREDATA     RECORDS
DENSITY         RETRIES
ERRORLIMIT      VOLUMECHECK
INTERLEAVE      WRITECHECK
```

You use these commands with the SELECT command to set default parameters for exercising a unit with an NFS volume. You can use all but BADBLOCKS within the SELECT command line to override the current default settings for the device being selected.

Table 4-2 shows the parameters for exercising different types of devices.

Table 4-2
Parameter Commands by Device Type for SELECT

NFS Disks	Cassettes	DECtapes	Magnetic Tapes
BADBLOCKS[1] BUFFERSIZE COMPAREDATA ERRORLIMIT INTERLEAVE RANDOM RANGE RETRIES VOLUMECHECK WRITECHECK	BUFFERSIZE COMPAREDATA ERRORLIMIT RECORDS	COMPAREDATA ERRORLIMIT	BUFFERSIZE ERRORLIMIT DENSITY RECORDS RETRIES

1. Cannot be used within SELECT command line.

4.6.2.3 **Parameter Commands for NFS Volumes with VERIFY** - The following commands affect the exercising of NFS disks that you select with the VERIFY command:

BADBLOCKS
BUFFERSIZE
ERRORLIMIT
INTERLEAVE
RANDOM
RANGE
RETRIES

You use these commands with the VERIFY command to set default parameters for testing a unit with an NFS disk. You can use all but BADBLOCKS within the VERIFY command line.

4.7 SETTING PARAMETERS FOR EXERCISING FILES-11 VOLUMES

This section shows you how to use IOX to exercise an RL02 with a Files-11 structure. Example 4-1 uses the following IOX commands:

PARAMETERLIST

Displays current default parameters, IOX buffer space statistics, and device-dependent parameters for devices you have selected for testing.

RUNTIME

Sets the length of time (in minutes) that IOX exercises the unit(s) you have selected for testing.

FILES11

Selects a disk with a mounted Files-11 volume for an I/O exercise that preserves the contents and structure of the disk.

TEMPORARYFILE

Sets the size of the temporary file that IOX uses for testing Files-11 disks with the FILES11 command.

START

Starts exercising units that you have selected for testing.

Example 4-1 Exercising an RL02 with the FILES11 Command

```
>ALL DL0:                          ❶
>MOU DL0: LOUIE                    ❷
>IOX                               ❸

IOX>PARAMETERLIST
BUFFERSIZE= 1024.                  COMPAREDATA= YES
ERRORLIMIT= 10.                    CONTROL_C= YES
INTERLEAVE= 4.                     LOGFILE= NO
PATTERN= 0.                        RANDOM= YES
RECORDS= 1024.                     REPORTERRORS= YES
RUNTIME= 5.                    ❼   RETRIES= YES                  ❹
SUMMARYTIME= 1.                    VOLUMECHECK= YES
TEMPORARYFILE= 500.            ❽   WAIT= YES
                                   WRITECHECK= NO

Buffer space usage= 3290.:1924.:1924.:1.

IOX>RUNTIME 3                      ❺
IOX>FILES11 DL0: TEMP=800          ❻
IOX>PARAMETER

Device  Mode  Buffersize  Filesize/Records/Range  Parameters                ❾
------  ----  ----------  ----------------------  ----------
DL0:    F11      512.             800.      ❿      COM RET RAN ERR=10.

BUFFERSIZE= 1024.                  COMPAREDATA= YES
ERRORLIMIT= 10.                    CONTROL_C= YES
INTERLEAVE= 4.                     LOGFILE= NO
PATTERN= 0.                        RANDOM= YES
RECORDS= 1024.                     REPORTERRORS= YES
RUNTIME= 3.                    ❼   RETRIES= YES
SUMMARYTIME= 1.                    VOLUMECHECK= YES
TEMPORARYFILE=500.             ❽   WAIT= YES
                                   WRITECHECK= NO

Buffer space usage= 3290.:1308.:1308.:1.

IOX>START                          ⓫
```

The command sequence in Example 4-1 does the following:

❶ Allocates drive DL0:.

❷ Mounts the disk with label "LOUIE" in DL0:.

❸ Invokes an installed version of IOX with default task name, ...IOX.

❹ Displays the initial default parameters with PARAMETERLIST.

❺ Changes the default RUNTIME to three minutes.

❻ Selects DL0: for FILES11 testing with a temporary file size of 800 blocks.

Note the difference between the defaults displayed by the PARAMETERLIST command before and after you changed RUNTIME and slected DL0:.

❼ Only the RUNTIME default has changed.

❽ The default for TEMPORARYFILE has not changed.

❾ After you selected DL0: for testing, PARAMETERLIST contains an extra line at the beginning that displays all device-dependent parameters for the selected device. Each selected device has its own display line. There are five fields of parameters.

❿ Field four shows that the temporary file size used for FILES11 testing for DL0: is 800. blocks instead of the 500.-block initial default. During FILES11 testing, this field refers only to temporary file size.

⓫ The START command begins FILES11 testing on DL0: for a RUNTIME of three minutes and uses a temporary file size of 800. blocks.

4.8 SETTING PARAMETERS FOR EXERCISING NFS VOLUMES

This section shows examples for the command sequences that test a magnetic tape with SELECT and an NFS disk with VERIFY. Both examples are explained in detail.

4.8.1 Exercising a Magnetic Tape with the SELECT Command

The command sequence in Example 4-2 tests a magnetic tape by using a buffer of 8K bytes, which contains the data pattern 052652, to transfer data.

Example 4-2 illustrates the following commands:

PARAMETERLIST

Displays current default parameters, IOX buffer space statistics, and device-dependent parameters for devices you have selected for testing.

PATTERN

Sets the pattern that IOX writes and reads during an exercise.

BUFFERSIZE

Sets the amount of data that IOX transfers for every read or write request when testing NFS disks, magnetic tapes, and cassettes.

DENSITY

Sets the density and characteristic word of the specified magnetic tape or displays the current density and characteristic word for the specified magnetic tape unit.

SELECT

Selects a unit with a mounted NFS scratch volume for an I/O exercise that destroys the contents of the volume

RETRIES

Determines (for magnetic tapes and NFS disks) whether an I/O driver repeats an I/O operation after the driver receives errors during the operation.

ERRORLIMIT

Sets the maximum number of errors that IOX tolerates between interval reports before terminating testing on a device.

RECORDS

Specifies the number of records to be exercised on magnetic tapes and cassettes.

START

Starts exercising units that you have selected for testing.

EXIT

Terminates IOX processing and exits from IOX.

Example 4-2 Exercising a Magnetic Tape with the SELECT Command

```
>ALL MM0:
>MOU MM0:                                    ❶
>IOX
IOX>PARAMETERLIST

BUFFERSIZE= 1024.          ❽      COMPAREDATA= YES
ERRORLIMIT= 10.                   CONTROL_C= YES
INTERLEAVE= 4.                    LOGFILE= NO
PATTERN= 0.                ❾      RANDOM= YES
RECORDS= 1024.                    REPORTERRORS= YES          ❷
RUNTIME= 5.                       RETRIES= YES
SUMMARYTIME= 1.                   VOLUMECHECK= YES
TEMPORARYFILE= 500.               WAIT= YES
                                  WRITECHECK= NO

Buffer space usage= 3290.:1924.:1924.:1.
```

(continued on next page)

Example 4-2 (Cont.) Exercising a Magnetic Tape with the SELECT Command

```
IOX>PATTERN 12                    ❸
IOX>BUFFER 8192                   ❹
IOX>DENS MM0:
  Density= 1600 BPI    Characteristic Word= 004004 (octal)   ❺
IOX>SELECT MM0: RETRIES=NO DENSITY=800 ERROR=100 REC=20000   ❻
IOX>PARAM

Device  Mode  Buffersize  Filesize/Records/Range  Parameters
------  ----  ----------  ----------------------  ----------
MM0:    NFS      8192.  ⓫        20000.      ⓬   COM ERR=10.  ⓾

BUFFERSIZE= 8192.          ❽     COMPAREDATA= YES
ERRORLIMIT= 10.                  CONTROL_C= YES
INTERLEAVE= 4.                   LOGFILE= NO
PATTERN= 12.               ❾     RANDOM= YES
RECORDS= 1024.                   REPORTERRORS= YES
RUNTIME= 5.                      RETRIES= YES
SUMMARYTIME= 1.                  VOLUMECHECK= YES
TEMPORARYFILE= 500.              WAIT= YES
                                 WRITECHECK= NO

Buffer space usage= 36058.:8188.:18292.:3.

IOX>DENSITY MM0:
  Density= 800 BPI     Characteristic Word= 000000 (octal)
IOX>START                 ⓭
IOX -- MM0: bad block, 112 (decimal) 160 (octal)

IOX -- Summary of exerciser activity at 20-JUL-81 16:50:40
  TIME -- Run: 1 min.   Elapsed: 1 min.   Remaining: 4 min.
  MM0: -- 230. R/W requests totaling 230. records with 1 errors
  AST's executed: 346.
  Idle-loop iterations: 344.
     .
     .
     .
     .

IOX -- MM0: bad block, 486 (decimal) 746 (octal)
IOX -- MM0: bad block, 551 (decimal) 1047 (octal)

IOX -- Summary of exerciser activity at 20-JUL-81 16:54:50
  TIME -- Run: 1 min.   Elapsed: 5 min.   Remaining: 0 min.
  MM0: -- 217. R/W requests totaling 217. records with 2 errors
  AST's executed: 325.
  Idle-loop iterations: 321.
 A C C U M U L A T E D   T O T A L S :
 - - - - - - - - - - -   - - - - - - -
  TIME -- Run: 5 min.   Elapsed: 5 min.   Remaining: 0 min.
  MM0: -- 1,121. R/W requests totaling 1,121. records with 3 errors
  AST's executed: 1,686.
  Idle-loop iterations: 1,670.

IOX>EXIT
$
```

❼

The command sequence in Example 4-2 does the following:

❶ Allocates drive MM0:, mounts the magnetic tape, and invokes an installed version of IOX.

On RSX11M-PLUS systems, you mount the tape wiht the /FOREIGN switch.

❷ Displays the initial defaults.

❸ Selects pattern number 12 to be written to the tape.

❹ Changes the default BUFFERSIZE from 1024. to 8192. (8K).

❺ Displays the default density and characteristic word on the magnetic tape mounted in drive MM0:.

❻ Selects MM0: for NFS testing with SELECT, overriding the defaults for RETRIES, ERRORLIMIT and RECORDS, and sets the tape density at 800.

❼ Checks the parameters with PARAMETERLIST and DENSITY.

❽ Changes the BUFFERSIZE default.

❾ Changes the PATTERN default.

❿ Displays the device display line for MM0:.

⓫ Shows the current BUFFERSIZE of MM0:.

⓬ Shows that the number or records to be tested is 20000.

⓭ Starts NFS testing with SELECT on MM0:.

The remainder of Example 4-2 shows interval reports, error reports, and an accumulated totals generated at your terminal during this exercise. These reports are discussed fully in Section 4.10. When the exercise has finished, you exit to the operating system monitor unless you want to do more testing.

4.8.2 Exercising with the VERIFY Command

The command sequence in Example 4-3 performs oscillating seeks between blocks 1 and 1000 to align disk heads. It reads only four bytes per data transfer, which is the minimum. No error or summary reporting is required. The exercise runs until you abort it.

Example 4-3 illustrates the following commands:

PARAMETERLIST

Displays current default parameters, IOX buffer space statistics, and device-dependent parameters for devices you have selected for testing.

VERIFY

Selects a mounted disk for an NFS exercise that reads buffers of data without writing on the disk, performing data comparisons, or destroying the contents of the disk.

RANGE

Sets the minimum and maximum block numbers for NFS testing on the specified disk.

BUFFERSIZE

Sets the amount of data that IOX transfers for every read or write request when testing NFS disks, magnetic tapes, and cassettes.

RANDOM

Directs IOX to select either random or sequential blocks for disk testing.

INTERLEAVE

Sets the number of blocks to be skipped before IOX writes the next buffer of data for disk testing (applies only to sequential testing which you determine with the RANDOM command).

ERRORLIMIT

Sets the maximum number of errors that IOX tolerates between interval reports before terminating testing on a device.

REPORTERRORS

Enables or disables error reports.

SUMMARYTIME

Determines how often (in minutes) IOX will output interval reports.

RUNTIME

Sets the length of time (in minutes) that IOX exercises the unit(s) you have selected for testing.

LOGFILE

Determines whether IOX directs activity and error reports to your terminal or to a log file.

START

Starts exercising units that you have selected for testing.

CTRL/C

With CONTROL_C enabled, allows you to enter IOX commands while an exercise is in progress.

ABORT

Terminates execution of an I/O exercise.

Example 4-3 Exercising an NFS RK07 with the VERIFY Command

```
>ALL DM0:
>MOU DM0:                                                    ❶
>IOX
IOX>PAR
BUFFERSIZE= 1024.                COMPAREDATA= YES
ERRORLIMIT= 10.                  CONTROL_C= YES
INTERLEAVE= 4.                   LOGFILE= NO
PATTERN= 0.                      RANDOM= YES
RECORDS= 1024.                   REPORTERRORS= YES
RUNTIME= 5.                      RETRIES= YES                ❷
SUMMARYTIME= 1.                  VOLUMECHECK= YES
TEMPORARYFILE= 500.              WAIT= YES
                                 WRITECHECK= NO

Buffer space usage= 3290.:1924.:1924.:1.

IOX>VERIFY DM0: RANGE=1:1000 BUF=4 RAND=N INTER=999 ERR=50000  ❸
IOX>REPORTERRORS NO
IOX>SUMMARY 0
IOX>RUNTIME 0                                                ❹
IOX>LOG Y
IOX>PAR

Device  Mode  Buffersize  Filesize/Records/Range  Parameters
------  ----  ----------  ----------------------  ----------
DM0:    VFY       4.      1. -> 1,000.            RET INT=999.ERR=50000.

BUFFERSIZE= 1024.                COMPAREDATA= YES
ERRORLIMIT= 10.                  CONTROL_C= YES
INTERLEAVE= 4.                   LOGFILE= YES
PATTERN= 0.                      RANDOM= YES                 ❺
RECORDS= 1024.                   REPORTERRORS= NO
RUNTIME= 0.                      RETRIES= YES
SUMMARYTIME= 0.                  VOLUMECHECK= YES
TEMPORARYFILE= 500.              WAIT= YES
                                 WRITECHECK= NO

Buffer space usage= 3290.:1912.:1912.:1.

IOX>ST
IOX>^C                                                       ❻
IOX>ABORT
```

The command sequence in Example 4-3 does the following:

❶ Allocates drive DM0:, mounts the RK07 for NFS testing, and invokes IOX.

On RSX11M-PLUS systems, mount disks with the /FOREIGN switch.

❷ Displays the initial default parameter.

❸ Selects DM0: for read-only NFS testing with VERIFY, overriding the following defaults:

a. RANGE -- blocks 1 through 1000

b. BUFFERSIZE -- minimum, four bytes

c. RANDOM block testing turned off and INTERLEAVE set at 999 so that IOX tests only blocks 1 and 1000

d. ERRORLIMIT set impossibly high (50000.) so that IOX will not deselect DM0: no matter how many errors are encountered during the exercise

❹ Turns off REPORTERRORS so that no error reports are generated. Turns SUMMARYTIME off so that no interval reports will be generated. Sets RUNTIME to zero so that IOX runs the exercise until you abort it. Sets LOGFILE YES to direct all output to the log file, IOX.LOG, instead of to the terminal.

❺ Shows that the RUNTIME, SUMMARYTIME, and LOGFILE defaults have been reset and that DM0: is being tested with VERIFY, using a buffersize of 4., over a range of 1000. blocks, with RETRIES enabled, INTERLEAVE set at 999, and ERRORLIMIT set at 50000.

❻ Shows how to start the exercise using the START command and enter Interactive Mode with CTRL/C so you can enter the ABORT command to abort the exercise.

Example 4-4 shows the contents of the log file for Example 4-3.

Example 4-4 Log File Output for Example 4-3

```
IOX -- 20-JUL-81 18:24:12 ******** EXERCISER STARTED ********

Device  Mode  Buffersize  Filesize/Records/Range  Parameters
------  ----  ----------  ----------------------  ----------
DM0:    VFY       4.        1. -> 1,000.          RET INT=999. ERR=50000.

BUFFERSIZE= 1024.              COMPAREDATA= YES
ERRORLIMIT= 10.                CONTROL_C= YES
INTERLEAVE= 4.                 LOGFILE= YES
PATTERN= 0.                    RANDOM= YES
RECORDS= 1024.                 REPORTERRORS= NO
RUNTIME= 0.                    RETRIES= YES
SUMMARYTIME= 0.                VOLUMECHECK= YES
TEMPORARYFILE= 500.            WAIT= YES
                               WRITECHECK= NO

Buffer space usage= 3290.:1812.:1812.:1.


IOX -- Summary of exerciser activity at 20-JUL-81 18:27:28
  TIME -- Run: 3 min.   Elapsed: 3 min.   Remaining: 65533 min.
  DM0: -- 3,075. R/W requests totaling 3,075. blocks with no errors
  AST's executed: 3,081.
  Idle-loop iterations: 2,982.


 A C C U M U L A T E D   T O T A L S :
 - - - - - - - - - - -   - - - - - - -
  TIME -- Run: 0 min.   Elapsed: 3 min.   Remaining: 65533 min.
  DM0:A -- 3,076. R/W requests totaling 3,076. blocks with no errors
  AST's executed: 3,082.
  Idle-loop iterations: 2,982.

IOX -- 20-JUL-81 18:27:31 ******** EXERCISER ABORTED ********
```

The log file output terminates when you abort the Exerciser. The log file begins with the time and date that the exercise began and the current default parameters as displayed by the PARAMETERLIST command.

4.9 IOX OPERATING MODES

IOX operates in three modes: Command, Execution, and Interactive. This section discusses the functions of these modes, how you enter and exit each mode, and what IOX commands are valid in each mode.

4.9.1 Command Mode

IOX is in Command Mode after you have invoked it and have received the first IOX prompt. (Section 4.2 discusses IOX prompts.) In Command Mode, no devices are being exercised, and tasks initiated by the EXECUTE command are queued but not active (see Section 4.12 for a complete description of the EXECUTE command.) You usually set parameters, select devices for testing, and initiate tasks to be executed in Command Mode.

When you press the RETURN key, you receive an IOX prompt. When you press CTRL/C, you exit to MCR for one command line.

4.9.2 Execution Mode

When you begin an I/O exercise by entering the START command, IOX exits Command Mode and enters Execution Mode. Exercises begin on selected devices, tasks initiated with the EXECUTE command begin executing, the IOX clock is turned on, and activity and error reporting begin.

You set the CONTROL_C parameter in Command Mode to allow or disallow you to enter IOX commands while an exercise is in progress. CONTROL_C YES is the initial default setting.

If you have set CONTROL_C NO, then CTRL/C in Execution Mode gives control to MCR and you receive the MCR prompt ">" every time you press the RETURN key and "MCR>" every time you press CTRL/C until the exercise has terminated. With CTRL/C disabled, you cannot interrupt the exercise. However, you can terminate it with the MCR ABORT command.

If you have set CONTROL_C YES, IOX does not accept any input from your terminal except CTRL/C. If you try to enter anything but CTRL/C, you receive an error message for each key you press:

```
IOX -- Only ^C Allowed
```

When you press CTRL/C, you receive the IOX prompt instead of the MCR prompt and IOX exits Execution Mode and enters Interactive Mode.

4.9.3 Interactive Mode

You have access to Interactive Mode only if CONTROL_C is enabled. When you press CTRL/C in Execution Mode, IOX exits Execution Mode and enters Interactive Mode. In Interactive Mode, the RETURN key and CTRL/C have the same functions that they have in Command Mode.

In Interactive Mode, the exercise and tasks initiated by the EXECUTE command are running. Activity reports and error reports directed to your terminal are suspended if you have set LOGFILE=NO. If you have set LOGFILE=YES, activity and error reports continue as in Execution Mode.

All IOX commands except the following are valid in Interactive Mode:

BUFFERSIZE
PATTERN
RUNTIME
START
RESTART

If you enter these commands, IOX prints the error message:

IOX -- Command valid only in command mode

You can use the BUFFERSIZE command within the SELECT and VERIFY command lines to override the current defaults for the device being selected, but the BUFFERSIZE command cannot be used by itself to reset the default.

Although you cannot use the PATTERN command to choose a different pattern to be written to the volumes being tested, you can use PATTERN with no argument to display all the patterns.

The following commands are valid only in Interactive Mode:

ABORT
PRINTSUMMARY
PROCEED
SPY

Although you can issue PRINTSUMMARY and SPY in Command Mode without generating an error message, these commands display meaningful information only when you issue them in Interactive Mode.

When you issue the ABORT command, IOX exits Interactive Mode and enters Command Mode. The ABORT command with no qualifiers terminates all exercising and all tasks initiated by the EXECUTE command. This form of ABORT is valid only in Interactive Mode because in Command Mode there can be no exercise or active tasks to abort.

The PROCEED command exits Interactive Mode and enters Execution Mode unless no activity remains when you issue PROCEED, in which case IOX aborts the exercise.

4.10 IOX OUTPUT

IOX generates two types of output: activity reports and error reports. Activity reports summarize IOX processing information, such as how long an exercise will run and how many ASTs have been executed. There are two kinds of error reports: data compare error reports, that show where IOX finds errors on the volume being tested, and I/O error reports, that indicate failed QIOs from a driver. You use the LOGFILE command (see detail in Section 4.12) to direct activity and error reports to your terminal or to a log file.

4.10.1 Activity Reports

There are three types of activity reports:

Accumulated Totals

An activity report at the end of an IOX exercise that reports the activity for the entire exercise

Interval Reports

An activity report that summarizes IOX activity in the time intervals set by the SUMMARYTIME command

Summary Reports

A report of activity between the last Interval Report and the time you enter the PRINTSUMMARY command to request a summary report.

Section 4.12 discusses the SUMMARYTIME and PRINTSUMMARY commands in detail.

4.10.1.1 **The Format of Activity Reports** - The formats of activity reports are indentical except for the first line. Interval and summary reports begin with the time and date of the report. For example:

```
IOX -- Summary of exerciser activity at 20-JUL-81 11:16:26
```

The accumulated totals report does not give the time or date of the report and begins:

```
A C C U M U L A T E D   T O T A L S :
- - - - - - - - - - -   - - - - - - -
```

4.10.1.2 **The Content of Activity Reports** - The content of all activity reports is identical. The following is an interval report of an exercise that tests SY0:.

```
IOX -- Summary of exerciser activity at 31-DEC-80 11:18:29
TIME -- Run: 1 min.    Elapsed: 2 min.    Remaining: 3 min.
SY0: -- 1,874. R/W requests totaling 1,874. blocks with no errors
ASTs executed: 1,875.
Idle-loop iterations: 1,803.
```

The activity report contains the following information:

- Timing information
  - Time between reports as set in the SUMMARYTIME command
  - Time that the exercise has been in progress to the nearest minute
  - Time remaining to the nearest minute before the exercise will finish
- Unit information for each unit being tested
  - Number of I/O requests to the unit
  - Number of block exercises on the unit
  - Number of error encounters on the unit
- Number of ASTs executed
- Number of iterations through the idle loop

You use the following commands to control the output of activity reports:

LOGFILE
PRINTSUMMARY
SPY
SUMMARYTIME

Section 4.12 discusses each of these commands in detail.

4.10.2 Error Reports

You use the REPORTERRORS command to enable or disable error reports. Section 4.12 discusses the REPORTERRORS command in detail. There are two kinds of error reports: data compare error reports and I/O error.

4.10.2.1 **Data Compare Error Reports** - A data compare error means that the driver returned a success I/O code to IOX but that the data written to the volume and the data read from the volume did not match. The following is an error report generated by a data compare error:

```
IOX -- DR1: data compare error at block 33,937 (decimal) 102221 (octal)
       Good Data: 005767   Bad Data: 155555 ❷                    ❶
       Word Position: 0 (decimal) 0 (octal) ❸
       Buffer contained 255. additional errors ❹
```

❶ Identifies the device on which IOX encountered an error. It reports the starting block (in both decimal and octal) in which the error was found.

❷ Shows the "good data" that IOX wrote to the volume and the "bad data" that IOX read from the volume. Only the first error encountered in the test buffer is reported.

❸ Shows the position of the first word in the test buffer that generated a mismatch during the current operation. Only the first error encountered in the test buffer is reported.

❹ Shows how many other errors were found in the test buffer.

4.10.2.2 **I/O Error Reports** - There are 12 I/O errors. If a QIO fails, the driver returns an I/O failure code to IOX. IOX generates a specific I/O error report as a result of the following I/O errors:

I/O Code	Description
IE.ABO	Operation aborted
IE.BBE	Bad block
IE.DAO	Data overrun
IE.DNR	Device not ready
IE.EOT	End-of-tape detected
IE.FHE	Fatal hardware error
IE.OFL	Device off-line
IE.PRI	Privilege violation
IE.VER	Unrecoverable error
IE.WCK	Write check error
IE.WLK	Write-lock error
N/A	Unexpected error

These error codes are shown in the RSX11M/M-PLUS I/O Drivers Reference Manual. Section 4.13 of this chapter presents the format and content of the 12 error messages that IOX generates because of these error codes.

4.11 THE TASK BUILD FILE

You can change the initial defaults for IOX parameters as well as other defaults of the IOX.TSK by editing the IOX task-build file, IOXBLD.BLD.

On RSX-11M systems, this file is located on your system disk in UFD [1,24] for mapped systems and [1,20] for unmapped systems.

On RSX-11M-PLUS systems, this file is located on your system disk in UFD [1,24].

4.12 COMMAND DESCRIPTIONS

This section provides complete descriptions of the commands in the IOX Command Language. Each command description includes:

- Kind of command by category: Function, Display, Parameter, and Control
- All command formats
- Qualifier descriptions
- Operating mode restrictions (where no mode restrictions are placed on a command, it is valid in Command Mode and Interactive Mode)
- Kinds of devices and testing to which the command applies
- Initial defaults if applicable

The commands are presented alphabetically.

Commands may be issued in all uppercase, all lowercase, or part upper and part lowercase. You can abbreviate commands using any combination of characters that appear unique to the IOX Command Language.

All commands are represented by unabbreviated uppercase letters. All lowercase words in the presentations of command syntax indicate that you substitute alphanumeric characters for the word.

All numerical qualifiers are decimal.

Brackets [ ] indicate optional elements in the command line.

ABORT

A[BORT] [/ALL]
[taskname]

The ABORT command is an IOX control command. You cannot use the ABORT command to abort a task that has attached your terminal, such as PIP.

There are three forms of ABORT:

ABORT	Terminates the execution of an I/O exercise and tasks initiated by the EXECUTE command. This form of ABORT is valid only in Interactive Mode (testing in progress) because in Command Mode (no testing in progress) there can be no running exercise or active tasks to abort.
ABORT /ALL	In Interactive Mode this form of ABORT terminates the execution of all tasks initiated by the EXECUTE command and removes the tasks from the list of tasks to be executed. In Command Mode, this form of ABORT removes all tasks from the list of tasks to be executed.
ABORT taskname	In Interactive Mode this form of ABORT terminates the execution of a task initiated by the EXECUTE command and removes the task from the list of tasks to be executed. In Command Mode, this form of ABORT removes the task from the list of tasks to be executed.

BADBLOCKS

BA[DBLOCKS] ddnn: [blocknumbers]

You can use BADBLOCKS as a device-dependent parameter command or a display command. BADBLOCKS applies only to NFS disk testing.

BADBLOCKS requires a device specification "dduu:". To use the BADBLOCKS command, the disk that you specify must already be selected for NFS operations.

BADBLOCKS as a Parameter Command

With the BADBLOCKS Parameter Command, you can omit the bad blocks on your disk from IOX testing. BADBLOCKS does not find bad blocks on your disk; it allows you to enter those blocks that the BAD utility has reported to be bad into an IOX badblocks list for the disk. When you exercise the disk with SELECT or VERIFY, the blocks that you entered in the list will not be exercised. IOX omits only those blocks that you listed as blocknumbers in the BADBLOCKS command line; blocks marked as bad by the BAD utility but not entered into the badblocks list with the BADBLOCKS command are treated as good blocks by IOX.

Once you have made entries in the badblocks list for a disk, the block numbers that you entered remain on the list until you deselect the disk.

There are two formats for entering bad blocks into the badblocks list. One format, using DM1: as an example, is:

```
BADBLOCKS DM1: 3
```

This command line enters block 3 in the badblocks list for DM1:. Another example:

```
BADBLOCKS DM1:  3,4,5,11,12,13,99,151
```

This command line enters blocks 3, 4, 5, 11, 12, 13, 99, and 151 into the badblocks list for DM1:.

The second format has the form:

```
beg:num
```

beg

Beginning block number

num

Number of sequential blocks

For example:

```
BADBLOCKS DM1:  3:3,10:4,100:50
```

This command line enters blocks 3 through 5, 10 through 13, and 100 through 149 into the badblocks list for DM1:. You can use this format for the length of a single command line.

BADBLOCKS (Cont.)

You can use the two formats together on the same command line. For example:

```
BADBLOCKS DM1:  3:3,10:4,99,100:50,151
```

This command line enters blocks 3 through 5, 10 through 13, 99, 100 through 149, and 151 into the BADBLOCK list for DM1:.

BADBLOCKS as a Display Command

BADBLOCKS with no blocknumbers specified is a Display Command. For example:

```
BADBLOCKS DM1:
```

This command line displays the badblocks list for DM1:. If you had typed:

```
BADBLOCKS DM1:  3:5,10:4,99,100:50,151
```

IOX would display:

```
000003:005
000010:004
000099:051
000151:001
```

IOX displays the badblocks list using a variation of the second format. It strings consecutive blocks together even if you did not enter them using the second format. For example, in the command line you entered:

```
99,100:50....
```

However, when IOX displays the badblocks list, it does not separate these because blocks 99 and 100 are consecutive blocks. It does not display two lines: one for block 99 and one for blocks 100 through 149. Instead, it displays (line three) the badblocks list in one line as if you had entered 51 blocks (99:51) beginning with block 99 in the badblocks list.

BUFFERSIZE

BU[FFERSIZE] buffersize

BUFFERSIZE is a device-dependent parameter command. You use BUFFERSIZE to determine the amount of data that IOX transfers for every read/write request when exercising NFS disks, magnetic tapes, and cassettes. IOX stores the specified size in bytes and rounds this number up to the next four-byte boundary.

When you use the random data pattern (PATTERN 0), BUFFERSIZE should never exceed 3400. bytes or IOX generates unpredictable data compare errors.

Although read/write buffers are used for FILES11 disk testing and for NFS testing with DECtapes, the buffersize for these devices is fixed. A 512.-byte buffer is used for all FILES11 testing. A 1024.-byte buffer is used for all DECtape testing.

The BUFFERSIZE command is valid only in Command Mode when used by itself. In addition, it is valid only if units are not currently selected; that is, you must set your default buffersize before you select any units for testing. However, you can use the buffersize command in the SELECT and VERIFY command lines to override the default buffersize for the selected device as long as you set the buffersize smaller than the default.

The initial default is 1024. bytes.

COMPAREDATA

COM[PAREDATA] Y[ES]/N[O]

COMPAREDATA is a device-dependent parameter command for FILES11 and SELECT that applies to the testing of devices.

COMPAREDATA enables or disables the checking of the data pattern. COMPAREDATA YES enables data pattern checking; COMPAREDATA NO disables data pattern checking. When you direct IOX to compare data patterns, it compares the data that it writes to the unit with the data it reads from the unit. If the data read does not match the data written, IOX prints a data compare error report. Section 4.10.2.1 discusses data compare error reports in detail.

You can use COMPAREDATA within the command lines of FILES11 and SELECT to override the current COMPAREDATA setting for the device being tested.

The initial default is COMPAREDATA YES.

CONFIGURE

CONF[IGURE] [dduu:[=type]] ....

CONFIGURE is a general parameter command and a display command. You use CONFIGURE as a parameter command to add devices to your test configuration and as a display command to display the current configuration.

IOX configures by default one of each supported device type as unit "0". You can add any device to the test configuration to which you can assign a logical device name by using the MCR ASN command. When you configure a device, IOX adds the device to its list of devices available for testing and creates an empty data structure for the device.

If a device is not already included in the test configuration, you must configure it before you can select it for testing. IOX does not unconfigure devices at the end of an exercise. Once you have included a device in the test configuration, only RESTART will unconfigure it.

CONFIGURE with no qualifiers displays the current test configuration. To find out if a device is already configured, type:

```
IOX>CONFIGURE
```

IOX responds:

```
Disks:
DB0  DD0  DF0  DK0  DL0  DM0  DP0  DR0  DS0  DU0
DX0  DY0  EM0  SY0

Tapes:
CT0  DT0  MF0  MM0  MS0  MT0
```

To add devices to the test configuration, type:

```
IOX>CONFIGURE DK1:  DB1:  DB2:  XX:  YY:  MM5:  MQ1:=TAPE
```

This command line configures devices DK1:, DB1:, DB2:, XX0:, and YY0: as disk units. It configures devices MM5: and MQ1: as tape units.

You must separate unit specifications with spaces or tabs.

Use the display form of CONFIGURE to check that these devices have been added to your configuration:

```
IOX>CONF

Disks:
DB0  DB1  DB2  DD0  DF0  DK0  DK1  DL0  DM0  DP0
DR0  DS0  DU0  DX0  DY0  EM0  SY0  XX0  YY0

Tapes:
CT0  DT0  MF0  MM0  MM5  MQ1  MS0  MT0
```

CONFIGURE (Cont.)

The device type defaults are as follows if you do not specify a device type when you configure a device:

1. If one unit of a device type has already been configured as a disk, then, if you add a unit with the same device specification, IOX also configures that unit as a disk. For example, since DK0: and DB0: in the previous example are configured as disks in the default configuration, IOX configures DK1:, DB1:, and DB2: as disks. By the same logic, IOX configures MM5: as a tape because MM0: has been configured as a tape unit by default.

   IOX maintains consistency in configuring device units. Therefore, you cannot configure DK1:, DB1:, or DB2: as tapes. The same holds true for MM5: -- you cannot configure it as a disk. In both cases, IOX generates the following error message:

   ```
   IOX -- dduu:  specified device type is inconsistent
   ```

2. For devices with mnemonics that have not been previously configured, the default is disk. Hence, in the following example, IOX configures both XX: and YY: as disks, and MQ1: as a tape.

   ```
   IOX>CONF DK1:  DB1:  DB2:  XX:  YY:  MM5:  MQ1:=TAPE
   ```

   IOX responds by displaying the following messages:

   ```
   IOX -- XX0:  device type not specified - defaulting to 'DISK'
   IOX -- YY0:  device type not specified - defaulting to 'DISK'
   ```

CTRL/C

CTRL/C is a control command. In Command Mode and Interactive Mode, CTRL/C generates an MCR prompt "MCR>" allowing you to enter one MCR command line. The function of CTRL/C in Execution Mode depends on whether CONTROL_C has been set YES or NO. If you have set CONTROL_C NO, then CTRL/C generates the MCR prompt "MCR>" as it does in Command and Interactive Modes. If you have set CONTROL_C YES, then CTRL/C exits IOX from Execution Mode and enters Interactive Mode.

Section 4.9 describes the three IOX operation modes in detail.

CTRL/C exits to MCR by default in Command and Interactive Modes. In Execution Mode, CTRL/C exits Execution Mode and enters Interactive Mode by default.

CONTROL_C

CONT[ROL_C] Y[ES]/N[O]

CONTROL_C is a general parameter command. You use CONTROL_C to enable or disable the use of CTRL/C to interrupt an I/O exercise.

Command Mode

CTRL/C gives control to MCR for one command line. The CONTROL_C parameter has no affect on CTRL/C in Command Mode.

Execution Mode

If you have set CONTROL_C NO, then CTRL/C gives control to MCR and you receive the MCR prompt ">" every time you press the RETURN key and "MCR>" every time you press CTRL/C until the exercise has terminated. With CTRL/C disabled, you cannot interrupt the exercise to enter IOX commands. However, you can terminate it with the MCR ABORT command and issue other MCR commands while an exercise is in progress.

If you have set CONTROL_C YES, IOX will not accept any input from your terminal except CTRL/C. If you try to enter anything but CTRL/C, you receive the following error message for each key you press:

```
IOX -- Only ^C allowed
```

When you press CTRL/C, you receive the IOX prompt instead of the MCR prompt and IOX exits Execution Mode and enters Interactive Mode.

Interactive Mode

You have access to Interactive Mode only if CONTROL_C is enabled. In Interactive Mode, you can enter most IOX commands. The RETURN key and CTRL/C have the same functions in Interactive Mode that they have in Command Mode.

Section 4.9 describes the three IOX operation modes in detail.

The initial default is CONTROL_C YES.

DENSITY

DEN[SITY] dduu: [n]

DENSITY is a device-dependent parameter command and a display command. The DENSITY command applies only to magnetic tape testing and requires a magnetic tape device specification for "dduu:".

You use DENSITY as a device-dependent parameter command with an argument "n" to set the density and characteristic word of a magnetic tape. (The device characteristic word is defined in the RSX11M Guide to Writing an I/O Driver.) You issue the DENSITY command before you select the tape for NFS testing with SELECT. You can also set the tape density for a magnetic tape unit by using the DENSITY command within the SELECT command line.

You set all tape densities in bits per inch (bpi) using decimal values for "n". The following tape densities are legal:

MF:	MM:	MS:	MT:
1600	800	1600	200
1250	1600		556
			800

You use DENSITY as a display command without an argument "n" to display the current density of the specified magnetic tape unit. For example:

```
IOX>DENSITY MM:

    Density= 1600 BPI Characteristic Word= 004004 (octal)
```

This command line displays the current density and characteristic word for MM0:. To change the tape density to 800, you type:

```
IOX>DENSITY MM: 800
```

The DENSITY command is the only way you can find out what the current DENSITY setting is on a magnetic tape, because DENSITY is not displayed by the PARAMETERLIST command.

To display the new tape density and characteristic word for MM0:, you type:

```
IOX>DENSITY MM:

     Density= 800 BPI Characteristic Word= 000000 (octal)
```

For tapes other than MT, MM, MS, and MF, you enter the density in decimal but IOX displays the density in octal as the characteristic word. For example:

```
IOX>DENS MQ:
  Density= ???? BPI    Characteristic Word= 004004 (octal)
IOX>DENS MQ: 800
IOX>DENS MQ:
  Density= ???? BPI    Characteristic Word= 001440 (octal)
```

DENSITY (Cont.)

This command sequence indicates that:

- IOX does not know the density of the tape designated as MQ:. The characteristic word is the default.
- When you set the density of MQ: to 800 bpi (decimal) and display the current density and characteristic word, the density appears in the characteristic word (octal).

Except for MT: devices, the initial default density for magnetic tape devices is 1600 bpi. The initial default for MT: devices is 800 bpi.

DESELECT

DES[ELECT] ddnn:

DESELECT is an IOX function command. You use DESELECT to "deselect" devices that you have selected for testing with SELECT, VERIFY, or FILES11. DESELECT requires device (dd) and unit (nn) specifications.

If you issue the DESELECT command in Interactive Mode (while testing of the specified device is in progress), testing on the device terminates and the device is removed from the list of devices to be tested. If you issue the command in Command Mode (while no testing is in progress), then DESELECT only removes the device from the list of devices to be tested.

For devices that you have selected for NFS testing, DESELECT does the following:

- Terminates I/O operations (if they are in progress)
- Deletes the badblocks list associated with the device (disks only)
- Deallocates all buffer space
- Detaches IOX from the device

For devices that you have selected for FILES11 testing, DESELECT does the following:

- Terminates I/O operations (if they are in progress)
- Closes and deletes the temporary file
- Deallocates all buffer space

The DESELECT command does not remove a device from the test configuration.

ERRORLIMIT

ER[RORLIMIT] n

ERRORLIMIT is a device-dependent parameter command that sets the maximum number of errors that IOX will tolerate between interval reports before deselecting a device.

You use the SUMMARYTIME command to determine how often IOX generates interval reports. If you request interval reports every two minutes (SUMMARYTIME 2) and set ERRORLIMIT at 10 errors, IOX will allow not more than 10 errors on a device in a two-minute interval before deselecting the device from testing.

You can also use the ERRORLIMIT command within the FILES11, SELECT, and VERIFY command lines to override the default error limit for the device being selected for testing.

The initial default is 10 errors.

EXECUTE

EXE[CUTE] taskname commandstring

EXECUTE is a function command. You use EXECUTE to direct a command line to the operating system for execution. The task you want to run must be installed. For the EXECUTE command to operate, your system must support task spawning. The rules for using the SPWN$ directive and entering command lines apply to the arguments for taskname and commandstring in the EXECUTE command line (see the RSX-11M/M-PLUS Executive Reference Manual).

When you issue the EXECUTE command in Interactive Mode (testing in progress), the task you have initiated with EXECUTE runs immediately. When you issue the EXECUTE command in Command Mode (no testing in progress), IOX queues the task for execution. The task begins executing when you start an exercise with the START command.

Tasks you initiate with EXECUTE run in parallel with I/O exercises. For example, you use the EXECUTE command to run the BAD utility to list at your terminal the bad blocks on an NFS disk. You can then enter this information in the IOX badblocks list so that IOX does not try to perform I/O operations on the blocks that are bad.

To run the BAD utility on NFS disk DL0:, type

```
IOX>EXE ...BAD #DL0:/LI
```

In this case, you must include the pound sign (#) as shown in the command line. For tasks such as BAD whose command line processing requires a leading space, IOX converts the pound sign to a space. A task such as PIP does not require a leading space in its command line. For these tasks you do not need to include the pound sign in the command line.

You can also use the EXECUTE command to run a task to do system testing while IOX is doing I/O testing. For example, if you have written a CPU exerciser task and installed the task as ...QWK, then you can run this task in parallel with the I/O Exerciser by entering the command line

```
IOX>EXECUTE ...QWK /CPU=1144
IOX>START
```

where /CPU=1144 is the command line to be passed to task ...QWK.

To remove a task from the list of tasks to be executed and terminate execution of the task (if the task is in progress), you use the ABORT command.

EXIT

EXI[T]

EXIT is a control command. You use the EXIT command to terminate IOX processing and exit from IOX to the operating system monitor.

FILES11

F[iles11] dduu: [command=qualifier] ...

FILES11 is a function command. You use FILES11 to select a Files-11 disk for an I/O exercise. Testing a disk with FILES11 protects the contents and structure of the disk, because IOX writes into a temporary file that IOX removes from the disk at the end of the test. You must first configure the disk if it is not already in the test configuration before you can select it for testing.

When you select a disk using FILES11, you can override the default parameters by using the following device-dependent parameter commands in the FILES11 command line:

```
COMPAREDATA
ERRORLIMIT
INTERLEAVE
RANDOM
TEMPORARYFILE
```

The following example uses some of them.

```
IOX>FILES11 DB0:  COMPAREDATA=YES RANDOM=NO TEMPORARYFILE=600
```

This command line selects DB0: for I/O exercising. IOX compares data read with data written, sequentially accesses the blocks in the file, and opens a 600-block temporary file. Example 4-1 in Section 4.7 illustrates and discusses how you test an RL02 with FILES11.

Once you have selected a device, the default parameters for that device are fixed. You cannot reset them without first deselecting the device.

When you issue the FILES11 command in Interactive Mode (while an exercise is in progress), IOX begins testing the specified device immediately. When you issue the FILES11 command in Command Mode (while no exercise is in progress), IOX includes the specified device in its list of devices to exercise but does not begin the exercise until you start the exercise with the START command.

If no commands are specified within the FILES11 command line, the device is exercised with the default parameters currently in effect. Commands can only be separated by spaces or tabs.

HELP

H[ELP]

HELP is a display command. You type HELP to display a summary of IOX commands at your terminal.

INTERLEAVE

I[NTERLEAVE] number

INTERLEAVE is a parameter command that sets a device-dependent parameter for disk testing. The INTERLEAVE command sets the number of blocks to be skipped before IOX writes the next buffer of data. The INTERLEAVE factor applies only to sequential testing: that is, if you have set RANDOM NO. If you have set RANDOM YES, IOX tests blocks on the disk randomly and ignores the INTERLEAVE factor. The initial default for RANDOM is RANDOM YES.

You can use the INTERLEAVE command within the FILES11, SELECT, and VERIFY command lines to override the current default for the disk being selected.

The initial default is four blocks.

LOGFILE

L[OGFILE] Y[ES]/N[O]

LOGFILE is a general parameter command. You use the LOGFILE command to direct activity and error Reports either to your terminal or to a log file in your default disk area. Section 4.10 discusses the content and format of activity and error reports in detail.

LOGFILE NO directs activity and error reports to your terminal. LOGFILE YES directs activity and error reports to the log file. Summary reports, which IOX generates when you issue the PRINTSUMMARY command, are displayed at your terminal regardless of how you have set LOGFILE.

IOX does not open a new log file each time you run an exercise with LOGFILE enabled. It creates the file when you first run an exercise and then appends to it. The file is in your area and is named:

yyyyyy.LOG

where yyyyyy is the name of the task currently running IOX on your terminal (IOX, another task name, or the name of your terminal, for example TT10).

In addition to activity and error reports, IOX inserts the following date and time information into the log file:

- When the exercise began followed by the output of the PARAMETERLIST command
- When the exercise ended
- When a log file was started
- When log file entries were terminated
- When an exercise was aborted

For example, all exercises begin with an event line such as:

IOX -- 20-JUL-81 11:15:24 ******** EXERCISER STARTED ********

and end with an event line such as:

IOX -- 20-JUL-81 11:20:34 ******** EXERCISER TERMINATED ********

The initial default is LOGFILE NO.

PARAMETERLIST

PAR[AMETERLIST]

PARAMETERLIST is a display command. PARAMETERLIST displays at least two kinds of information. For example:

```
IOX>PARAMETERLIST

BUFFERSIZE= 1024.          COMPAREDATA= YES
ERRORLIMIT= 10.            CONTROL_C= YES
INTERLEAVE= 4.             LOGFILE= NO
PATTERN= 0.                RANDOM= YES
RECORDS= 1024.             REPORTERRORS= YES
RUNTIME= 5.                RETRIES= YES
SUMMARYTIME= 1.            VOLUMECHECK= YES
TEMPORARYFILE= 500.        WAIT= YES
                           WRITECHECK= NO

Buffer space usage= 3290.:1924.:1924.:1.
```

❶ The current default parameters

❷ IOX buffer space statistics

At most, PARAMETERLIST displays four kinds of information:

- Current execution default parameters
- IOX buffer space statistics

IOX may also list the:

- List of tasks initiated by the EXECUTE command
- Devices selected for testing with FILES11, SELECT, and VERIFY

Example 4-5 describes the information in the four parts of the PARAMETERLIST display that correspond to these four kinds of information.

Example 4-5 PARAMETERLIST Display Shown in Four Parts

```
IOX>EXECUTE ...BAD $DL0:/LI                                    ❾
IOX>FILES11 SY: TEMPORARYFILE=800 ERRORLIMIT=50                ❺
IOX>SELECT MM0: RECORDS=350 BUFFERSIZE=700 RETRIES=NO
IOX>VERIFY DM0: RANGE 200:1000 RANDOM=NO INTERLEAVE=8          ❼
IOX>PARAMETERLIST

Device  Mode  Buffersize  Filesize/Records/Range  Parameters
------  ----  ----------  ----------------------  ----------

DM0:    VFY       1024.     200. -> 1,000.       RET INT=8. ERR=10. ❻
MM0:    SEL        700.                350.      COM ERR=10.
SY0:    F11        512.                800.      COM RET RAN ERR=50. ❹
```

❸

(continued on next page)

PARAMETERLIST (Cont.)

Example 4-5 (Cont.) PARAMETERLIST Display Shown in Four Parts

```
Task     Status    Command line
----     ------    ------------
...BAD   QUEUED    DL0:/LI
```

❽

```
BUFFERSIZE= 1024.          COMPAREDATA= YES
ERRORLIMIT= 10.            CONTROL_C= YES
INTERLEAVE= 4.             LOGFILE= NO
PATTERN= 0.                RANDOM= YES
RECORDS= 1024.             REPORTERRORS= YES
RUNTIME= 5.                RETRIES= YES
SUMMARYTIME= 1.            VOLUMECHECK= YES
TEMPORARYFILE= 500.        WAIT= YES
                           WRITECHECK= NO
```

❶

```
Buffer space usage= 5338.:1020.:1588.:2.
```

❷

❶ Current defaults at the time you entered the PARAMETERLIST command. None of the defaults have been changed by the command sequence in the example. These are the initial default parameters. You can alter the default for these parameters. See the description for the appropriate command in this section.

❷ IOX buffer space statistics. The form of these statistics is the following:

totalmemory:freeblock:freebytes:fragments

totalmemory

Size of IOX buffer space in bytes.

freeblock

Size of the largest free block of IOX buffer space in bytes.

freebytes

Total amount of free bytes available in IOX buffer space.

fragments

Number of memory fragments in IOX buffer space.

In the examples, these statistics have changed to reflect the buffer allocations for the three devices that you selected for testing.

❸ Selected devices and their device-dependent parameters. These parameters correspond to the default parameters unless you have overridden the defaults in the command line that you used to select the devices.

PARAMETERLIST (Cont.)

This part of the display has five fields that are labeled by function. They are:

device

Device mnemonic of the selected device.

mode

Testing mode selected, either VERIFY (VFY), SELECT (SEL), or FILES11 (F11).

buffersize

Buffersize used for reading data from the device and writing data to the device. You can alter the buffersize for magnetic tapes, NFS disks, and cassettes. Files-11 disks have a fixed buffersize of 512. bytes. DECtapes have a fixed buffersize of 1024. bytes.

filesize/records/range

One of the following:

- Temporary file size in blocks for disks selected with FILES11.
- Number of records to test for magnetic tapes and cassettes selected with the SELECT command.
- Range of blocks to be tested on NFS disks that you selected with the SELECT or VERIFY command.

parameter

Device dependent parameters. These either specify a value, such as INT=8. or ERR=10. (INTERLEAVE 8 or ERRORLIMIT 10), or their presence indicates that you have set a YES/NO parameter to YES. For example, in (4) RAN means that you set RANDOM YES; and, in (6), the absence of RAN means that you set RANDOM NO. See Section 4.6 for a list of what parameters apply to what devices and modes of testing.

❹ This display line corresponds to the command line discussed next.

❺ This is the command line that you used to select SY: for FILES11 testing, overriding the TEMPORARYFILE and ERRORLIMIT defaults. IOX tests SY: as a Files-11 device, using a 512-byte data buffer (fixed) and a TEMPORARYFILE size of 800 blocks, with COMPAREDATA enabled (by default), RETRIES enabled (by default), RANDOM YES (by default), and an error limit of 50 errors.

❻ The display line corresponding to the command line discussed next.

PARAMETERLIST (Cont.)

❼ The command line you used to select DM0: with VERIFY. IOX tests DM0:, using the VERIFY mode with the default buffersize of 1024. bytes, over the range of blocks 200 through 1000, with RETRIES enabled (by default), RANDOM disabled, the INTERLEAVE factor set to 8, and the default error limit of 10.

❽ The list of tasks initiated by the EXECUTE command corresponding to the command line discussed next.

This part of the display has the following fields:

- Task Installed -- taskname (...BAD)
- Status Execution -- status (QUEUED, ACTIVE, SUCCESS, WARNING, ERROR, SEVERE ERROR)
- Command Line -- line passed to the task

❾ The command line you used to execute the BAD utility.

In Command Mode you can use CTRL/O to suppress the output of PARAMETERLIST. In Interactive Mode, CTRL/O has no affect on the output.

PATTERN

PAT[TERN] [patternnumber]

PATTERN is a general parameter command and a display command. You use PATTERN as a parameter command (with an argument) to set the pattern that IOX writes and reads during I/O testing. The PATTERN display command (with no argument) displays all 13 patterns and shows the pattern that is the current default with an asterisk (*).

The following example illustrates the default pattern when you use PATTERN as a display command:

```
IOX>PATTERN
   0 = Random Data   *
   1 = 000000,000000
   2 = 177777,177777
   3 = 125252,052525
   4 = 165555,133333
   5 = 163126,163126
   6 = 055555,155555
   7 = 022222,122222
   8 = 111111,111111
   9 = 007417,007417
  10 = 021042,021042
  11 = 104210,104210
  12 = 052652,052652
```

To change the default pattern from random data to "125252,052525", which is pattern 3, you type:

```
IOX>PATTERN 3
```

Using the display form of PATTERN now shows that the asterisk moved from pattern 0 to pattern 3 after you changed the default:

```
IOX>PATTERN
   0 = Random Data
   1 = 000000,000000
   2 = 177777,177777
   3 = 125252,052525 *
   4 = 165555,133333
   5 = 163126,163126
   6 = 055555,155555
   7 = 022222,122222
   8 = 111111,111111
   9 = 007417,007417
  10 = 021042,021042
  11 = 104210,104210
  12 = 052652,052652
```

The form of the PATTERN command that changes the pattern default is valid only in Command Mode.

Pattern 0 is random data from pure portions of the I/O Exerciser itself. When you use the random data pattern, the buffersize should never exceed 3400. bytes or IOX will generate unpredictable data compare errors.

The initial default pattern is random data, pattern 0.

PRINTSUMMARY

PRI[NTSUMMARY]

PRINTSUMMARY is a display command that is valid only in Interactive Mode.

IOX outputs activity reports and error reports. Activity reports summarize IOX processing information, such as how long an exercise will run and how many ASTs have been executed. There are three kinds of activity reports: accumulated totals, interval reports, and summary reports. (For a complete description of these three kinds of activity reports, see Section 4.10.)

Summary reports and interval reports have the same content and format. However, IOX generates interval reports at the time intervals that you have set with the SUMMARYTIME command. IOX generates summary reports when you enter the PRINTSUMMARY command in Interactive Mode. A summary report summarizes IOX activity between the last interval report and the time you enter the PRINTSUMMARY command. IOX displays summary reports at your terminal even if LOGFILE is enabled.

PROCEED

PRO[CEED]

PROCEED is a control command that you use only in Interactive Mode. (Section 4.9 describes the three IOX operating modes in detail.) When you enter the PROCEED command, IOX exits Interactive Mode, enters Execution Mode, and checks for IOX activity. If no exercises are in progress and no tasks initiated by IOX are executing, then IOX terminates the exercise.

If there is activity, IOX resumes display of activity and error reports to your terminal if you have set LOGFILE NO. (With LOGFILE disabled, IOX suspends activity and error reports to your terminal in Interactive Mode. With LOGFILE enabled, IOX continues recording activity and error reports in the log file.)

Use the PROCEED command only if you want IOX to terminate an exercise as soon as activity terminates or if you want IOX to resume display of activity and error reports at your terminal with LOGFILE disabled.

RANDOM

RAND[OM] Y[ES]/N[O]

RANDOM is a device-dependent parameter command that applies to disk testing. The RANDOM command directs IOX to select either random or sequential blocks for disk testing.

RANDOM YES directs IOX to use its random number generator to choose the blocks on the selected disk for exercising. RANDOM NO directs IOX to sequentially test blocks on the disk.

If you set RANDOM NO, then IOX skips the number of blocks specified with the INTERLEAVE command before testing the next block. For example:

```
IOX>INTERLEAVE 8
IOX>RANDOM NO
```

In this example, IOX tests every eighth block on the disk (that is, block 0, 7, 15, ...).

You can use the RANDOM command within the FILES11, SELECT, and VERIFY command lines to override the current default parameter for selecting blocks for testing.

The initial default is RANDOM YES.

RANGE

RANG[E] dduu: [minimum:maximum]

RANGE is a device-dependent parameter command and a display command. The RANGE command applies only to disks that you have selected for NFS testing with SELECT or VERIFY.

RANGE as a parameter command sets the minimum and maximum block numbers for NFS testing on the specified disk unit. For example:

```
IOX>RANGE DL0: 300:500
```

IOX tests blocks in the range 300 through 500 on DL0. This form of the RANGE command can be used in the SELECT or VERIFY command lines to specify the range of blocks to be tested.

RANGE as a display command displays the current range of blocks to be tested on the specified disk unit. The device must have been selected for testing.

For example:

```
IOX>RANGE DL0:
   0. -> 20,420.
```

This is the default for RL02s. After you issue the RANGE command in the previous example, you can use the RANGE command to display the current range of blocks to be tested:

```
IOX>RANGE DL0:
   300. -> 500.
```

This form of the RANGE command cannot be used within the SELECT or VERIFY command lines.

The default range for all disks that have been selected is zero to the maximum block number on the disk minus sixty (0:maximum-60).

RECORDS

REC[ORDS] recordcount

RECORDS is a device-dependent parameter command that applies to the testing of magnetic tapes and cassettes with SELECT. RECORDS specifies a recordcount, which is the number of records on the magnetic tape or cassette to be exercised. For example:

```
IOX>RECORDS 200
```

This command line directs IOX to exercise 200 records on all magnetic tapes and cassettes. RECORDS 0 directs IOX to test the records on magnetic tapes and cassettes until it detects the end of the tape, at which time IOX displays an end of tape message and deselects the tape unit.

You can use RECORDS in the SELECT command line to override the default recordcount for the magnetic tape or cassette being selected.

The initial default is 1024. records.

REPORTERRORS

REP[ORTERRORS] Y[ES]/N[O]

REPORTERRORS is a general parameter command that you use to enable (REPORTERRORS YES) or disable (REPORTERRORS NO) IOX error reports. There are two kinds of error reports: data compare error reports and I/O error reports.

If a success code is returned from the device driver (meaning that a QIO succeeded) but the data that IOX wrote does not match the data that IOX read, then a data compare error has occurred. Section 4.10.2.1 illustrates and discusses data compare error reports.

I/O error codes are returned from the device drivers (meaning that a QIO failed). These error codes generate one of 12 error messages discussed in Sections 4.10 and 4.13.

IOX directs error reports to your terminal, but with the following restrictions:

- Error reports never interrupt activity reports.
- If LOGFILE is enabled, IOX enters error reports in the log file instead of displaying them at your terminal.
- In Interactive Mode, IOX suppresses the printing of error reports.

The initial default is REPORTERRORS YES.

RESTART

RES[TART]

RESTART is a control command. The RESTART command is valid only in Command Mode. When you restart the I/O Exerciser, you effectively reinvoke IOX without exiting to the operating system monitor.

The RESTART command performs the following functions:

- Deselects devices that are selected for testing.
- Resets all parameters to their initial defaults.
- Removes all tasks queued for execution by the EXECUTE command from the list of tasks to be executed.
- Unconfigures all devices that you added to the default configuration.

RETRIES

RET[RIES] Y[ES]/N[O]

RETRIES is a device-dependent parameter command that applies to the testing of magnetic tapes and NFS disks. RETRIES YES directs a device driver to use its standard recovery techniques to attempt to recover from an error. RETRIES NO inhibits an I/O driver from using its standard recovery techniques when it encounters an error.

You can use the RETRIES command within the SELECT and VERIFY command lines to override the current default for the device being selected.

The initial default is RETRIES YES.

RUNTIME

RUN[TIME] minutes

RUNTIME is a general parameter command. RUNTIME is valid only in Command Mode (no exercising in progress).

You use the RUNTIME command to specify how many minutes you want IOX to test the devices(s) that you have selected for testing. If you specify RUNTIME 0, IOX continues testing the selected devices until you abort the exercise with either the IOX ABORT command or the MCR ABORT command.

The initial default is five minutes.

SELECT

SE[LECT] dduu: [command=qualifier] ....

SELECT is a function command. The SELECT command selects a device with an NFS volume mounted for an I/O exercise that destroys the contents and structure of the volume. Before using select, you must first configure the device if it is not in the test configuration.

When you use the SELECT command to select a device to be tested as an NFS volume, you can override the current default parameters by using the commands in Table 4-3 within the SELECT command line.

Table 4-3
Parameter Commands by Device Type for the SELECT Command

NFS Disks	Cassettes	DECtapes	Magnetic Tapes
BUFFERSIZE COMPAREDATA ERRORLIMIT INTERLEAVE RANDOM RANGE RETRIES VOLUMECHECK WRITECHECK	BUFFERSIZE COMPAREDATA ERRORLIMIT RECORDS	COMPAREDATA ERRORLIMIT	BUFFERSIZE ERRORLIMIT DENSITY RECORDS RETRIES

The following command line illustrates some of these commands:

```
IOX>SEL DL0:  RANG=100:4000 RET=YES RAND=NO INT=8 BUF=2000
```

This command line:

- Selects DL0: for I/O exercising
- Tests only blocks 100 through 4000
- Allows the driver to retry I/O operations if it encounters errors
- Accesses the blocks on the volume sequentially testing every eighth block
- Uses a data buffer of 2000 bytes for data transfer.

Example 4-2 in Section 4.8.1 illustrates and describes how you test a magnetic tape unit with the SELECT command.

SELECT (Cont.)

Once you have selected a device, the default parameters for that device are fixed. You cannot reset them without first deselecting the device.

When you issue the SELECT command in Interactive Mode (while an exercise is in progress), IOX begins testing the specified device immediately. When you issue the SELECT command in Command Mode (while no exercise is in progress), IOX includes the specified device in its list of devices to exercise but does not begin the exercise until you start the exercise with the START command.

If no further parameter commands are specified within the SELECT command line, the device is exercised with the parameters currently in effect. Commands can only be separated by spaces or tabs.

SPY

SP[Y] [dduu:]

SPY is a display command. You issue the SPY command only in Interactive Mode (exercising in progress) to display status information on device(s) being tested.

SPY without an argument displays status information on all devices being tested. SPY with a device and unit specification displays status information only on the specified device. For example:

```
>SPY DB0:
```

IOX outputs the following:

```
DB0:  -- TYPE=Fll   VBN=305.   I/O=AST   queued   FUNCTION=WRITE
```

This output indicates:

- Device being exercised (DB0:)
- Type of testing being performed on the device: FILESll (Fll), SELECT (SEL), VERIFY (VFY)
- Virtual block number being tested (305. in this example) or, for NFS volumes, the logical block number
- I/O request status: AST queued, in progress, or idle
- Type of I/O function being performed: READ or WRITE

SPY always outputs to your terminal, even if LOGFILE is enabled.

START

ST[ART]

START is a control command. START is valid only in Command Mode. When you issue the START command, IOX exits Command Mode and enters Execution Mode. Section 4.9 discusses the three IOX operating modes in detail.

After you have set your test configuration and selected the devices you want to test, you begin exercising the selected units with the START command. START also turns on the IOX clock, begins activity and error reporting (if you enabled error reporting with REPORTERRORS YES), and begins executing tasks queued by the EXECUTE command.

SUMMARYTIME

SU[MMARYTIME] minutes

SUMMARYTIME is a general parameter command you use to specify how often (in minutes) IOX outputs interval reports. (See Section 4.10 for a complete discussion of interval reports). You can suppress interval reports by setting SUMMARYTIME 0.

The initial default is one minute.

TEMPORARYFILE

T[EMPORARYFILE] filesize

TEMPORARYFILE is a device-dependent parameter command that applies to the testing of Files-11 disks with the FILES11 command. The FILES11 command protects the contents of the disk by writing a temporary file that IOX deletes at the end of the exercise. You specify the filesize in blocks.

You can use the TEMPORARYFILE command in the FILES11 command line to override the current default for the disk being selected.

The initial default is 500 blocks.

VERIFY

VE[RIFY] dduu: [command=qualifier] ....

VERIFY is a function command you use to select a mounted NFS disk for an exercise. VERIFY reads buffers of data without writing on the disk, performing data comparisons, or destroying the contents of the disk.

VERIFY accepts the following device-dependent parameter commands within its command line to override the default parameters for the device being tested:

BUFFERSIZE
ERRORLIMIT
INTERLEAVE
RANDOM
RANGE
RETRIES

The following example illustrates how you use the ERRORLIMIT and RANGE commands within the VERIFY command line to override the current defaults for testing DM0:

```
IOX>VERIFY DM0:  RANGE 1000:3000 ERRORLIMIT=1000
```

This command line selects DM0: for read only testing of blocks 1000 through 3000 (inclusive) with a maximum of 1000 errors before IOX deselects the disk. Example 4-3 in Section 4.8.2 illustrates and explains how you test an NFS disk with VERIFY.

Once you have selected a device, the default parameters for that device are fixed. You cannot reset them without first deselecting the device.

When you issue the VERIFY command in Interactive Mode (while an exercise is in progress), IOX begins testing the specified device immediately. When you issue the VERIFY command in Command Mode (while no exercise is in progress), IOX includes the specified device in its list of devices to exercise but does not begin the exercise until you start the exercise with the START command.

If no further parameter commands are specified within the VERIFY command line, the disk is exercised with the parameters currently in effect. Commands can only be separated by spaces or tabs.

VOLUMECHECK

VO[LUMECHECK] Y[ES]/N[O]

VOLUMECHECK is a device-dependent parameter command that applies to NFS disk testing with SELECT. VOLUMECHECK enables or disables checking of disks mounted as NFS volumes for a home block and a Files-11 structure. If VOLUMECHECK is enabled (VOLUMECHECK YES), then IOX will not allow you to use the SELECT command to select a volume with a Files-11 structure. With VOLUMECHECK disabled (VOLUMECHECK NO), IOX will allow you to use the SELECT command to select the disk without warning you that the test will destroy the contents of the disk. You should use VOLUMECHECK YES unless you are intentionally writing over a Files-11 disk.

You can use the VOLUMECHECK command within the SELECT command line to override the current default for the disk being selected.

The initial default is VOLUMECHECK YES.

WAIT

WA[IT] Y[ES]/N[O]

WAIT is a general parameter command you use to determine whether or not IOX waits for an event flag while an exercise is in progress. The WAIT command has no affect on tasks that you initiated with the EXECUTE command.

WAIT YES requires IOX to wait for an event flag to signal that an I/O operation has completed. During this time, the Executive may be able to execute another task.

If you set WAIT NO, IOX executes an idle loop during I/O operations. Setting WAIT NO can monopolize system resources. Therefore, you should not set WAIT NO unless you intentionally want to monopolize system resources.

The initial default is WAIT YES.

WRITECHECK

WR[ITECHECK] Y[ES]/N[O]

WRITECHECK is a device-dependent parameter command that applies only to testing NFS disks with SELECT. You use the WRITECHECK command to determine whether or not I/O drivers perform write checks to NFS disks.

If WRITECHECK is enabled, IOX writes data to the disks by issuing an IO.WLC (write logical block with check) to the driver instead of IO.WLB (write logical block). This means that the disk controller reads, word-by-word, a copy of the data from the data buffer while simultaneously reading the same data from the disk. The disk controller compares the two copies of the data. If the copies do not match and RETRIES is enabled, the driver repeats the original write operation and performs another write check. This process continues until the WRITECHECK function succeeds or the retry count for the driver is exceeded. If RETRIES is disabled or if RETRIES is enabled but the retry count for the driver is exceeded, then the driver issues the hard error message, IE.WCK, and IOX generates an error message.

You can use the WRITECHECK command within the SELECT command line to override the current default for the device being selected.

The initial default is WRITECHECK NO.

4.13 ERROR MESSAGES

IOX generates two types of error messages: error messages with device specifications and error messages without device specifications. IOX displays device specifications as appropriate.

The format for an error message with a device specification is:

taskname -- ddnn: message

taskname

Name of the IOX task that is currently running. In the error messages that follow, taskname is always IOX.

ddnn

Device specification text of the message.

message

Information specific to this message.

The format for an error message without a device specification is:

taskname--message

where taskname and message are the same as defined in the previous format.

All error messages use the letter "x" to indicate that IOX substitutes the appropriate command, task name, or value for "x" within the message.

4.13.1 Error Messages with Device Specifications

The error messages are presented in alphabetical order by the first letter following the device specification.

IOX -- dduu: bad block, xxx (decimal) xxx (octal)

Explanation: IE.BBE error code returned to IOX (bad block error).

User Action: Use the BADBLOCKS command (only for NFS disks to enter the block number into the badblocks list.

IOX -- dduu: block number out of range

Explanation: You used the BADBLOCKS command to enter a block number in the badblocks list that is outside the range of blocks that you specified with the RANGE command.

IOX -- dduu: buffer size is greater than default

Explanation: The buffersize that you specified in the SELECT or VERIFY command line is greater than the current default set with the BUFFERSIZE command by itself.

User Action: Either raise the default buffersize (using BUFFERSIZE by itself) or lower the buffersize override for the device (using BUFFERSIZE within the SELECT or VERIFY command line).

IOX -- dduu: data compare error at block xxx (decimal) xxx (octal)

```
Good data: xxxxx bad data: xxxxx
Word position: xxx (decimal) xxx (octal)
Buffer contained xx. additional errors
```

Explanation: IOX found an error when comparing the data read with the data written. This error message shows the first block number in which the error occurred during the transfer, the good and bad data, the word position of the first error found in the buffer, and the number of additional errors found in the buffer. Section 4.10.2.1 discusses data compare error reports in detail.

User Action: None. If the error persists, enter the block in the badblocks list by using the BADBLOCKS command (for NFS disks only).

IOX -- dduu: data overrun at block xxx (decimal) xxx (octal)

Explanation: IE.DAO error code returned to IOX (data overrun). The record size given was greater than the record size read. This error occurs on tape devices only.

User Action: None.

IOX -- dduu: device type not specified - defaulting to 'DISK'

Explanation: This message tells you that you are adding a device to your test configuration with the CONFIGURE command that IOX configures as a disk by default.

User Action: None.

IOX -- dduu: did not respond - unit now deselected

Explanation: The unit did not respond to the IOX command within one minute. The device may have been accidentally put off line or a tape drive may have lost its column vacuum.

User Action: Check the device.

IOX -- dduu: does not support the x command

Explanation: The x command is incompatible with the device specification, dduu:.

User Action: None.

IOX -- dduu: duplicate block number

Explanation: The block number you entered with the BADBLOCKS command is already in the badblocks list for the device.

User Action: None.

IOX -- dduu: end of tape detected - unit now deselected

Explanation: IO.EOT error code returned to IOX (end-of-tape detected). IOX attempted to write a record past the end-of-tape marker.

User Action: This is an informational message only.

IOX -- dduu: error threshold exceeded - unit now deselected

Explanation: The number of errors set with the ERRORLIMIT command has been exceeded.

User Action: None.

IOX -- dduu: error while reading home block

Explanation: When VOLUMECHECK is enabled, this message occurs if IOX detects an error while attempting to read the home block of a Files-11 device.

User Action: The disk you selected may be defective. If you are testing a scratch disk, select the disk again with VOLUMECHECK NO.

IOX -- dduu: failed to attach

Explanation: IOX could not attach the unit.

User Action: The device may be mounted or attached by someone else. Make sure you have allocated and mounted the device.

For RSX11M-PLUS systems, mount devices with the /FOREIGN switch.

IOX -- dduu: failed to detach

Explanation: An error occurred while IOX was attempting to detach a unit that it had previously attached.

User Action: None. This is a possible operating system problem.

IOX -- dduu: failed to open temporary file

Explanation: IOX could not find enough free blocks on the disk to open a temporary file as large as the one you specified with the TEMPORARYFILE command.

User Action: Specify a smaller temporary file size either by resetting the default with the TEMPORARYFILE command by itself or by using TEMPORARYFILE within the FILES11 command line to override the default TEMPORARYFILE size for each device.

If you are using the default (500 blocks) when IOX generates this message, then you need to make room on the disk.

IOX -- dduu: fatal hardware error - unit now deselected

Explanation: IE.FHE error code returned to IOX (fatal hardware error).

User Action: Check the hardware.

IOX -- dduu: illegal density

Explanation: The density that you specified with the DENSITY command, for the magnetic tape is illegal.

User Action: Reenter the command, using a legal density. See Section 4.12.

IOX -- dduu: illegal range

Explanation: The range of blocks which you specified on the device to be tested with the RANGE command is out of bounds.

User Action: Use RANGE with no qualifier to check the default range on the device; then specify a range within bounds.

IOX -- dduu: invalid block number

Explanation: The block number that you entered with the BADBLOCKS command was not a valid block number for the device.

User Action: None.

IOX -- dduu: I/O is unsatisfied at timeout - function: xxxx

Explanation: When IOX finishes processing, it waits five seconds for I/O to stop. If a long tape has not finished rewinding, this message may occur. The function that timed out is xxxx.

User Action: None.

IOX -- dduu: I/O request aborted - unit now deselected

Explanation: IE.ABO error code returned to IOX (operation aborted).

User Action: None.

IOX -- dduu: is already configured

Explanation: You tried to configure with the CONFIGURE command a unit that is already in the test configuration.

User Action: Use the CONFIGURE command as a display command to check which devices are configured.

IOX -- dduu: is already selected

Explanation: You tried to select a unit that has already been selected.

User Action: Use the PARAMETERLIST command to display information on all units that are selected.

IOX -- dduu: is not configured

Explanation: You issued a command that is valid only for configured devices.

User Action: Use the CONFIGURE command to configure the device.

IOX -- dduu: is not mounted or allocated for NFS operations

Explanation: You have selected a device for testing with SELECT or VERIFY that is not properly allocated and mounted for NFS testing.

User Action: Allocate and mount the device.

For RSX11M-PLUS systems, mount NFS devices with the /FOREIGN switch.

IOX -- dduu: is not ready - unit now deselected

Explanation: IE.DNR error code returned to IOX (device not ready).

User Action: Select the device again.

IOX -- dduu: is not selected

Explanation: You issued a device-dependent parameter command (BADBLOCKS or RANGE) that is valid only after you have selected the device.

User Action: Select the device; then reenter the command.

IOX -- dduu: is offline

Explanation: IE.OFL error code returned to IOX (device off line). The device was taken off line while testing was in progress or you tried to select a device that was off line. If testing was in progress on the device when it was taken off line, IOX appends " - unit now deselected" to the error message.

User Action: Ensure that the device is on line.

IOX -- dduu: is write locked - unit now deselected

Explanation: IE.WLK error code returned to IOX (write-lock error). The device is write-locked.

User Action: Check the device write-lock/write-enable switch and the MCR write-lock status.

IOX -- dduu: LUN assignment failure

Explanation: This probably means that the device, ddnn:, is not physically present in your system. It may mean that you have exceeded the number of devices that IOX can support. The default is 28 devices; the maximum number of devices is 246.

User Action: If the device is not physically present in your system then take no action. If you have exceeded the number of devices that IOX can support, then edit the Task Builder command file, IOX.CMD (see Section 4.11), to allow IOX to support more devices.

IOX -- dduu: privilege violation - unit now deselected

Explanation: IE.PRI error code returned to IOX (privilege violation). The device was not allocated or it was mounted by someone else.

User Action: Check the ownership of the device.

IOX -- dduu: specified device type is inconsistent

Explanation: You tried to configure device "dduu" as a tape when a "dd" device had already been configured as a disk. The same holds true if "dd" had already been configured as a tape and you tried to configure another "dd" device as a disk.

User Action: Use the CONFIGURE command as a display command to check on the devices in your test configuration. Configure the unit using a different device mnemonic.

IOX -- dduu: unexpected error #xx

Explanation: IOX received an I/O error code that it could not process. The error code is xx (octal).

User Action: None.

IOX -- dduu: unexpected request to queue I/O - request aborted

Explanation: IOX received an AST from a device after it had deselected the device.

User Action: Check the device.

IOX -- dduu: unrecoverable error at block xxx (decimal) xxx (octal)

Explanation: IE.VER error code returned to IOX (unrecoverable error). This code indicates an unrecoverable error.

User Action: Check the device and media.

IOX -- dduu: VOLUMECHECK failure. Unit contains a home block

Explanation: The unit is file-structured because the device contains a home block. This error message is generated by select if you try to select a Files-11 volume while VOLUMECHECK is enabled.

User Action: IOX destroys information on devices that you select for exercising with the SELECT command. If you want to destroy the contents of the disk, disable VOLUMECHECK (VOLUMECHECK NO) and select the device again.

IOX -- dduu: write check error at block xxx (decimal) xxx (octal)

Explanation: IE.WCK error code returned to IOX (write check error). The write check operation failed at block xxx.

User Action: None.

4.13.2 Error Messages Without Device Specifications

These error messages are presented in alphabetical order by the first letter following the IOX task name.

IOX -- Indirect command file syntax error

Explanation: There is an error in the indirect command file.

User Action: Check the indirect command file.

IOX -- Indirect command file nesting level exceeded

Explanation: IOX accepts indirect command files with only one level of nesting.

User Action: Rewrite the indirect command file so that it does not have more than one level of nesting.

IOX -- Aborting due to user command

Explanation: You aborted IOX by using the MCR ABORT command.

User Action: Reinvoke IOX if you want to continue testing.

IOX -- EXECUTE command is not supported

Explanation: You attempted to use the EXECUTE command in a system that does not support task spawning.

User Action: Do another SYSGEN if you require the EXECUTE command to properly test your I/O devices.

IOX -- Command input I/O error

Explanation: IOX detected an error when it tried to read the command that you typed at the terminal.

User Action: Reenter the command.

IOX - Command valid only in Command Mode

Explanation: You tried to enter a command that cannot be entered while an I/O exercise is in progress.

User Action: Either abort the exercise and enter the command or wait until the exercise has finished to enter the command.

IOX -- Command valid only in Interactive Mode

Explanation: You tried to enter a command that cannot be entered unless IOX is running.

User Action: Type START; then reenter the command.

IOX -- Command not unique

Explanation: The command abbreviation that you entered did not distinguish it from another command.

User Action: Reenter the command by using a longer abbreviation.

IOX -- Failed to close log file. F.ERR= -x

Explanation: IOX could not close the log file in your directory. The FCS error returned was "-x".

User Action: Look up the I/O error code returned to IOX by FCS in the IAS/RSX-11 I/O Operations Reference Manual and take the appropriate action.

IOX -- Failed to open indirect command file

Explanation: The FCS GCML routine failed to open the indirect command file.

User Action: Check to see that the file exists and that you specified the file correctly; then resubmit the file to IOX.

IOX -- Failed to open log file. F.ERR= -x

Explanation: IOX could not open the log file. The FCS error returned was "-x".

User Action: The disk may be write protected or may not have enough blocks.

IOX -- Failed to open existing log file. F.ERR= -x

Explanation: IOX could not open a log file that already exists in your directory. The FCS error returned was "-x".

User Action: Look up the I/O error code returned to IOX by FCS in the IAS/RSX-11 I/O Operations Reference Manual and take the appropriate action.

IOX -- Failed to truncate/close log file. F.ERR= -x

Explanation: IOX could not truncate or close the log file. The FCS error returned was "-x".

User Action: Look up the I/O error code returned to IOX by FCS in the IAS/RSX-11 I/O Operations Reference Manual and take the appropriate action.

IOX -- Fatal file system error. F.ERR= xxx PC+2= xxx

Explanation: IOX exited because an error occurred while IOX was trying to issue a read or write. F.ERR is the octal error code in the File Descriptor Block indicating the kind of file error. PC+2 is the octal location of the failure in the IOX code.

User Action: Look up the I/O error code returned to IOX by FCS in the IAS/RSX-11 I/O Operations Reference Manual and take the appropriate action.

IOX -- Fatal system error. DSW= xxx PC+2= xxx

Explanation: IOX immediately exits because a directive that IOX issued failed. DSW is the Directive Status word in octal. PC+2 is the octal location of the failure in the IOX code.

User Action: Look up the I/O error code returned to IOX by FCS in the IAS/RSX-11 I/O Operations Reference Manual and take the appropriate action.

IOX -- Invalid pattern number

Explanation: You tried to set a data pattern using a pattern number greater than 12. The valid arguments for PATTERN are 0 through 12.

User Action: Enter a valid pattern number.

IOX -- Invalid task name

Explanation: The task name that you specified with the EXECUTE command is invalid. The task name is either incorrect or the task does not exist.

User Action: Reenter the task name correctly or determine if the task exists.

IOX -- No activity remains - aborting

Explanation: In Execution Mode, IOX determines whether or not any exercises are in progress or any tasks are executing at one minute intervals. If IOX finds no activity, it aborts the exercise and generates this informational message.

In Interactive Mode, IOX determines whether or not any exercises are in progress or any tasks are executing only when you enter the PROCEED command. If there is no activity when you enter PROCEED, IOX aborts the exercise and issues this informational message.

User Action: None.

IOX -- No activity to start

Explanation: When you issued the START command, no devices were selected for testing and no tasks were initiated for execution.

User Action: Select devices that you want to test, and use the EXECUTE command to initiate tasks that you want to execute.

IOX -- No buffer space available

Explanation: IOX does not have adequate buffer space to accommodate all the devices you selected.

User Action: Either deselect some devices from testing or reinstall or reinvoke IOX with a greater increment.

IOX -- No such command - type H for help

Explanation: You entered a command that IOX did not recognize.

User Action: Enter H (abbreviation for the HELP command), which displays a summary all IOX commands.

IOX -- No tasks to abort

Explanation: You tried to abort tasks using the ABORT command when no tasks had been initiated by the EXECUTE command.

User Action: None.

IOX -- Only CTRL/C allowed

Explanation: You set CONTROL_C YES to enable you to enter IOX commands while exercises are in progress. Therefore, IOX only accepts CTRL/C in Execution Mode.

User Action: Enter CTRL/C to exit Execution Mode, then IOX is in Interactive Mode and you can enter commands.

IOX -- Syntax error

Explanation: The command line that you entered contained a syntax error.

User Action: If necessary, check the syntax of the command line that you entered. See Section 4.12 or type H (for HELP).

IOX -- Task never executed

Explanation: The task that you tried to abort is not on the list of tasks initiated by the EXECUTE command.

User Action: Use the PARAMETERLIST command to check the list of tasks to be executed.

IOX -- Task "xxxxxx" is currently active

Explanation: You tried to execute a task that was already active. The name of the task is "xxxxxx".

User Action: None.

IOX -- Task "xxxxxx" is completed. Status= nnnnnnn

Explanation: The task has completed execution. The name of the task is "xxxxxx". There are four task statuses represented by "nnnnnnn": SUCCESS, WARNING, ERROR, and SEVERE ERROR.

User Action: None.

IOX -- Task "xxxxxx" is not installed

Explanation: Task "xxxxxx" is not installed and cannot be initiated using the EXECUTE command.

User Action: Install the task from a privileged terminal.

IOX -- Task "xxxxxx" is unable to execute

Explanation: Task "xxxxxx" could not execute.

User Action: Try to execute the task again.

IOX -- Units are selected - command ignored

Explanation: The BUFFERSIZE command is valid only if no units are selected for testing.

User Action: You can use the BUFFERSIZE command within the SELECT and VERIFY command lines to override the default for each device. However, if you require a larger BUFFERSIZE for all devices, use the DESELECT command to deselect all units; then reissue the BUFFERSIZE command.

CHAPTER 6

RESOURCE MONITORING DISPLAY (RMD)

The Resource Monitoring Display (RMD) is a privileged task that displays information about the resources in your system. This information includes the active tasks, their location in memory, the amount of memory they occupy, and available pool space. On video terminals such as VT52s and VT100s, RMD provides dynamic displays. On hardcopy terminals such as LA180s, RMD provides "snapshot" displays. RMD alters the display format according to terminal type without your having to set switches or rebuild the task.

6.1 INTRODUCTION

RMD consists of "pages". A page consists of 24 lines, which, on CRT terminals, is one screen. The program contains two kinds of pages: display pages and setup pages.

6.1.1 Display Pages

There are four display pages available on both RSX-11M and RSX-11M-PLUS systems:

- Memory (M)
- Active Task List (A)
- Task Header (T)
- Help (H)

There are two additional display pages that are available only on RSX-11M-PLUS systems:

- I/O Counts (I)
- System Statistics (S)

You press the terminal keys indicated in parentheses to switch display pages.

6.1.2 Setup Pages

There are three setup pages. Each setup page is associated with a display page. (The Help Display Page has no associated setup page.) You access a setup page from a display page by pressing the ESCAPE key on your terminal. The setup page documents and prompts you for setup commands, which alter the content of the information displayed on the associated display page.

6.2 INSTALLING AND INVOKING RMD

Individual users can invoke RMD at their terminals. For large systems that support many terminals, you may want to run RMD continuously on a slaved terminal near the computer. You use different procedures to install and invoke RMD for use by individual users and to run RMD on a slaved terminal.

6.2.1 Using RMD at Individual Terminals

On RSX-11M systems, RMD is installed if you selected RMD at system generation.

On RSX-11M-PLUS, RMD is installed.

6.2.1.1 **Installing RMD for Use at Individual Terminals** - You must install RMD if you have removed it. To install RMD, use the MCR INSTALL command at a privileged terminal:

```
>INS $RMD
```

6.2.1.2 **Invoking RMD for Use at Individual Terminals** - From any terminal, you invoke an installed copy of RMD from MCR as follows:

```
>RMD [page][,setupcommand] . . .
```

page

One of the display page abbreviations (M, A, T, or H). The default page is the Memory Display (M).

setupcommand

A valid setup command for the display page that you have selected. The setup commands are the same as those available to you from the setup page associated with the display page you specified. The default setup commands are discussed in Sections 6.4 through 6.6, which describe the content of each display page and how you use setup commands to alter display parameters.

From a privileged terminal only, you can invoke an uninstalled copy of RMD using the MCR RUN command. However, MCR command line parsing is available only if RMD is installed.

6.2.2 Running RMD on a Slaved Terminal

To run RMD on a slaved terminal and also allow users to invoke RMD at their own terminals, RMD must be installed twice using different task image files and different task names.

On RSX-11M systems, if you selected RMD at system generation, you will find RMD.TSK, which has been task built at system generation with task name ...RMD, in SYSUIC on your system disk.

On RSX-11M-PLUS systems, you will find RMD.TSK, which has been task built at system generation with task name ...RMD, in SYSUIC on your system disk.

RMD.TSK is the copy of RMD that the system installs for individual users to invoke. LUNs 1 and 2 in this task image have been assigned to TI: so that a user at any terminal can invoke RMD.

To run RMD on a slaved terminal, do the following:

1. From a privileged terminal, make a copy of RMD.TSK in SYSUIC on your system disk named RMDEMO.TSK. If your SYSUIC is [1,54] and your system disk is DR0:, use the following command sequence:

```
>SET /UIC=[1,54]
>ASN DR0:=SY:
>PIP RMDEMO.TSK=RMD.TSK
```

2. Either from a privileged terminal or from a command file to be executed at a privileged terminal, install RMDEMO using the following command sequence:

```
>INS RMDEMO/TASK=RMDEMO
>REA RMDEMO 1 ttnn:
>REA RMDEMO 2 ttnn:
>SET /SLAVE=ttnn:
>RUN RMDEMO
```

   where ttnn: is the terminal at which you want to run RMD.

   This command sequence installs RMD with task name RMDEMO, reassigns LUNs 1 and 2 to the terminal you want to slave to the task, slaves the terminal, and invokes the task.

   Because the REASSIGN command alters the task image on disk, you must use two task images rather than simply installing RMD twice with different task names.

6.3 THE HELP DISPLAY

The Help Display documents how you switch display pages. You switch display pages by pressing a terminal key as follows:

Key	Explanation
M	Accesses the Memory Display
A	Accesses the Active Task Display
T	Accesses the Task Header Display
I	Accesses the I/O Counts Display
S	Accesses the System Statistics Display

The Help Display also documents how to exit from RMD to MCR and use the ESCAPE key to access setup pages from their associated display pages. (Because there is nothing to alter on the Help Display, no setup page is available from the Help Display.)

The Memory, Active Task, and Task Header Displays use the entire screen. Therefore, you receive no prompts or documentation on display pages. To find out how to access a setup page from a display page or how to switch display pages, press the H key (for Help) to access the Help Display.

6.4 THE MEMORY DISPLAY

The Memory Display graphically represents the entire system memory, including the approximate size and locations of partitions and active tasks. The display also shows pool statistics, the name of the task that is currently executing, and other information about the status of your operating system. You access the Memory Display from another display page by pressing the M key (for Memory). To access the Memory Display from the MCR command line, you type:

```
>RMD M
```

If you invoke RMD without specifying a display page:

```
>RMD
```

RMD defaults to the Memory Display.

Figures 6-1 and 6-2 show "snapshots" of the Memory Display for RSX-11M and RSX-11M-PLUS respectively. The reverse numbers in each figure (white numbers in black circles) match the reverse numbers in the explanations that follow the two figures. Where a display field is the same for both RSX-11M and RSX-11M-PLUS, it is designated by the same reverse number in Figures 6-1 and 6-2. The matching explanation, however, appears only once.

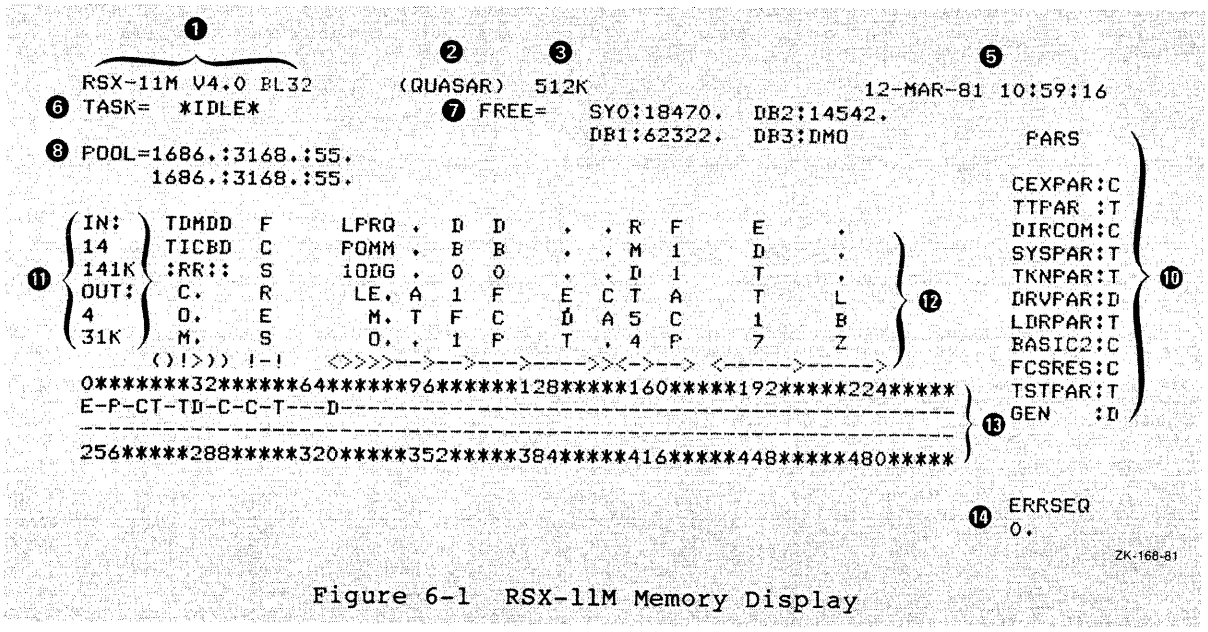


Figure 6-1 RSX-11M Memory Display

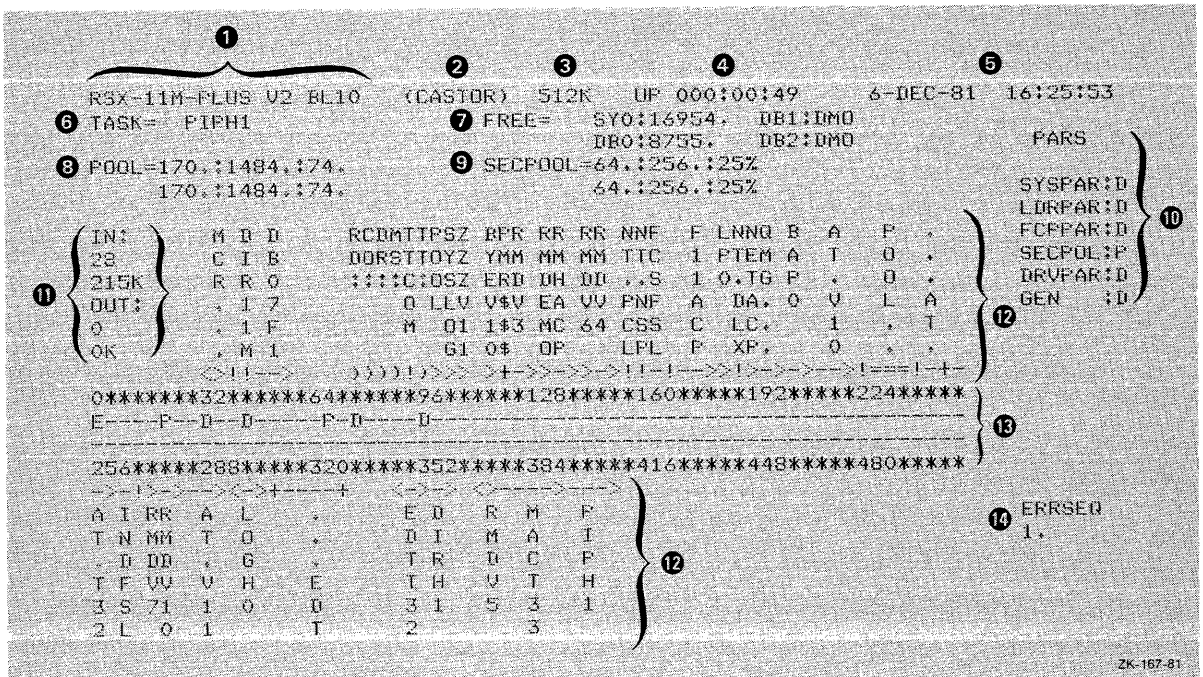


Figure 6-2 RSX-11M-PLUS Memory Display

❶ Operating system type, version number, and base level.

❷ Six-character name that is a DECnet node name (if DECnet is running on your system) or the system name that you selected in SYSGEN (if DECnet is not running on your system).

❸ Size in K words of the system memory.

❹ For RSX-11M-PLUS systems, time elapsed in units of days, hours, and minutes, since the system was last booted.

❺ Current date and time.

❻ Name of the task that is currently executing or, if none is executing, *IDLE* (Executive executing the idle loop).

❼ Number of free blocks on the first four Files-11 devices in your system. If a device is dismounted, RMD displays: "DMO". If a device is off line, RMD displays: "OFL".

❽ Pool (dynamic storage region) information in the format:

POOL=X:Y:Z

X

Number of words in the largest free block in pool

Y

Number of free words in pool

Z

Number of fragments in the pool free list

The second line records the worst case of pool since you invoked RMD. This line is most useful if RMD has been running on a slaved terminal since the system was last booted.

❾ On RSX-11M-PLUS systems, secondary pool information in the format:

SECPOOL=A:B:C%

A

Number of free blocks in secondary pool

B

Total number of blocks in secondary pool

C%

Percentage of secondary pool that is free

The second line records the worst case of secondary pool since you invoked RMD. This line is most useful if RMD has been running on a slaved terminal since the system was last booted.

❿ Partitions in the system using the format:

partitionname:type

type

For RSX-11M systems (Figure 6-1), one of the following:

T	Task partition
S	Subpartition
D	System-controlled (dynamic) partition
C	Common partition

For RSX-11M-PLUS systems (Figure 6-2), one of the following:

D	System-controlled (dynamic) partition
P	Secondary pool partition

⓫ Number of tasks in memory and amount of memory they use, and number of active tasks swapped or checkpointed out of memory and the amount of memory they would require.

⓬ Name of each task, common, or driver in memory and its location in memory using the following symbols to designate size, type (task, common, or driver), and other attributes:

Symbol	Attribute
< >	Active task
[ ]	Task not active, yet occupies memory
! !	Named common
+ +	Unnamed common (displayed name is first attached task)
( )	Loaded driver using device mnemonic
-----	Task not fixed in memory
=====	Task fixed in memory

The hyphens and equal signs represent the amount of memory that each task, driver, or common occupies. Where the display shows only one delimiter and no hyphens or equal signs, the open delimiter is in the same location as the closing delimiter of the preceding task.

⓭ Partition size and location. The beginning of each partition is marked with the same symbols as those listed previously plus the following additions for both operating systems:

E Executive
P Pool

The lines of asterisks are proportional representations of the amount of memory occupied by each partition. The numbers are in 1K word increments. Each numerical character also represents the same amount of memory as an asterisk. RMD always divides the system memory into groups of eight units.

⓮ System error count sequence recorded by the Error Logger (always zero if Error Logger not present in your system).

6.4.1 Altering the Memory Display from the Setup Page

To alter the Memory Display, you press the ESCAPE key, which accesses the setup page for the Memory Display. The setup page documents and prompts you for commands which you use to alter the Memory Display. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the Memory Display are:

- FREEx=ddnn: where x is a number from 0 to 3 and ddnn: is a device name and number
- RATE=s where s is the replot rate in seconds

You can truncate these commands to their shortest unique forms. These setup parameters stay in effect until you alter them, even if you switch to another display page and back to the Memory Display.

6.4.1.1 **The FREE Command** - You use this command to determine the four Files-11 devices for which you want the Memory Display to show the available free blocks. This information is (7) in both Figures 6-1 and 6-2. The default is your system disk (SY:) and the next three Files-11 devices in your configuration.

6.4.1.2 **The RATE Command** - You use this command to determine how often RMD replots the Memory Display if you are running RMD on a CRT. The default replot rate is once per second.

6.4.2 Altering the Memory Display from the MCR Command Line

You can enter the FREE and RATE commands as part of the MCR command line. (The general format of the MCR command line is discussed in Section 6.2.1.2.) For example, when you enter the MCR command line

```
>RMD M,FREE3=DM0:,RATE=3
```

RMD displays the number of free blocks on DM0: instead of the fourth Files-11 device in your configuration and replots the display every three seconds instead of every second.

To change the second and third Files-11 devices to DB1: and DB2: respectively, use the MCR command line:

```
>RMD M,FREE1=DB1:,FREE2=DB2:
```

These setup parameters stay in effect until you alter them by using setup commands from the setup page. Even if you switch to another display page and back to the Memory Display, RMD does not restore the default setup parameters.

6.5 THE ACTIVE TASK DISPLAY

The Active Task Display shows you the active tasks in the system. You access this display from another display page by pressing the A key (for Active Task). To access this display from the MCR command line, you type:

```
>RMD A
```

This display has six fields:

- Name of the task
- Length of the task in octal bytes
- Terminal that issued the task
- Running priority of the task
- Outstanding I/O count
- Status flags

The status flags use the same mnemonics as the MCR ATL command. See the RSX-11M/M-PLUS MCR Operations Manual for the definitions of the status flags.

6.5.1 Altering the Active Task Display from the Setup Page

To alter the Active Task Display, you press the ESCAPE key, which displays the setup page for the Active Task Display. The setup page documents and prompts you for commands that you use to alter the Active Task Display. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the Active Task Display are:

- OWNER=ttnn: where ttnn: is the terminal that issued the task
- PRIORITY=p where p is a task's running priority
- RATE=s where s is the replot rate in seconds
- TASK=taskname where taskname is the name of the task whose header you want to display

You can truncate these commands to their shortest unique forms. These setup parameters stay in effect until you alter them, even if you switch to another display page and back to the Active Task Display.

6.5.1.1 **The OWNER Command** - This command allows RMD to display only those tasks that have been issued by a particular terminal. The default is ALL, which displays tasks issued from all terminals.

6.5.1.2 **The PRIORITY Command** - The active task list may be too long to fit on one screen. You use the PRIORITY command to determine the highest priority tasks that you want to see. The default is 250, the highest possible priority.

6.5.1.3 **The RATE Command** - This command allows you to determine how often RMD replots the Active Task Display, if you are running RMD on a CRT. The default replot rate is once per second.

6.5.1.4 **The TASK Command** - This command allows you to look at a specific task header. This command is an exception because it is the only setup command that switches display pages. There is no default for the TASK command. The Task Header Display is discussed in Section 6.6.

6.5.2 Altering the Active Task Display from the MCR Command Line

You can enter the same setup commands discussed in the previous section as part of the MCR command line. For example, when you enter the MCR command line:

```
>RMD A,PRIORITY=200,RATE=3
```

RMD displays the Active Task Display using 200 as the maximum task priority and replots the display every three seconds.

To display only those tasks issued by your terminal, type:

```
>RMD A,OWNER=TI:
```

6.6 THE TASK HEADER DISPLAY

The Task Header Display shows you the task header of the task you specify. You access this display from another display page by pressing the T key (for Task Header). If no task is currently specified, RMD shows you the setup page first so that you can specify the task whose task header you want RMD to display.

To access the Task Header Display from the MCR command line, you type:

```
>RMD T,TASK=taskname
```

where taskname is the name of the task whose header you want RMD to display.

If you invoke RMD from the MCR command line without specifying the name of a task:

```
>RMD T
```

RMD displays the setup page first so that you can specify the name of the task whose header you want RMD to display.

The Task Header Display shows you the following information about the specified task:

- Name of the task
- Name of the partition in which the task runs
- Status flags, which have the same mnemonics as in the Active Task display
- Owner of the task by terminal number
- Outstanding I/O count
- Default priority
- Running priority
- Swapping priority
- Length in decimal words
- Contents of the six general purpose registers, the program counter, and the Processor Status Word
- Contents of the Directive Status Word ($DSW)
- Local event flags
- Logical unit number (LUN) assignments to a maximum of 26 LUNs

When RMD displays file names in the list of LUN assignments, the filename and UIC displayed are the filename and UIC of the file when it was created. If the file has been renamed, the RMD display may not reflect the current UIC and/or filename.

6.6.1 Altering the Task Header Display from the Setup Page

To alter the Task Header Display, you press the ESCAPE key, which displays the setup page for the Task Header Display. The setup page documents and prompts you for commands that alter the Task Header Display. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the Task Header Display are:

- RATE=s where s is the replot rate in seconds
- TASK=taskname where taskname is the name of the task whose header you want to display

You can truncate these commands to their shortest unique forms. These setup parameters remain in effect until you alter them, even if you switch to another display page and back to the Task Header Display.

6.6.1.1 **The RATE Command** - This command allows you to determine how often RMD replots the Task Header Display if you are running RMD on a CRT. The default replot rate is once per second.

6.6.1.2 **The TASK Command** - This command changes the task header to be displayed. There is no default. The TASK command has the same function as the TASK command on the Active Task Display, except that here it does not switch display pages.

6.6.2 Altering the Task Header Display from the MCR Command Line

Both the RATE and TASK commands are available to you from the MCR command line. If you invoke RMD using the following command line:

```
>RMD T,RATE=2,TASK=...EDT
```

RMD displays the Task Header Display for EDT, replotting the page every two seconds. You must give the full task name. In this case the taskname is "...EDT", not "EDT".

If you do not specify a task name in the MCR command line:

```
>RMD T,RATE=2
```

RMD first displays the setup page so that you can select a task whose header you want RMD to display. The replot rate will be set to two seconds.

6.7 DISPLAYS AVAILABLE ONLY ON RSX-11M-PLUS

Two displays are available only on RSX-11M-PLUS systems: the I/O Counts Display and the System Statistics Display. These displays are invoked by the commands RMD I and RMS S, respectively.

6.7.1 The I/O Counts Display

The I/O Counts display shows you I/O and error logging counts for up to six error logging devices. By default, the first six error logging devices in your hardware configuration are displayed. You access this display from another display by pressing the I key (for I/O). To access this display from the MCR command line, you type:

```
> RMD I
```

The I/O Counts Display has ten fields for each device:

- Device name of the displayed device
- Total number of I/O requests to the device
- Number of I/O requests issued to the device in the last second
- Average number of I/O requests issued to the device per second
- Average number of cylinders crossed per I/O request during the last second
- Average number of cylinders crossed per I/O request
- Number of words transferred during the last second
- Average number of words transferred per second
- Count and limit of soft I/O errors
- Count and limit of hard I/O errors

6.7.1.1 **Altering the I/O Counts Display from the Setup Page** - To alter the I/O Counts Display, you press the ESCAPE key, which displays the setup page for the I/O Counts Display. The setup page lists the available commands you can use to alter the I/O Counts Display, and prompts you for a command. You can enter multiple commands after each prompt by using commas as separators. The setup commands available for altering the I/O Counts Display are:

- DEVICEx=ddnn: where x is a number from 0 to 6 and ddnn: is an error logging device
- RATE=s where s is the replot rate in seconds

You can truncate these commands to their shortest unique forms. These setup parameters stay in effect until you alter them, even if you switch to another display page and then return to the I/O Counts Display.

6.7.1.1.1 **The DEVICE Command** - This command allows you to select which error logging devices are to be displayed. The device specified must be an error logging device. If you do not use the DEVICE command to specify a device or devices, the first six error logging devices in your hardware configuration are displayed by default. See the RSX-11M/M-PLUS Error Logging Manual for a list of the error logging devices.

6.7.1.1.2 **The RATE Command** - This command allows you to determine how often RMD replots the I/O Counts Display, if you are running RMD on a video display terminal. The default replot rate is once per second.

6.7.1.2 **Altering the I/O Counts Display from the MCR Command Line** - You can enter the DEVICE and RATE commands as part of the MCR command line. For example, when you enter the following command line, RMD displays I/O count information for DB3: and replots the display every two seconds:

```
> RMD I,DEVICE2=DR3:,RATE=2
```

The information for DR3: is displayed in the position that would normally display information for the third error logging device in your hardware configuration.

To change the first and second device display positions to DB1: and DB2:, respectively, use the following MCR command line:

```
> RMD I, DEVICE0=DB1:,DEVICE1=DB2:
```

These setup parameters stay in effect until you alter them by using setup commands from the setup page. Even if you switch to another display page, and then return to the I/O Counts Display, RMD does not restore the default parameters.

6.7.2 The System Statistics Display

The System Statistics Display shows you general information about the operation of your system. Much of this information comes from RSX-11M-PLUS Resource Accounting, so this display is only useful if Resource Accounting was included when your system was generated. You access this by pressing the S key. To access this display from the MCR command line, you type:

```
> RMD S
```

The display has 25 fields.

- Total number of tasks run
- Number of currently active tasks
- Total number of user logons
- Number of users currently logged on
- Total number of runs of the shuffler
- Error sequence count
- Pool information (in the same format as in the Memory Display)
- Secondary pool information (in the same format as in the Memory Display)
- Percentage of system memory used
- Percentage of system checkpoint files used

- Total number of directives issued
- Number of directives issued during the last second
- Average number of directives issued per second
- Total number of QIOs issued
- Number of QIOs issued during the last second
- Average number of QIOs issued per second
- Total number of running CPU ticks
- Number of running CPU ticks during the last second
- Average number of running CPU ticks per second
- Total number of zero CPU intervals
- Number of zero CPU intervals during the last second
- Average number of zero CPU intervals per second
- Total number of task checkpoints
- Number of task checkpoints during the last second
- Average number of task checkpoints per second

6.7.2.1 **Altering the System Statistics Display from the Setup Page** - You can alter the replot rate on the System Statistics Display by pressing the ESCAPE key, which displays the setup page for the System Statistics Display. The setup page lists the command format for altering the replot rate, and prompts you for a command. The command for altering the replot rate of the System Statistics Display is:

```
RATE=s  where s is the replot rate in seconds
```

You can truncate this command to the single letter R. This setup parameter stays in effect until you alter it, even if you switch to another display page and then return to the System Statistics Display.

The default replot rate is once per second.

6.7.2.2 **Altering the System Statistics Display from the MCR Command Line** - You can enter the RATE command as part of the MCR command line. For example, when you enter the following MCR command line, RMD replots the System Statistics Display once every two seconds:

```
> RMD S,RATE=2
```

This setup parameter stays in effect until you alter it by using the RATE=n command from the setup page. Even if you switch to another display page and then return to the System Statistics Display, RMD does not restore the default replot rate.

6.8 MODIFYING THE TASK-BUILD COMMAND FILE

You can modify the task-build command file to change the defaults for RMD.

On RSX-11M systems, this file is LB:[1,24]RMDBLD.CMD for mapped systems and LB:[1,20]RMDBLD.CMD for unmapped systems.

On RSX-11M-PLUS systems, this file is LB:[1,24]RMDBLD.CMD.

6.9 ERROR MESSAGES

RMD generates the following error messages:

RMD - Allocated screen buffer too small for this device

Explanation: RMD requires more internal memory to display the requested display on the type of terminal on which you are running RMD.

User Action: Rebuild RMD with a larger screen buffer by modifying the task-build command file as discussed in Section 6.8.

RMD - Illegal command - xxxxx

Explanation: You entered an illegal command xxxxx either on the MCR command line or in response to the COMMAND> prompt on a setup page.

User Action: Enter the correct command as documented in this chapter.

RMD - Page does not exist

Explanation: You requested a display page from the MCR command line that does not exist.

User Action: Enter the command line again specifying a correct display mnemonic.

RMD - Segment 'xxxxxx' not found

Explanation: The module xxxxxx was not found in the task image for RMD. This denotes an error in how RMD was task-built.

User Action: Check that RMD has been built with all the required modules.

RMD - Terminal type not defined

Explanation: The operating system and RMD do not recognize your terminal type.

User Action: Check your terminal type using the MCR command SET /TERM=TI:. If this setting is incorrect, use the MCR SET command to correct your terminal type setting. If the setting is correct, move to a different type of terminal.

RMD - Terminal type not set

Explanation: You did not build RMD to display the requested display page on the type of terminal to which your terminal is set.

User Action: Determine your terminal type setting using the MCR SET /TERM=TI: command. If this setting is incorrect, use the MCR SET command to correct your terminal type setting. If the setting is correct, move to a different type of terminal.

RMD - Terminal type not yet supported

Explanation: RMD does not recognize your terminal type.

User Action: Determine your terminal type setting using the MCR SET /TERM=TI: command. If this setting is incorrect, use the MCR SET command to correct your terminal type setting. If the terminal type is correct, you can write a terminal characteristics module for the terminal similar to the one in [14,10]V52CHA.MAC. Then assemble the module, include the new module in the ODL file and rebuild RMD.

CHAPTER 7

SETTING UP AND RUNNING THE QUEUE MANAGER

This chapter is directed to the system manager or operator who will be setting up and running the Queue Manager and batch processors on an RSX-11M or RSX-11M-PLUS system. Except where noted, all commands in this chapter are privileged.

Most RSX-11M/M-PLUS systems will include the Queue Manager for the convenience of multistream line printer spooling.

The Queue Manager on RSX-11M-PLUS also supports transparent spooling and batch processing. You can have the Queue Manager without batch processing, but you cannot have batch processing without the Queue Manager.

In UFD [1,2] on the system disk (LB:), you will find the file STARTUP.CMD. This file includes commands for setting up the Queue Manager on a standard configuration. This sets up the QMG subsystem on a baseline system with one print processor, and, on RSX-11M-PLUS systems, one batch processor. The information in the present chapter will help you alter STARTUP.CMD to set up a Queue Manager tailored to your installation.

This chapter consists of a narrative introduction and review of the process of setting up the Queue Manager and batch processors and detailed descriptions of all commands to the Queue Manager.

The queue management subsystem consists of three major components on RSX-11M. RSX-11M-PLUS systems have a fourth major component. Here are the components:

A command line interpreter (QMGCLI) that processes both print requests and queue access commands as described in the RSX-11M/M-PLUS Batch and Queue Operations Manual. QMGCLI communicates with the Queue Manager by sending data packets to it.

The Queue Manager (QMG) that controls the queueing and dequeueing of print jobs. QMG communicates with QMGCLI and the despooler tasks to process print jobs.

The despool prototype task (LPP) that attaches a device and despools print jobs to it. An LPP must be installed for each device to be used as a spool device with the task name ddn, where dd is the generic device name and n is the unit number. A despool device is a device under the control of QMG.

On RSX-11M-PLUS systems only, QMG also controls batch processing. The RSX-11M-PLUS QMGCLI accepts batch requests as well as print requests. The RSX-11M-PLUS QMG controls the queueing and dequeueing of batch jobs. The RSX-11M-PLUS QMG also includes a **batch processor prototype task, BPR.** This task creates a virtual terminal that passes commands to the batch processor and to a system CLI, either MCR or DCL, or a

user-written CLI. (Note that QMGCLI is not a CLI in this sense.) The BPR task can be installed as many as 16 times with task names in the form BAPnnn, where nnn represents one to three Radix-50 characters.

On RSX-11M-PLUS only, the source files in SY0:[121,10] make up the print processor task. You can use this as a model if you wish to write output despoolers for your applications.

7.1 INTRODUCTION AND REVIEW

The following narrative explains how to bring QMG into your system. Both MCR and DCL commands are included.

This can be done interactively, but in most cases you will include it in an indirect command file invoked as part of the system's STARTUP procedures.

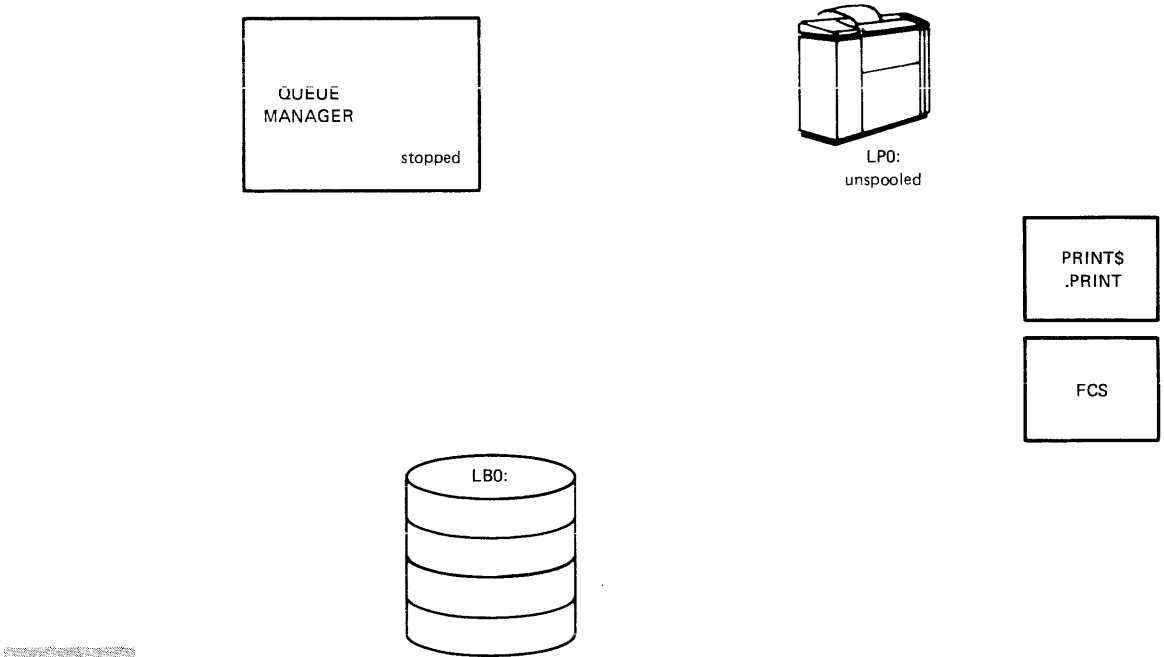
NOTE

On RSX-11M-PLUS systems with DCL as a CLI, the DCL form of these commands can be used on both MCR and DCL terminals. This is for compatibility with RSX-11M-PLUS V1.0.

Each of the following steps includes a block diagram of the Queue Manager, processors, and devices as they exist at that point in the procedure. Those parts of the subsystem brought into being by that step are shown in red.

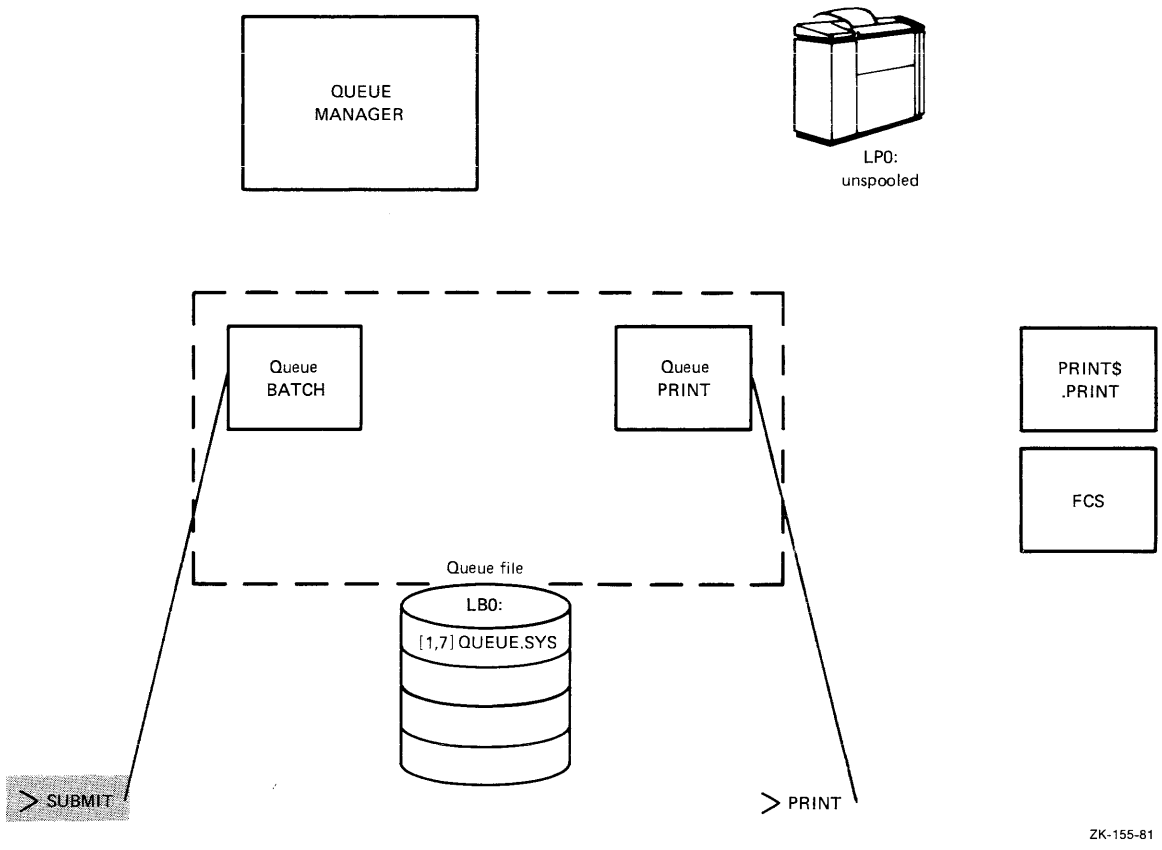
1. Install the Queue Manager. It will be found with the privileged utility tasks.

```
DCL>INSTALL/CHECKPOINT $QMG

MCR>INSTALL $QMG/CKP=YES
```



2. Start QMG. The command starts the Queue Manager and initialize the default queues PRINT and BATCH. It creates the queue file on LB0: if the file is not there. The command also clears all assignments previously recorded. If the queue file was intact, all queues will still contain their jobs, but are not assigned to processors. (Processors will have to be reinitialized, but queues will not.)

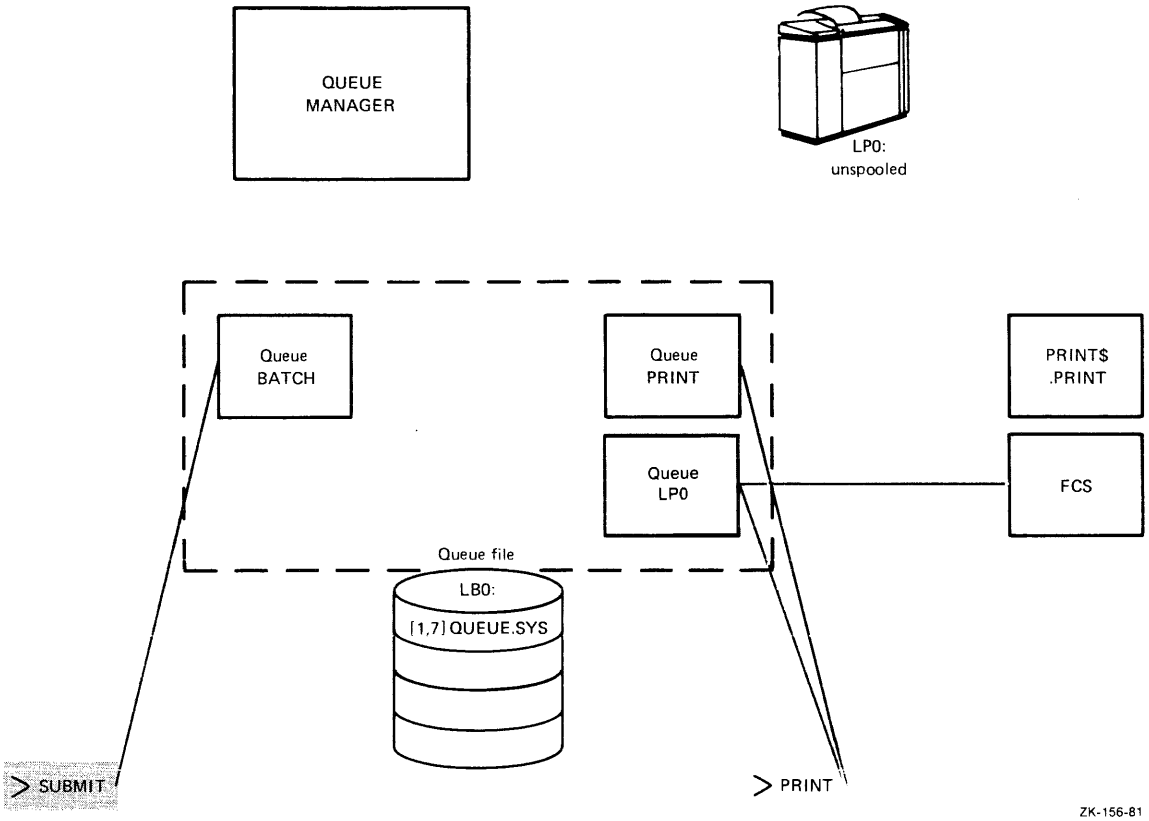
```
DCL>START/QUEUE/MANAGER

MCR>QUE /STA:QMG
```



3. The default print queue PRINT was initialized in the previous step, but you must initialize at least one more queue named for each line printer. This queue will serve the many system tasks that conventionally send output to LP: by directing this output to QMG. This is known as a device-specific queue. PRINT is a general queue.

   A device-specific queue must be initialized before the associated device is initialized (spooled). For LP0:, the device-specific queue is named LP0.

```
DCL>INITIALIZE/QUEUE LP0/PRINT

MCR>QUE LP0:/CR:P
```

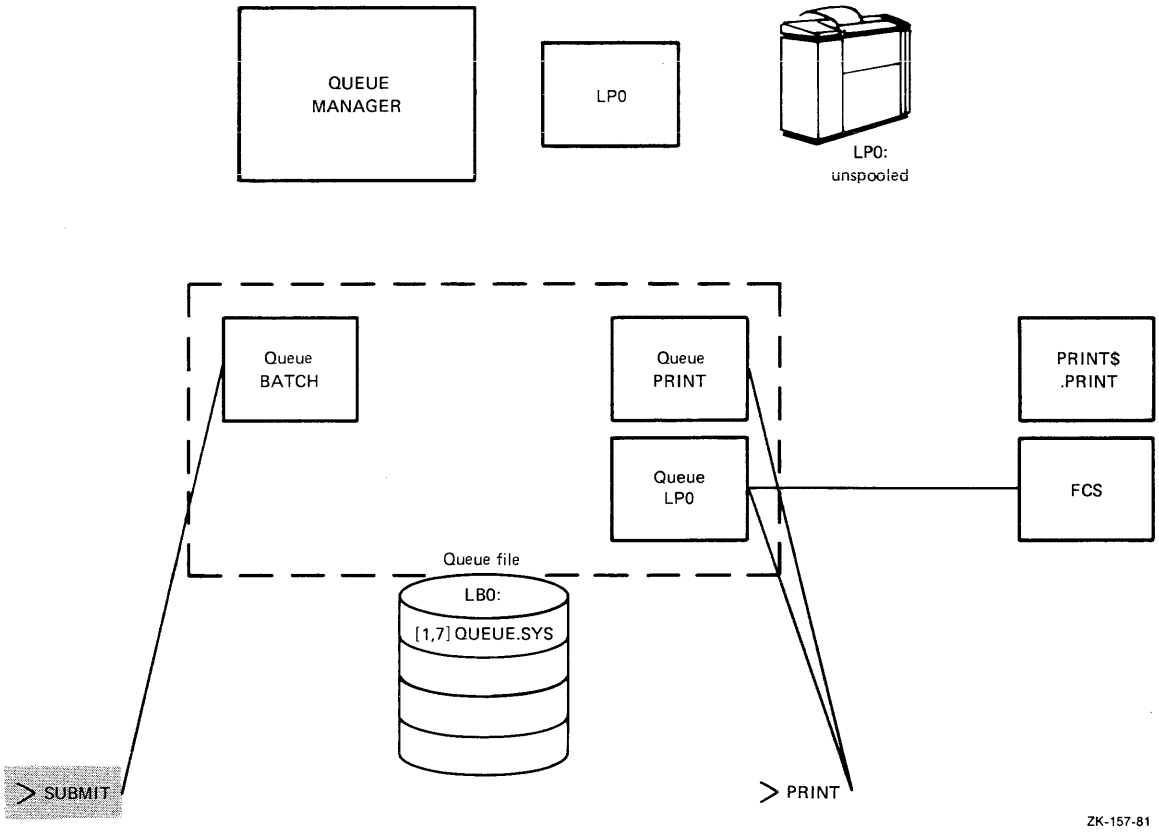


Initialization names, creates, and starts a queue.

You will need this extra queue even if your system has only one printing device. You can initialize as many as 14 additional print queues for a total of 16. One must be the default queue PRINT. The remaining 15 can be device-specific or general queues. There must be a device-specific queue for every spooled (initialized) output device. Queues for application processors are print queues.

4. Now you must install the print processors. You must install a print processor for each physical printing device that will be set spooled (by means of INITIALIZE). This processor "owns" the printer; the processor name must be derived from the printer name. LP0: is owned by LP0. LPP.TSK installs automatically as LP0. The print processors will be found with the privileged utility tasks.

```
DCL>INSTALL $LPP

MCR>INS $LPP
```



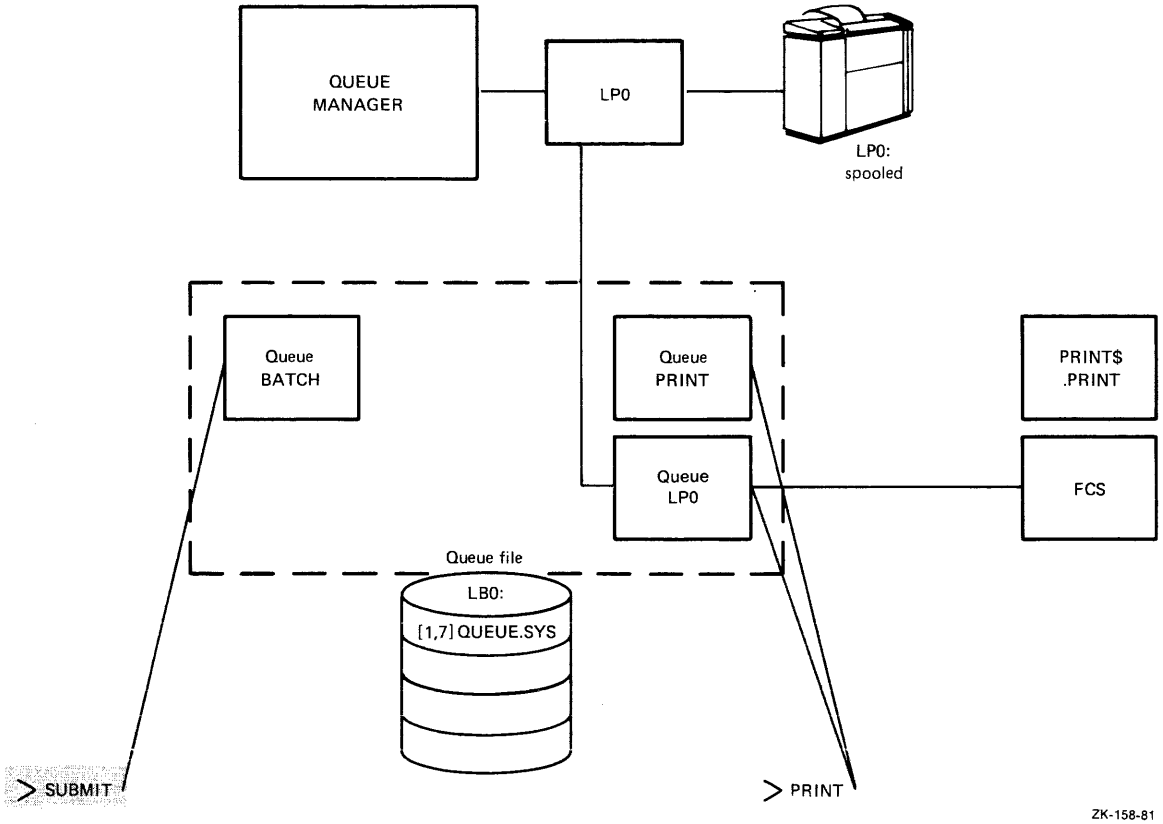
ZK-157-81

If you have additional output devices, use the /TASK qualifier to install their print processors under the proper names. You can initialize as many as 15 print processors. Applications processors are considered print processors for this count.

```
DCL>INSTALL/TASK:LP1 $LPP

MCR>INS $LPP/TASK=LP1
```

5. You must initialize your print processors, setting whatever attributes you wish. Any initialization of a print processor sets the associated output device spooled.

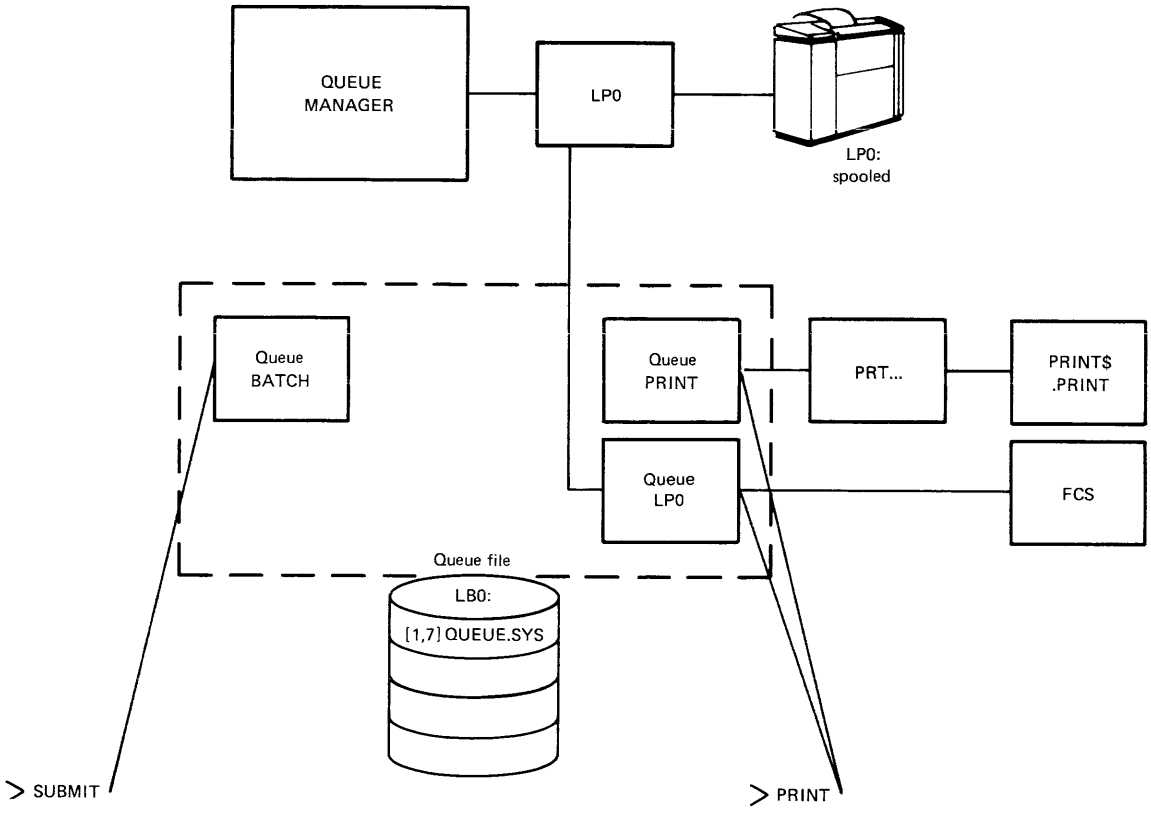
```
DCL>INITIALIZE/PRINTER LP0/FLAG_PAGE:2/FORM:0/LOWERCASE

MCR>QUE LP0:/SP/FL:2/FO:0/LOW
```



The command assigns the queue LP0 to despooler LP0. The queue must exist before you initialize the processor LP0.

6. Next, you must install the interface between the Queue Manager and other system tasks. This interface is handled by $QMGPRT.TSK. It must be installed as PRT.... It will be found with the nonprivileged utility tasks.

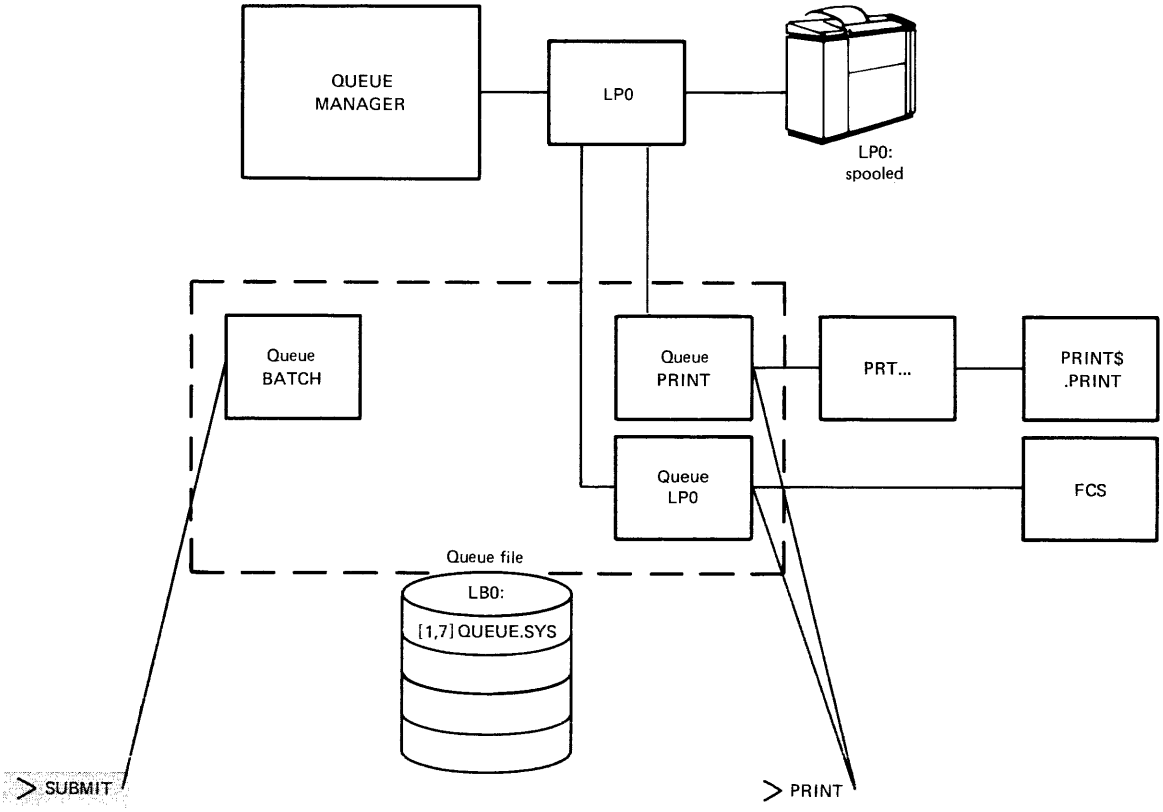
```
DCL>INSTALL/TASK:PRT... $QMGPRT

MCR>INS $QMGPRT/TASK=PRT...
```



ZK-159-81

7. Now, you must assign queues to print processors. Assignment establishes the path between queue and print processor. The Queue Manager can send jobs only to assigned processors.

```
DCL>ASSIGN/QUEUE PRINT LP0

MCR>QUE LP0:/AS:PRINT
```



ZK-160-81

Since queue LP0 was assigned to processor LP0 at initialization, this means two queues are being assigned to the same processor.

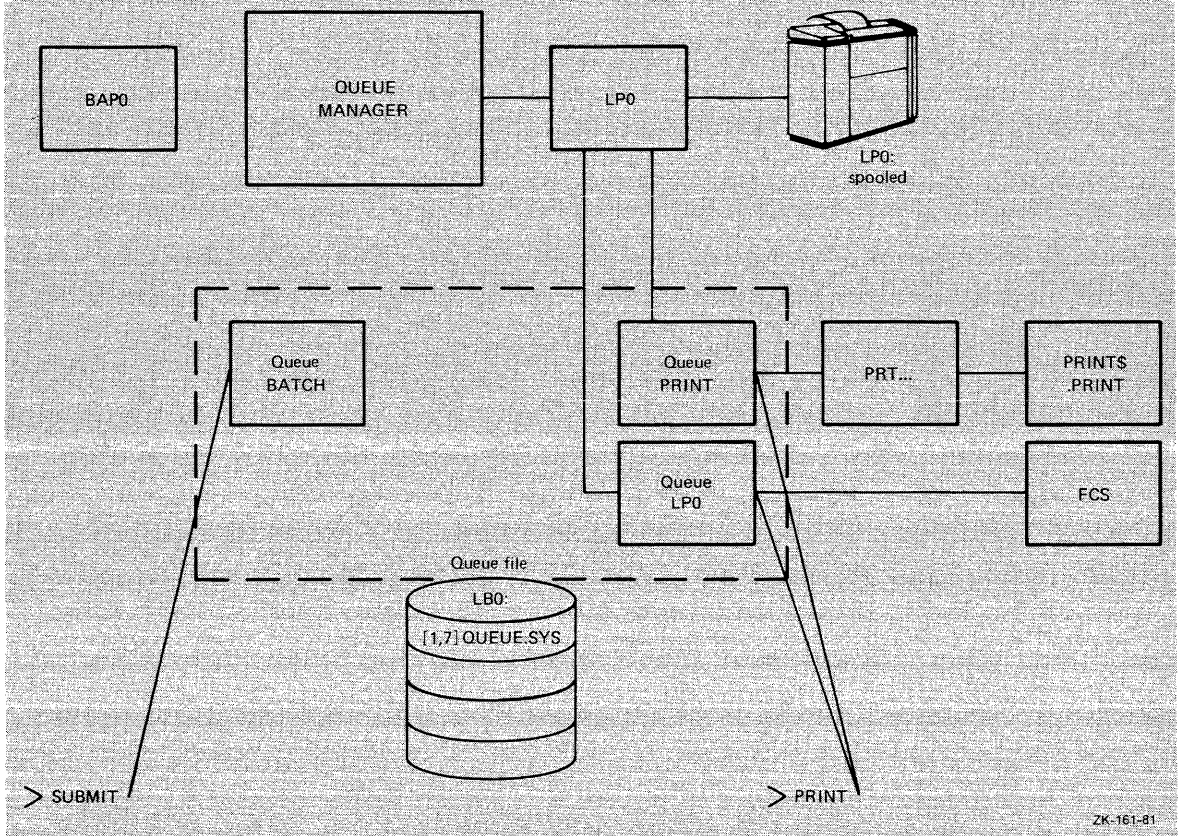
You can assign a queue to more than one processor and you can assign more than one queue to a processor.

If you initialize a queue, but do not assign it, jobs sent to that queue will not be processed until the queue is assigned.

This completes the installation process for the RSX-11M QMG. The remaining steps apply to RSX-11M-PLUS only.

8. You must now install the batch processor. This task will be found with the privileged utility tasks.

```
DCL>INSTALL $BPR

MCR>INS $BPR
```

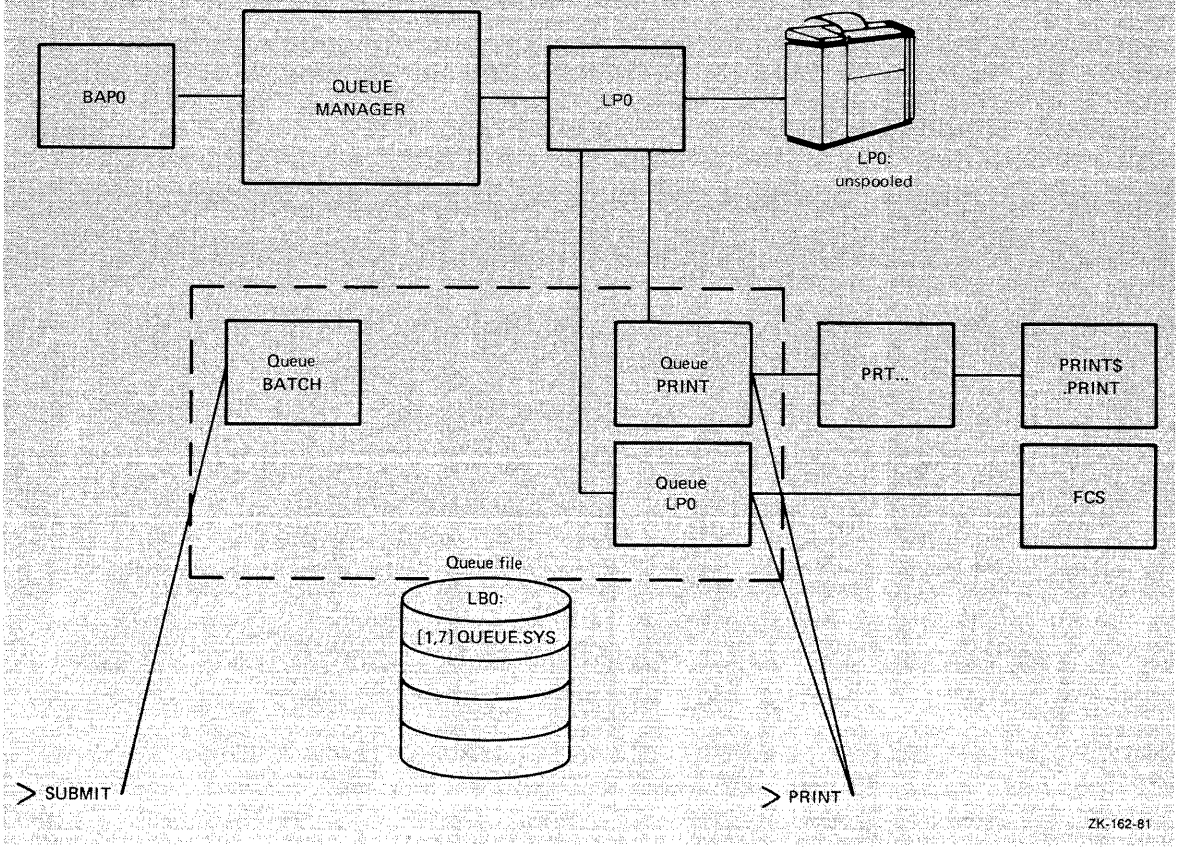


BPR.TSK installs automatically as BAP0.

You can install as many as 16 batch processors. They must have names in the form BAPn.

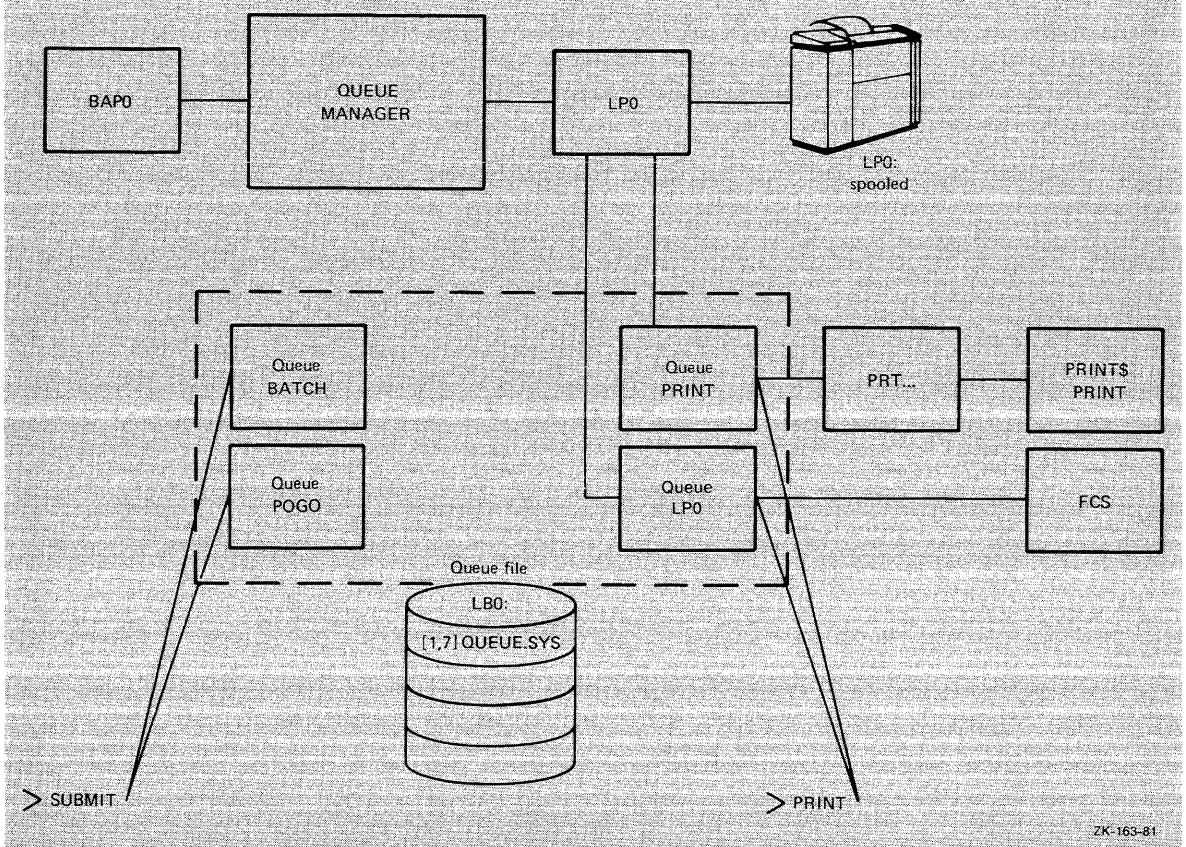
9. Now you initialize the batch processors.

```
DCL>INI/BATCH_PROCESSOR BAP0

MCR>QUE BAP0:/BATCH
```



10. The default queue BATCH was initialized by START/QUEUE/MANAGER, but you may initialize as many as 15 additional batch queues with names of as many as six alphanumeric characters in any form.

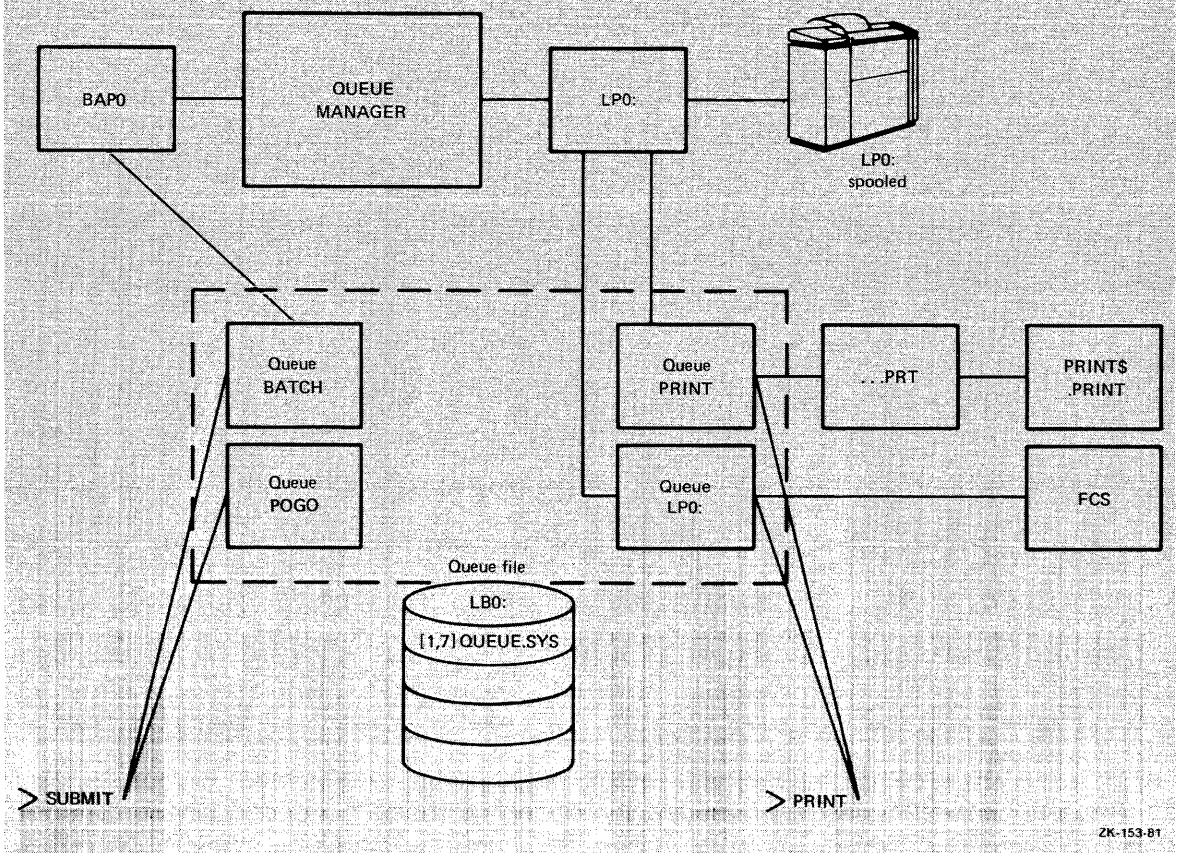
```
DCL>INI/Q POGO/BATCH

MCR>QUE POGO:/CR:B
```



Note that batch processors do not need device-specific queues because no devices are associated with them.

11. Finally, you must assign batch queues to batch processors.

```
DCL>ASSIGN/Q BATCH BAP0

MCR>QUE BAP0:/AS:BATCH
```



If you initialize a queue but do not assign it to a processor, jobs can be queued, but will not be processed until the queue is assigned.

7.2 QUEUE MANAGER COMMAND DESCRIPTIONS

The operations of QMG are controlled through a number of DCL and MCR commands. The functions invoked are identical, but the command syntax is different. Use whichever command syntax you are accustomed to. In this manual, both forms of the syntax appear in the command descriptions. The DCL commands and command elements are to the left and the MCR to the right. In the section titles, MCR commands or command elements appear in parentheses. In the examples, commands are identified explicitly by the DCL> and MCR> prompts.

The processors, devices, and device-specific queues all have names derived from the name of the spooled device. Thus, if LP1: is the spooled device, the processor that "owns" it is called LP1 and the device-specific queue is also called LP1. The colon (:) portion of the device name can be omitted from references in DCL commands to processors, devices, and queues. The colon portion must be included in references in MCR commands to processors, devices, and queues. You can include the colon portion in any DCL command, but it is required only in MCR commands. Further, in MCR commands, all queue names must be terminated by a colon, whether they are derived from device names or not. Thus, if you wish to refer to a queue named XPRESS in an MCR command, you must refer to that queue as XPRESS:.

The general format of DCL QMG commands is:

```
DCL>DCLcommand/QMGqualifier[s] parameter[s][/qualifier[s]]
```

The general format of MCR QMG commands is:

```
MCR>QUE ddnn:  /function[:option[s]][/sw[s]]
```

Command descriptions are given in the order they are used in the narrative, except that any counteracting commands are described just after the set-up commands they counteract.

7.2.1 INITIALIZE/ (QUE /CR/SP)

These commands create, name, and start a queue or processor.

When you initialize an output processor, you are setting its corresponding device spooled. A spooled device can be accessed by the Queue Manager. The processor or task must be installed.

7.2.1.1 INITIALIZE/QUEUE (QUE /CR) - INITIALIZE/QUEUE creates, names, and starts a queue.

Format

```
DCL>INITIALIZE/QUEUE queuename/qualifier
```

DCL Parameter Qualifiers

```
/BATCH
/PRINT
```

```
MCR>QUE ddnn:/CR:typ
```

MCR Queue Types

```
PRINT
BATCH
```

Parameter

queuename

Specifies the name of the queue to be initialized.

Queue names BATCH and PRINT are reserved, because the BATCH and PRINT queues are initialized by the START/QUEUE/MANAGER (QUE /STA:QMG) command.

The queue named BATCH is initialized on RSX-11M systems, but is not used, and is not accessible or displayed.

You must initialize a queue for every output device to which the Queue Manager is to send print jobs. These are called device-specific queues.

Otherwise, any six alphanumeric characters make an acceptable queue name for a general queue. If you have no processor LP1, LP1 is then a general queue name.

Device-specific queues must be initialized before the associated processor.

Parameter Qualifiers

This qualifier can be either /BATCH or /PRINT. The qualifier tells the Queue Manager what kind of queue is being initialized. Queues for applications processors are print queues.

The default qualifier is /PRINT. For the sake of clarity, you should always include the qualifier when initializing a queue.

Examples

```
DCL>INITIALIZE/QUEUE XPRESS/PRINT

MCR>QUE XPRESS:/CR:PRINT
```

This example creates, names, and starts a print queue named XPRESS. The queue can now accept print jobs, but it must be assigned to an output processor before jobs can be taken from the queue.

```
DCL>ASSIGN/QUEUE XPRESS LP0:

MCR>QUE LP0:/AS:XPRESS
```

This command assigns the queue initialized in the previous example to a print processor.

```
DCL>INITIALIZE/QUEUE LP0/PRINT
DCL>INITIALIZE/PRINTER LP0:

MCR>QUE LP0:/CR
MCR>QUE LP0:/SP
```

This example creates, names, and starts a print queue named LP0. This is a device-specific queue. Its name is derived from the name of an output device. This queue must be initialized before the output device despooler can be initialized. After both the queue and output device despooler are initialized, the queue can accept print jobs and the named printer is set spooled. Jobs can be placed in the queue.

Notes

No more than 16 print queues (including queues for applications processors) can be initialized.

On RSX-11M-PLUS, no more than 16 batch queues can be initialized.

These counts include the default PRINT and BATCH queues.

7.2.1.2 **INITIALIZE/PROCESSOR (QUE /SP)** - This command creates, names, and starts an output despooler or batch processor. When you initialize an output despooler, this task "owns" a hardware device. This is setting the device spooled. A spooled device can be accessed by the Queue Manager.

Format

```
DCL>INITIALIZE/processortype          processorname/[/qualifier]

MCR>QUE processorname:/SP[/sw[s]]
```

DCL Processor Types	MCR Function
/PROCESSOR	/SP
/PRINTER	/SP
/DEVICE	/SP
/BATCH_PROCESSOR	/BA
/APPLICATIONS_PROCESSOR	/SP/EX

DCL Qualifiers	MCR Switches
/FLAG_PAGE:n	/FL:n
/LOWERCASE	/LOW
/UPPERCASE	/NOLO
/FORMS:n	/FO:n
/SHAREABLE	/SHR

Processor Types

```
/PROCESSOR  (/SP)
/PRINTER    (/SP)
/DEVICE     (/SP)
```

The DCL processor types fall in three groups. One group consists of /PROCESSOR, /PRINTER, and /DEVICE. These three are synonyms. The effect of each is the same. The synonyms are provided to assure unambiguous commands and documentation. Use /PRINTER for line printers and /DEVICE for other output devices.

If you use one of these three synonyms, a device-specific queue must exist before the output despooler task can be initialized. See Notes.

The MCR /SP switch performs the same function. There are no synonyms in the MCR form of this command. A device-specific queue must exist before the output despooler task can be initialized. See Notes.

/BATCH_PROCESSOR (/BA)

Use this processor type (or option) when initializing a batch processor.

/APPLICATIONS_PROCESSOR (/SP/EX)

Use this processor type when the destination of the output is not a physical device, but rather some software application. Use /APPLICATIONS_PROCESSOR when you are initializing a user-written output despooler for which the physical device is not currently in the system. This makes it possible for the Queue Manager to despool output to an applications task, to a network, or to some other instance of down-line loading.

In MCR format, you must combine the /SP and /EX options.

When you use this command in either form, the Queue Manager does not check for the existence of a physical device before initializing the processor.

Parameter

processorname

Remember, the names of the device, processor, and queue are the same, and interchangeable. MCR users must include the colon (:) but DCL users can omit it.

You must initialize device-specific queues before you initialize the print processor with the same number.

If initializing a batch processor, its name must be in the form BAPn.

DCL Qualifiers and MCR Functions

These qualifiers and functions have meaning for initializing print processors only.

With the exception of setting the number of flag pages, these qualifiers have no effect on the printed output of the printer being initialized. Rather, these qualifiers define the kind of print jobs the processor can accept.

Different kinds of print jobs are established by the PRINT command which places them in the queue.

/FLAG_PAGE:n (/FL:n)

Specifies how many flag pages are to precede jobs and files printed by this processor. Default is 0; n must be 0, 1, or 2.

If the processor is being initialized to accept special forms, you may want to set this qualifier to 0.

If you wish, you can change the value set here with the /FLAG qualifier to the START/ command (/FL switch to /STA option).

/LOWERCASE (/LOW)

If you initialize a printer as lowercase, that printer will accept print jobs with lowercase or uppercase specified in the PRINT command. Normally, you should not use this qualifier unless the hardware involved has both the uppercase and lowercase character sets.

/UPPERCASE (/NOLO)

If you initialize a printer as uppercase, that printer will not accept print jobs with lowercase specified in the PRINT command. Normally, you should not use this qualifier unless the hardware involved does not have the lowercase character set. This is the default.

/FORMS:n (/FO:n)

This number can be anything from 0 through 255. The default is 0.

These numbers are to be specified by your users in their PRINT commands.

You should determine which forms will be used on your system and assign each a number. You should then edit the file LB:[1,24]LPPBLD.CMD to set up the forms table for the print processor to correspond to the numbers you have assigned to the forms. Then you must build the processor.

If the right forms are not in an available printer, then their jobs will be waiting until the printer has been stopped, the proper forms put in place, and the printer restarted with a new value for /FORMS (/FO).

Forms differ by their length and width. Both values can be set in the forms table.

Setting width is simply a matter of determining how many columns are needed to fill in the form. There are no standard widths for forms, but 80 and 132 are the most common.

Setting length requires you to determine whether your forms are of a standard length.

"Standard length" is any form length that can be set on your printing hardware. If the hardware can handle the form by setting a response to a form-feed character, then it is a standard length form, and the form feed is a "real," that is, hardware, form feed.

If the form length cannot be set on the hardware, then it is a nonstandard length. In this case, the form feed must be replaced by an appropriate number of line feeds. This is called a "simulated," that is, software, form feed.

The following table shows the forms requirements for a hypothetical installation. The form names describe different uses that special forms might be put to, but have no significance otherwise.

Form	Assigned n	Width	Length	Standard?
Accordion	0	132	66	Y
Paychecks	1	40	10	Y
Invoices	2	80	66	Y
Stationery	3	80	66	Y
Moon Diary	4	22	43	N
Star Log	5	112	113	N

The forms table can be changed at LB:[1,24]LPPBLD.CMD. This procedure is explained in the comments in this file.

As shipped, FORMS:0 is 66 lines long by 132 columns with real form feeds. FORMS:1 is 1 line long by 132 columns with simulated form feeds; FORMS:2 is 2 lines long by 132 columns with simulated form feeds; and so on through FORMS:66 which is 66 lines long by 132 columns with simulated form feeds. The remaining FORMS:n entries are set to 1 line long by 132 columns with simulated form feeds. Thus, if you specify /FORMS:43, when the printer encounters a form feed, it will count the number of lines on the page and subtract that number from 43. The printer will then reel off a number of line feeds equal to the difference. You may find this arrangement satisfactory if your installation rarely uses forms other than standard line-printer paper, but if you have more than one standard form, you will probably want to make the changes in the forms table entries.

/SHAREABLE (/SHR)

Establishes the processor as shareable. This means the physical device is not "owned" by a processor if it is not printing a job. Normally, the output processor attaches the device when it is initialized and no other job can access the device until it is detached.

If you initialize the processor as shareable, the device is attached at the beginning of the QMG print job and detached at the end of the job, leaving the device free for other uses.

On RSX-11M-PLUS, if you initialize a processor as shareable, transparent spooling to that device will not work.

Examples

```
DCL>INITIALIZE/PROCESSOR LP0:

MCR>QUE LP0:/SP
```

LP0: is set spooled. Jobs can be passed to it from the Queue Manager.

You must initialize a device-specific queue named LP0 before you can initialize the processor for device LP0:.

```
DCL>INITIALIZE/PRINTER LP1:/FLAG_PAGE:2

MCR>QUE LP1:/SP/FL:2
```

This command initializes a print processor to control LP1: and specifies that jobs run on this printer will be preceded by two flag pages. Jobs specifying flag pages in the PRINT command will have two flag pages at the head of the job.

```
DCL>INITIALIZE/PRINTER LP1/NOLOWER

MCR>QUE LP1:/SP/NOLOW
```

This command initializes a print processor to control LP1: and states that the printer will only accept jobs specifying uppercase in the PRINT command.

```
DCL>INI/BATCH_PROCESSOR BAP0

MCR>QUE BAP0:/BA
```

This command creates, names, and starts batch processor BAP0, on an RSX-11M-PLUS system.

```
DCL>INI/PRO LP0/FLAG:2/FORMS:0/LOWER

MCR>QUE LP0:/SP/FL:2/FO:0/LOW
```

This command creates, names, and starts print processor LP0. The printer has the lower-case character set. Jobs will have two flag pages when printed.

Notes

No more than 16 batch processors on an RSX-11M-PLUS system, or 16 print processors (including applications processors) can be initialized.

If you are initializing an output despooler that attaches a physical device, the system checks for the existence of the device and of its device-specific queue before the command is executed. Therefore, the following procedure must be followed.

1. The device must be part of the current system.

2. You must initialize a device-specific queue for the device before you initialize an output despooler of the same type and number.

   When you initialize an output processor, it is named and started and has its device-specific queue assigned to it.

3. For each print processor, there must be a print queue, called the device-specific queue, with a corresponding name. The printer named LP0: is owned by the output despooler named LP0. There must also be a device-specific queue named LP0. The processor cannot be named until the device-specific queue has been initialized. This queue will be assigned automatically to the processor whose name it shares, but the device-specific queue can be deassigned and assigned elsewhere after initialization.

   This procedure does not apply to initializing batch processors or applications processors. Neither a physical device nor a device-specific queue is needed to initialize a batch processor or applications processor.

   There can also be print queues whose names do not correspond to output devices. These are called general queues.

The lowercase/uppercase attributes have no effect on the nature of the printer's output. They do not cause jobs to be printed all uppercase. These qualifiers set up a "mask" that must be matched by the attributes of print jobs before they can be sent to the printer. A printer without the lowercase character set can be initialized lowercase and it will accept print jobs with lowercase specified in the PRINT command. This might be done if a printer with the lowercase character set was temporarily unavailable. As a general rule, however, you should only initialize printers with the characteristics they actually have to avoid confusion.

(The line printer driver does, however, affect the printer's output, through lowercase-to-uppercase translation. See the discussion of SET /LOWER in the RSX-11M/RSX-11M-PLUS MCR Operations Manual, or the discussion of SET DEVICE /LOWER in the RSX-11M/M-PLUS Command Language Manual.)

7.2.2 DELETE/ (QUE /DEL)

You can delete queues or processors.

7.2.2.1 **DELETE/QUEUE (QUE /DEL:Q)** - DELETE/QUEUE deletes queues by name.

Formats

DCL>DELETE/QUEUE queuename/ERASE

MCR>QUE queuename:/DEL:Q

Parameters

queuename/ERASE (/DEL:Q)

Specifies the name of a queue to be deleted. Only a privileged user can delete a queue using this parameter. The default queues BATCH and PRINT cannot be deleted.

If the queue has jobs in it, it will be marked for delete. When the last job leaves the queue, the queue itself is deleted. No new jobs can be entered.

You cannot delete device-specific queues until after you have deleted the processor with the same number. See next section.

Examples

DCL>DELETE/QUEUE MONA/ERASE

MCR>QUE MONA:/DEL:Q

This privileged command deletes the queue named MONA if it is empty. If the queue has entries, it is marked for delete, and will be deleted as soon as the last job is removed from the queue. Note the colon in the queue name in the MCR example.

7.2.2.2 **DELETE/PROCESSOR (QUE /UNSP)** - DELETE/PROCESSOR deletes print processors, output despoolers, or batch processors by name. The command also sets the device unspooled.

Format

DCL>DELETE/processortype processorname

MCR>QUE processorname:/UNSP

DCL Processor Types	MCR Functions
/PROCESSOR	/UNSP
/PRINTER	/UNSP
/DEVICE	/UNSP
/BATCH_PROCESSOR	/UNBA
/APPLICATIONS_PROCESSOR	/UNSP

Parameters

processorname

For physical devices, the name of the processor is identical with the name of the physical device. Thus, the device LP1: is

controlled by processor LP1:. DCL commands do not require that you include the colon (:), but you must include the colon in all MCR commands.

Batch processors have names in the form BAPnnn, where nnn is from one through three Radix-50 characters.

Applications processors have names of six Radix-50 characters.

Processor Types

/PROCESSOR (/UNSP)
/PRINTER
/DEVICE

The DCL processor types fall in three groups. One group consists of /PROCESSOR, /PRINTER, and /DEVICE. These three are synonyms. The effect of each is the same. The synonyms are provided to assure unambiguous commands and documentation. Use /PRINTER for line printers and /DEVICE for other output devices.

If you use one of these three synonyms, the device-specific queue cannot be deleted until after the output despooler task is deleted.

These distinctions do not apply in MCR.

/BATCH_PROCESSOR (/UNBA)

Use this processor type when deleting a batch processor.

/APPLICATIONS_PROCESSOR (/UNSP)

Use this processor type when the destination of the output is not a physical device, but rather some software application. Use /APPLICATIONS_PROCESSOR when you are deleting a user-written output despooler.

Examples

```
DCL>DELETE/PROCESSOR LP0

MCR>QUE LP0:/UNSP
```

This example deletes processor LP0, the owner of line printer LP0:.

```
DCL>DELETE/BATCH_PROCESSOR BAP2

MCR>QUE BAP2:/UNBA
```

The commands are equivalent; they delete the batch processor named BAP2, on an RSX-11M-PLUS system. Note the colon in the MCR command.

```
DCL>DELETE/PROCESSOR LP0
DCL>DELETE/QUEUE LP0/ERASE

MCR>QUE LP0:/UNSP
MCR>QUE LP0:/DEL:Q
```

This example shows the order you must follow to delete a device-specific queue.

Notes

DELETE/PROCESSOR (/UNSP) counteracts INITIALIZE/PROCESSOR (/SP).

For maximum clarity, operators using DCL should use the specific qualifiers /PRINTER and /DEVICE rather than the synonym /PROCESSOR when deleting output despoolers.

7.2.3 ASSIGN/QUEUE (QUE /AS:)

ASSIGN/QUEUE (QUE /AS:) establishes a path from a queue to a processor.

Format

DCL>ASSIGN/QUEUE queuename processorname

MCR>QUE processorname:/AS:queuename

Parameters

queuename

Specifies the queue that is to be assigned to the processor.

processorname

Specifies the processor to which the queue is to be assigned.

When assigning queues to output devices, use the the device name in the form ddnn:.

Notes

This command can be used to redirect output from one processor to another.

Other uses include grouping related processors, such as plotters, under a single queue.

7.2.4 DEASSIGN/QUEUE (QUE /DEA)

DEASSIGN/QUEUE (QUE /DEA) counteracts ASSIGN/QUEUE (/AS). It is used to eliminate the path from a queue to a processor.

Format

DCL>DEASSIGN/QUEUE queuename processorname

MCR>QUE processorname:/DEA:queuename

Parameters

queuename

Specifies the queue that is to be deassigned from the processor.

processorname

Specifies the processor from which the queue is to be deassigned.

When deassigning queues from output devices, use the device name in the form ddnn:.

7.2.5 STOP/ (QUE /STO)

STOP/ (/STO) is used to stop queues, the Queue Manager, or a processor.

7.2.5.1 STOP/QUEUE (QUE /STO:QUE) - STOP/QUEUE stops queues.

Format

DCL>STOP/QUEUE queuename

MCR>QUE queuename:/STO:QUE

Parameter

queuename

Specifies the queue to be stopped. Following the command, no jobs will be taken from the queue, but jobs can still be added to the queue.

If a job is active at the time this command is issued, the job will be allowed to complete processing but no further jobs will be dequeued.

Note the colon (:) in the MCR form of the command.

7.2.5.2 STOP/QUEUE/MANAGER (/STO:QMG) - STOP/QUEUE/MANAGER (/STO:QMG) stops the Queue Manager and deletes all processors, whether batch processors or output despoolers.

Format

DCL>STOP/QUEUE/MANAGER[/ABORT]

MCR>QUE /STO:QMG[:AB]

Notes

If you issue this command without the /ABORT (:AB) qualifier, QMG is marked for stop. As soon as all currently active jobs on all processors are done, QMG will stop. No more jobs can be entered in the queue. See also example in Section 7.2.6.2.

If you include the /ABORT (:AB) qualifier, all active jobs are held immediately and QMG exits without further delay.

7.2.5.3 **STOP/PROCESSOR (QUE /STO)** - STOP/PROCESSOR (/STO) stops a processor.

Format

DCL>STOP/processortype processorname[/qualifier]

MCR>QUE processorname:/STO[:opt]

DCL Processor Types	MCR Functions
/PROCESSOR	/STO
/PRINTER	/STO
/DEVICE	/STO
/APPLICATIONS_PROCESSOR	/STO
/BATCH_PROCESSOR	/STO

DCL Parameter Qualifiers	MCR Options
/PAUSE	<no equivalent>
/JOB_END	:EOJ
/FILE_END	:EOF
/ABORT	:AB

Parameter

processorname

When stopping print processors, use the device name in the form ddnn:.

This command has no effect on the status of jobs in queues, but no jobs will be dequeued while the processor is stopped.

Processor Types

/PROCESSOR (/STO)
/PRINTER
/DEVICE

The DCL processor types are synonyms. The effect of each is the same. These synonyms are provided to assure unambiguous commands and documentation. Use /PRINTER for line printers, /DEVICE for other output devices, and /BATCH_PROCESSOR for batch processors. MCR users use /STO across the board.

/BATCH_PROCESSOR (/STO)

Use this processor type when stopping a batch processor.

/APPLICATIONS_PROCESSOR (/STO)

Use this processor type when the destination of the output is not a physical device, but rather some software application. Use /APPLICATIONS_PROCESSOR when you are stopping a user-written output despooler.

Qualifiers

Only one qualifier is permitted. The choice of qualifier here affects the restarting of jobs when the processor is started again.

/PAUSE <no equivalent>

This is the default; it causes the processor to stop at the end of the current line. There is no MCR qualifier; /STO is sufficient.

/JOB_END (:EOJ)

Causes the processor to stop at end of the current job.

/FILE_END (:EOF)

Causes the processor to stop at end of the current file.

/ABORT (:AB)

Causes the processor to stop immediately. The current job is held in its queue.

Notes

This command is counteracted by START/PROCESSOR (/STA). You cannot issue START/PROCESSOR (/STA) until the processor actually stops.

7.2.5.4 **STOP/ABORT (QUE /KIL)** - STOP/ABORT (/KIL) deletes the active job on a given processor. Privileged users can delete any job; nonprivileged users can delete their own jobs. You do not need to know the queue name or job name to delete the job, only the name of the processor.

Format

```
DCL>STOP/ABORT processorname

MCR>QUE processorname:/KIL
```

processorname

Specifies the processor whose active job you wish to delete. Note that the MCR command format requires a colon (:) after the processor name.

You can stop jobs running on any processor under the control of QMG, including RSX-11M-PLUS batch processors.

Examples

```
DCL>STOP/ABORT
Processor?  LP0
```

This example shows how to stop a print job on LP0: The currently active job is deleted from the queue and the next eligible job is queued.

Notes

Use this command to stop a processor fast, such as a line printer printing nothing but form feeds.

As soon as the active job is deleted, QMG passes the next eligible job to the processor. The processor has not been aborted or killed, only the active job on that processor.

You can also delete the active job on a card-reader processor with this command.

7.2.6 START/ (QUE /STA)

START/ (QUE /STA) is used to start a queue, the Queue Manager, or a processor. START/ (QUE /STA) counteracts STOP/ (/STO).

7.2.6.1 START/QUEUE (QUE /STA) - START/QUEUE (QUE /STA) starts a queue.

Format

DCL>START/QUEUE queuename

MCR>QUE queuename:/STA

Parameter

queuename

Specifies the queue to be started. This command starts a queue that has been stopped. Note the colon (:) in the MCR format.

7.2.6.2 START/QUEUE/MANAGER (QUE /STA:QMG) - START/QUEUE/MANAGER (QUE /STA:QMG) starts the Queue Manager, initializes the default queues PRINT and BATCH, and creates the queue file LB0:[1,7]QUEUE.SYS if it does not exist. The command also clears all queue assignments. If the queue file was intact, all queues will still contain their jobs, but will not be assigned to processors.

Format

DCL>START/QUEUE/MANAGER

MCR>QUE /STA:QMG

Examples

```
DCL>START/QUEUE/MANAGER

MCR>QUE /STA:QMG
```

Starts Queue Manager. Initializes the default queues PRINT and BATCH, if they are not already initialized. Clears all assignments of queues to processors.

Any jobs that were active at the time the Queue Manager was stopped will be held when it is started again. All other jobs retain the status they had when QMG was stopped.

All processors must be reinitialized.

The default queue BATCH is initialized on RSX-11M systems as well as RSX-11M-PLUS, but it is neither accessible nor displayed.

```
DCL>STOP/QUEUE/MANAGER

MCR>QUE /STO:QMG
```

With this command, the Queue Manager is stopped, and all processors are deleted.

```
DCL>START/QUEUE/MANAGER

MCR>QUE /STA:QMG
```

This command clears all assignments of queues to processors.

```
DCL>INITIALIZE/PRINTER LP0
DCL>INITIALIZE/PRINTER LP1
DCL>INITIALIZE/BATCH_PROCESSOR BAP0
DCL>INITIALIZE/BATCH_PROCESSOR BAP1
DCL>ASSIGN/QUEUE PRINT LP0
DCL>ASSIGN/QUEUE PRINT LP1
DCL>ASSIGN/QUEUE BATCH BAP0
DCL>ASSIGN/QUEUE BATCH BAP1

MCR>QUE LP0:/SP
MCR>QUE LP1:/SP
MCR>QUE BAP0:/BA
MCR>QUE BAP1:/BA
MCR>QUE LP0:/AS:PRINT
MCR>QUE LP1:/AS:PRINT
MCR>QUE BAP0:/AS:BATCH
MCR>QUE BAP1:/AS:BATCH
```

These commands recreate the network of queues and processors. Note that RSX-11M systems have no batch processors.

```
DCL>RELEASE/JOB PRINT BOBO

MCR>QUE PRINT:BOBO/REL
```

This command releases a held job that was active at the time the system crashed.

7.2.6.3 **START/PROCESSOR (QUE /STA)** - START/PROCESSOR (QUE /STA) starts a batch processor or output despooler.

Format

```
DCL>START/processortype                processorname    [/qualifier[s]]
```

DCL Processor Types	DCL Qualifiers
/PROCESSOR	/FORMS:n
/PRINTER	/FLAG:n
/DEVICE	/CONTINUE
/BATCH_PROCESSOR	/RESTART
/APPLICATIONS_PROCESSOR	/NEXT_JOB
	/TOP_OF_FILE
	/BACKSPACE:n
	/FORWARDSPACE:n
	/AT_PAGE:n
	/ALIGN

MCR>QUE processorname:/STA[:opt[s]]

MCR Options

/FO:n
/FL:n
<no equivalent>
:NE
:RES
:PA:0
:BA:n
:FW:n
:PA:n
/AL

Parameters

processorname
ddnn:

Specifies the processor to be started. When starting a stopped output processor, use the device name in the form ddnn:.

Processor Types

/PROCESSOR
/PRINTER
/DEVICE
/BATCH_PROCESSOR
/APPLICATIONS_PROCESSOR

These are DCL synonyms. The effect of each is the same. These synonyms are provided to assure unambiguous commands and documentation. Use /PRINTER for line printers, /DEVICE for other output devices, /BATCH_PROCESSOR for batch processors, and /APPLICATIONS_PROCESSOR for applications processors. In MCR format, use the /STA option.

Qualifiers

Some qualifiers have meaning for starting either batch or print processors, and some have meaning for print processors only. See descriptive text.

This field is restricted. You can include either /FORMS: (/FO:) or /FLAG: (/FL:) or both, or you can include any one of the remaining qualifiers, if permitted.

/FORMS: (/FO:) and /FLAG: (/FL:) have meaning for print processors only.

/FORMS:n (/FO:n)

Overrides the forms value set when the processor was initialized. n can be any number from 0 through 256.

/FLAG:n (/FL:n)

Overrides the flag value set when the processor was initialized.

When the processor starts again, the Queue Manager sends the next job that matches the attributes of the processor.

/CONTINUE

Specifies that the processor carry on from wherever it was stopped. This is the default. MCR format requires no switch.

/RESTART (:RES)

Specifies that interrupted jobs be restarted from the beginning of the file that was being processed when the processor was stopped. This qualifier affects only jobs that were ACTIVE at the time the processor stopped.

This qualifier may conflict with the desires of your users who have specified /NORESTART in their PRINT or SUBMIT commands. This qualifier causes such jobs to be restarted at the start of the file being processed when the processor was stopped. You should check the queue display for such jobs before using this qualifier. The /CONTINUE (default) qualifier will not affect such jobs.

The action of this qualifier further depends on the qualifier to the STOP/ (/STO) command that stopped the processor in the first place.

If the qualifier was /PAUSE (the default for /STO), then any jobs active at that time will be restarted at their beginning.

If the qualifier was /FILE_END (:EOF), then any jobs active at that time will be restarted at their beginning, unless the active file was the last one in the job.

If the qualifier was /ABORT (:AB) (or if the system crashed), the current active job was held in its queue, so the restart qualifier will have no effect.

Likewise, if the qualifier was /JOB_END, (:EOJ) there will have been no current job, so the restart qualifier will have no effect.

/NEXT_JOB (:NE)

Specifies that the processor start at the beginning of the next job.

/TOP_OF_FILE (:PA:0)

Specifies that the processor start at the top of the current file in the current job.

The qualifiers /CONTINUE (the default for /STA), /RESTART (/RES), /NEXT_JOB (/NE), and /TOP_OF_FILE (PA:0) are appropriate for both print and batch processors.

/BACKSPACE:n (:BA:n)

Specifies that the processor start n pages back in the current job.

/FORWARDSPACE:n (:FW:n)

Specifies that the processor start n pages forward in the current job.

/PAGE:n (:PA:n)

Specifies that the processor start at specified page in the job that was active when the processor was stopped.

These page-related qualifiers have meaning for print processors only. They will have no effect unless the user specified a page length qualifier in the PRINT command.

/ALIGN (:AL)

Informs the print processor that the paper in the line printer is currently at the top of a form. It is used when you have stopped the print processor to change forms. The print processor maintains a count of how many lines it has gone down on a form. The /ALIGN (/AL) qualifier clears that count for new forms.

Examples

```
DCL>START/PRINTER LP0

MCR>QUE LP0:/STA
```

This example starts print processor LP0:. Printing continues from the point at which it was stopped.

```
DCL>START/PRINTER LP0/FLAG:2

MCR>QUE LP0:/STA/FL:2
```

This example starts print processor LP0:. The processor will put two flag pages at the head of the job and at the head of each file if the user specifies flag pages in a PRINT command.

```
DCL>START/BATCH_PROCESSOR BAP0

MCR>QUE BAP0:/STA
```

This example starts batch processor BAP0. Batch processing continues from the point at which it was stopped.

```
DCL>START/BATCH BAP0/CONTINUE

MCR>QUE BAP0:/STA
```

The two examples are equivalent.

```
DCL>START/PRINTER LP1/RESTART

MCR>QUE LP1:/STA:RES
```

This example starts print processor LP1:. The job active at the time the processor was stopped is restarted.

Notes

You should choose with care between the /RESTART (:RES) and /CONTINUE (MCR default) operations. A restart may conflict with the desires of users with print or batch jobs active at the time a processor is stopped.

Users can specify /NORESTART (/-RES) in their PRINT or SUBMIT commands. If a print or batch job with this attribute is active when the processor is stopped, and you start the processor with the /RESTART (:RES) qualifier, you will be restarting the jobs at the top of the file that was being processed when the processor was stopped, which the user did not want to happen. The /CONTINUE (MCR default) qualifier will continue such jobs as the user expected.

If you are restarting a processor to handle special forms, you will probably want to set the /FLAGS (:FL) attribute to 0 to avoid having a banner page printed on your special forms.

CHAPTER 8

THE SHUTUP PROGRAM

SHUTUP is the system program that enables orderly shutdown of an RSX-11M/M-PLUS system. SHUTUP sends warning messages, aborts nonprivileged tasks, and dismounts devices on the system.

8.1 PREREQUISITES TO RUNNING SHUTUP

To shut down a system using SHUTUP, the following tasks must be installed:

- INDIRECT, the MCR Indirect Command Processor.
- ACS, if dynamic checkpoint space is established.
- BYE, if multiuser protection support is included.
- DMO, if there are any mounted devices.
- ELI, if error logging is active.
- In an RSX-11M-PLUS system that includes Resource Accounting, ACC must be installed.

If LB:[1,2]SHUTUP.CMD is present, you may want it to install ACS, BYE, DMO, ELI, and ACC. See Section 8.5.

8.2 INVOKING SHUTUP

Invoke SHUTUP from a privileged terminal as follows:

```
>RUN $SHUTUP RET
```

On RSX-11M systems, SHUTUP displays the following message when invoked:

```
RSX-11M SHUT DOWN PROGRAM
```

On RSX-11M-PLUS systems, SHUTUP displays the following message when invoked:

```
RSX-11M-PLUS SHUT DOWN PROGRAM
```

8.3 SHUTUP INPUT

SHUTUP prompts for the number of minutes to wait before the system is to be shut down, between warning messages, and before disabling logins. If you specify less than four minutes to wait before shutdown, SHUTUP does not ask the last two questions. Instead, SHUTUP sets the interval between messages to one minute and disables logins immediately.

The minutes between messages must be greater than zero and less than the minutes before shutdown minus three. If you specify either value incorrectly, SHUTUP rejects your answer and reprompts.

The following is an example of SHUTUP prompts and user input:

```
Enter minutes to wait before shut down: 15 RET
Enter minutes between messages: 3 RET
Enter minutes to wait before disabling logins: 5 RET
```

After receiving your input, SHUTUP asks for confirmation as follows:

```
OK to shutdown? [Y/N]:
```

SHUTUP terminates without taking any action if your response to the question is N. If your response is Y, SHUTUP sends warning messages to terminals. Section 8.4 explains the messages.

8.4 SHUTUP OUTPUT

SHUTUP sends warning messages to terminals. If the system does not support multiuser protection, SHUTUP sends warning messages to all terminals. If the system does support multiuser protection, SHUTUP sends warning messages only to logged-in terminals. On multiuser systems, a terminal set to NOBROADCAST receives the warning messages only during the last five minutes before shutdown.

SHUTUP sends the following warning messages:

```
Please finish up, 15 minutes before shutdown -- KERMIT
```

Three minutes later, SHUTUP sends another message as a result of the example shown in Section 8.3.:

```
Please finish up, 12 minutes before shutdown -- KERMIT
```

The messages include a six-character name that is a DECnet node name (if DECnet is active) or the system name that was selected in SYSGEN (if DECnet is not active). In the previous messages, KERMIT is the six-character name.

SHUTUP continues to issue the warning messages at the interval specified. When three minutes remain before system shutdown, SHUTUP changes the interval between messages to one minute.

If the system supports multiuser protection and the specified delay before disabling logins has expired (measured from the time SHUTUP is started), SHUTUP displays the following message on the invoking terminal:

```
All further logins are disabled
```

After SHUTUP logs off all terminals, aborts all nonprivileged tasks, and dismounts all devices, it displays the following message on the invoking terminal:

```
SHUTUP operation complete
```

SHUTUP then halts the processor.

At this point, you can either boot another system or resume operation of the current system by pressing the Continue switch on the CPU console. After you press the CONT switch (continue), logins are enabled and the following message appears on the invoking terminal:

```
Logins are now enabled
```

To continue operation of the current system, you must mount the devices and follow normal system start-up procedures.

8.5 THE SHUTUP.CMD FILE

The indirect command file LB:[1,2]SHUTUP.CMD is for user-specific functions. SHUTUP submits the file, if present, to the Indirect Command Processor for execution. The SHUTUP task-build file specifies the amount of time (the timeout interval) that SHUTUP.CMD has to execute. The default timeout interval is 120 seconds or 170(8). It is possible to change the timeout interval for SHUTUP.CMD in the task-build file.

On unmapped RSX-11M systems, the task build file is [1,20]SHUBLD.CMD. On mapped RSX-11M systems, the file is [1,24]SHUBLD.CMD.

For RSX-11M-PLUS systems, the task-build file is [1,24]SHUBLD.CMD.

If the timeout interval expires and SHUTUP.CMD is not finished, SHUTUP asks if you want to wait for SHUTUP.CMD to finish. For example:

```
Command (AT.) Timeout - Continue waiting? [Y/N]:
```

If you enter N, SHUTUP continues without waiting any longer. If you enter Y, SHUTUP waits an additional 30 seconds. SHUTUP continues without waiting any longer if SHUTUP.CMD has not finished at the end of the 30 seconds.

You may want SHUTUP.CMD to install ACS, BYE, DMO, ELI, and ACC (these tasks must be installed to enable SHUTUP to execute properly). If DECnet is installed in your system, you may use SHUTUP.CMD to shut down the network.

The following is an example of a SHUTUP.CMD file:

```
;THIS SHUTUP.CMD FILE INSTALLS ELI
;AND ACS IF THEY ARE NOT INSTALLED,
;AND ALSO ACTIVATES A COMMAND FILE TO
;SHUT DOWN THE NETWORK IF THE NETWORK
;IS ACTIVE.
;
.ENABLE SUBSTITUTION

     .IFNINS ...ELI INS $ELI
     .IFNINS ...ACS INS $ACS

     .IFNACT NETACP .GOTO 20
     @LB:[1,2]NETSTOP

.20:
```

8.6 RSX-11M SYSTEM SHUTDOWN

The following sections describe the shut-down procedure for RSX-11M systems. An annotated example of a system shutdown supplements the description.

8.6.1 RSX-11M System Shutdown Procedure

When the delay before system shutdown expires, SHUTUP performs the following functions:

1. Logs off logged-in terminals (multiuser protection systems only).
2. Stops the Console Logger (if active).
3. Redirects the console terminal to the TI: of SHUTUP.
4. If the indirect command file LB:[1,2]SHUTUP.CMD is present, submits it to the Indirect Command Processor for execution.
5. Stops the Queue Manager (if active).
6. Stops the Error Log Task (ERRLOG) (if active).
7. Deallocates checkpoint space and dismounts devices.
8. Halts the processor.

At this point, you can either boot another system or resume operation of the current system.

8.6.2 Example of an RSX-11M System Shutdown

The following is an example of an RSX-11M system shutdown:

```
>RUN $SHUTUP

RSX-11M SHUT DOWN PROGRAM

Enter minutes to wait before shutdown: 10 (RET)
Enter minutes between messages: 3 (RET)
Enter minutes to wait before disabling logins: 5 (RET)
OK to shutdown? [Y/N]: Y (RET)

17-FEB-81 16:35 Please finish up, 10 minutes before shutdown -- QUASAR

17-FEB-81 16:38 Please finish up, 7  minutes before shutdown -- QUASAR

All further logins are disabled

17-FEB-81 16:40 Please finish up, 5  minutes before shutdown -- QUASAR

17-FEB-81 16:42 Please finish up, 3  minutes before shutdown -- QUASAR

17-FEB-81 16:43 Please finish up, 2  minutes before shutdown -- QUASAR

17-FEB-81 16:44 Please finish up, 1  minute  before shutdown -- QUASAR

17-FEB-81 16:45 System is now shutting down -- QUASAR

SET /COLOG=OFF ❶
>
@LB:[1,2]SHUTUP ❷
>INS $ELI
>INS $ACS
>@ <EOF>
>
QUE /STO:QMG ❸
>
 ELI /NOLOG ❹
16:46:01 ERRLOG -- Error Logging stopped
>
 ACS DB0:/BLKS=0, ❺
ACS -- Checkpoint file now inactive
DMO DB0:/DEV ❻
>
DMO -- System disk being dismounted
DMO -- SYSTEM dismounted from DB0:  *** Final dismount initiated ***
16:47:10  *** DB0: -- Dismount complete
>
DMO DS0:/DEV ❻
DMO -- Warning - Tasks installed from DS0:
DMO -- SYSTEM dismounted from DS0:  *** Final dismount initiated ***
16:47:20  *** DS0: -- Dismount complete
>
SHUTUP operation complete ❼
```

To shut down the system in this example, SHUTUP:

❶ Stops the Console Logger.

❷ Invokes the indirect command file LB:[1,2]SHUTUP.CMD.

SHUTUP.CMD installs ELI and ACS. The @<EOF> message indicates the end of the command file SHUTUP.CMD.

❸ Stops the Queue Manager.

❹ Runs the task ELI to terminate the Error Log Task (ERRLOG).

❺ Deallocates checkpoint files on DB0:.

❻ Dismounts the disks on DB0: and DS0:.

❼ Displays a message when SHUTUP has finished executing.

8.7 RSX-11M-PLUS SYSTEM SHUTDOWN

The following sections describe the shut-down procedure for RSX-11M-PLUS systems. An annotated example of a system shutdown supplements the description.

8.7.1 RSX-11M-PLUS System Shutdown Procedure

When the delay before system shutdown expires, SHUTUP performs the following functions:

1. Stops the Console Logger (if active).
2. Redirects the console terminal to the TI: of SHUTUP.
3. If the indirect command file LB:[1,2]SHUTUP.CMD is present, submits it to the Indirect Command Processor for execution.
4. Logs off logged-in terminals (multiuser protection systems only).
5. Stops the Queue Manager (if active).
6. If Resource Accounting is active, stops Resource Accounting.
7. Stops the Error Log Task (ERRLOG) (if active).
8. Checkpoints all read/write commons.
9. Deallocates checkpoint space and dismounts devices.
10. Halts the processor.

At this point, you can either boot another system or resume operation of the current system.

8.7.2 Example of an RSX-M-PLUS System Shutdown

The following is an example of an RSX-11M-PLUS system shutdown:

```
>RUN $SHUTUP

RSX-11M-PLUS SHUT DOWN PROGRAM

Enter minutes to wait before shutdown: 10 RET
Enter minutes between messages: 5 RET
Enter minutes to wait before disabling logins: 2 RET
OK to shutdown? [Y/N]: Y RET

05-MAR-81 16:34 Please finish up, 10 minutes before shutdown -- CASTOR

All further logins are disabled

05-MAR-81 16:39 Please finish up, 5  minutes before shutdown -- CASTOR

05-MAR-81 16:41 Please finish up, 3  minutes before shutdown -- CASTOR

05-MAR-81 16:42 Please finish up, 2  minutes before shutdown -- CASTOR

05-MAR-81 16:43 Please finish up, 1  minute  before shutdown -- CASTOR

05-MAR-81 16:44 System is now shutting down -- CASTOR

@LB:[1,2]SHUTUP ❶
>INS $ELI
>INS $ACS
>@ <EOF>
>
QUE /STO:QMG ❷
>
STOP/ACCOUNTING SHUTUP ❸
>
 ELI /NOLOG ❹
16:50:20: ERRLOG -- Error Logging stopped
>
 ACS DB7:/BLKS=0. ❺
ACS -- Checkpoint file now inactive
DMO DB7:/DEV/LOCK=V ❻
>
DMO -- System disk being dismounted
DMO -- SYSTEM dismounted from DB7:  *** Final Dismount Initiated ***
16:47:19  *** DB7: -- Dismount complete
>
SHUTUP operation complete ❼
```

To shut down the system in this example, SHUTUP:

❶ Invokes the indirect command file LB:[1,2]SHUTUP.CMD.

SHUTUP.CMD installs ELI and ACS. The @<EOF> message indicates the end of the command file SHUTUP.CMD.

❷ Stops the Queue Manager.

❸ Stops Resource Accounting.

❹ Runs the task ELI to terminate the Error Log Task (ERRLOG).

❺ Deallocates checkpoint files on DB7:.

❻ Dismounts the disk on DB7:.

The Virtual option, /LOCK=V, is privileged. When you specify the Virtual option, the DISMOUNT command does not clear volume as valid and does not spin down the disk. (Refer to the RSX-11M/M-PLUS MCR Operations Manual for additional information.)

❼ Displays a message when SHUTUP has finished executing.

CHAPTER 9

USER ENVIRONMENT TEST PACKAGE (UETP)

The User Environment Test Package (UETP) verifies the integrity and operation of a newly generated RSX-11M/M-PLUS system. The program consists of several Indirect command files that verify the presence and operation of devices, test the basic Executive features, and verify the presence of system utilities.

UETP does not replace the hardware diagnostic tests that DIGITAL Field Service performs on your devices nor does it substitute for software testing during software development. UETP only verifies that your system generation has produced a working operating system with the configuration you intended.

9.1 STRUCTURE OF THE UETP

UETP consists of seven modules: five test modules, an initialization module, and a cleanup module. During initialization, the program allows you to:

- Select the number of times you wish to run UETP
- Select which of the five tests you wish to run
- Include or omit extended comments at the start of each test
- Exclude devices from testing

UETP time-stamps the start and end of the output for each test module. If a module cannot run because of any of the restrictions discussed in Sections 9.1.1 and 9.1.2, UETP outputs an error message.

The five test modules are:

- Load Test
- I/O Exerciser Test
- Utilities Test
- MCR Command Test
- Interactive Utilities Test

Each of these test modules is discussed briefly in the following sections.

9.1.1 The Load Test

The Load Test exercises the multitasking and multiprogramming capabilities of your system. The Load Test:

- Determines whether the memory encompassed by the system partition, GEN, is available to the system
- Uses the Send and Receive Executive Directives
- Tests swapping and round-robin scheduling if they are present in your system

On RSX-11M systems, your system must support all of the following features to enable the Load Test to run:

- ASTs
- Memory mapping
- Send/Receive directives
- Alter Priority directive
- Get Partition Parameters directive
- Dynamic checkpointing

UETP determines the presence of these features. It outputs a message to your terminal if the Load Test cannot be run because one or more of these features is not present.

9.1.2 The I/O Exerciser Test

This test runs the I/O Exerciser (IOX), which is a software tool for detecting and diagnosing I/O problems on the disk and tape units in your system's hardware configuration. IOX performs I/O testing on disks, magnetic tapes, cassettes, and DECtapes. IOX exercises disk and tape units that have recording media in place and are not allocated to other users. If units are not mounted, IOX mounts them.

IOX destroys the contents of tapes and nonfile-structured disks during testing. IOX exercises Files-11 disks without destroying their contents, because IOX creates and deletes its own files during testing.

For RSX-11M systems, the I/O Exerciser can only test systems that support ASTs.

9.1.3 The Utilities Test

This test invokes some of the RSX-11M/M-PLUS utilities: CMP, FLX, DMP, CREF, PMD, PRT, DSC, PAT, SLP, and LBR. It verifies their presence and performs a simple test of their functions.

9.1.4 The MCR Command Test

This module tests some of the MCR disk-access commands: ALL, DEA, MOU, DMO, INI, UFD, and VFY and the BAD utility.

9.1.5 The Interactive Utilities Test

This test invokes EDI, EDT, ZAP, and ACNT. You must be present at the terminal from which the UETP is running to exit from the interactive utilities that this test module invokes.

9.2 OPERATING ENVIRONMENT

You run UETP following system generation. To ensure the integrity of the testing, UETP should have unlimited access to all system features and devices. You should therefore run UETP on a stand-alone basis for two reasons: (1) users may change the state of the system, thereby invalidating or modifying the results of UETP and (2) UETP puts a heavy load on the system, which may interfere with other work on the system.

To run the UETP, you need only one terminal. It must be a privileged terminal. It is best to run the program from a hard-copy terminal so that you have a record of any error messages generated during the execution of UETP. If you have a second terminal that is a CRT terminal, you can use it to run the Memory Display of RMD which monitors how the system memory is being allocated while UETP is executing. This is particularly useful during the Load Test.

For RSX-11M systems, the command files that comprise UETP are located in UFD[2,300] on one of the disks in your system as follows:

Disk Type	Volume Label
Big Disks	
RP04/05/06	RSXM32
RM02/03	RSXM32
RK06/07	CLISRC
RL01/RL02	HLPDCL
RK05	HLPSYS

For RSX-11M-PLUS systems, the command files that comprise UETP are located in UFD[2,300] on the disk labeled RSX11MPBL10.

For RSX-11M systems on RL01/RL02 and RK05 kits, you must have completed SYSGEN Phase III to build the nonprivileged tasks that you want UETP to test. These tasks are tested in the Utilities Test, the MCR Command Test, and the Interactive Utilities Test (see Sections 9.1.3 through 9.1.5 for a list of these tasks). You must have also built IOX if you want to run the I/O Exerciser Test.

To run the Utilities Test, you must have a scratch disk or DECtape. To run the MCR Command Test, you must have a scratch disk. To run the DSC in the MCR Command Test module, you must have a floppy disk or DECtape.

The utilities BAD, TKB, and MAC must be present on the system disk or already installed for UETP to run.

9.3 OPERATING PROCEDURES

Before running the UETP, create the necessary operating environment as described in Section 9.2. After you log in on the privileged terminal from which you wish to run UETP do the following:

1. Place scratch media on all DECtape, cassette, floppy, and magnetic tape drives that you want to test. Place initialized Files-11 disk packs on all disk drives that you want to test. The I/O Exerciser Test destroys data on the nonfile-structured disks and tapes; therefore you must use scratch media. IOX does not destroy data on Files-11 hard disks; therefore they need not be scratch disks.

2. Assign your system disk to LB: by typing

```
>ASN ddnn:=LB:
```

ddnn:

Logical device name of your system disk.

3. For RSX-11M systems that do not have a big disk, the system disk and the disk containing the UETP files are not the same. If your system disk has enough free blocks to accommodate all the UETP command files plus 300 to 400 additional blocks for files that UETP creates, then you may want to copy the UETP files to your system disk.

   Assign the disk that contains the UETP files to SY: by typing

```
>ASN ddnn:=SY:
```

ddnn:

Logical device name of the disk that contains the UETP files.

4. Allocate checkpoint space for the system. You need two blocks for each 1K bytes of memory on your system.

```
>ACS SY:/BLKS=512.
```

5. Execute the UETP.CMD command file by typing

```
>@[2,300]UETP
```

9.4 ERROR REPORTING

Tasks that detect errors may output error messages to your terminal. IOX does its own error reporting. Tasks that run during the MCR Command Test, the Utilities Test, and the Interactive Utilities Test output their own error messages.

The DEFINE task, which gets the system configuration information for UETP, and some of the tasks that comprise the Load Test generate error messages. For the most part, these tasks use the following format for error messages:

```
taskname ERROR   nn ccc
```

taskname

Name of the task that caused the error.

nn

Error number specific to that task (defined in the following sections for each task).

ccc

Low byte in octal of the $DSW for Executive Directive errors or F.ERR(R0) for FCS errors. (For more information on the $DSW, see the RSX-11M/M-PLUS Executive Reference Manual. For more information on F.ERR(R0), see IAS/RSX-11 I/O Operations Reference Manual.)

The error numbers for each task identify which kind of macro instruction caused the error. These are either FCS macros or Executive Directive macros.

9.4.1 The System Configuration Module (DEFINE.TSK)

The error messages for the system configuration module, DEFINE.TSK, are as follows:

Error No.	Macro	Explanation
1	OPEN$W	Open to create CONFIG.CMD file failed.
2	OPEN$R	Open file TAILOR.DAT for read (FDB=FDTAIL) failed.
3	READ$	Read of file TAILOR.DAT failed (FDB=FDTAIL).
4	WAIT$	Wait for file TAILOR.DAT (FDB=FDTAIL) failed.
5	CLOSE$	Close of file TAILOR.DAT (FDB=FDTAIL) failed.
6	CLOSE$	Close failed for CONFIG.CMD file.
8	PUT$	Output to CONFIG.CMD file failed.
19	EXIT$S	Exit failed.

9.4.2 The Monitoring Task for QT Load Task (LOGGER.TSK)

The error messages for the monitoring task for QT Load Task, LOGGER.TSK, are as follows:

Error No.	Macro	Explanation
1	OPEN$W	Failure of open for write to RUNLOG.DAT file.
2	OPEN$W	Failure of open for write to ERRLOG.DAT file.

3	PUT$	Failure of write to ERRLOG.DAT or RUNLOG.DAT file.
4	PUT$	Failure of write to ERRLOG.DAT file.
5	PUT$	Failure of write to ERRLOG.DAT or RUNLOG.DAT file.
6	PUT$R	Failure of write to RUNLOG.DAT or ERRLOG.DAT file.
7	WSIG$S	Wait for Significant Event directive failure.
10	RCVD$C	Receive Data directive failure.
11	ALUN$S	Assign LUN directive failure.
12	WTSE$C	Wait for Significant Event directive failure.
13	SETF$S	Set Event Flag directive error.
14	SDAT$S	Send Data Directive failure.
15	CLOSE$	Close failure of RUNLOG.DAT or ERRLOG.DAT file.
16	GTIM$C	Get Time directive failure.
17	CLEF$S	Clear Event Flag directive failure.

9.4.3 The Startup QT Load Task (QTRUN.TSK)

The error messages for the Startup QT Load Task, QTRUN.TSK, are as follows:

Error No.	Macro	Explanation
1	OPEN$W	Open failure on DVLREM.CMD or DVLSET.CMD file.
2	OPEN$R	Open failure on TAILOR.DAT file.
3	READ$	Read failure on TAILOR.DAT file.
4	WAIT$	Wait failure on TAILOR.DAT file.
5	CLOSE$	Close failure on TAILOR.DAT file.
6	CLOSE$	Close failure on DVLSET.CMD file or DVLREM.CMD file.
7	GPRT$C	Get Partition Parameters Executive directive failure.
8	PUT$	Write failure for .CMD files created by QTRUN.
9	RSUM$S	Resume Executive directive failure for DUMMY task.
10	RQST$S	Request Executive directive failure for LOGGER task.
13	WSIG$S	Wait for Significant Event Executive directive failure.
14	RSUM$S	Resume Executive directive failure.
15	RQST$S	Request Executive directive failure.
16	SDAT$S	Send Data Executive directive failure (failure of Send Data to QT task).
17	SDAT$S	Send Data Executive directive failure (failure of Send Data to LOGGER task).
18	RSUM$S	Resume Executive directive failure.
19	EXIT$S	Exit Executive directive failure.
20	GTIM$C	Get Time Executive directive failure.
21	ALTP$C	Alter Priority Executive directive failure (alteration of Indirect command file priority failed).
22	MRKT$C	Mark Time Executive directive failure.

9.4.4 The NUL1 Form of the QT Load Task (NUL1.TSK)

The error reporting format for NUL1.TSK differs from the general error reporting format. When a directive error occurs, the QT Load Task prints

```
TASK QTnn ABORTING
```

where "nn" is the task number that identifies the task that caused the error.

The error number is then displayed as

```
ERROR   xx
```

where "xx" corresponds to one of the error conditions described in the following table.

After displaying the error number, the task displays the $DSW.

Error No.	Macro	Explanation
0	SPND$S	Suspend directive issued by QT Load Task failed.
1	RCVD$C	Receive Data directive failed (receive from CONFIG).
3	ALUN$S	Assign Logical Unit Number directive failure to use TT number received from CONFIG.TSK.
4	MRKT$C	Mark time directive failure.
5	EXIF$S	Exit directive failure.
6	DSCP$S	Disable Checkpointing directive failure.
7	SDAT$S	Send Data directive failure (send to LOGGER before exiting). Insufficient dynamic memory can cause this error.
8	ASTX$S	AST Service Exit directive failure.
12	CLEF$C	Clear Event Flag directive failure.

9.5 SAMPLE UETP CONSOLE DIALOG

The console dialog in Example 9-1 is from a session of running the UETP on an RSX-11M system.

On RSX-11M-PLUS systems, mount all scratch disks using the /FOREIGN switch for the MOUNT command. Mount all tapes.

Example 9-1 Sample UETP Console Dialog for a System Using RL01s

```
>ASN DL:=LB:
>ASN DL1:=SY:
>ACS SY:/BLKS=512.
>
>@[2,300]UETP
>;DISK = DL: UIC = [2,300]
>SET /UIC=[2,300]
>;***** THINKING *****
>; NEEDED UTILITIES WILL BE INSTALLED NOW.
>INS LB:$MAC/CKP=YES
>INS LB:$TKB/CKP=YES
>INS LB:$CMP/CKP=YES
>INS LB:$LBR/CKP=YES
>INS LB:$PIP/CKP=YES
>INS LB:$EDI/CKP=YES
>;
>;       ************************************************************************
>;       *                                                                      *
>;       * RSX-11M V4.0 UETP PROCEDURE STARTED   AT 13:57:34 ON 20-JUL-81  *
>;       *                                                                      *
>;       ************************************************************************
>;
>* ENTER NUMBER OF PASSES FOR UETP [0 R:1-100 D:1]: 1
>;
>* RUN LOAD TEST? [Y/N]: Y
>;
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>* RUN IO EXERCISER? [Y/N]: Y
>;
>* RUN UTILITIES TEST? [Y/N]: Y
>;
>* RUN MCR COMMAND TEST? [Y/N]: Y
>;
>* RUN INTERACTIVE UTILITIES TEST? [Y/N]: Y
>;
>* EXTENDED COMMENTS DESIRED? [Y/N]: Y
>;
>* IF PRESENT, ACTIVATE ERROR LOGGER? [Y/N]: Y
>;
>;
>* DO YOU WISH TO RUN RMDEMO? [Y/N]: Y
>;
>* ENTER TERMINAL NUMBER FOR RMDEMO [0]: 2
>;
>INS LB:$RMD/TASK=T2RMD
>REA T2RMD 1 TT2:
>REA T2RMD 2 TT2:
>RUN T2RMD
>ELI /LOG
ERRLOG -- Error Logging initialized
>;
>* PAUSE INITIALIZE DISKS OR MOUNT TAPES? [Y/N]: Y
>;
...AT. -- Pausing. To continue type 'UNS ...AT.<ESC>'

>ALL MM0:
>ALL DM0:
>MOU DM0:/OVR
>ALL DK0:
>MOU DK0:/OVR
>
>UNS ...AT.

...AT. -- Continuing
>
>MAC @UETPASM
>TKB @UETPBLD
>MAC DEFINE=LB:[11,10]RSXMC/PA:1,SY:[2,300]DEFINE
>TKB DEFINE/PR/CP=DEFINE,LB:[1,54]RSX11M.STB,[1,1]EXELIB/LB
>PIP NUL1.OBJ;*,NUL2.OBJ;*,NUL3.OBJ;*,LOGGER.OBJ;*,QTRUN.OBJ;*/DE/NM
>PIP DUMMY.OBJ;*/DE,DEFINE.OBJ;*,*.DAT;*,ERRLOG.LST;*/NM
>PIP CRTEST.LST;1/DE,CRTEST.OBJ;1/DE,CRTEST.TSK;1/DE/NM
>PIP *.*/PU
>INS DEFINE/TASK=DEFINE
>PIP TAILOR.DAT=DEVM.SAM/AP
>RUN DEFINE
>
>;
>; THESE ARE THE DEVICES INCLUDED IN YOUR SYSTEM
>;
>DEV
IP0: OFFLINE UNLOADED
UD0: OFFLINE
XM0: OFFLINE LOADED
XP0: OFFLINE LOADED
XQ0: OFFLINE LOADED
LP0: OFFLINE LOADED
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
DB0: LOADED TYPE=RP04
DK0: TT0: - PRIVATE MOUNTED LOADED LABEL=TESTA
DL0: PUBLIC MOUNTED LOADED LABEL=RSXM32 TYPE=RL01
DL1: MOUNTED LOADED LABEL=HLPDCL TYPE=RL01
DM0: TT0: - PRIVATE MOUNTED LOADED LABEL=TESTB TYPE=RK06
DM1: LOADED TYPE=RK06
DY0: OFFLINE LOADED TYPE=RX02
DY1: OFFLINE LOADED TYPE=RX02
MM0: TT0: - PRIVATE LOADED
CO0:    TT0:
TT0: [2,300] - LOGGED ON  LOADED
TT1: LOADED
TT2: LOADED
TT3: LOADED
TT4: LOADED
TT5: LOADED
TT6: LOADED
TT7: LOADED
TT10: LOADED
NL0:
TI0:
CL0:    TT0:
LB0:    DL0:
SY0:    DL0:
>;
>; IF YOU DO NOT WISH TO TEST OR DO NOT HAVE MEDIA FOR ANY
>; OF THE ONLINE DEVICES, SPECIFY THEM NOW (ONE AT A TIME)
>; TYPE <CR> WHEN FINISHED.
>;
>* SPECIFY TAPE OR DISK TO BE EXCLUDED [S]: DB
>* SPECIFY TAPE OR DISK TO BE EXCLUDED [S]: DM1
>* SPECIFY TAPE OR DISK TO BE EXCLUDED [S]: (RET)
>;
>;
>;      TO RUN THE MCR, FLX, AND DSC TESTS AT LEAST ONE SCRATCH DISK (OR
>;      DECTAPE) MUST BE SPECIFIED.  ENTER SCRATCH DISK(S), AND TYPE
>;      <CR> WHEN FINISHED.
>;
>* SPECIFY A SCRATCH DISK [S]: DK
>* THE CONTENTS OF DK0 WILL BE LOST !! APPROVE ? [Y/N]: Y
>* SPECIFY A SCRATCH DISK [S]: (RET)
>;
>;      * LOAD TEST STARTED AT 14:27:24  *
>;
>; FOR THE NEXT FEW MINUTES A NUMBER OF COMPUTE-BOUND TASKS WILL BE
>; EXERCISING THE MULTI-TASKING CAPABILITIES OF THE SYSTEM,
>; AND, IF INCLUDED IN THIS SYSTEM, THE ROUND-ROBIN SCHEDULER,
>; THE SHUFFLER AND CHECKPOINTING.  THE ACTION OF THESE LAST THREE
>; FEATURES CAN BE SEEN ON THE RMDEMO SCREEN.  IF THE SYSTEM IS
>; PERFORMING PROPERLY, ALL OF THE 'GEN' PARTITION IS USED AND THE 'QT'
>; TASKS ARE SHARING SYSTEM RESOURCES IN A MANNER CONSISTENT
>; WITH THE FEATURES AVAILABLE.  NOTE THAT CHECKPOINTING IS DISABLED
>; FOR THE SECOND HALF OF THE TEST.
>;
>;      *** THIS TEST RUNS FOR A MAXIMUM OF 6 MINUTES ***
>;
>INS DUMMY/TASK=DUM00/PRI=40.
>INS LOGGER/TASK=LOG00
>INS QTRUN /TASK=QTR00
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>RUN DUM00

...AT. -- DELAYING

...AT. -- Continuing
>RUN QTR00
>
>INS NUL2/TASK=QT0000
>INS NUL2/TASK=QT0100
>INS NUL2/TASK=QT0200
>INS NUL2/TASK=QT0300
>;      >>>>>> THE LOAD TEST IS RUNNING <<<<<<
>REM QT0000
>REM QT0100
>REM QT0200
>REM QT0300
>REM DUM00
>REM LOG00
>REM QTR00
>;
>;      * LOAD TEST COMPLETED AT 14:34:17  *
>;
>;
>;      * IOX TEST STARTED AT 14:34:17  *
>;
>;              **** NOTE ****
>;
>; TEST MEDIA MUST BE LOADED ON ALL TAPE AND DISK DRIVES.  USE SCRATCH TAPES
>; AND FILES-11 FORMAT DISKS.
>;
>; ALL MAGTAPES BEING TESTED ARE OVER-WRITTEN.  THE DISKS ARE
>; TESTED WITH FILE-STRUCTURED I/O WITH NO NEGATIVE EFFECT ON DATA
>; OR STRUCTURE.
>;
>; BEFORE PERFORMING THE FILE-STRUCTURED I/O TEST, ALL UNMOUNTED
>; DISKS ARE MOUNTED AUTOMATICALLY.
>;
>; DEVICES LISTED AS 'OFFLINE' ARE NOT PHYSICALLY PRESENT AND ARE NOT
>; TESTED.  INDIVIDUAL MAG TAPE, DEC TAPE AND RK05 UNITS CAN
>; ERRRONEOUSLY BE SHOWN AS ONLINE WHEN THE CONTROLLER IS ONLINE.
>; THEY ARE TESTED AND THE TEST FAILS AFTER SEVERAL MINUTES
>; OF ERROR MESSAGES.
>;
>;
>; IF ANY DRIVERS ARE UNLOADED, THEY WILL BE LOADED NOW.
>;
>; **********
>; ************************************************************
>INS $IOX/TASK=...SIO
>;
>;
>;  THE SYSTEM I/O EXERCISER WILL RUN FOR 3 MINUTES
>SIO @IOXCN
SIO -- Summary of exerciser activity  at 20-JUL-81 14:34:10
  TIME -- Run: 3 min.   Elapsed: 3 min.   Remaining: 0 min.
  DK0: -- 3,357. R/W requests totaling 3,357. blocks with no errors
  DL0: -- 2,881. R/W requests totaling 2,881. blocks with no errors
  DL1: -- 2,930. R/W requests totaling 2,930. blocks with no errors
  DM0: -- 5,901. R/W requests totaling 5,901. blocks with no errors
  MM0: -- 651. R/W requests totaling 651. records with no errors
  AST's executed: 15,720.
  Idle-loop iterations: 6,798.
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
A C C U M U L A T E D   T O T A L S :
- - - - - - - - - - -   - - - - - - -
  TIME -- Run: 3 min.   Elapsed: 3 min.   Remaining: 0 min.
  DK0: -- 3,351. R/W requests totaling 3,358. blocks with no errors
  DL0: -- 2,882. R/W requests totaling 2,882. blocks with no errors
  DL1: -- 2,931. R/W requests totaling 2,931. blocks with no errors
  DM0: -- 5,902. R/W requests totaling 5,902. blocks with no errors
  MM0: -- 652. R/W requests totaling 652. records with no errors
  AST's executed: 15,725.
  Idle-loop iterations: 6,798.


>REM SIO
>;
>;       * IOX TEST COMPLETED AT 14:34:54  *
>;       *  UTILITIES TEST STARTED AT 14:34:54  *
>;       * CMP VERIFICATION *
>;
>TIM
14:34:55 20-JUL-81
>PIP COMPAR.CMD=SIOE.CMD
>! INVOKE CMP TO COMPARE IDENTICAL FILES
>CMP TI:=COMPAR.CMD,SIOE.CMD
  1)  DL1:[2,300]COMPAR.CMD;1
  2)  DL1:[2,300]SIOE.CMD;1

  No differences found
TI:=COMPAR.CMD,SIOE.CMD
>! NO DIFFERENCES SHOULD HAVE BEEN FOUND
>!
>! INVOKE CMP TO COMPARE DIFFERING FILES
>CMP TI:=FILE1.,FILE2.
**************************************************
  1)  DL1:[2,300]FILE1.;1
    6   .; IF CMP IS FUNCTIONING, THIS LINE FROM FILE1. IS DIFFERENT.
    7   .; THE REST OF THE LINES ARE IDENTICAL.
***************
  2)  DL1:[2,300]FILE2.;1
    6   .; THIS LINE IS THE FILE2. LINE WHICH DIFFERS FROM THE ABOVE.
    7   .; THE REST OF THE LINES ARE IDENTICAL.

    1 differences found
TI:=FILE1.,FILE2.
>! ONE DIFFERENCE SHOULD HAVE BEEN FOUND
>PIP COMPAR.CMD;*/DE
>;
>TIM
14:34:59 20-JUL-81
>;
>;       * CMP VERIFICATION COMPLETE *
>;
>;       * FLX VERIFICATION *
>INS LB:[1,54]FLX/TASK=...FLX
DM0 -- TT0:    Dismounted from DK0:    *** Final dismount initiated ***
14:35:14  *** DK0: -- Dismount complete
>FLX DK0:/ZE/DO
>FLX DK0:/DO=NUL1.MAC/RS,NUL2.MAC,NUL3.MAC
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>FLX DK0:/DO/LI


Directory       DK:[2,300]
20-JUL-81

NUL1.MAC        19.      20-JUL-81  <233>
NUL2.MAC        19.      20-JUL-81  <233>
NUL3.MAC        19.      20-JUL-81  <233>

 Total of 57. blocks in 3. files

>FLX DK0:/ZE/RT
>FLX DK0:/RT=NUL1.MAC,NUL2.MAC,NUL3.MAC
>FLX DK0:/RT/LI


DIRECTORY         DK0:
20-JUL-81


NUL1   .MAC    19. 20-JUL-81
NUL2   .MAC    19. 20-JUL-81
NUL3   .MAC    19. 20-JUL-81
< Unused > 4721.

 4721. Free blocks

 Total of 57. blocks in 3. files

>REM FLX
>;       * FLX VERIFICATION COMPLETE *
>;
>;       * DMP VERIFICATION *
>;
>TIME
14:35:09 20-JUL-81
>INS LB:[1,54]DMP/TASK=...DMP
>DMP TI:=ASRC.XXX/AS/RC
Dump of DL1:[2,300]ASRC.XXX;1 - File ID 55,1,0
                  Record number 01. - Size 16. bytes


000000    D  M  P     W  O  R  K  S  !


*** EOF ***


>REMOVE ...DMP
>TIME
14:35:11 20-JUL-81
>;
>;       * DMP VERIFICATION COMPLETE *
>;
>;       * CREF, PMD AND PRT VERIFICATION *
>;
>;       THIS ROUTINE OPERATES THE FOLLOWING UTILITIES:
>;       MAC,TKB,CREF,PMD,PRT
>;
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>;      IF ALL ARE FUNCTIONAL, AN EXAMPLE OF A MACRO LISTING WITH SYMBOL
>;      AND MACRO CROSS-REFERENCE LISTINGS ATTACHED IS SPOOLED TO
>;      THE LINE PRINTER, A TASK WILL BE RUN AND IT WILL CRASH AND A
>;      POST-MORTEM DUMP LISTING OF THE CRASHED TASK IS SPOOLED.
>;
>TIM
14:35:12 20-JUL-81
>INS LB:[1,54]CRF/TASK=...CRF
UFD SY:[1,4]
>;
>MAC CRTEST,CRTEST/CR=CRTEST
>TKB CRTEST/FM=CRTEST
>INS CRTEST
>;
>;
>RUN CRTEST
>;
14:35:20  Task "CRTEST" terminated
          Odd address or other trap four
          R0=000000
          R1=111111
          R2=022222
          R3=033333
          R4=044444
          R5=055555
          SP=001172
          PC=001236
          PS=170001
          Postmortem Dump will be generated
>REM CRF
>TIM
14:35:20 20-JUL-81
>;
>;      * CREF, PMD AND PRT VERIFICATION COMPLETE *
>;
>;
>;      * DSC VERIFICATION *
>;
>TIM
14:35:21 20-JUL-81
>INS LB:[1,54]DSC
>SET /UIC=[2,300]
>;
>; BECAUSE YOU DO NOT HAVE A FLOPPY DRIVE OR DEC TAPE,
>; AN AUTOMATIC VERIFICATION OF DSC OPERATION IS DEEMED
>; IMPRACTICAL.
>; THE DSC TEST IS BEING SKIPPED.
>;
>REM DSC
>PIP *.TMP;*/DE
PIP -- No such file(s)
SY0:[2,300]*.TMP;*
>;
>;      * LBR VERIFICATION *
>;
>TIME
14:53:52 20-JUL-81
>! Create an object module library.
>LBR OBJLIB/CR:::OBJ
>! Insert three object modules into the library.
>LBR OBJLIB/IN=OBJCT1,OBJCT2,OBJCT3
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>! Insert an existing module into the library (error expected).
>LBR OBJLIB/IN=OBJCT2
LBR -- *FATAL*-Duplicate module name 'OBJ2  ' IN OBJCT2.OBJ;2
>! Replace two existing modules.
>LBR OBJLIB/RP=OBJCT1,OBJCT2
Module 'OBJ1  ' replaced

Module 'OBJ2  ' replaced

>! Display the contents of the object module library.
>LBR OBJLIB/LE/FU

Directory of file OBJLIB.OLB;1
Object module library created by:  LBR V06.00
Last insert occurred 20-JUL-81 AT 14:54:02
MNT entries allocated: 256; AVAILABLE: 253
EPT entries allocated: 512; AVAILABLE: 506
File space available: 21897 WORDS
Recoverable deleted space: 00150 WORDS
** Module:OBJ1    Size:00075  Inserted:20-JUL-81 Ident:ID 1


  ENT$A1  ENT$B1


** Module:OBJ2    Size:00075  Inserted:20-JUL-81 Ident:ID 2


  ENT$A2  ENT$B2


** Module:OBJ3    Size:00075  Inserted:20-JUL-81 Ident:ID 3


  ENT$A3  ENT$B3



>PIP OBJLIB.*;*/DE
>TIME
14:54:05 20-JUL-81
>;      * LBR VERIFICATION COMPLETE *
>;
>;      * PAT VERIFICATION *
>;
>TIME
14:54:06 20-JUL-81
>INSTALL LB:[1,54]PAT/TASK=...PAT
>! Create an updated object file.
>PATCH PATCHD.PAT=OBJCT1,PATCH1
>! Compare the updated object file with one of known quality.
>CMP TI:=PATCHD.PAT,PATCHD.XXX
  1)  DL1:[2,300]PATCHD.PAT;1
  2)  DL1:[2,300]PATCHD.XXX;1

  No differences found
TI:=PATCHD.PAT,PATCHD.XXX
>PIP *.PAT;*/DE
>REMOVE ...PAT
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>TIME
14:54:10 20-JUL-81
>;       * PAT VERIFICATION COMPLETE *
>;
>;       * SLP VERIFICATION *
>;
>TIME
14:54:10 20-JUL-81
>; Invoke the SLP Editor to edit a data file.
>SLP @SLPCMD
>; Verify that the editing was properly performed.
>CMP TI:=OUTPUT.SLP,SLPOUT.XXX
  1)  DL1:[2,300]OUTPUT.SLP;1
  2)  DL1:[2,300]SLPOUT.XXX;1

  No differences found
TI:=OUTPUT.SLP,SLPOUT.XXX
>PIP *.SLP;*/DE
>TIME
14:54:14 20-JUL-81
>;       * SLP VERIFICATION COMPLETE *
>;       *  UTILITIES TEST COMPLETED AT 14:54:14  *
>;       *  MCR COMMANDS TEST STARTED AT 14:54:14  *
>;       * ALL, BAD, INI, MOU, UFD, VFY, DMO AND DEA VERIFICATION *
>;
>; THIS ROUTINE CHECKS THE DISK-ACCESSING MCR COMMANDS ALL, INI, BAD,
>; MOU, UFD, VFY, DMO AND DEA.
>; ALL AND DEA ARE TESTED ONLY FOR A MULTI-USER SYSTEM.
>; THIS ROUTINE REQUIRES EITHER ONE SCRATCH DISK DRIVE OR DECTAPE
>; OR ANSI SUPPORT FOR MAGTAPE.  IT WILL NOT USE A PUBLIC DISK.
>; IF ONLY MAGTAPE IS AVAILABLE, A SUB-SET OF THE TEST IS RUN.
>;
>INS LB:$BAD/TASK=...BAD
>INS LB:$VFY/TASK=...VFY/INC=10000
>DEV DK:
DK0: TT0: - PRIVATE LOADED
>ACS DK0:/BLKS=0.
ACS -- File not in use
>DMO: DK0:/DEV
DMO -- Volume not mounted DK0:
>DEA DK0:
>ALL DK0:
>BAD DK0:
BAD -- DK0: Total bad blocks= 1.
>INI DK0:SCRTCH/BAD=[AUTO]/EXT=4/FPRO=[R,RW,RWE,RWED]/INDX=BEG/VI
** VOLUME INFORMATION FOR DK0: **

Label    = SCRTCH
/BAD     = [AUTO]
/CHA     = []
/EXT     = 4.
/FPRO    = [R,RW,RWE,RWED]
/INDX    = BEG
/INF     = 147.
/LRU     = 3.
/MXF     = 294.
/PRO     = [RWED,RWED,RWED,RWED]
/UIC     = [1,1]
/WIN     = 7.

THEORETICAL MAXIMUM FILES = 2357.
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>INI DK0:SCRTCH/INDX=BEG/INF=20/LRU=3/MXF=100/PRO=[R,RW,RWE,RWED]/VI

** VOLUME INFORMATION FOR DK0: **

LABEL    = SCRTCH
/BAD     = [AUTO]
/CHA     = []
/EXT     = 5.
/FPRO    = [RWED,RWED,RWED,R]
/INDX    = BEG
/INF     = 16.
/LRU     = 3.
/MXF     = 64.
/PRO     = [R,RW,RWE,RWED]
/UIC     = [1,1]
/WIN     = 7.

THEORETICAL MAXIMUM FILES = 2357.

>INI DK0:SCRTCH/UIC=[2,300]/WIN=7/VI/INF=20

** VOLUME INFORMATION FOR DK0: **

LABEL    = SCRTCH
/BAD     = [AUTO]
/CHA     = []
/EXT     = 5.
/FPRO    = [RWED,RWED,RWED,R]
/INDX    = MID
/INF     = 16.
/LRU     = 3.
/MXF     = 294.
/PRO     = [RWED,RWED,RWED,RWED]
/UIC     = [2,300]
/WIN     = 7.

THEORETICAL MAXIMUM FILES = 2357.

>MOU DK0:SCRTCH/VI/OVR
Volume Information
Class:        Files-11
Device:       DK00
Volume label:SCRTCH
Owner:        [2,300]
Protection:   [RWCD,RWCD,RWCD,RWCD]
Default:      [RWED,RWED,RWED,R]
Processor:    F11ACP
>DEV DK:
DK0: TT0: - PRIVATE MOUNTED LOADED LABEL=SCRTCH
>PIP DK0:[0,0]/FU
Directory DK0:[0,0]
20-JUL-81 14:56
INDEXF.SYS;1        (1,1)          24./24.          20-JUL-81 14:56
  [1,1]    [RWED,RWED,RWED,R]
BITMAP.SYS;1        (2,2)          3./3.            20-JUL-81 14:56
  [1,1]    [RWED,RWED,RWED,R]
BADBLK.SYS;1        (3,3)          2./2.            20-JUL-81 14:56
  [1,1]    [RWED,RWED,RWED,R]
000000/DIR;1        (4,4)          1./1.            20-JUL-81 14:56
  [1,1]    [RWED,RWED,RWED,R]
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
CORIMG.SYS;1        (5,5)           0./0.           20-JUL-81 14:56
  [1,1]    [RWED,RWED,RWED,R]

Total of 30./30. blocks in 5. files
>PIP DK0:/FR

DK0: has 4770. blocks free, 30. blocks used out of 4800.
Largest contiguous space = 2383. blocks
289. file headers are free, 5. headers used out of 294.
>VFY DK0:/LO

        0. Lost files were found

>VFY DK0:/LI
Listing of index on DK0:

File ID 000001,000001 INDEXF.SYS;1  Owner [1,1]
File ID 000002,000002 BITMAP.SYS;1  Owner [1,1]
File ID 000003,000003 BADBLK.SYS;1  Owner [1,1]
File ID 000004,000004 000000.DIR;1  Owner [1,1]
File ID 000005,000005 CORIMG.SYS;1  Owner [1,1]
>VFY DK0:/FR

DK0: has 4770. blocks free, 30. blocks used out of 4800.
>VFY DK0:/RC
Read check of files on DK0: - Blocking factor 13.

>UFD DK0:[2,300]/ALLOC=34./PRO=[R,RW,REW,REWD]
>PIP DK0:[2,300]=DK0:[0,0]BADBLK.SYS
>PIP DK0:[2,300]/FU
Directory DK0:[2,300]
20-JUL-81 14:56
BADBLK.SYS;1        (7,1)           2./2.           20-JUL-81 14:56
  [2,300]  [RWED,RWED,RWED,R]
Total of 2./2. blocks in 1. file
>DMO DK0:
DMO -- TT0:    dismounted from DK0:    ***Final dismount initiated***
14:56:32  ***DK0: -- Dismount complete

>DEA DK0:
>DEV DK:
DK0: LOADED
>REM VFY
>REM BAD
>;      * ALL, BAD, INI, MOU, UFD, VFY, DMO AND DEA *
>;             * VERIFICATION COMPLETE *
>;      *  MCR COMMANDS TEST COMPLETED AT 14:58:58  *
>;
>;      *  INTERACTIVE UTILITIES TEST STARTED AT 14:58:59  *
>;
>;      * INTERACTIVE UTILITIES VERIFICATION *
>;
>;      DEPRESS CARRIAGE RETURN TO CONTINUE FROM EACH EDITOR
>;      SUCCESSFULLY INVOKED AND CTRL Z TO EXIT FROM ACNT.
>TIM
14:58:59 20-JUL-81

>EDI QQQQ.XXX
[Creating new file]
Input
TESTING TEST ING
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
>ABO EDI

14:59:20  Task "...EDI" terminated
          Aborted via directive or CLI
          And with pending I/O requests

>;
>PIP QQQQ.XXX;*/DE
>INS LB:[1,54]EDT
>EDT
EDT>
>ABO EDT

14:59:32  Task "...EDT" terminated
          Aborted via directive or CLI
          And with pending I/O requests

>;
>REM EDT
>INS LB:[1,54]ZAP
>ZAP
ZAP>
>ABO ZAP
14:59:42  Task "...ZAP" terminated
          Aborted via directive or CLI
          And with pending I/O requests

>;
>REM ZAP
>INS LB:[1,54]ACNT/TASK=ACNT
>RUN ACNT
 * ACCOUNT FILE MAINTENANCE PROGRAM *
Options are: A - ADD, C - CREATE FILE, D - DELETE, E - EXAMINE,
L - LIST, M - MODIFY, S - SORT, CTRL/Z - EXIT
Enter option: ^Z
OPERATION COMPLETE

>
>REM ACNT
>TIM
15:00:14 20-JUL-81
>;
>;      *  INTERACTIVE UTILITIES TEST COMPLETED AT 15:00:15  *
>;
>; IF ANY ERRORS HAVE BEEN LOGGED BY THE ERROR LOGGER, YOU SHOULD INVESTIGATE
>; FURTHER BY RUNNING THE ERROR LOGGER REPORT GENERATOR (RPT)
>; AFTER THE UETP COMMAND FILE EXITS.
>ELI /SHOW

        Error Logging Status    20-JUL-81  15:00:27

Logging: On     Limiting: On

Log File: SY0:[1,6]LOG.ERR    File ID: DL0: 320,111

Backup File: SY0:[1,6]BACKUP.ERR
```

(continued on next page)

Example 9-1 (Cont.) Sample UETP Console Dialog for a System Using RL01s

```
Device  Hard Error      Soft Error      QIO
Name    Count/Limit     Count/Limit     Count

DB0:    0./5.           0./8.           0.

DK0:    0./5.           0./8.           0.

DL0:    0./5.           0./8.           12292.
DL1:    0./5.           0./8.           6229.

DM0:    0./5.           0./8.           5970.
DM1:    0./5.           0./8.           0.

DY0:    0./5.           0./8.           0.      (Offline)
DY1:    0./5.           0./8.           0.      (Offline)

MM0:    0./5.           0./8.           19.
>ELI /-LOG
ERRLOG -- Error Lossing stopped
>REM ELI
>; RESTORE SYSTEM TO PRE-UETP STATE
>REM MAC
>REM TKB
>REM CMP
>REM LBR
>REM EDI
>;
>;      ************************************************************************
>;      *                                                                      *
>;      * RSX-11M  V4.0 UETP PROCEDURE COMPLETED AT 15:01:05 ON 20-JUL-81 *
>;      *                                                                      *
>;      ************************************************************************
>PIP *.*/PU:4
>REM DEFINE
>ABO T2RMD
15:01:11  Task 'T2RMD ' terminated
          Aborted via directive or CLI

>REM T2RMD
>REMOVE CRTEST
>PIP TAILOR.DAT;*/DE/NM
>PIP DVLSET.CMD;*,DVLREM.CMD;*,REMUTL.CMD;*,CONFIG.CMD;*/DE/NM
>PIP UETPASM.CMD;*,UETPBLD.CMD;*,RUNLOG.DAT;*,ERRLOG.DAT;*,IOXCNF.DAT;*/DE/NM
>PIP CRTEST.OBJ;*,OBJCT1.OBJ;*,OBJCT2.OBJ;*,OBJCT3.OBJ;*,DUMMY.TSK;*/DE/NM
>PIP LOGGER.TSK;*,QTRUN.TSK;*,NUL1.TSK;*,NUL2.TSK;*,NUL3.TSK;*/DE/NM
>PIP CRTEST.LST;*,DEFINE.TSK;*,CRTEST.TSK;*/DE/NM
>SET /UIC=[2,300]
>@ <EOF>
>
```

CHAPTER 10

VIRTUAL MONITOR CONSOLE ROUTINE (VMR)

The Virtual Monitor Console Routine (VMR) programs are privileged system tasks on RSX-11M and RSX-11M-PLUS that allow you to configure an RSX-11M or RSX-11M-PLUS system image file, respectively. VMR allows you to issue Monitor Console Routine (MCR) commands that are directed to the disk image of a system rather than to the current running system. The system image file that you configure using VMR can later be booted.

VMR supports a number of MCR commands described in the RSX-11M/M-PLUS MCR Operations Manual. With the exception of the SAVE command, VMR commands have the same syntax and semantics as their MCR counterparts.

Error messages for VMR are given in Section 10.6.

10.1 STARTING VMR

You can start VMR four ways. The first three methods apply when you have installed VMR and it is ready to be executed. The fourth method installs VMR automatically, executes it, and then removes it on exit.

Method 1:

```
>VMR @indirect
```

The system loads VMR and executes the specified indirect command file. VMR then exits.

Method 2:

```
>VMR
```

The system loads VMR, and VMR responds with the following prompt:

```
ENTER FILENAME:
```

Method 3:

```
>RUN VMR
```

The system loads VMR, and VMR responds with the following prompt:

```
ENTER FILENAME:
```

Method 4:

```
>RUN $VMR
```

The system installs VMR from the pseudo device LB: under the system UFD (normally, [1,50] for unmapped systems and [1,54] for mapped systems) and loads it. VMR then responds with the following prompt:

```
ENTER FILENAME:
```

In all starting methods except Method 1, VMR prompts

```
ENTER FILENAME:
```

At this point, you specify the file that contains the image of the RSX-11M or RSX-11M-PLUS system you want to modify.

VMR expects to find a symbol definition file of the same name as that of the system image file, but with a file type of .STB, on the same device and under the same UFD as those of the system image file (except when you use the /IM switch; see Section 10.2). For example, suppose you enter the following file specification in response to the prompt:

```
RSX11M.SYS
```

A corresponding file, RSX11M.STB, must exist that contains the symbol definitions for the RSX11M.SYS system image.

NOTE

If the system image file you wish to modify is not on the device assigned as SY: and LB:, you must assign both SY: and LB: to the system image file's physical device.

The system image and symbol definition files are in the standard format produced by the RSX-11M/M-PLUS Task Builder.

Upon receiving a valid file specification, VMR opens and reads the symbol definition file. It also opens the system image file and verifies that the file is actually a system image.

On RSX-11M-PLUS, if the system image file you specify does not have Executive data space support, VMR prompts with:

```
VMR>
```

If the system image file you specify does have Executive data space support, VMR loads the data space and prints the following message:

```
Loading Executive data space
Data space loading completed
```

VMR then prompts with:

```
VMR>
```

You can now enter any valid VMR command or terminate VMR by typing CTRL/Z.

If you used Method 1 to start VMR, the first line in the specified indirect command file must contain a file specification for the system image file that is to be modified.

In addition to specifying an indirect command file to start VMR, you can also enter the following any time that VMR prompts for input:

```
@filespecification
```

This directs VMR to extract command input from the specified file until it reaches an end-of-file and then to return to the terminal for further input. This feature of VMR is useful when you want to repeat a series of commands for several generated systems (INSTALL commands, for example).

VMR allows two levels of indirect command files. This means you can invoke a second command file from within the first file. Trying to invoke another file from within the second file results in an error message.

10.2 VMR FILE SPECIFICATIONS

VMR accepts the standard RSX-11M/M-PLUS file specification format:

```
ddnn:[g,m]filename.type;ver[/IM]
```

ddnn:

The device unit on which the volume containing the desired file resides (for example, DM0: or DT1:). The name consists of two ASCII characters, followed by an optional 1- or 2-digit octal unit number. The default device is SY0:.

[g,m]

The UFD in which the file is listed. The variables g and m are octal numbers from 1 to 377 that represent the file owner's group and member number, respectively. The brackets are a mandatory part of the UFD syntax. The default is the UIC under which VMR is running.

filename

The name of the file. An RSX-11M/M-PLUS file name can consist of up to nine alphanumeric characters. The file name and file type are always separated by a period (.). You must specify the file name. No default is applied.

type

The file type, consisting of up to three alphanumeric characters. The file type and version number are always separated by a semicolon (;). The defaults are .SYS for system image files, .TSK for task image files, and .CMD for indirect command files.

ver

The version number in the range of 0 through 77777(8). The default is the highest version in the UFD.

/IM

A switch that, when applied to the system image file specification, overrides the VMR requirement for the symbol definition file. When you use the /IM switch with the file specification, the only command VMR will accept is the SAVE command.

This is the only switch provided with VMR. You use it to write non-RSX-11M/M-PLUS system images to secondary media in bootstrappable format. The switch is also useful when you use an RSX-11M/M-PLUS system as a host system for developing small, memory-resident systems that must be loaded from a serial medium (such as paper tape).

10.3 COMMAND SYNTAX

You do not have to give an entire command name to VMR. VMR accepts three characters and then searches for a space or tab followed by the command parameters or by a RETURN if there are no parameters. Because VMR requires a space or a tab preceding parameters, VMR does not allow embedded spaces in a command name. If you enter a command incorrectly, VMR displays an error message at your terminal.

The following example illustrates the format of the TIME command. Note that the square brackets in the example indicate that the "E" is optional:

```
VMR>TIM[E]RET
```

10.3.1 Keywords

Some commands use keywords that apply to the command argument. A keyword consists of a slash (/), followed by an alphanumeric identification, and, optionally, followed by an equal sign (=) and the value of the keyword, as follows:

```
/keyword=value
```

You can enter keywords in any order. For example, the INSTALL command requires a file name parameter specifying the task to be installed. You can append keywords to the file name. One such keyword, /TASK, specifies the name under which the task is to be installed.

```
VMR>INS TEST.TSK/TASK=SUPER
```

This command line causes the task contained in the file named TEST.TSK to be installed with the name SUPER. Keywords are command-specific and are defined with each command (Section 10.5).

10.3.2 Comments

VMR treats a line of text as a comment if the first character in the line is a semicolon (;). In addition, you can use the exclamation mark (!) to delimit comments within a command line. The first exclamation mark starts the comment and the next exclamation mark or RETURN terminates the comment. VMR ignores all text between the two exclamation marks. For example:

```
VMR>;  THIS LINE IS A COMMENT

VMR>TAS !THIS IS A COMMENT STRING!
```

Comments are especially useful to clarify commands in VMR indirect command files. (Refer to the RSX-11M/M-PLUS MCR Operations Manual for more information on indirect command files.)

10.4 COMMAND SUMMARY

The VMR commands described in this chapter are listed below by category. The detailed descriptions of the VMR commands are given in Section 10.5 in alphabetical order.

Initialization Commands

ASSIGN	Define, delete, or display a global logical device assignment. List current assignments on console terminal.
CONFIGURATION	(RSX-11M-PLUS systems only.) Display system configuration, or change a device's CSR or vector address.
INSTALL	Install a task into the system image.
LOAD	Load a device driver.
REDIRECT	Redirect I/O requests from one physical device to another.
SAVE	Write the system image file onto a secondary medium in bootstrappable format.
SET	Set a variety of system and terminal characteristics. This command has both device characteristics keywords and memory allocation keywords.
TIME	Enter time and/or date into the system image; display the time and date.
UNLOAD	Unload a device driver.

Informational Commands

DEVICES	Display a list of the peripheral devices defined in the system image.
LUNS	Display the list of LUN assignments for an indicated task.

PARTITIONS	Display the list of partitions defined in the system image.
TASKLIST	Display a list of the tasks installed in the system image.

On RSX-11M-PLUS, the display lists tasks installed in primary pool and then tasks installed in secondary pool.

Task Control Commands

ALTER	Alter the priority of a task.
CANCEL	Cancel the time-based initiation requests for a task.
FIX	Fix a task in the system image.
REASSIGN	Change a task's LUN assignment.
REMOVE	Remove a task from the system image.
RUN	Schedule a task's activation. The task may run after one clock tick, after a delay, or in synchronization with the system clock. Periodic rescheduling is optional.
UNFIX	Unfix a task from the system image.

10.5 VMR COMMAND DESCRIPTIONS

The following subheadings are used in the detailed command descriptions in this section. Subheadings that do not apply to a particular command are omitted from the command description.

Command Acronym

This subheading consists of the minimal form of the command name in capital letters. The acronym is at the outside margin of each page.

Command Name

This subheading consists of the full command name in capital letters. The text following the name describes the command's function.

Format

The command line format is given and all parameters are listed.

Parameters

Parameters are listed and described.

Keywords

Keywords are listed and described.

Examples

Examples of the command output are shown.

Notes

A list of special considerations that may assist you in using the command properly appears under this subheading.

ALT

ALTER

Use the ALTER command to change the static priority of an installed task.

Format

ALT[ER] taskname /PRI=priority

Parameters

taskname

The name of the task whose priority is to be altered.

/PRI

The keyword that changes the task's static priority.

priority

The new priority to be assigned to the specified task, from 1 to 250 (decimal).

Example

```
VMR>ALT TEST /PRI=248.
```

Alters the static priority of task TEST to 248(10).

ASN

ASSIGN

Use the ASSIGN command to define, delete, or display logical-device assignments. Logical-device assignments are a way to associate logical names with physical, pseudo, or logical devices. When you assign a logical name to a pseudo or logical device, the system resolves the assignment to the associated physical device.

There are three types of logical-device assignments: global, local, and login. VMR supports the manipulation of global assignments only. Global assignments apply to all tasks in the running system. See the RSX-11M/M-PLUS MCR Operations Manual for information on local and login assignments.

A logical device name has the same syntax as that of a physical device unit. It consists of a 2-character ASCII name (alphabetic) and an optional 1- or 2-digit octal unit number, followed by a colon (:). The 2-character name can be either equivalent to a standard RSX-11M/M-PLUS device name (for example, DK:) or formed by two letters picked at random (for example, ZZ:).

When you install a task or the system executes an ASSIGN LUN directive, the system scans the logical device table before it scans the physical device table in searching for a specified device. As a result, you can define logical device names that are identical to physical device names even though the logical and physical device names do not refer to the same physical device. The system resolves a specified device name to the first matching table entry found. If a logical device name is identical to a physical device name, the logical name has precedence.

Formats

ASN ppnn:=llnn:/GBL

Define a global assignment that associates the logical name llnn: with device ppnn:.

ASN /GBL

Display all global assignments for the system.

ASN =[llnn:]/GBL

Delete the specified global logical device assignment or, if you omit llnn:, delete all global assignments.

Parameters

pp

A physical, logical, or pseudo device name.

nn

A unit number.

ll

A logical device name.

ASN (Cont.)

Examples

```
VMR>ASN DM:=GB0:/GBL
```

Defines the global logical device name GB0: and assigns it to DM:.

```
VMR>ASN /GBL
GB0: DM0:  Global
```

Lists all global assignments.

```
VMR>ASN =/GBL
```

Deletes all global assignments.

CAN

CANCEL

Use the CANCEL command to cancel time-based initiation requests for a task. These requests result from the RUN$ directive or any of the time-synchronized variations of the VMR RUN command.

CANCEL does not affect the execution of the task that may be active in the saved system image. It removes only the time-based schedule requests still in the queue.

Format

CAN[CEL] taskname

Parameter

taskname

The name of the task whose time-based initiation requests are to be canceled.

Example

```
VMR>CAN XKE
```

Cancels all periodic rescheduling and time-based initiation requests for task XKE.

CON

CONFIGURATION

(RSX-11M-PLUS systems only.) Use the CONFIGURATION command to display system configuration information on your terminal, or to change the CSR or vector address of a device.

Format 1

```
CON DIS[PLAY] [keyword][FOR string]
```

Parameters

keyword

One of the following keywords:

CON[TROLLERS]	CPU
UNI[TS]	FUL[L]

FOR string

A parameter that restricts the display to the devices matching the specified string.

Keywords

CON[TROLLERS]

Displays every device controller that is in the current configuration.

UNI[TS]

Displays every device unit that is in the current system.

CPU

Displays the CPUs that are in the current system.

FUL[L]

Displays the CPUs, the controllers, and the device units in the current configuration. If you do not specify any keyword, CON DISPLAY defaults to FULL. The controller's display contains (from left to right) the following information:

- The controller name
- Whether it is on line or off line
- The UNIBUS Run Mask (URM)
- The controller's CSR address
- The controller's starting vector
- The controller's priority

CON (Cont.)

The units display contains (from left to right) the following information:

- The logical device name
- The physical device name (including physical names of both ports for dual-ported devices)
- Whether the device is on line or off line
- A field indicating "driver" is loaded

Examples

Single-Processor Systems

```
VMR>CONFIGURATION DISPLAY CONTROLLERS
RHA     OFL     CPA     CSR=176700      VEC=254 PRI=5
RHB     OFL     CPA     CSR=176300      VEC=150 PRI=5
RHC     OFL     CPA     CSR=172040      VEC=204 PRI=5
RHD     OFL     CPA     CSR=172440      VEC=224 PRI=5
  .
  .
  .
YMA     OFL     CPA     CSR=170500      VEC=300 PRI=5
YZA     OFL     CPA     CSR=160100      VEC=340 PRI=5
VMR>

VMR>CON DIS UNI
DB0:    RHA0:           OFL     DRIVER
DB1:    RHA1:           OFL     DRIVER
DB2:    RHA2:           OFL     DRIVER
DB3:    RHA3:           OFL     DRIVER
  .
  .
  .
NL0:                    OFL     DRIVER
VT0:                    OFL     DRIVER
RD0:                    ONL     DRIVER
VMR>

VMR>CON DIS CPU
CPA

VMR>CON DIS FUL
CPA
RHA     OFL     CPA     CSR=176700      VEC=254 PRI=5
RHB     OFL     CPA     CSR=176300      VEC=150 PRI=5
RHC     OFL     CPA     CSR=172040      VEC=204 PRI=5
RHD     OFL     CPA     CSR=172440      VEC=224 PRI=5
  .
  .
  .
DB0:    RHA0:           OFL     DRIVER
DB1:    RHA1:           OFL     DRIVER
DB2:    RHA2:           OFL     DRIVER
DB3:    RHA3:           OFL     DRIVER
  .
  .
  .
```

CON (Cont.)

```
NL0:                    OFL     DRIVER
VT0:                    OFL     DRIVER
RD0:                    ONL     DRIVER
VMR>
```

Multiprocessor Systems

```
VMR>CONFIGURATION DISPLAY CONTROLLERS FOR RH
RHB     OFL     CPA     CSR=176300      VEC=150 PRI=5
RHC     OFL     CPA     CSR=176700      VEC=254 PRI=5
RHD     OFL     CPA     CSR=172440      VEC=224 PRI=5
RHH     OFL     CPB     CSR=176700      VEC=254 PRI=5
RHM     OFL     CPC     CSR=176700      VEC=254 PRI=5
RHN     OFL     CPC     CSR=172440      VEC=224 PRI=5
RHR     OFL     CPD     CSR=176300      VEC=150 PRI=5
RHS     OFL     CPD     CSR=176700      VEC=254 PRI=5

VMR>CON DIS CPU
CPA
CPB
CPC
CPD

VMR>CON DIS FULL
CPA
CPB
CPC
CPD
  .
  .
  .
BS0:  BSA0:  BSB0:  OFL  DRIVER
BS1:  BSC0:  BSD0:  BSE0:  BSF0:  OFL  DRIVER
  .
  .
  .
DB0:  RHH0:  RHM0:  OFL  DRIVER
  .
  .
  .

VMR>CON DIS FOR YH
YHA     OFL     CPA     CSR=160020      VEC=310 PRI=5
YHB     OFL     CPA     CSR=160040      VEC=320 PRI=5
YHC     OFL     CPA     CSR=160060      VEC=330 PRI=5
VMR>
```

Format 2

```
CON SET controller-name CSR=value

   or

CON SET controller-name VEC=value
```

Parameters

controller-name

The name of the controller for which the CSR or vector address is to be changed.

CON (Cont.)

CSR=value

The new CSR address for the controller.

VEC=value

The new vector address for the controller.

NOTE

You may enter only one value per command line to the VMR CON SET command.

Examples

```
VMR>CON SET YHC CSR=160100
```

Sets the CSR address for the third DH-11 controller to 160100.

```
VMR>CON SET YHC VEC=340
```

Sets the vector address for the third DH-11 controller to 340.

DEV

DEVICES

Use the DEVICES command to display the symbolic names of all device units or, optionally, all of a particular device type defined in the system image. The device names appear in a single vertical column. A second column contains any additional information about each device.

Format

```
DEV[ICES] [dd:]
```

Parameter

dd:

The particular device type to be displayed. If you omit this parameter, the symbolic names of all device units are displayed.

Examples

```
VMR>DEV
LP0:    Loaded
DB0:    Loaded
DB1:    Loaded
DB2:    Loaded
DD0:    Loaded
DD1:    Loaded
DK0:    Loaded
DK1:    Loaded
DK2:    Loaded
DK3:    Loaded
DL0:    Loaded
DL1:    Loaded
DL2:    Loaded
DM0:    Loaded
DM1:    Loaded
DM2:    Loaded
DR0:    Loaded
DR1:    Loaded
DS0:    Offline  Loaded
DS1:    Offline  Loaded
DT0:    Loaded
DT1:    Loaded
DX0:    Loaded
DX1:    Loaded
EM0:    Unloaded
MM0:    Loaded
MM1:    Loaded
CO0:  TT0:
TT0:    Loaded
TT1:    Loaded
TT2:    Loaded
TT3:    Loaded
TT4:    Loaded
  .
  .
  .
```

DEV (Cont.)

```
NL0:
TI0:
CL0:  LP0:
LB0:  DR0:
SY0:  DR0:
VMR>
```

Notes

Note that the pseudo device SY: is not redirected to the booted device until the system is booted.

The following list describes the messages that can appear in the second column. More than one message can appear on the same line.

- On RSX-11M, OFFLINE indicates that, although the system tables contain entries for the device, the system as saved did not contain the related device.
- On RSX-11M-PLUS, OFFLINE indicates that a device has yet to be brought on line by the reconfiguration task. All physical devices will have this attribute.
- LOADED indicates that the driver for the device is loadable and is currently loaded.
- UNLOADED indicates that the driver for the device is loadable and is currently not loaded.
- A device name in the second column is the device to which the corresponding device in the first column has been redirected.
- The absence of either LOADED or UNLOADED in the second column indicates that the driver is permanently resident; it is linked to the Executive.

FIX

FIX-IN-MEMORY

Use the FIX-IN-MEMORY command to load a task or, on RSX-11M-PLUS systems, a region into its partition in the system image. The system can service requests for running the task more quickly because the task is memory-resident and does not have to be loaded from the disk before it can run.

VMR can fix a task in the system image only if the partition in which it is to be fixed is available. (On RSX-11S systems, all installed tasks must be fixed since they cannot be loaded into the running RSX-11S system.)

Fixed tasks remain in memory even after they exit. Therefore, they do not have to be reloaded when a request is made to run them. However, note that reexecuting fixed tasks is not always practical. If you expect to reexecute a fixed task, you should write it such that it initializes all impure data and LUN assignments at run time.

Only an UNFIX or REMOVE command can free the occupied partition.

The following restrictions apply to the fixing of tasks:

- On RSX-11M systems, checkpointable tasks cannot be fixed. On RSX-11M-PLUS systems, checkpointable tasks should not be fixed because fixing a checkpointable task makes the task noncheckpointable.
- On RSX-11M-PLUS systems, tasks whose names are in the form ...xxx and xxx$$$ cannot be fixed because that copy of the task is never executed. (The task is a prototype task.)
- Active tasks cannot be fixed.
- A task must be installed before it can be fixed.

Format

FIX taskname[/keyword]

Parameter

taskname

The task or region that you wish to fix in memory. If you do not specify a keyword, VMR assumes that the task is a single-user task.

/keyword

One of the following keywords (RSX-11M-PLUS systems only):

/DIR
/NSF
/REG
/RON

FIX (Cont.)

Keywords

/DIR

Specifies that the task to be fixed is a directive common.

Directive commons are used to hold some of the Executive directives instead of the directives being kept in executive addressing space. Because the commons contain Executive code, they must be installed and fixed in the system image before the system can be booted.

The commons are installed and fixed in the system image by commands in SYSVMR.CMD as part of the system generation procedure. There are two directive commons used by RSX-11M-PLUS: DIR11M and DR211M. These commons are installed and fixed automatically by SYSVMR.CMD during system generation. DIR11M must be fixed in memory before any other directive commons can be fixed. Other directive commons must be fixed contiguously to DIR11M.

Directive commons are nonshuffleable, that is, the shuffler cannot move them around in memory after they have been fixed.

To eliminate the directive commons, use the REMOVE command.

/NSF

Specifies that the task to be fixed will be nonshuffleable. The shuffler will not be able to move the tasks around in memory after the task has been fixed.

For information about the shuffler, see Chapter 14.

/REG

Specifies that the task to be fixed is a common region.

/RON

Specifies that the task to be fixed is a multiuser task, the read-only portion of which is to be fixed in memory. You must specify the entire task name.

Examples

```
VMR>FIX XKE
```

Fixes task XKE in its partition.

```
VMR>FIX TTCOM/REG/NSF
```

Fixes resident common TTCOM in its partition and specifies that the task will be nonshuffleable.

```
VMR>FIX ...EDI/RON
```

The editor EDI is built as a multiuser task. This command line fixes its read-only portion in its partition.

INS

INSTALL

Use the INSTALL command to make a specified task known in the system image or, on RSX-11M systems, to load a common into a common partition. An installed task is dormant until the Executive receives a request to run it. The optional keywords either provide parameters that were not specified at task-build time or override parameters that were specified.

When a task is installed, VMR creates a Task Control Block (TCB) for the task in a system image table called the System Task Directory (STD). The task parameters contained in the TCB enable the system to run the task when requested to do so by an MCR command, VMR command, or a system directive. A task cannot be run until you install it.

A TCB contains the address of the load-device Unit Control Block (UCB) and the logical block number (LBN) of the task image file. Because task image files are contiguous, the Executive can load the task image in one transfer when an MCR command or system directive requests an installed task. (Loading requires more than one transfer if the task is larger than 32K words.)

When VMR installs a task in a virgin system, it stores the task's LBN in the TCB. When you save a system, the SAVE command places the file identification, rather than the file's LBN, in the TCB. When the system is rebooted, it opens the task file and stores the task's new LBN in its TCB. If a task has been deleted, the system cannot open the task file when the system is rebooted. The system then automatically removes the task's TCB from the STD.

When VMR installs a task in a saved system image, it stores the task's file identification, rather than the LBN, in the TCB to allow a reboot to occur successfully. It is important that a saved system image not retain the physical disk addresses of the installed tasks because the Disk Save and Compress Utility Program (DSC) or the Backup and Restore Utility (BRU) can compress and copy a disk containing a saved system, thus moving the task files to different physical locations. Because the TCB entries contain task file identifications rather than LBNs, the system can function normally when it is rebooted.

Format

```
INS[TALL] filespec[/keyword(s)]
```

Parameters

filespec

A task image file specification in the form:

```
LB:[g,m]filename.type;version
```

The device specification must be LB:, the file type defaults to TSK, and standard defaults apply to the remaining fields.

INS (Cont.)

/keyword(s)

One or more of the following keywords:

/AFF=[urm]
/CKP=option
/CLI=option
/FIX=option
/INC=size
/IOP=option
/PAR=pname
/PMD=option
/PRI=number
/PRO=[system,owner,group,world]
/RON=option
/ROPAR=pname
/SEC=option
/SLV=option
/TASK=taskname
/UIC=[group,member]
/XHR=option

Keywords

/AFF=processor or /AFF=urm(s)

(RSX-11M-PLUS multiprocessor systems only.) The processor or UNIBUS Run Mask (URM) affinity of the task. If the latter, you must specify at least one URM. If you specify more than one URM, you must separate them by commas. The legal processor specifications are:

CPA
CPB
CPC
CPD

The legal URM specifications are:

UBE	UBK	UBP
UBF	UBL	UBR
UBH	UBM	UBS
UBJ	UBN	UBT

When you specify the /AFF keyword, the installed task runs only on the processor to which the specified URMs are connected.

/CKP=option

The checkpointability of the task on systems that support the dynamic allocation of checkpoint space. The options are YES and NO. If /CKP=YES (default for checkpointable tasks), checkpointing is enabled for the task. If /CKP=NO, checkpointing is disabled for the task.

The /CKP keyword overrides the /CP switch specified at task build.

The task image file need not contain allocated checkpoint space. The /CKP=YES keyword instructs the system to allocate checkpoint space for the task, when required, from a checkpoint file.

INS (Cont.)

On RSX-11M systems, if a task was built with the /XH switch, the /CKP keyword is disabled for that task.

Default: /CKP=YES.

/CLI=option

Specifies that the task being installed is a command line interpreter (CLI). The options are YES and NO. You must install a CLI with the YES option before you can set a terminal to that CLI (MCR SET /CLI) or work with the CLI by means of the MCR CLI command.

/FIX=option

(RSX-11S systems only.) Specifies whether memory-resident tasks on RSX-11S will be fixed in memory. The options are YES and NO. When you specify /FIX=YES, the task is both installed and fixed in memory.

/INC=size

(Mapped systems only.) The extension size; that is, the number of additional words of address space to be allocated to a task that runs in a system-controlled partition. The extension size is usually expressed in octal but may also be expressed in decimal by appending a period (.). This keyword overrides the Task Builder's EXTTSK or PAR options.

Tasks such as MAC, TKB, PIP, VFY, and AT. (Indirect) use the space from the end of the task image to the end of their address space as dynamic memory in which to allocate symbol tables and buffers. The /INC keyword defines the number of words in this dynamic region.

Default: the size specified at task-build time or zero (0).

/IOP=option

Specifies that a privileged task can overmap the I/O page. The options are YES and NO. If /IOP=YES, INSTALL assumes that the task will overmap the I/O page (APR 7) and, if the task is larger than 8K words, issues a warning message. If /IOP=NO, INSTALL assumes that the task does not need to use the I/O page. No warning message is issued.

The /IOP keyword complements the Task Builder /IP switch. Specifying /IP informs INSTALL that the task is intentionally overmapping the I/O page.

Default: /IOP=YES.

/PAR=pname

A partition name that overrides the partition specified at task-build time.

On RSX-11M systems, if the partition specified for the task is a user-controlled partition, the size of the task is extended to the size of the partition. Consequently, if the task is built with checkpoint space allocated in the task image file and the partition is extended (thereby extending the size of the task when it is reinstalled), the allocated checkpoint space may no longer be large enough to hold the task.

INS (Cont.)

If the keyword specifies a nonexistent partition, VMR issues a warning message and attempts to install the task in the GEN partition.

Default: the Task Builder uses the name GEN. (See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for an explanation of multiuser system defaults.)

/PMD=option

Specifies whether a Postmortem Dump is required. The options are YES and NO. If /PMD=YES, the system will perform a Postmortem Dump of the task if it aborts the task because of an SST error condition. If /PMD=NO (the default from a task build), the system will not perform a dump following an SST abort. To generate a dump, you must install the Postmortem Dump task PMD... in the system. This keyword overrides the Task Builder's /PM switch.

Default: specified at task-build time.

/PRI=number

The installed priority of the task. The number is the priority at which the task is to be installed. The value range is 1 through 250(10), where 250 is the highest priority. Standard RSX-11M/M-PLUS number conventions apply to this parameter: octal by default, decimal if followed by a period.

Default: PRI=50(10).

/PRO=[system,owner,group,world]

(RSX-11M-PLUS systems only.) The protection mask for a common. The square brackets are required syntax.

A task linked to a common region with improper access cannot be installed.

The user categories are defined as:

System

All system tasks running under a UIC group number less than or equal to 10(8).

Owner

Tasks that run under the same UIC as the file's owner.

Group

Any task that is running under the same UIC group as the file's owner.

World

Any task that does not fit in one of the above categories.

INS (Cont.)

The /PRO keyword establishes the access rights for each category. If an access code is not specified, the access that the code represents is denied. The four access codes are:

- Read
- Write
- Extend
- Delete

Default: /PRO=[RWED,RWED,RWED,RWED].

Note that you cannot specify protection options for a read-only common region or the read-only portion of a multiuser task.

/RON=option

(RSX-11M-PLUS systems only.) Establishes whether a common region is to be read-only. The options are YES and NO. If /RON=YES, the region is installed as read-only. If /RON=NO, the region is installed with read/write access. This keyword overrides the effect of the /PRO keyword.

Default: /RON=NO; the region is installed with read/write access.

/ROPAR=pname

(RSX-11M-PLUS systems only.) The read-only partition into which the read-only portion of a multiuser task is to be installed. The pname is the name of a read-only system partition. This keyword overrides the Task Builder options. If the named partition does not exist, VMR issues a warning message and attempts to install the read-only portion of the task into the partition containing the read/write portion of the task.

Default: specified at task-build time.

/SEC=option

(RSX-11M-PLUS systems only.) Places the Task Control Block (TCB) for the task being installed in secondary pool. The keyword is legal only for tasks whose names are in the form ...xxx or xxx$$$. The options are YES and NO. If /SEC=NO, the TCB for the task is placed in primary pool instead of secondary pool.

Default: the TCB is placed in secondary pool.

/SLV=option

The task's slave status. The options are YES and NO. If the command specifies /SLV=YES, data sent to the installed task is marked with the TI: of the sending task. When the installed task receives the data, the system sets the task's TI: to that associated with the data. This keyword overrides the Task Builder's /SL switch.

If you specify /SLV=NO, the TI: of the task does not change when the task receives data.

INS (Cont.)

On RSX-11M-PLUS, the /SLV keyword cannot be used with tasks whose names are in the form ...xxx. These tasks are prototype tasks and cannot be installed as slaved tasks.

Default: specified at task-build time.

/SYNC=option

(RSX-11M-PLUS multiprocessor systems only.) Synchronizes task use of a common. The options are YES and NO. This keyword permits several tasks to share a common and use the common for synchronization. If /SYNC=YES for two or more tasks, only one of the tasks can execute at one time. If one of the tasks becomes blocked, the others compete to execute.

The /SYNC keyword prevents inconsistent data that occurs as the result of race conditions. For example, if two tasks access a common, the /SYNC keyword ensures that only one task executes at one time. This prevents the second task from reading or writing the common while the first task is executing.

Default: /SYNC=NO.

/TASK=taskname

The name to be used to install the task.

Default: the name specified at task-build time or the first six characters of the file name. (See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for an explanation of multiuser system defaults.)

/UIC=[group,member]

The UIC for the task. The square brackets are required syntax. You can also override the UIC with the RUN command.

The task UIC determines the file-protection class (system, owner, group, world) to which a task belongs and thus directly influences file access. Note that file protection may differ from file to file.

Default: specified at task-build time. This default applies only to tasks started by a RUN$ directive or a RUN command that specifies a time delay before run time. Tasks started by typing the name as a command (for example, PIP and TKB) run under the default UIC of the terminal from which the command was issued.

/XHR=option

(RSX-11M-PLUS systems only.) Specifies that the Executive's copy of the task's header will be in an area external to the system's dynamic storage region (pool).

The options are YES and NO. If /XHR=YES, the task is installed with an external header. If /XHR=NO, the task is installed with a resident header.

When you specify /XHR=YES, the task's header is not kept in pool. Instead, a copy of the header is placed in a reserved area immediately below the task image. The Executive uses the copy of

INS (Cont.)

the header rather than the actual header. When the task is checkpointed, the system writes the task image and the copy of the header into the checkpoint file. The header in the task image is left unchanged.

The /XHR keyword correlates with the /XH option for the Task Builder. The default for /XHR and the ability for a user to override the option specified at task-build time depend on how the INSTALL task was built at system generation. See your system manager for this information.

Examples

```
VMR>INSTALL SCAN
```

Installs task SCAN using the default priority and UIC.

```
VMR>INSTALL DM1:[1,111]SCAN.TSK;4/PRI=103
```

Installs task SCAN, file type .TSK, version 4, from device DM1:, group number 1, member number 111, with a priority of 103(8).

```
VMR>INS MAC/INC=4096./PAR=SYSCTL
```

Installs task MAC in the partition SYSCTL and increases the task size by 4096(10) words.

LOA

LOAD

Use the LOAD command to load a nonresident device driver into the system image. LOAD either links a driver to an already resident data base or makes a loadable data base resident. This command eliminates the need to link infrequently used device drivers permanently into the Executive.

Executive support of loadable device drivers is a system generation option on RSX-11M but is always included on RSX-11M-PLUS.

Format

```
LOA[D] dd:[/keyword(s)]
```

Parameters

dd:

A 2-character ASCII device name.

/keyword(s)

One or two of the following keywords:

/CTB=cc[a,b,...] (RSX-11M-PLUS only)

/PAR=pname

/SIZE=psize

Keywords

/CTB=cc[a,b,...]

(RSX-11M-PLUS systems only.) The cc is the name of the multidriver controller (CTB) that should be used instead of $ddCTB in the loadable data base. [a,b,...] are slots in the CTB's KRB table that should point at the KRBs in the loadable data base. The slots are named A,B,C,D,E,F,H,J,K,L,M,N,P,R,S, and T.

The /CTB keyword is used to load a driver that references a split data base. Refer to the RSX-11M-PLUS Guide to Writing an I/O Driver for more information.

/PAR=pname

The partition into which the driver is to be loaded. The pname is a partition name. The /PAR keyword overrides the partition specified at task-build time.

/SIZE=psize

The minimum size (octal words) of the partition into which the driver is to be loaded. The psize is the partition size. VMR rounds psize up to the next largest unit of 100(8). Psize must be less then 77741.

LOA (Cont.)

Examples

```
VMR>LOAD  MM:
```

Loads the TU16 driver (MMDRV).

```
VMR>LOA  DK:/PAR=DRVPAR
```

Loads the RK05 driver (DKDRV) into DRVPAR.

```
VMR>LOA  DR:/CTB=RHB
```

Loads the RM03 driver (DRDRV), using the RHB multidriver controller.

Notes

1. Guidelines for building user-written loadable device-driver tasks are defined in the RSX-11M and RSX-11M-PLUS Guide to Writing an I/O Driver.

2. The LOAD command requires that a driver's .TSK and .STB files reside in the same UFD and on the same disk as the system image upon which you are working. For example, the following command requires that the files DTDRV.TSK and DTDRV.STB reside on SY: under the current UIC:

   ```
   LOA DT:
   ```

3. The LOAD command requires pool space as follows:

   a. If the driver data base is being loaded, the data base needs contiguous pool space equal to its size.

   b. In a mapped system, each interrupt entry in the driver needs pool space for an Interrupt Control Block (ICB).

   c. If the driver is being loaded into a system-controlled partition, the Partition Control Block (PCB) needs pool space.

LUN

LOGICAL UNIT NUMBERS

Use the LUN command to display the static LUN assignments for a specified task. Static assignments are those recorded in the task's disk image file (for example, those assignments specified at task-build time). The display consists of a list of physical device units in one column and their corresponding LUNs in an adjoining column.

Format

```
LUN[S] taskname
```

Parameter

taskname

The task for which the LUNs are to be displayed.

Example

```
VMR>LUN XKE
SY0:  1
SY0:  2
CL0:  3
TT3:  4
TT3:  5
DK0:  6
TT3:  7
```

The display shows static assignments for LUNs 1 through 7 as recorded in the task image file header. No other LUNs are statically assigned to the task XKE.

PAR

PARTITION DEFINITIONS

Use the PAR command to display a description of each memory partition in the system image. The display consists of six columns that specify:

1. Partition name
2. Partition Control Block address (octal)
3. Partition base address (octal)
4. Partition size (octal)
5. Partition type
6. Name of partition occupant(s) -- task, driver, common, and so forth.

On RSX-11M-PLUS, the symbols accompanying the name have the following meanings:

< > active task

[ ] in memory but inactive

+ + multiuser tasks (read-only sections)

! ! read-write named commons

( ) driver

Format

PAR[TITIONS]

Example (of a saved system)

```
Name   PCB     Base      Size       PAR/TASK Name

VMR>PAR
SYSPAR 035720 00200300 00010000  MAIN
SPLPAR 035654 00210300 00013000  MAIN
DRVPAR 035610 00223300 00140000  MAIN
       035324 00223300 00003500  DRIVER (DB:)
       035260 00227000 00001300  DRIVER (DK:)
       035214 00230300 00003600  DRIVER (DR:)
       035150 00234100 00001600  DRIVER (DS:)
       035104 00235700 00004200  DRIVER (DM:)
       034774 00246400 00001300  DRIVER (DT:)
       034730 00247700 00001600  DRIVER (DX:)
       034664 00251500 00001200  DRIVER (LP:)
       034620 00252700 00035500  RW COM !TTCOM !
       034534 00310400 00020000  DRIVER (TT:)
       034470 00330400 00001400  DRIVER (RD:)
       034424 00332000 00001300  DRIVER (VT:)
       034360 00333300 00003100  DRIVER (CO:)
       034314 00336400 00000100  DRIVER (NL:)
       035040 00336500 00004400  DRIVER (MM:)
```

PAR (Cont.)

```
LDRPAR 035544 00363300 00003000  MAIN
       033774 00363300 00003000  TASK   <LDR...>
TSTPAR 035500 00366300 00040000  MAIN
       036454 00366300 00040000  TASK   <SAVT0 >
SECPOL 035434 00426300 00050000  SEC POOL
GEN    035370 00476300 04701500  MAIN
VMR>
```

REA

REASSIGN

Use the REASSIGN command to reassign a task's logical unit numbers (LUNs) from one physical device unit to another. The reassignments affect only the static assignments recorded in the task's disk image file.

The command does not affect a memory-resident task. Therefore, REASSIGN has no effect on the assignments for a task that is fixed or loaded in memory.

Format

REA[SSIGN] taskname lun new:

Parameters

taskname

The name of the task for which the LUNs are to be reassigned.

lun

The logical unit number.

new:

The new device unit, which can be a physical, logical, or pseudo device.

Examples

VMR>REA LODN 3 TT0:

Reassigns LUN 3 of task LODN to physical device TT0:.

VMR>REA TEST 3 XX:

Reassigns LUN 3 of task TEST to logical device XX:.

RED

REDIRECT

Use the REDIRECT command to redirect all I/O requests from one physical device unit to another physical device unit.

You cannot redirect the following device units:

- The pseudo device TI: (nor can a device be redirected to TI:)
- An attached device

In addition, you cannot redirect a pseudo device to itself, nor can you redirect a terminal to the null device.

You can redirect a logged-in terminal to a device other than a terminal (except for the null device), but it is not generally advisable to do so because terminal I/O would be written to the device and that could destroy the boot block. If you do redirect a terminal to another type of device, you should redirect the terminal to itself before you log out.

Format

```
RED[IRECT] new:=old:
```

Parameters

new:

The new device unit to which requests will be redirected.

old:

The old device unit from which requests will be redirected.

Examples

```
VMR>RED TT3:=TT6:
```

Redirects all I/O requests for device TT6: to device TT3:.

```
VMR>RED TT:=LP:
```

Redirects all I/O requests for device LP0: to device TT0:.

REM

REMOVE

Use the REMOVE command to delete a task name or, on RSX-11M-PLUS systems, a region name from the System Task Directory. Removing a task from the system image makes the task unknown to the system. The action resulting from this command is the complement of INSTALL.

The REMOVE command cannot remove an active task from a saved system. An example of such a task is SAV; an attempt to remove SAV from a saved system results in the error: Task active.

Formats

REM[OVE] name[/REG]

Parameters

name

The name of the task, region, or directive common to be removed.

/REG

(RSX-11M-PLUS systems only.) Specifies that name is the name of a region.

You must specify this keyword when removing the directive commons. To remove the commons, use the following command line:

```
VMR>REM DIR11M/REG
```

Note that when you remove DIR11M, you are removing all of the directive commons in the system.

Example

```
VMR>REM SCAN
```

Removes the task named SCAN from the System Task Directory. The task image file remains unaffected.

Notes

1. If a task or region that is the object of a REMOVE command is fixed, the command automatically unfixes the task from the system image.

2. REMOVE automatically cancels all time-based schedule requests for the specified task.

REM (Cont.)

3. To remove a command line interpreter (CLI) task, you must first eliminate the CPB for the task with the MCR CLI /ELIM command on the running system. Then use the MCR SAVE command to update the system image file, invoke VMR, and remove the CLI. However, it is recommended that you work with CLIs exclusively with MCR on the running system -- this includes eliminating and removing old CLIs, and installing and initializing new ones -- and then save the system.

   An exception to this note is the primary MCR task (MCR...). The CPB for this task can never be eliminated and the task itself can only be removed with VMR.

RUN

RUN

Use the RUN command to initiate the execution of an installed task. This command, which has four formats, can cause a task to be:

- Started one clock tick after system boot
- Started at a time increment from the system image time
- Started at a time increment from clock unit synchronization
- Started at an absolute time of day

All of these options are available with or without rescheduling.

On RSX-11M-PLUS, you cannot run a task using the ...xxx form of the task name, that is, you cannot specify RUN ...xxx. The ...xxx (and xxx$$$) form is used for the prototype copy of the task, which is never executed.

Format 1

```
RUN taskname [/keyword(s)]
```

Runs taskname one clock tick after system boot.

Parameters

taskname

The 1- through 6-character task name of the task that you want the system to run.

/keyword(s)

One or both of the following keywords:

```
/RSI=magu
/UIC=[uic]
```

Keywords

/RSI=magu

The reschedule interval. The format is the same as that for dtime, which is detailed under Format 2. The reschedule interval specifies how often you want the system to rerun the task. Each time the interval of time you have specified lapses, the system makes an initiation request for the task. The default is no rescheduling.

If the magnitude field is equal to 0, then the reschedule interval is set to 1. Specifying /RSI=0H is the same as specifying /RSI=1H.

RUN (Cont.)

/UIC=[uic]

The UIC under which the task is requested. It determines which files the task can access. The UIC is in the format [g,m], where g and m represent the group and member numbers, respectively. Legal group and member numbers range from 1 through 377(8). The square brackets are required syntax.

Default: the UIC established for pseudo device CO:.

Format 2

```
RUN taskname dtime [/keyword(s)]
```

Runs taskname at a time increment from the system image time.

Parameters

taskname

The 1- through 6-character task name.

dtime

The time at which the task is initiated in time units from the current system image time. It has the form:

magu

mag

The magnitude, which is the number of units to be clocked. The magnitude value cannot exceed 24 hours in the specified unit and is decimal by default.

u

Time units specified as follows:

T for ticks. A tick is a clock interrupt. The rate at which interrupts occur depends on the type of clock installed in the system.

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

For a programmable clock, a maximum of 1000 ticks per second is available. (The frequency is selected at system generation.)

T can be any positive value up to a maximum of 15 bits.

S for seconds. S can be any positive value up to a maximum of 15 bits.

M for minutes. M can by any positive value from 0 through 1440(10).

H for hours. H can by any positive value from 0 through 24(10).

RUN (Cont.)

/keyword(s)

One or both of the following keywords:

/RSI=magu
/UIC=[uic]

These keywords are described under Format 1.

Format 3

RUN taskname sync [dtime][/keyword(s)]

Runs taskname at a time increment from clock unit synchronization.

The system determines the task start-time by waiting for the specified time unit (sync) to increment and then waiting for the specified time (dtime) to elapse.

Parameters

taskname

The 1- through 6-character task name.

sync

The synchronization unit. Permissible values are:

H - Synchronize on the next hour
M - Synchronize on the next minute
S - Synchronize on the next second
T - Synchronize on the next tick

dtime

The value of dtime, if present, is added to the synchronization unit to produce the actual run time for running the task. The format for specifying dtime is given in Format 2.

/keyword(s)

One or both of the following keywords:

/RSI=magu
/UIC=[uic]

These keywords are described under Format 1.

Format 4

RUN taskname atime [/keyword(s)]

Runs taskname at an absolute time of day.

RUN (Cont.)

Parameters

taskname

The 1- through 6-character task name.

atime

The absolute time of day when this task will be initiated. Input format is hh:mm:ss, where:

hh - Hours (decimal assumed)
mm - Minutes (decimal assumed)
ss - Seconds (decimal assumed)

/keyword(s)

One or both of the following keywords:

/RSI=magu
/UIC=[uic]

These keywords are described under Format 1.

Examples

```
VMR>RUN XKE 15M
```

Runs task XKE 15 minutes from the current system image time.

```
VMR>RUN XKE 15M/RSI=90S/UIC=[3,1]
```

Runs task XKE 15 minutes from the current system image time, rescheduling it every 90 seconds. The task is running under UIC [3,1].

Note

The correct value to be inserted in the clock queue for Formats 2 through 4 is computed from the current value of the system time stored in the system image. Thus, the system image time must be set before any RUN requests are made or they may not occur at the correct time after the system is booted.

SAV

SAVE

Use the SAVE command to write the image of a system onto a sequential medium in bootstrappable format. With this command, you can copy an RSX-11S or a foreign system onto a tape or a floppy disk. You can then transport the tape or floppy to another computer and bootstrap it, using a hardware bootstrap. (Refer to the description of the /IM switch in Section 10.2.)

On paper tape, system images are created in absolute loader format and must be bootstrapped with the absolute loader.

Format

SAV[E] ddnn:[label][/keyword(s)]

Parameters

ddnn:

The device unit onto which the system image is to be written. Valid devices are as follows:

CT:	Tape cassette
DD:	TU58 DECtape II
DT:	DECtape
DU:	RX50 flexible disk/RA60 disk/RC25 disk (removable)
DX:	RX01 flexible disk
DY:	RX02 flexible disk
MM:	TE/TU16/TU45/TU77 magnetic tape
MS:	TS11/TSV05/TU80 magnetic tape
MT:	TE/TU10/TS03 magnetic tape
PP:	Paper tape

label

The file label name that must be specified for all secondary media except paper tape. It consists of from 1 to 12 alphanumeric characters.

Keywords:

/BOOT=type:

Specifies the type of boot block being used when the system image is saved. The options for type are DX: (for RX01s) and DY: (for RX02s).

The keyword allows you to boot on an RX01 a system that was saved on an RX02 in single-density mode and vice versa.

If you do not use /BOOT, the boot block for the type of device being used is written to the device.

/DENS=1600

Specifies the bits per inch (bpi) density for the following 9-track magnetic tapes: TE16, TU16, TU45, and TU77. These tapes use the device mnemonic MM:

SAV (Cont.)

Example

```
VMR>SAV MT:MYSYS
```

In this example, the system image is written onto MT: in bootstrappable format. System changes made by other VMR commands are also saved with the system image.

Notes

1. When you save RSX-11S systems on paper tape and you do not apply the /IM switch to the system's file name (refer to Section 10.2 for a description of the /IM switch), the last 1000(8) bytes of the system image are not written out. This allows space to accommodate the absolute loader. When you specify the /IM switch with the system's file name, the entire system image is written to paper tape. In this case, it is your responsibility to ensure that the system image is not so large that it overlays the absolute loader.

2. When you save a system on paper tape, contiguous blocks that contain only zero and that begin on a 32-word address are compressed into a 2-word descriptor. This compression significantly reduces both the amount of paper tape required and the time expended to create the tape and to read it into memory. After the entire image has been loaded into memory, a short routine zeroes the appropriate blocks of memory before passing control to the bootstrapped system.

3. When you bootstrap systems on media other than paper tape, regardless of the size of the system image, the last 1000(8) bytes of physical memory are not copied to accommodate the bootstrap.

4. Systems saved on a TE/TU10/TS03 9-track magnetic tape drive can be booted from a TE/TU16 drive and vice versa.

5. When you specify the /IM switch with the system file specification, the SAVE command always uses a bootstrap that does not enable memory management.

6. On RSX-11M, the maximum memory size of a system is 124K words. The SAVE command imposes maximum sizes on saved images of mapped systems for certain devices. If a mapped system is saved and its memory size is greater than the maximum size imposed by the SAVE command for the specified device, only the amount of memory equal to the maximum size (starting at physical location 0) is saved; no diagnostic message is issued.

   The devices for which the SAVE command imposes a maximum size on saved system images are:

Paper tape punch (PP:)	28K words
Cassette tape (CT:)	28K words
RX01 flexible disk (DX:)	123K words
RX02 flexible disk (DY:)	123K words

SAV (Cont.)

7. None of the bootstraps require that a target RSX-11S system have a driver for the bootstrap device.

8. Unless you specify /DENS=1600, when you save a system image to a magnetic tape drive that supports multiple densities, the drive is forced to 800 bpi before the SAVE operation is started.

SET

SET

Use the SET command to alter a number of system-wide characteristics. The following list describes the available options. The SET command allows you to:

- Display the status (determined by the keyword specified) for all terminals
- Establish device characteristics for any device in the system
- Establish the default UIC for any terminal in the system
- Establish the system UIC used by the LOAD command and the install-run-remove option of the MCR RUN command
- Set the status of any terminal in the system
- Create and eliminate partitions
- Add space to the system dynamic memory
- Enable write checking and seek optimization on devices that support it

Improper use of the SET command can result in a situation in which the system will not accept privileged user input. The following situations should be avoided:

- The system does not have any nonslaved, privileged terminals. This is unacceptable because nonprivileged terminals cannot enter privileged commands and slaved terminals cannot enter unsolicited input to MCR. See the description of the /SLAVE keyword.
- The buffer length of any nonslaved, privileged terminal is shorter than 14 bytes on RSX-11M and shorter than 2 bytes on RSX-11M-PLUS. In this case, the terminal buffer is too small to enter many important commands required either to continue system operation or alter the status of other terminals. See the description of the /BUF keyword.

On RSX-11M, a number of SET command options (for example, /CRT, /ESCSEQ, /LOWER, and /WCHK) require Executive features that must be included at system generation. If you specify an option that has not been generated into the system image, the SET command has no effect and results in an error message.

Also, some of the keywords are only supported with the full-duplex terminal driver. The full-duplex terminal driver is a system generation option in RSX-11M systems but is always included in RSX-11M-PLUS systems. Note that if you select the full-duplex driver on RSX-11M, you must also select the get/set multiple characteristics options to set or display any terminal-related characteristics.

VMR permits only one keyword per command line. You can prefix most keywords by NO to negate or disable the function of the keyword. VMR returns a syntax error message when NO precedes a keyword for which it is illegal.

SET (Cont.)

Examples of using the SET command follow the description of the keywords.

Format

SET /keyword=value

Parameter

/keyword=value

One of the keywords listed in Table 10-1. The keywords are described in alphabetical order following the table.

Table 10-1
SET Keywords

Type	Keywords
Device Characteristics	/ABAUD[=ttnn:] /BRO[=ttnn:] /BUF=ddnn:[size] /CRT[=ttnn:] /EBC[=ttnn:] /ECHO[=ttnn:] /ESCSEQ[=ttnn:] /FDX[=ttnn:] /FORMFEED[=ttnn:] /HFILL=ttnn:[value] /HHT[=ttnn:] /LINES=ddnn:[value] /LOWER[=ttnn:] /OPT[=ddnn:opttyp] /PRIV[=ttnn:] /PUB[=ddnn:] /REMOTE[=ttnn:[baud]] /RPA[=ttnn:] /SLAVE[=ttnn:] /SPEED=ttnn:[recv:xmit]

(continued on next page)

SET (Cont.)

Table 10-1 (Cont.)
SET Keywords

Type	Keywords
Device Characteristics (Cont.)	/TERM=ttnn:[type] /TYPEAHEAD[=ttnn:[size]] /UIC[=[uic][:ttnn:]] /VFILL=[ttnn:] /WCHK[=ddnn:] /WRAP[=ttnn:]
Enable/Disable Login	/LOGON
Library or System UICs	/LIBUIC[=[uic]] /NETUIC[=[uic]] /SYSUIC[=[uic]]
Memory Allocation (see notes before examples)	/AFF=[UNIBUS-mask] /BOT=pname:value /MAIN=pname[:base:size:type] /NOMAIN=pname /MAXEXT[=size] /PAR=pname[:base:size[:type]] /NOPAR=pname /POOL[=top] /POOLSIZE[=value] /SECPOL /SUB=mname:pname[:base:size] /NOSUB=mname:pname /TOP=pname:value
Pool Monitor Task	/PLCTL[=[high][:[low][:[frsiz][:basep]]]]
System Tuning	/RNDC[=nn] /RNDH[=nn] /RNDL[=nn] /SWPC[=nn] /SWPR[=nn]

SET (Cont.)

Keywords:

/ABAUD[=ttnn:]

(Full-duplex terminal driver only.) Enables autobaud detection for the remote dial-up line for the specified terminal. The terminal driver samples the line's first input character, tries to determine the incoming baud rate, and sets the interface speed accordingly.

When you omit the terminal specification, VMR displays all the terminals with autobaud detection.

/NOABAUD[=ttnn:]

(Full-duplex terminal driver only.) Disables autobaud detection for the specified terminal. The terminal driver does not attempt to determine the incoming baud rate for the remote dial-up line.

When you omit the terminal specification, VMR displays all the terminals that do not have autobaud detection.

/AFF=[UNIBUS-run-mask,...]

(RSX-11M-PLUS multiprocessor systems only.) Specifies which I/O page will contain the device partition. The keyword is required for creating device partitions on the system.

UNIBUS-run-mask

A 3-letter name of the UNIBUS Run Mask (URM) with which the device partition is associated. There is one URM for each processor, as follows:

CPA
CPB
CPC
CPD

There are also up to 12 URMs for the switched UNIBUS runs, as follows:

UBE	UBK	UBP
UBF	UBL	UBR
UBH	UBM	UBS
UBJ	UBN	UBT

You must specify at least one URM. If you specify more than one URM, you must separate them by commas.

You must enclose the URM specification in brackets.

For example, to create a device partition that is mapped to CPA's I/O page, enter the following command:

```
SET /PAR=DEVP:177600:200:DEV/AFF=[CPA]
```

SET (Cont.)

/BOT=pname:value

Directs VMR to move the bottom boundary of the specified system-controlled partition up or down the amount indicated by the value parameter. The keyword modifies the bottom boundary of a partition even though it has tasks installed in it. If you modify the partition so that it becomes too small to hold the tasks installed in it, VMR displays a warning message, but it will modify the partition anyway.

pname

The 1- to 6-character alphanumeric partition name.

value

The number of 64-byte blocks.

The value parameter can be supplied in five formats: +value, -value, value, +* or *, and -*.

+value

Moves the bottom boundary up by the specified amount. Moving the bottom boundary up makes the partition smaller.

NOTE

You can supply the value in the same units as those of the base and size parameters for the /PAR keyword.

-value

Moves the bottom boundary down by the specified amount. Moving the bottom boundary down makes the partition larger.

value

Establishes the total partition size equal to the specified value by moving the bottom boundary.

+* or *

Moves the bottom boundary up as far as possible. Note that using this format will cause the bottom boundary to move up to the bottom of the first fixed task or fixed driver loaded into the partition at the time the /BOT keyword is executed. If there is nothing loaded into the partition, it assumes a size of one 64-byte block.

-*

Moves the bottom boundary down as far as possible. Using this format will cause the bottom boundary of the partition to move down to the top boundary of the partition below it, or to the top of the Executive pool if the partition is the lowest partition in memory.

SET (Cont.)

/BRO[=ttnn:]

Enables the broadcast option for the specified terminal. The terminal can receive messages sent from other terminals with the MCR BROADCAST command.

When you omit the terminal specification, VMR displays all the terminals that can receive messages.

The /BRO keyword is valid only on systems with multi-CLI support.

/NOBRO[=ttnn:]

Disables the broadcast option for the specified terminal. The terminal cannot receive messages sent from other terminals.

When you omit the terminal specification, VMR displays all the terminals that cannot receive messages.

The /NOBRO keyword is valid only on systems with multi-CLI support.

/BUF=ddnn:[size]

Sets the default buffer size of the specified device. The size must be greater than zero and less than or equal to 255(10) (for a terminal, the buffer size must be greater than or equal to 15 decimal). The /BUF keyword is particularly useful for defining line printer width (80 or 132 columns).

When you omit the size specification, VMR displays the current buffer size of the device.

/CRT[=ttnn:]

Defines the specified terminal as a cathode ray tube (CRT) terminal and enables the backwards deletion option. When the backwards deletion option is enabled, pressing the RUBOUT or DELETE key moves the printing position one space to the left and erases any character displayed in that position.

When you omit the terminal specification, VMR displays all the terminals that are defined as CRTs.

/NOCRT[=ttnn:]

Deletes the CRT status of the specified terminal and disables the backwards deletion option.

When you omit the terminal specification, VMR displays all the terminals that are not defined as CRTs.

/EBC[=ttnn:]

(Full-duplex terminal driver only.) Enables the 8-bit character option. The terminal driver passes all eight bits of the input characters to the specified terminal.

When you omit the terminal specification, VMR displays all the terminals that have the 8-bit character option enabled.

SET (Cont.)

/NOEBC[=ttnn:]

(Full-duplex terminal driver only.) Disables the 8-bit character option. The terminal driver strips off the parity bits of the input characters before passing them to the device.

When you omit the terminal specification, VMR displays all the terminals that do not have the 8-bit character option enabled.

/ECHO[=ttnn:]

Enables the echo option. The terminal driver displays each character that is typed on the specified terminal.

When you omit the terminal specification, VMR displays all the terminals that have the echo option enabled.

/NOECHO[=ttnn:]

Disables the echo option. The terminal driver does not display the characters that are typed on the specified terminal.

When you omit the terminal specification, VMR displays all the terminals that have the echo option disabled.

/ESCSEQ[=ttnn:]

Enables the recognition of escape sequences from the specified terminal. The terminal driver treats the ESC character as the beginning of an escape sequence rather than as a line terminator. (See the RSX-11M/M-PLUS I/O Drivers Reference Manual for further information about escape sequences.)

When you omit the terminal specification, VMR displays all the terminals that recognize escape sequences.

/NOESCSEQ[=ttnn:]

Disables the recognition of escape sequences at the specified terminal.

When you omit the terminal specification, VMR displays all the terminals that do not support the recognition of escape sequences.

/FDX[=ttnn:]

(Full-duplex terminal driver only.) Enables the full-duplex option. The terminal driver accepts input from the specified terminal while simultaneously outputting to it.

When you omit the terminal specification, VMR displays all the terminals that have the full-duplex option enabled.

/NOFDX[=ttnn:]

(Full-duplex terminal driver only.) Disables the full-duplex option. The terminal driver will not process input and output from the specified terminal simultaneously.

When you omit the terminal specification, VMR displays all the terminals that do not have the full-duplex option enabled.

SET (Cont.)

/FORMFEED[=ttnn:]

Indicates that the specified terminal has a hardware form feed.

When you omit the terminal specification, VMR displays all the terminals that have hardware form feeds.

/NOFORMFEED[=ttnn:]

Indicates that the specified terminal will not have a hardware form feed.

When you omit the terminal specification, VMR displays all the terminals on the system that do not have hardware form feeds.

/HFILL=ttnn:[value]

(Full-duplex terminal driver only.) Specifies the number of fill characters (value) that the terminal driver is to place after a carriage return when outputting to the specified terminal. The value is a number from 0 through 7.

When you omit the value parameter, VMR displays the number of fill characters currently being output by the terminal driver to the specified terminal.

/HHT[=ttnn:]

(Full-duplex terminal driver only.) Indicates that the specified terminal has a hardware horizontal tab.

When you omit the terminal specification, VMR displays all the terminals on the system that have a hardware horizontal tab.

/NOHHT[=ttnn:]

(Full-duplex terminal driver only.) Indicates that the specified terminal will not have a hardware horizontal tab.

When you omit the terminal specification, VMR displays all the terminals that do not have a hardware horizontal tab.

/LIBUIC[=[uic]]

(RSX-11M-PLUS systems only.) Directs the INSTALL command to search the specified UIC in response to a RUN $file or INS $file command when the file is not in the system UIC.

The UIC is in the format [g,m], where g and m are octal numbers from 1 through 377(8) that represent a group and member number, respectively. The square brackets are required syntax. The default LIBUIC is [3,54].

When you omit the UIC specification, VMR displays the current library UIC.

SET (Cont.)

/LINES=ddnn:[value]

(Full-duplex terminal driver only.) Establishes the number of lines per page on the specified device. The value is a number from 1 through 255(10). If you specify a number that is larger than the number of lines per page possible for a given device, VMR defaults to the largest number possible for that device.

When you omit the value parameter, VMR displays the lines per page for the specified device.

/LOGON

Enables logging in on terminals in a multiuser protection system. (See /NOLOGON for more information.)

/NOLOGON

Disables logging in on terminals in a multiuser protection system. /LOGON and /NOLOGON clear and set a flag checked by the HELLO command when a user attempts to log in on a terminal. /NOLOGON provides a means of preventing users from accessing the system during such activities as system maintenance or account file maintenance.

/LOWER[=ttnn:]

Specifies that lowercase characters are not to be converted to uppercase when they are received from the specified terminal.

When you omit the terminal specification, VMR displays all the terminals for which lowercase characters are not converted.

/NOLOWER[=ttnn:]

Resets the terminal characteristics of the specified terminal so that lowercase characters received on input are automatically converted to uppercase and echoed as uppercase.

When you omit the terminal specification, VMR displays all the terminals for which conversion to uppercase is specified.

/MAIN=pname[:base:size:type]

(RSX-11M systems only.) Establishes a main partition.

pname

The 1- to 6-character alphanumeric partition name.

base

The physical base address of the partition specified as a number of 64-byte blocks.

size

The size of the partition specified as a number of 64-byte blocks. The maximum size is 0 to 1920.K words for a task, common, or main partition.

SET (Cont.)

type

SYS for a main partition, DEV for a common partition that maps the device registers, COM for a common partition, or TASK for a task partition.

Main partitions are allocated for user tasks and loadable drivers.

Device partitions are allocated for device commons that are used by tasks to access the device registers on the I/O page.

You can enter base and size in any of the following formats:

Format	Calculated Value
nnnn	Octal (nnnn*100)
nnnn.	Decimal (nnnn.*64.)
nnnK	Octal K (nnn*4000)
nnn.K	Decimal K (nnn.*2048.)

where nnnn is a number to which a multiplication factor is applied to determine the value used. Valid ranges of calculated values are:

Octal	0<=base<177777	0<size<2000
Decimal	0.<=base<65535	0.<size<1024.
Octal K	0K<=base<3777K	0K<size<40K
Decimal K	0.K<=base<2047.K	0.K<size<32.K

For system-controlled partitions, size may range from 0 to 1920.K or any valid representation of these limits.

You can specify the base address of a partition through a wildcard (*) for TASK or SYS partitions in a mapped system. When you specify a wildcard (*), VMR creates a partition at the lowest address possible for a partition of the size you have specified.

You can also specify the size of a partition through a wildcard (*) for SYS partitions. When you specify the wildcard (*), VMR creates the largest possible partition starting at the base address you have specified in the base parameter.

When you specify wildcards for both the base and size, VMR will create a partition that will completely fill the first gap in the existing partition layout.

If you do not specify base, size, or type, VMR displays the values of these parameters for the named partition. If the named partition is a main task-controlled partition, VMR also displays the parameters of all its subpartitions.

/NOMAIN=pname

Eliminates a named main partition from the system. The partition you specify is eliminated from the list of partitions and the system. If any tasks or commons are installed in or attached to the partition, the partition is not eliminated and VMR generates an error message.

SET (Cont.)

/MAXEXT[=size]

Establishes the maximum size to which a task can extend itself beyond its mapped array area by means of the EXTEND TASK directive. (See the RSX-11M/M-PLUS Executive Reference Manual for a description of the directive.) The size is given in units of 32(10)-word blocks. The maximum size allowed is 1777(8) for systems without PLAS and 177777(8) for systems with PLAS.

You can specify size in the following five formats:

Specified Size	Resulting Size
n	(n*100) octal
n.	(n*64.) decimal
nK	(n*4000) octal
n.K	(n*2048.) decimal
*	1777(8) maximum for systems without PLAS; 177777(8) maximum for systems with PLAS

When you omit the size specification, VMR displays the current maximum extension size in 100(8)-byte units.

The /MAXEXT keyword is valid only on mapped RSX-11M systems that have selected the EXTEND TASK directive option during system generation and on all RSX-11M-PLUS systems. However, you can use both the unmapped and mapped versions of VMR to establish the maximum extension size for tasks.

/NETUIC[=[uic]]

Specifies the UIC in which all DECnet-related tasks are stored. The keyword applies only to systems that selected the external communication products option during system generation. The UIC is in the format [g,m], where g and m are octal numbers from 1 to 377(8) that represent a group and member number, respectively. The square brackets are required syntax.

When you omit the UIC specification, VMR displays the current network UIC.

/OPT[=ddnn:opttyp]

(RSX-11M-PLUS systems only.) Enables disk I/O queue optimization for the specified device. For more information on optimization and on the algorithm (opttyp) parameter for this keyword, see Chapter 16.

ddnn:

The device that will use I/O queue optimization. Legal devices are DB:-, DL:-, DM:-, and DR:-type devices. When you do not specify a device, VMR displays all the devices that use optimization.

SET (Cont.)

opttyp

The algorithm for optimization. The algorithms are:

NEAR[EST]
ELEV[ATOR]
CSCAN

The default algorithm set at system generation is NEAR.

/NOOPT[=ddnn:]

(RSX-11M-PLUS systems only.) Disables disk I/O queue optimization for the specified device.

When you omit the device specification, VMR displays all the devices that do not use I/O queue optimization.

/PAR=pname[:base:size[:type]]

(RSX-11M-PLUS systems only.) Establishes a partition.

pname

The 1- through 6-character alphanumeric partition name.

base

The physical base address of the partition specified as a number of 64-byte blocks or as a wildcard (*).

size

The size of the partition, specified as a number of 64-byte blocks or as a wildcard (*). The maximum size is 0 to 1920.K words for a task, common, or main partition.

type

SYS for a main partition, SECPOL for a secondary pool partition, DEV for a common partition that maps the device registers, or CPU for CPU partitions.

Main partitions are allocated for user tasks and loadable drivers.

Pool partitions are allocated for secondary pool.

Device partitions are allocated for device commons that tasks use to access the device registers on the I/O page. Note that creating a device partition automatically creates a main partition named IO PAR, which spans the entire I/O page. The partition you specify with the /PAR keyword then becomes a subpartition of IO PAR. On multiprocessor systems, you must use the /AFF (affinity) keyword to specify which I/O page contains the device partition.

CPU partitions are allocated on multiprocessor systems to store vectors for all the devices associated with the particular processor.

Partition type is an optional parameter. If you do not specify it, SYS is the default.

SET (Cont.)

You can enter base and size in any of the following formats:

Format	Calculated Value
nnnn	Octal (nnnn*100)
nnnn.	Decimal (nnnn.*64.)
nnnK	Octal K (nnn*4000)
nnn.K	Decimal K (nnn.*2048.)

where nnnn is a number to which a multiplication factor is applied to determine the value used.

For SYS partitions, size can range from 0 to 1920.K or any valid representation of these limits. For example, the following inputs for size allocate a 2048(10)-byte partition:

```
40
32.
1K
1.K
```

You can also specify the base address of a partition with a wildcard (*). When you specify a wildcard (*), VMR creates a partition at the lowest address possible for a partition of the size you have specified.

You can also specify the size of a partition with a wildcard (*). When you specify the wildcard (*), VMR creates the largest possible partition starting at the base address you have specified in the base parameter. When you specify a wildcard for a CPU partition, VMR creates a partition of size (n-1)*4K, where n is the number of processors specified during SYSGEN.

When you specify wildcards for both the base and size, VMR creates a partition that completely fills the first gap in the existing partition layout.

If you do not specify base, size, or type, VMR displays the values of these parameters for the named partition. If the named partition is a main partition, VMR also displays the parameters of all its subpartitions.

When you are establishing base and size for a CPU partition, note that the entire CPU partition must exist below 124K.

/NOPAR=pname

(RSX-11M-PLUS systems only.) Eliminates a named partition from the system. The partition you specify is eliminated from the list of partitions and the system. If any tasks or commons are installed in or attached to the partition, the partition is not eliminated and VMR generates an error message.

/PLCTL[=high][:[low][:[frsiz][:basep]]]

(Systems with pool monitoring support only.) Sets the pool limit parameters used by the Pool Monitor Task (PMT). The following list defines the parameters:

high

The high pool limit in bytes.

SET (Cont.)

low

The low pool limit in bytes.

frsiz

The minimum byte size of the largest free pool block required for avoiding low pool actions by the Pool Monitor Task (PMT...).

basep

The base task priority, which is the lowest priority a nonprivileged task can have and still be eligible for memory contention during times of low pool; this priority remains in effect until pool conditions improve. (Note that only those tasks requested to run while the system is in a low pool state are affected).

The following list defines the limits of the parameters:

84. <=low <=high <=total size of system pool

84. <=frsiz <=high

0. <=basep <=250.

The following list gives the defaults for the parameters:

high 1600. bytes

low 600. bytes

frsiz 200. bytes

basep 51.

See the RSX-11/M-PLUS System Management Guide for information on determining the values for the parameters.

When you omit the parameter specifications, VMR displays the current parameters for the system.

/POOL[=top]

Increases the size of the dynamic storage region (pool).

top

The first location in memory used for partition allocation. You specify top in units of 64-byte blocks, in octal or decimal.

When you do not specify top, VMR displays the virtual address of the top of the Executive, the size of the longest block of pool space in words, the total number of words in the pool, and, on RSX-11M-PLUS, the lowest physical address at which a partition can start. (You can use the address when calculating partition layouts for systems supporting kernel data space. If your system is not a kernel data space system, this value is always equal to the virtual address of the top of pool.) The display's format is:

```
POOL=top:max:total[:par]
```

SET (Cont.)

where top is given in units of 64-byte blocks, max and total are in decimal words, and par is the lowest physical address (32-word blocks).

You can also specify top by using a wildcard (*). The wildcard directs VMR to supply the maximum amount of pool possible. Specify the wildcard as follows:

```
/POOL=*
```

If you request more pool than can be supplied, VMR prints a warning message and supplies the maximum amount of pool possible.

/POOLSIZE[=value]

Specifies a total amount of pool rather than a top address.

If you do not specify a value, VMR displays the virtual address of the top of the Executive, the size of the longest block of pool space in words, the total number of words in the pool, and the lowest physical address at which a partition can start (32-word blocks).

/PRIV[=ttnn:]

Sets the specified terminal to privileged status.

When you omit the terminal specification, VMR displays all the privileged terminals in the system.

/NOPRIV[=ttnn:]

Sets the specified terminal to nonprivileged status.

When you omit the terminal specification, VMR displays all the nonprivileged terminals in the system.

/PUB[=ddnn:]

Establishes the specified device as a public device.

When you omit the device specification, VMR displays all the public devices in the system.

/NOPUB[=ddnn:]

Causes the specified device to lose its public status.

When you omit the device specification, VMR displays all the nonpublic devices in the system.

/REMOTE[=ttnn:[speed]]

Declares that the specified terminal is connected to a modem and can be connected to the system by means of a dial-up network. The terminal has a line to a DH11, DHV11, DL11, DLV11, DZ11, or DZV11 multiplexer.

Specifying speed establishes the initial baud rate (answer speed) of the remote dial-up line for the specified terminal. You may specify an answer speed only for DH11s, DHV11s, DZ11s, and

SET (Cont.)

DZV11s, which are variable-speed multiplexers. (You cannot specify an answer speed for DL11s or DLV11s.)

Valid speeds for the DH11 are 0, 50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, EXTA (user-specified baud rate A), and EXTB (user-specified baud rate B).

Valid speeds for the DHV11 are 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, and 19200.

Valid speeds for the DZ11 and DZV11 are 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, and 9600.

When you omit the terminal specification, VMR displays all the remote terminals. When you omit the speed specification, the current setting for the terminal remains in effect.

/NOREMOTE[=ttnn:]

Establishes the terminal's line as a local line not connected to a modem.

When you omit the terminal specification, VMR displays all the local terminals.

/RNDC[=nn]

- Defines the length of the Executive round-robin scheduling interval in ticks. The parameter nn can be an octal value, or decimal value if followed by a period (.). The minimum value for nn is 0. See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for more information.

When you omit the length specification, VMR displays the current value for the interval.

/RNDH[=nn]

- Defines the highest priority (1 through 250 decimal) that will be considered for Executive round-robin scheduling. The priority class must be higher than the one specified with the /RNDL keyword. See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for more information.

When you omit the priority class, VMR displays the current value for the priority.

/RNDL[=nn]

- Defines the lowest priority (1 through 250 decimal) that will be considered for Executive round-robin scheduling. The priority class must be lower than the one specified with the /RNDH keyword. See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for more information.

When you omit the priority class, VMR displays the current value for the priority.

/RPA[=ttnn:]

(Full-duplex terminal driver only.) Enables the read-pass-all option. The terminal driver passes all characters input at the keyboard to the terminal's input buffer.

SET (Cont.)

When you omit the terminal specification, VMR displays all the terminals on the system that have the read-pass-all option enabled.

/NORPA[=ttnn:]

(Full-duplex terminal driver only.) Disables the read-pass-all option. The terminal driver does not pass special function characters (for example, CTRL/C) input at the keyboard to the terminal's input buffer.

When you omit the terminal specification, VMR displays all the terminals on the system that have the read-pass-all option disabled.

/SECPOL

(RSX-11-M-PLUS systems only.) Displays secondary pool use in the system. The display is in the following format:

SECPOL=secfr:secsiz:pctfr

secfr

The number of free blocks in secondary pool in units of 32-word (decimal) blocks.

secsiz

The size of secondary pool in units of 32-word (decimal) blocks.

pctfr

The percentage of free blocks in secondary pool.

/SLAVE[=ttnn:]

Establishes the specified terminal as one that can enter data only if it is solicited by a task. Thus, the specified terminal always rejects unsolicited input (other than CTRL/O, CTRL/Q, and CTRL/S).

When you omit the terminal specification, VMR displays all the terminals currently classified as slaved.

/NOSLAVE[=ttnn:]

Sets the specified terminal to nonslaved status.

When you omit the terminal specification, VMR displays all the terminals currently classified as nonslaved.

/SPEED=ttnn:[recv:xmit]

Establishes the receive and transmit baud rate for terminals attached to the system through a DH11, DHV11, DZ11, or DZV11 variable-speed multiplexer. The recv argument is the baud rate at which characters are input to the computer from the terminal. The xmit argument is the baud rate at which characters are output to the terminal from the computer. You must specify both arguments when setting the speed. If you do not specify either argument, VMR will display the current settings.

SET (Cont.)

Valid baud rates for the DH11 are 0, 50, 75, 110, 134.5, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, EXTA (user-specified baud rate A), and EXTB (user-specified baud rate B).

Valid baud rates for the DHV11 are 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, and 19200.

Valid baud rates for the DZ11 and DZV11 are 50, 75, 110, 134.5 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, and 9600.

If you have a DZ11 or DZV11, the receive and transmit baud rates must be the same because these multiplexers do not support split speeds.

/SUB=mname:sname[:base:size]

(RSX-11M systems only.) Establishes a subpartition. Subpartitions can only be established in user-controlled main partitions.

mname
: The 1- through 6-character main partition name. The subpartition being defined becomes a subpartition in the named main partition.

sname
: The 1- through 6-character subpartition name. The subpartition being defined becomes a subpartition of the main partition specified in mname.

base
: The physical base address of the subpartition, specified as a number of 64-byte blocks.

size
: The size of the subpartition, specified as a number of 64-byte blocks.

Base and size may be entered in any of the following formats; nnnn is a number to which the system applies a multiplication factor to determine the value used.

Format	Calculated Value
n	Octal (nnnn*100)
n.	Decimal (nnnn.*64.)
nK	Octal K (nnn*4000)
n.K	Decimal K (nnn.*2048.)

Valid ranges for calculated values are:

Octal	0<=base<10000	0<size<2000
Decimal	0.<=base<4096.	0.<size<1024.
Octal K	0K<=base<200K	0K<size<40K
Decimal K	0.K<=base<2047.K	0.K<size<32.K

Example:

Any of the following inputs for size will allocate a 2048(10)-byte subpartition.

SET (Cont.)

```
        40
        32.
        1K
        1.K

    If base and size are omitted, VMR displays the current values for
    the named subpartition.

/NOSUB=mname:sname

    Eliminates the specified subpartition from the main partition.
```

/SWPC[=nn]

Defines the number of clock ticks for a single Executive swapping interval. The parameter nn is in the range 0 through 45,568(10) and can be an octal value, or decimal if followed by a period (.). See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for more information.

When you omit the clock ticks specification, VMR displays the current value for the interval.

/SWPR[=nn]

Defines a priority range for Executive swapping. The parameter nn is in the range 0 through 127(10) and can be an octal value, or decimal if followed by a period (.). The highest number is the highest priority.

The value for the /SWPR keyword affects the installed priority of all tasks. Each task is initiated at its installed priority plus the value for nn (priority + nn). While the task is executing, its priority is decremented until it becomes the installed priority minus the value for nn (priority - nn). When the priority reaches this lower limit, the task is swapped out so that another task with a higher priority can execute. The priority for the new task changes in the same way. See the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for more information.

When you omit the range specification, VMR displays the current value for the priority range.

/SYSUIC[=[uic]]

Establishes the UIC for the system and all system tasks. The UIC has the format [g,m], where g and m are octal numbers that represent a group and member number, respectively. On most unmapped systems, the system UIC is [1,50]; on most mapped systems, the system UIC is [1,54]. These UIC values are the initial defaults on virgin systems.

The UIC specified with the /SYSUIC keyword also becomes the default UIC used by the LOAD command and the install-run-remove option of the MCR RUN command.

When you omit the UIC specification, VMR displays the current system UIC.

SET (Cont.)

/TERM=ttnn:[type]

(Full-duplex terminal driver only.) Establishes the terminal type of the specified terminal. When VMR sets the terminal type, it automatically sets the HFILL, VFILL, CRT, FORMFEED, and HHT characteristics for the terminal. The standard terminal types are as follows:

ASR33	LA30S	LA120	VT55	VT105
ASR35	LA34	LA180S	VT61	VT125
KSR33	LA36	VT05B	VT100	VT131
LA12	LA38	VT50	VT101	VT132
LA30P	LA100	VT52	VT102	

If you specify a terminal type that is unknown to VMR, VMR does not set any device characteristics for the terminal. However, VMR stores the value you have specified in the Unit Control Block (UCB).

When you omit the type parameter, VMR displays the terminal type of the specified terminal.

VMR also accepts an alternate form of the command. You can specify the terminal type and equate it to the desired terminal. This automatically associates certain characteristics of the particular terminal type with the specified terminal. (See the RSX-11M/M-PLUS I/O Drivers Reference Manual for a list of the implicit characteristics for each terminal type.) The command is in the format:

```
SET /term-type=TTnn:
```

where term-type is one of the standard terminal types. Note that the terminal types are mutually exclusive.

The half-duplex terminal driver also accepts the /term-type form of the command. However, the only legal terminal types for the half-duplex driver are /ASR33, /LA30S, and /VT05B. The command associates implicit characteristics of the terminal type with the specified terminal, but, unlike the full-duplex terminal driver, the types are not mutually exclusive.

The following list describes the effects of the keywords for the half-duplex terminal driver:

- /ASR33 — Converts lowercase letters to uppercase before they are output to the terminal.
- /LA30S — Indicates that fill characters are used after a carriage return. The number of fill characters used depends on the current carriage position at the time the carriage return occurs.
- /VT05B — Sets a vertical file of three nulls after a line feed on 60 Hz systems and of four nulls on 50 Hz systems.

SET (Cont.)

/TOP=pname:value

Directs VMR to move the top boundary of the specified partition up or down the amount indicated by the value parameter. The keyword modifies the top boundary of a system-controlled partition even if the partition has tasks installed in it. If you modify the partition so that it becomes too small to hold the tasks installed in it, VMR displays a warning message but modifies the partition anyway.

pname

The 1- through 6-character alphanumeric partition name.

value

The number of 64-byte blocks.

The value parameter can be supplied in five formats: +value, -value, value, +* or *, and -*.

+value

Moves the top boundary up by the specified amount. Moving the top boundary up makes the partition larger.

NOTE

You can supply the value in the same units as the base and size parameters for the /PAR keyword.

-value

Moves the top boundary down by the specified amount. Moving the top boundary down makes the partition smaller.

value

Establishes the total partition size equal to the specified value by moving the top boundary.

+* or *

Moves the top boundary up as far as possible. This format causes the top boundary to move to the bottom boundary of the partition above it or to the top of memory if the partition is the top partition in memory.

-*

Moves the top boundary down as far as possible. This format causes the top boundary to move down to the top of the first fixed task, common, or driver in the partition at the time the /TOP keyword is executed. If nothing is loaded into the partition, the partition assumes a size of one 64-byte block.

SET (Cont.)

/TYPEAHEAD[=ttnn:[size]]

(Full-duplex terminal driver only.) Enables the typeahead option for the specified terminal. The terminal driver stores input characters in a buffer before passing them to a requesting attached task. Thus, characters input during periods between requests are not lost.

If size is specified (RSX-11M-PLUS systems that support kernel data space only), the typeahead buffer size is set accordingly. The value for size can be from 0 through 255(10). If you specify 0 or 1, the terminal will not have the typeahead option enabled (same effect as /NOTYPEAHEAD). The default for size is 86(10).

When you omit the terminal specification, VMR displays all the terminals that have the typeahead option enabled.

/NOTYPEAHEAD[=ttnn:]

(Full-duplex terminal driver only.) Disables the typeahead option for the specified terminal. The terminal driver does not store characters to prevent their loss because the size of the buffer is forced to 1.

When you omit the terminal specification, VMR displays all the terminals that do not have the typeahead option enabled.

/UIC[=[uic]:ttnn:]

Establishes the specified UIC as the default UIC for the specified terminal (the default is TI:). All tasks run from the terminal are run with the UIC of the terminal, except in a multiuser protection system. In such a system, a privileged user can override the terminal UIC with the /UIC keyword of the RUN command. Furthermore, external MCR function tasks are requested with this UIC. These tasks are the MOUNT, INSTALL, DMO, and UFD commands (which run as tasks), and all system-supplied software.

When you specify only the terminal, VMR displays the UIC for that terminal.

/VFILL[=ttnn:]

(Full-duplex terminal driver only.) Enables the vertical fill characters option for the specified terminal. The terminal driver adds four fill characters following each line feed.

When you omit the terminal specification, VMR displays all the terminals on the system that have the VFILL option enabled.

/NOVFILL[=ttnn:]

(Full-duplex terminal driver only.) Disables the vertical fill characters option for the specified terminal. The terminal driver does not add any vertical fill characters following line feeds.

When you omit the terminal specification, VMR displays all the terminals on the system that do not have the VFILL option enabled.

SET (Cont.)

/WCHK[=ddnn:]

Specifies for all Files-11 devices, except DX:, DY:, and magnetic tapes, that all write operations are to be followed by a write check. The checks assure the reliability of data transfers to the specified disk.

On RSX-11M, Executive support for write checking is a system generation option and must be selected for this command to have any effect. On RSX-11M-PLUS, support for write checking is always included.

When you omit the device specification, VMR displays all Files-11 disk drives with write checking enabled.

/NOWCHK[=ddnn:]

Specifies that write checking is to be disabled for the specified device.

When you omit the device specification, VMR displays all disks that support write checking but have the option disabled.

/WRAP[=ttnn:]

(Full-duplex terminal driver only.) Enables the wrap-around option for the specified terminal. The terminal driver automatically generates a carriage return/line feed sequence when the number of input characters exceeds the buffer size of the specified terminal. Both the carriage return and line feed are transparent and do not appear in the input buffer.

When you omit the terminal specification, VMR displays all the terminals on the system that have the wrap-around option enabled.

/NOWRAP[=ttnn:]

(Full-duplex terminal driver only.) Disables the wrap-around option for the specified terminal. The terminal driver does not accept input characters beyond the input buffer size for the terminal.

When you omit the terminal specification, VMR displays all the terminals on the system that have the wrap-around option disabled.

Notes on Memory Allocation Keywords

1. If a command attempts to eliminate a partition in which tasks are installed or drivers loaded, VMR rejects the command until the tasks have been removed.

2. When you define a partition, the name must not already be defined as a partition. In addition, a partition cannot overlap any other partition.

   On RSX-11M, subpartitions can overlap other subpartitions.

3. The address specified with the /POOL keyword must be lower than the base address of the first partition in the system.

   On RSX-11M, an exception is the partition LDRPAR for a resident loader. LDRPAR is in the Executive.

SET (Cont.)

4. Once you have allocated space to the dynamic storage region (pool), you cannot recover the space for use in partitions.

5. When the highest memory addresses of a system are allocated, the size of the partition is modified automatically to reflect the amount of memory present when the system is bootstrapped.

6. On RSX-11M-PLUS, SET /TOP and SET /BOT work correctly for secondary pool partitions.

7. If the last partition is system-controlled, its top address is modified by the SAVE command at boot time to reflect the actual top of physical memory of the hardware in use.

Examples

```
VMR>SET /ABAUD=TT4:
```

Enables the autobaud detection option for TT4:.

```
VMR>SET /ABAUD
ABAUD=TT2:
ABAUD=TT4:
ABAUD=TT5:
```

Displays all the terminals that have the autobaud detection option enabled.

```
VMR>SET /BRO=TT5:
```

Enables the broadcast option for TT5:.

```
VMR>SET /BRO
BRO=TT2:
BRO=TT4:
BRO=TT5:
```

Displays all the terminals that have the broadcast option enabled.

```
VMR>SET /BUF=LP0:
BUF=LP0:132.
```

Displays the current buffer size of LP0:.

```
VMR>SET /BUF=TT1:40.
```

Sets the buffer size of TT1: to 40(10).

```
VMR>SET /CRT=TT3:
```

Enables the backwards deletion option on the display terminal whose number is TT3:.

```
VMR>SET /CRT
CRT=TT3:
CRT=TT7:
CRT=TT12:
```

Displays all the terminals that have the backwards deletion option enabled.

SET (Cont.)

```
VMR>SET /ESCSEQ=TT3:
```

Establishes terminal TT3: as a terminal that can send and receive escape sequences.

```
VMR>SET /ESCSEQ
ESCSEQ=TT3:
ESCSEQ=TT7:
ESCSEQ=TT16:
```

Displays all the terminals that can send and receive escape sequences.

```
VMR>SET /NOESCSEQ=TT3:
```

Disables support of escape sequences for TT3:.

```
VMR>SET /LA30S=TT2:
VMR>SET /LA30S
LA30S=TT0:
LA30S=TT2:
```

Sets TT2: as an LA30S, then displays all LA30S terminals.

```
VMR>SET /NETUIC
NETUIC=[[20,3]]
```

Displays the current network UIC.

```
VMR>SET /NETUIC=[25,6]
```

Sets the network UIC to [25,6].

```
VMR>SET /OPT=DM1:NEAR
```

Enables I/O queue optimization for DM1:, using the NEAR algorithm.

```
VMR>SET /OPT
OPT=DB0:NEAR
OPT=DB1:NEAR
OPT=DM0:NEAR
OPT=DM1:NEAR
```

Displays all the disk devices that have I/O queue optimization enabled.

```
VMR>SET /PLCTL
PLCTL=1600.:600.:200.:51.
```

Displays the current pool limit parameters used by the Pool Monitor Task.

```
VMR>SET /PRIV=TT0:
```

Sets TT0: to be a privileged terminal.

```
VMR>SET /PRIV
PRIV=TT0:
PRIV=TT1:
```

Displays all privileged terminals.

SET (Cont.)

```
VMR>SET /REMOTE=TT21:
```

Sets the terminal line corresponding to TT21: as a remote dial-in line.

```
VMR>SET /REMOTE
REMOTE=TT1:
REMOTE=TT2:
REMOTE=TT3:
REMOTE=TT4:
REMOTE=TT21:
```

Displays all the terminals that are currently set as remote dial-in lines.

```
VMR>SET /RNDC
RNDC=6.
```

Displays the current length of the round-robin scheduling interval.

```
VMR>SET /RNDC=10.
```

Sets the length of the round-robin scheduling interval to 10(10) ticks.

```
VMR>SET/SECPOL
SECPOL=285.:640:44%
```

Displays the amount of secondary pool currently being used.

```
VMR>SET /SLAVE=TT3:
```

Sets TT3: to be a slaved terminal.

```
VMR>SET /SWPR
SWPR=5.
```

Displays the current value for the Executive swapping priority range.

```
VMR>SET /SWPR=10.
```

Sets the Executive swapping priority range to 10(10).

```
VMR>SET /VT05B
VT05B=TT4:
VT05B=TT5:
VT05B=TT6:
```

Displays all VT05B terminals.

```
VMR>SET /WCHK=DK1:
```

Enables write checking on DK1:.

```
VMR>SET /PAR=SYSPAR:420:140:SYS
```

Defines a partition called SYSPAR whose base address is 42000(8) and whose length is 14000(8) bytes.

SET (Cont.)

```
VMR>SET /NOPAR=SYSPAR

Eliminates the main partition SYSPAR.

VMR>SET /PAR=GEN:*:*

Creates a partition named GEN, setting the base address as low as
possible and making the partition as large as possible.
```

```
VMR>SET /POOL=420
```

Establishes the top of the dynamic storage region (pool). Following this command, no user-defined partition can begin below 42000(8).

```
VMR>SET /POOL
POOL=1200:11470.:11738.:2003
```

Shows that the last virtual address in the Executive is 120000, the longest free block is 11470(10) words, and the total of all pool space is 11738(10) words. The lowest physical address at which a partition can start is 2003 (32-word blocks).

TAS

TASKLIST

Use the TASKLIST command to display a description of each installed task in the system. The display contains several columns that give, from left to right, the following information:

1. Task name
2. Task version identification
3. Task Control Block address
4. Partition name
5. Task priority
6. Size of task in bytes (octal)
7. Load device identification
8. Disk address logical block number (in octal) on virgin systems, or the image or task file identification on saved systems
9. Task memory state (saved systems only)

On RSX-11M-PLUS, the display lists the tasks installed in primary pool and then those installed in secondary pool. The tasks in secondary pool are indicated by a plus sign (+).

Format

TAS[KLIST][taskname]

Parameters

taskname

The name of a single installed task, the description of which you would like to be displayed.

Examples

This example is from a virgin system:

```
VMR>TAS
TKTN    03.14   035300 GEN     248. 00010000 LB0:-00631420
...MCR  01.01   035174 GEN     160. 00040000 LB0:-00630300
MCR...  3.3     035070 GEN     160. 00010000 LB0:-00627721
...INS  4       035614 GEN     100. 00040000 LB0:-00627333
...LOA  03.3    035510 GEN      50. 00040000 LB0:-00627643
...UNL  03.4    035404 GEN      50. 00040000 LB0:-00070512
VMR>
```

On a system image generated for a PDP-11/70 with more than 124K words of memory, the size field (sixth column) contains 8-digit rather than 6-digit numbers.

TAS (Cont.)

This example is from a saved system:

```
VMR>TAS
LDR... 08.03    034210 LDRPAR  248. 00003000 LB0:- FILE ID:(1111,12) FIXED
TKTN   03.14    115424 GEN     248. 00010000 LB0:- FILE ID:(31540,27)
MTAACP 0010     115214 GEN     200. 00013500 LB0:- FILE ID:(31461,24)
MCR... 3.3      116464 SYSPAR  160. 00010000 LB0:- FILE ID:(31454,12)
...DCL 0113A    115634 GEN     160. 00040000 LB0:- FILE ID:(31344,53)
DCL... 0113A    115530 GEN     160. 00010000 LB0:- FILE ID:(31345,5)
...MCR 01.01    036770 GEN     160. 00040000 LB0:- FILE ID:(30546,34)
...MOU 23.00    036250 GEN     160. 00040000 LB0:- FILE ID:(5424,15)
F11ACP M0320    117674 GEN     149. 00065700 LB0:- FILE ID:(30101,47)
DB00F1 M0320    117570 GEN     149. 00065700 LB0:- FILE ID:(31370,6)
COT... X00.01   117360 GEN     145. 00015400 LB0:- FILE ID:(31336,120)
...DMO 23.00    117254 GEN     140. 00040000 LB0:- FILE ID:(31346,23)
...INI 20.01    116674 GEN     140. 00040000 LB0:- FILE ID:(31424,55)
...UFD V0410    116044 GEN     140. 00040000 LB0:- FILE ID:(31541,34)
PMD... 3.1      114630 GEN     140. 00023600 LB0:- FILE ID:(31464,70)
HRC... 01.01    036560 GEN     140. 00050400 LB0:- FILE ID:(31240,21)
SHF... 03.01    115320 SYSPAR  105. 00010000 LB0:- FILE ID:(31521,40)
FXR... 01       117104 GEN     100. 00003100 LB0:- FILE ID:(31371,113)
...INS 4        116570 GEN     100. 00027700 LB0:- FILE ID:(31435,12)
...SAV 04.35    116254 TSTPAR  100. 00040000 LB0:- FILE ID:(31475,5)
SAVT0  04.35    036130 TSTPAR  100. 00040000 LB0:- FILE ID:(31475,5)
...PRV 4.30     036664 GEN     100. 00040000 LB0:- FILE ID:(3433,163)
...AT. 04.25    037254 GEN      65. 00057700 LB0:- FILE ID:(31421,131)
...BOO 04.08    117464 GEN      50. 00040000 LB0:- FILE ID:(31323,6)
...ACS 01.1     115740 GEN      50. 00040000 LB0:- FILE ID:(31321,73)
...HEL 01.25    037360 GEN      50. 00044100 LB0:- FILE ID:(31416,15)
VMR>
```

The display for a saved system differs from the display for a virgin system in that the file-ID of the task file, rather than the logical block number, is displayed.

FIXED indicates that the task is fixed in memory.

This example is from an RSX-11M-PLUS saved system that has tasks installed in secondary pool:

```
VMR>TAS
     .
     .
     .
BAP0   02       113440  GEN     80. 00045700 LB0:- FILE ID:(4332,27)
QMG... 1.7      112450  GEN     75. 00031400 LB0:- FILE ID:(4366,2)
LP0    1.9      112734  GEN     70. 00015400 LB0:- FILE ID:(4354,135)
LP1    1.9      112604  GEN     70. 00015400 LB0:- FILE ID:(4354,135)
SHC... 01.01    110600  GEN     60. 00051700 LB0:- FILE ID:(4360,2)
SHUTUP 02       110320  GEN     50. 00011200 LB0:- FILE ID:(4320,65)
...DMO 23.20    006025+ GEN    160. 00015500 LB0:- FILE ID:(4167,35)
...MCR 2.4      006037+ GEN    160. 00032000 LB0:- FILE ID:(2132,227)
...MOU 2502     006041+ GEN    160. 00040000 LB0:- FILE ID:(4205,13)
...DCL 00       006130+ GEN    160. 00035600 LB0:- FILE ID:(2535,1525)
                  .
                  .
                  .
```

TIM

TIME

Use the TIME command to set the time of day and the date or to display the time and date.

Formats

```
TIM[E] [hrs:mins[:secs]] [m1/day/year]

     or

TIM[E] [hrs:mins[:secs]] [day-m2-year]
```

Parameters

hrs

Hours (range 0 through 23).

mins

Minutes (range 0 through 59).

secs

Seconds (range 0 through 59). This parameter is optional; the default is zero.

m1

Numeric representation of month (range 1 through 12).

m2

Three-letter abbreviation for month.

day

Day (range 1 through 31).

year

Year (range 0 through 99). Year is relative to 1900; thus, 1983 is entered as 83.

Examples

```
>TIM
10:23:31 03-FEB-83
```

Displays the current system image time and date.

```
>TIM 14:30 4/11/83
```

Sets the time to 14:30:00 and the date to 11-APR-83.

TIM (Cont.)

Notes

1. If you specify neither time nor date, VMR displays the current system image time and date.

2. If you specify the time and date, VMR sets the system image clock and calendar. If you specify only the time, VMR sets only the clock; if you specify only the date, VMR sets only the calendar.

3. You can specify the time and date in any order.

4. The form used to specify the date does not affect the display format. The date is always displayed as day-month-year.

5. All numeric values are decimal. No terminating period is required.

UNF

UNFIX

Use the UNFIX command to free a fixed task from the virtual memory of the system image. UNFIX is the complement of the FIX command.

If a fixed task exits or aborts, it still occupies the physical memory in the partition.

Format

```
UNF[IX] taskname[/keyword]
```

Parameters

taskname

The task that you want to unfix from memory.

/keyword

One of the following keywords (RSX-11M-PLUS systems only):

/REG
/RON

Keywords

/REG

Specifies that the task to be unfixed is a common region.

/RON

Specifies that the task is a multiuser task, the read-only segment of which is to be unfixed from memory.

Example

```
VMR>UNFIX XKE
```

Unfixes task XKE, freeing the partition in which it resides.

UNL

UNLOAD

Use the UNLOAD command to remove a loadable device driver from the system image. If a device is attached, its driver cannot be unloaded.

To unload a device driver from the system image, the driver's symbol definition file must reside on LB: under the same UFD as the UFD of the system image file. By convention, the system UIC is [1,50] for an unmapped system and [1,54] for a mapped system.

The UNLOAD command cannot remove a loadable data base from the system image, even if the data base was loaded by means of the LOAD command.

Format

UNL[OAD] dd:

Parameter

dd:

A 2-character ASCII device name

Example

VMR>UNL LP:

Unloads the line printer driver (LPDRV).

10.6 VMR ERROR MESSAGES

The following are the VMR command error messages. Although just the text of the messages is given here, the messages actually appear in one of the following formats:

VMR -- *DIAG* message

VMR -- message

VMR -- *FATAL* message

A diagnostic message usually indicates that something unexpected may have happened, but it does not interfere with VMR's operation.

An unmarked message usually indicates that VMR rejected the command line. If the offending line is in an indirect command file, VMR ignores the rejected line and attempts to execute the remaining command lines.

A fatal message also indicates that VMR rejected the command line. In this case, however, VMR either exits or resets itself to the beginning (that is, ENTER FILENAME:). If the offending line is in an indirect command file, VMR aborts without attempting to execute the remaining command lines.

VMR -- Access to common block denied

Explanation: You attempted to install a task that has specified illegal access to a shared region.

VMR -- Addressing extensions not supported

Explanation: You attempted to install a task that was built using the VSECT (virtual section) directive into a system image that does not support the VSECT feature.

VMR -- Alignment error

Explanation: The base address or size of the partition being created with the SET command conflicts with existing partitions or physical memory.

VMR -- Base address must be on a 4K boundary

Explanation: The virtual base address of the task being installed is not aligned on a 4K boundary.

VMR -- Base mismatch common block <commonname>

Explanation: The base address of the partition does not match that of the common block.

VMR -- Cannot fix an I/D space task

Explanation: You attempted to fix a task that is built with I- and D-space. Such tasks cannot be fixed.

VMR -- Cannot fix a task in a system controlled partition

Explanation: This is a VMR restriction. To fix a task in a system-controlled partition, use the MCR FIX command and then save the system image.

VMR -- Cannot have multiple pool or CPU partitions

Explanation: You attempted to create a secondary pool partition in a system that already has one. There can be only one secondary pool partition in a system.

VMR -- Cannot install tasks or commons from other than LB:

Explanation: SAVE does not allow saving of a system that has tasks and/or commons that were not installed from the library device LB:. Therefore, a task or common cannot be installed from other than LB:.

VMR -- Cannot LOAD/UNLOAD a pseudo device

Explanation: When loading or unloading a device driver, the physical name of the driver must be specified.

VMR -- Cannot UNFIX a mapped common region

Explanation: You attempted to unfix a common region that is currently mapped by a task.

VMR -- Cannot UNFIX the directive common

Explanation: RSX-11M-PLUS directive commons can be removed but not unfixed. Because directive commons are merged with DIR11M as they are fixed, the size of a particular common is unkown.

VMR -- Checkpoint area too small

Explanation: The area allocated for checkpointing in the task image file is smaller than the partition into which the task is being installed. The system image does not support the dynamic allocation of checkpoint space.

VMR -- Checkpoint space too small, using checkpoint file

Explanation: This is a warning message. The checkpoint space allocated in the task image file is too small to hold the task (usually because of the /INC keyword). The system image supports the dynamic allocation of checkpoint space. It will use the checkpoint file to store the task when it rolls it out to disk. Since no checkpoint file exists when the system is booted, you should consider this message to be a fatal error if it is issued by any of the system tasks while the system is being booted. These tasks are MCR, MOU, and F11ACP.

VMR -- Circular redirect error

Explanation: The attempt to redirect a device failed because it would result in a circular device list.

VMR -- Command I/O error

Explanation: The system detected an I/O error during a read from an indirect command file.

VMR -- Command syntax error <command-line>

Explanation: The syntax of the command line is incorrect.

VMR -- Common block currently installed

Explanation: You attempted to install a common block that has already been installed.

VMR -- Common block is task partition <commonname>

Explanation: A task's request for access to a common block has been rejected because the requested partition is a task partition.

VMR -- Common block not loaded <commonname>

Explanation: The common block specified by VMR in the error message has been linked to the task, but the common block has not yet been installed.

User Action: The common block must be installed before the task can be installed.

VMR -- Common block occupied

Explanation: You attempted to install a task in a common block that was already occupied.

VMR -- Common block parameter mismatch <commonname>

Explanation: The parameters of the common block named in the error message do not match the parameters in the label block of the task that references the common block.

VMR -- Common block segment count error

Explanation: You attempted to install a resident library with overlays, but the number of overlay segments in the task header differs from the number specified in the segment descriptors.

VMR -- CTB <name> does not exist

Explanation: The Controller Table name that you specified with the LOAD /CTB keyword does not exist in the resident data base.

VMR -- CTB name <name> is a duplicate

Explanation: The loadable data base contains a Controller Table (CTB) with a name that is the same as a CTB name in the resident data base. CTB names must be unique.

VMR -- CTB <name> not supported by driver -- not loaded

Explanation: This is a warning message. Two things could be at fault:

1. The Driver Dispatch Table in the driver does not have all the Controller Table mnemonics that the rest of the data base (that is, DCBs and CTBs) implies that it should have.

2. The CTB is defective. A defective CTB may contain a wrong name or it may point at the wrong DCB.

VMR -- Data space file exceeds available space in system image

Explanation: The RSX11M.SYS files are not large enough to contain the data space file DSP11M.SYS.

VMR -- DCB table for CTB <name> is full

Explanation: LOAD attempted to write the address of the Device Control Block (DCB) of the loadable data base into the DCB table of the Controller Table for a multicontroller device. There were no null entries in the DCB table. (See the RSX-11M-PLUS Guide to Writing an I/O Driver.)

VMR -- Device <ddnn:> is attached

Explanation: You attempted to unload a driver that has one or more of its device units attached. The first attached unit is ddnn:. You cannot unload a driver that has device units attached.

VMR -- Device <ddnn:> is not in system

Explanation: This message can occur for the following reasons:

1. You attempted to save an image on a device that does not exist on the host system.

2. You attempted to redirect I/O requests to a device that does not exist on the host system.

3. You specified a device in the LOAD command line for which there is no data base. This is caused by one of two conditions:

   a. The device does not exist in the system device tables.

   b. You did not declare the device to be loadable and therefore LOAD cannot find a loadable data base for it.

4. You specified a device in the command line that has not been defined in the system image.

5. You specified a device in the DEVICES command line that does not exist in the system image.

VMR -- Device not mounted

Explanation: You did not mount the device that contains the system image file.

VMR -- Device not redirectable

Explanation: You attempted to redirect a device that is marked (in its UCB) as not redirectable.

VMR -- Device not terminal

Explanation: You attempted to set terminal characteristics for a device that is not a terminal.

VMR -- Device not variable speed multiplexer

Explanation: You attempted to set the baud rate for a terminal that is not attached to a DH11, DHV11, DZ11, or DZV11 multiplexer.

VMR -- Directive common already loaded

Explanation: You attempted to fix in the RSX-11M-PLUS system image a directive common that is already fixed.

VMR -- Directive common not contiguous to DIR11M

Explanation: Directive commons must be loaded contiguous to DIR11M to work.

VMR -- Directive common(s) not fixed in memory

Explanation: Not all of the directive commons that were installed in the RSX-11M-PLUS system image have been fixed.

VMR -- Driver already resident

Explanation: You attempted to load a device driver that is already loaded or that is permanently resident in the system image.

VMR -- Driver built with wrong STB file

Explanation: The symbol definition file for the driver you attempted to load is not compatible with the Executive symbol definition file for the current system image. This means that the driver has been built for another system and must be rebuilt before you can load it into the current system.

Explanation: The symbol definition file for the driver you attempted to unload is not compatible with the symbol definition file of the current Executive.

VMR -- Driver cannot be unloaded

Explanation: You attempted to unload a permanently resident driver (that is, a driver linked permanently to the Executive).

VMR -- Driver dispatch table is inconsistent

Explanation: You attempted to load a driver whose driver dispatch table is illegally formatted. See the RSX-11M or RSX-11M-PLUS Guide to Writing an I/O Driver.

VMR -- Driver not loaded

Explanation: You specified a driver in the UNLOAD command line that is not resident in memory.

VMR -- Driver requires running system for LOAd/UNLoad

Explanation: The driver you requested has the symbols $ddLOA or $ddUNL in its source. Support of driver load/unload calls can only be provided by the MCR LOAD/UNLOAD commands.

VMR -- Executive common partition eliminated

Explanation: This is a warning message. One of the executive common partitions has been removed. A system generated with executive common support must have both partitions installed to run.

VMR -- Executive common(s) not installed

Explanation: This is a warning message. For a system generated with executive common support, you should install one (or both) of the commons before you exit VMR. Otherwise, the system is unusable.

VMR -- Executive too large

Explanation: The Executive is too large to be accommodated in the system image. This message indicates that the Executive is larger than 20K. The system is unusable.

VMR -- External headers not supported

Explanation: You attempted to install a task built with external headers in a system that was not generated with external header support.

VMR -- Feature not supported in system image

Explanation: You either attempted to create secondary pool in a system that does not support it, or you attempted to set a device public in a system that does not support multiuser protection, or you attempted to enable/disable the broadcast option for a terminal, but the system is not a multiuser/multi-CLI system.

VMR -- File <name> has illegal format

Explanation: The driver's symbol definition file contains illegal object code or data.

VMR -- File <name> has inconsistent data

Explanation: This message can occur for the following reasons:

1. You attempted to fix a partition or a task that is logically beyond the end of the system image file.
2. The partition for loadable drivers is logically beyond the end of the system image file. The system image is probably corrupt.
3. The partition of the common library you are attempting to load is logically beyond the end of the system image file.

VMR -- File <name> not a valid driver task image

Explanation: One of three things has occurred:

1. The driver's task image has been overlaid.
2. The driver has a header.
3. The driver has referenced a resident library.

VMR -- File not contiguous

Explanation: This message can occur for the following reasons:

1. You attempted to install a task from a noncontiguous file. Task images must be contiguous files.

2. The file ddDRV.TSK is not contiguous. This file must be contiguous for a device driver to be loaded.

VMR -- File not found

Explanation: You requested a file that is not in the directory of the volume you specified.

VMR -- File not task image

Explanation: The data in the label block of the task you tried to install is incorrect, indicating that the file is not a task image.

VMR -- Illegal device/volume

Explanation: You attempted to save your system image on an invalid device. Valid devices are:

CT: Tape cassette

DD: TU58 DECtape II

DT: DECtape

DU: RX50 flexible disk/RA60 disk/RC25 disk (removable)

DX: RX01 flexible disk

DY: RX02 flexible disk

MM: TE/TU16/TU45/TU77 magnetic tape

MS: TS11/TSV05/TU80 magnetic tape

MT: TE/TU10/TS03 magnetic tape

PP: Paper tape

VMR -- Illegal driver task APR usage

Explanation: The device driver being loaded must be built for APR 5 and must be less than 4K words.

VMR -- Illegal error severity code <code>

Explanation: This message indicates an internal failure in VMR. If this error message persists, contact a DIGITAL software representative.

VMR -- Illegal file specification <filespec>

Explanation: The file specification printed following the error message has an illegal format or contains wildcards.

VMR -- Illegal first APR

Explanation: The task you tried to install was built using an incorrect base APR. The following rules apply:

1. A nonprivileged task must be built with APR 0 as its base APR.
2. A privileged task that does not map into the Executive must be built with APR 0 as its base APR.
3. If the Executive is 16K words or less, a privileged task that is to map to the Executive routines and data structures must be built with APR 4 as its base APR.
4. If the Executive is 20K words (or less on RSX-11M systems), a privileged task that is to map to the Executive routines and data structures must be built with APR 5 as its base APR.

VMR -- Illegal function

Explanation: You entered a command that VMR does not recognize, or the command line refers to a feature that is not supported in the system.

VMR -- Illegal Get Command Line <error code>

Explanation: This message indicates a system failure. If the error persists, contact a DIGITAL software representative.

VMR -- Illegal keyword value

Explanation: You entered a value for a keyword that is out of range.

VMR -- Illegal operation for unmapped system

Explanation: The SET /POOLSIZE keyword is valid only on mapped systems.

VMR -- Illegal parameter for executive common partition

Explanation: You specified a parameter with SET /MAIN that is illegal for an executive common. The partition size must be 200(8) and the partition type must be COM.

VMR -- Illegal priority

Explanation: You specified a priority for the task being installed that is out of range (that is, not 1 through 250 decimal).

VMR -- Illegal slave attribute

Explanation: You attempted to install a task with the /SLV keyword, but the task's name is in the form ...xxx. The task is a prototype task and cannot be installed as a slaved task.

VMR -- Illegal switch <switch>

Explanation: The switch following the error message is incorrect in the context of the command line you have entered.

VMR -- Illegal use of <pname> partition or common

Explanation: You attempted to install a task in a CPU or secondary pool partition.

VMR -- Illegal use of <pname> partition or region

Explanation: You attempted to load a driver into a common or secondary pool partition.

VMR -- Illegal use of wildcard

Explanation: On an unmapped system, you cannot specify a wildcard (*) with the following SET keywords: /TOP, /BOT, and /POOL. You also cannot use a wildcard for the size parameter of task partitions.

VMR -- Illegal value for symbol <symname> in file <filename>

Explanation: LOAD or UNLOAD has determined that symbol <symname> in the STB <filename> has been defined to an illegal value. This can occur for several reasons, some general and some specific to individual symbols. For example, most symbols generally cannot be defined to zero or to an odd number. A specific example is the symbol $INTSV, which must not be referenced by mapped loadable drivers.

VMR -- Incorrect assignment of SY: or LB:

Explanation: This is a warning message. The device SY: or LB: is not assigned to the same device as the system image file you are trying to work with in VMR.

VMR -- Indirect command syntax error <command-line>

Explanation: You entered an improper indirect command file specification.

VMR -- Indirect file depth exceeded <command-line>

Explanation: You attempted to reference more than two levels of indirect command files.

VMR -- Indirect file open failure <command-line>

Explanation: The indirect command file specified in the command line could not be opened.

VMR -- Install device not LB0:

Explanation: You attempted to install a task from other than the boot device.

VMR -- Installed tasks or commons may no longer fit in partition

Explanation: You shortened a partition so that the tasks and/or commons that are installed in it may no longer fit. This is a warning message. VMR shortens the partition even though this situation exists.

VMR -- Interrupt vector already in use

Explanation: The interrupt vector for the driver being loaded does not point to one of the nonsense interrupt entry points in the system image.

VMR -- Invalid driver data base at offset xxxx in file <filename>

Explanation: The driver name has an invalid data base value at offset xxxx relative to the symbol $ddDAT. (See the RSX-11M or RSX-11M-PLUS Guide to Writing an I/O Driver.

VMR -- Invalid interrupt vector

Explanation: You specified an interrupt vector address in the driver data base that is too high for the system image.

VMR -- Invalid keyword

Explanation: You specified a keyword that is incorrect in the context of the command.

VMR -- Invalid record address for file <filename>

Explanation: LOAD has attempted to write outside of the system image file or at a byte boundary. Three possible causes are:

1. A corrupted system image
2. An STB file that does not match the system image
3. An internal problem in VMR

VMR -- Invalid speed

Explanation: This message can occur for two reasons:

1. You requested a speed that the multiplexer line you specified does not support.
2. You specified unequal receive and transmit speeds for a DZ11 or DZV11. The DZ11 and DZV11 multiplexers do not support split speeds.

VMR -- Invalid time parameter

Explanation: The time field you specified in the RUN command line is incorrect.

VMR -- Invalid UIC

Explanation: You specified an illegal value for either a group number or a member number. The numbers must be from 1 through 377(8).

VMR -- I/O error on input file <filename>

Explanation: An input I/O error occurred on the indicated file.

VMR -- I/O error on output device

Explanation: SAVE has encountered an I/O error on the output device while writing the system image.

VMR -- I/O error on output file <filename>

Explanation: An output I/O error occurred on the indicated file.

VMR -- Keyword not supported for this terminal driver

Explanation: The terminal driver selected during system generation does not support the specified SET keyword. For example, typeahead is a full-duplex terminal driver option only and your system uses the half-duplex driver.

VMR -- KRB <KRBname> interrupt vector <octal-number> in use

Explanation: You attempted to load a driver that specifies an interrupt vector that is not pointing to a nonsense interrupt entry point address. This is usually caused by a previously loaded driver that has used this vector.

When you get this message, LOAD has loaded the driver correctly, but it has not overwritten the interrupt vector.

VMR -- KRB <KRBname> interrupt vector <octal-number> too high

Explanation: You attempted to load a driver that specifies an interrupt vector address that is higher than the highest permissible vector address in the system.

When you get this message, LOAD has loaded the driver correctly, but it has not overwritten the interrupt vector.

VMR -- KRB <KRBname> not in loadable data base

Explanation: You attempted to load the driver for a multicontroller device with a loadable data base. The special symbol (KRBname) that defines the location of the KRB in the multicontroller device's CTB table is not defined in the loadable data base. See the RSX-11M-PLUS Guide to Writing an I/O Driver.

VMR -- KRB table of CTB <CTBname> will not accept KRB <KRBname>

Explanation: You attempted to load the driver for a multicontroller device with a loadable data base. If this operation is to be successful, two conditions must exist:

1. The appropriate slot in the CTB (Controller Table) must exist.

2. The slot in the CTB must be unused.

When VMR issues this error message, one of these conditions does not exist. See the RSX-11M-PLUS Guide to Writing an I/O Driver.

VMR -- Length mismatch common block <blockname>

Explanation: The length parameter for the common block, as described in the label block for the task image, does not match the corresponding length parameter defined in the system image. A task's label block data must match system data for that task before it can be installed.

VMR -- Line not DZ11

Explanation: You attempted to set a line to remote that is not attached to a DZ11, but your system uses the half-duplex terminal driver.

VMR -- Loadable driver larger than 4K

Explanation: This is a warning message. You are loading a driver that is larger than 4K words.

VMR -- Loadable driver support not in system

Explanation: You attempted to load a device into a system image that does not contain the Executive routine $INTSI. The routine is needed to support loadable device drivers.

VMR -- Logical device not in system

Explanation: The device you specified has not been defined and therefore ASSIGN could not find it in the logical device assignment table.

VMR -- LUN out of range

Explanation: You attempted to reassign a LUN for a task that is higher than the maximum number of LUNs allocated for the task when it was built.

VMR -- No checkpoint space, assuming not checkpointable

Explanation: This is a warning message. You attempted to install a task as checkpointable that was not built as checkpointable. This means that there is no checkpoint space allocated in the task image file. Since the system image does not support the dynamic allocation of checkpoint space, INSTALL does not consider the task to be checkpointable.

VMR -- No checkpoint space or dynamic checkpoint file

Explanation: You attempted to install a task built with external headers or as checkpointable, but the task does not have any checkpoint space and the system does not have a checkpoint file.

VMR -- No LUNs

Explanation: The task that you specified as an argument of the LUN command does not have any logical units assigned to it. This is not an error message; rather, it is simply an indication that there are no assignments to display.

VMR -- No pool space

Explanation: The pool space required to load the driver is not available in the system image.

VMR -- No room available in STD for new task

Explanation: There is no dynamic storage available to make an entry in the System Task Directory (STD). The task cannot be installed.

VMR -- No table space for optional directive common

Explanation: You attempted to fix a directive common in the RSX-11M-PLUS system image, but the extra slots built into the system data base for unrequired directive commons have all been used.

VMR -- Nonexistent memory

Explanation: You attempted to define a partition in nonexistent memory.

VMR -- Not enough APRs for task image

Explanation: The Task Builder allows you to specify the virtual base address of a task image as a multiple of 4K. Privileged tasks that access the Executive start at either virtual address 100000(8) for a 16K Executive or at virtual address 120000(8) for a 20K Executive. If the virtual base address of the task is set too high, there are not enough APRs remaining to map the task image. (See the RSX-11M/M-PLUS Task Builder Manual for information on privileged tasks.)

VMR -- Old device attached

Explanation: You attempted to redirect an attached device.

VMR -- Open failure on file <filename>

Explanation: The indicated file cannot be opened.

VMR -- Operation cannot extend above first 124.K of memory

This message can occur for the following reasons:

1. On RSX-11M, you attempted to fix a task above 124K words. The largest system that can be saved is 124K words. If FIX allowed you to fix a task above this limit, the task would be lost when the system was saved.

2. You attempted to create with the SET command a CPU partition whose upper limit is above 124K words.

VMR -- Operation not allowed for <ptype> partitions

Explanation: You attempted a SET operation that is not allowed for the type of partition specified in the error message.

VMR -- Optional directive common successfully loaded

Explanation: VMR has successfully fixed in the system image a directive common that is not one of the commons required by RSX-11M-PLUS (DIR11M and DR211M).

VMR -- Partition already exists

Explanation: This message can occur for the following reasons:

1. You attempted to define a partition, using the SET command, with a name already in use as a partition name.

2. You attempted to define a subpartition, using the SET command, but there are no more subpartition slots available.

VMR -- Partition or common region is busy

Explanation: This message can occur for the following reasons:

1. You attempted to fix a task or common region in a partition that is full. The task or region cannot be fixed.

2. You attempted to load a device driver into a partition, but the partition is full.

3. You attempted to remove a partition that still has tasks installed in it. The partition cannot be removed.

VMR -- Partition <pname> is a common

Explanation: You attempted to load a device driver into a common partition. This cannot be done.

VMR -- Partition <pname> is not a common

Explanation: Partition <pname> was found, but it is not a common partition.

VMR -- Partition <pname> is too small

Explanation: You attempted to load the driver into a partition that is either too small for the driver or that does not currently have enough space for the driver. Or, you attempted to install a task that is too large for the specified partition.

VMR -- Partition <pname> not in system

Explanation: This message can occur for the following reasons:

1. The partition you specified in the command line is not in the system image.

2. You attempted to load a driver into a partition that does not exist in the system, or LOAD found a symbol $xxCOM in the driver and partition xxCOM does not exist in the system.

VMR -- Partition <pname> not in system, defaulting to GEN

Explanation: This is a warning message. You attempted to install a task in a partition that does not exist. VMR will try to install the task in the partition GEN.

VMR -- Partition not system controlled

Explanation: The SET command keywords /TOP and /BOT are valid only for system-controlled partitions.

VMR -- Partition reduced to executive common size

Explanation: This is an informational message. When an executive common is installed in its partition, the partition is reduced to the top of the common to eliminate any unused space in the partition.

VMR -- Privileged task overmaps I/O page

Explanation: This message pertains to the installation of a privileged task.

A privileged task that is mapped into the Executive is often mapped to the I/O page as well. However, many privileged tasks do not require access to the I/O page and can use the extra 4K of address space that would be allocated to the I/O page for the task image. This message warns that a privileged task that may require access to the I/O page does not have that access. If the task does not require access to the I/O page, this message can be ignored.

VMR -- Privileged task larger than 12K

Explanation: You attempted to install a privileged task that is larger than 12K words. (See the RSX-11M/M-PLUS Task Builder Manual for information on privileged tasks.)

VMR -- Pool link error at xxxx Block=yyyy zzzz

Explanation: When SET /POOL traced the pool linkage, it detected an error at location xxxx in the system image. The contents of xxxx is yyyy and the contents of xxxx+2 is zzzz.

VMR -- Pseudo device assignment error

Explanation: You attempted to assign a logical device name to a pseudo device. Logical device names cannot be assigned to pseudo devices.

VMR -- Pseudo device redirect error

Explanation: You attempted to redirect one pseudo device to another pseudo device. This is not allowed.

VMR -- R/O partition <pname> not in system, defaulting to task's

Explanation: You specified a read-only partition that does not exist in the system. INSTALL will place the read-only portion of your multiuser task into the same partition as the read/write portion of the task.

VMR -- Receive data or by reference list not empty

Explanation: A task with entries in its receive queues cannot be removed from the system image.

VMR -- Space used

Explanation: You attempted to create a partition in a storage area that is already occupied.

VMR -- Specified partition for common block

Explanation: You attempted to install a task in a common block.

VMR -- Specified partition too small

Explanation: This message can occur for the following reasons:

1. You attempted to install a task into a partition that is smaller than the task.

2. You attempted to install a driver into a partition that is smaller than the driver.

VMR -- Symbol <symname> is doubly defined by file <filename>

Explanation: The symbol symname is defined twice in the specified file. Duplicate symbols are illegal. The driver is not loaded or unloaded.

VMR -- Symbol <symname> is undefined in file <filename>

Explanation: LOAD found the symbol symname in the specified file, but the symbol is not defined.

VMR -- Syntax error

Explanation: You have typed the command line incorrectly.

VMR -- System image is not RSX-11M-PLUS

Explanation: You attempted to use RSX-11M-PLUS VMR on a non-RSX-11M-PLUS system image file.

VMR -- System image is RSX-11M-PLUS

Explanation: You attempted to use RSX-11M VMR on an RSX-11M-PLUS system image file.

VMR -- Task active

Explanation: The task you specified as an argument of the command is active.

VMR -- Task and partition bases mismatch

Explanation: The base of the partition does not match that of the task being installed or of the driver being loaded. This message applies only to unmapped systems.

VMR -- Task or common region already fixed

Explanation: You attempted to fix a task or common region in memory that is already fixed.

VMR -- Task has attached common regions

Explanation: This message can occur for the following reasons:

1. You attempted to fix or unfix a task in memory that is linked to a shared region. VMR cannot fix or unfix such tasks.

2. You attempted to remove a task that is linked to a shared region from the system image. VMR cannot remove such tasks.

VMR -- Task image currently installed

Explanation: You attempted to install a checkpointable task that has checkpoint space allocated in its task image. The task has already been installed.

The system allows you to install checkpointable tasks more than once subject to the following rules:

1. You must allocate checkpoint space for the task in a system checkpoint file, or

2. You must use a different name for the task each time you install it.

VMR -- Task image I/O error

Explanation: The task cannot be installed or the driver cannot be loaded. VMR could not read the task image file or could not rewrite the task image header. The specified device may be write-locked.

VMR -- Task image virtual address overlaps common block <blockname>

Explanation: The virtual addresses you reserved for the task image overlap those you reserved for the common block specified in the error message. Corruption of the task image file may have caused the overlap.

VMR -- Task is checkpointable

Explanation: You attempted to fix a checkpointable task in memory. Checkpointable tasks cannot be fixed.

VMR -- Task is not multi-user

Explanation: You attempted to fix in memory the read-only portion of a multiuser task, but the task you named in the command line is not a multiuser task.

VMR -- Task name already in use

Explanation: You attempted to install a task of the same name as that of one already installed in the system image.

VMR -- Task not installed with external header

Explanation: This is a warning message to indicate that the specified task is installed with its header in pool (the dynamic storage region).

VMR -- Task not in system

Explanation: You referenced a task that is not installed in the system.

VMR -- Task not removed, CLI has messages enabled

Explanation: You attempted to remove a CLI task that is able to receive messages from the system. The task may have messages that it has not received yet, so the remove request is aborted.

VMR -- Task or common not in memory

Explanation: You attempted to remove a task or common from the system that is checkpointed and, therefore, not in memory.

VMR -- Task or common region is not fixed

Explanation: You attempted to unfix a task that is not fixed in the system image.

VMR -- Task or common region not in system

Explanation: You referenced a task or common region that has not been installed in the system or, on RSX-11M-PLUS, you attempted to run a prototype task.

VMR -- Task requires system controlled partition

Explanation: You attempted to install a task that was built with external header (/XH) support and as checkpointable (TKB /CP, INS /CKP=YES). Such tasks must be installed in a system-controlled partition.

VMR -- TI redirect error

Explanation: You attempted to redirect the pseudo device TI:. TI: cannot be redirected.

VMR -- TT redirect error

Explanation: You attempted to redirect a terminal to the null device. This combination is not allowed.

VMR -- Too many common block requests

Explanation: You attempted to install a task that contains too many common block requests. A task is limited to seven common block references.

VMR -- Too many LUNs

Explanation: You attempted to install a task that contains more than 250(10) LUN assignment requests.

VMR -- Too many symbols of the form $xxTMO in file name

Explanation: LOAD is attempting to extract symbols of the form $xxTMO from the driver's symbol definition file. There are more symbols of this form than LOAD can process.

VMR -- Top of pool set to <value>, requested amount not available

Explanation: You requested more pool space than the system can provide. The system provides as much as possible in this case.

VMR -- Undefined common block <blockname>

Explanation: You attempted to install a task that references a common block that is not defined in the system image. This message may indicate that the task was built for another system.

VMR -- Unknown partition

Explanation: You attempted an operation on a partition that does not exist in the system.

VMR -- User D-space not supported

Explanation: An attempt was made to install a task with separate instruction and data space on a system that does not support user D-space. (User D-space is a system generation option.)

VMR -- Virtual terminal error

Explanation: You attempted to enable, using the SET command, a terminal option for a virtual terminal.

VMR -- Write check not supported for device

Explanation: You attempted to enable write checking for a device that does not support it. Write checking is supported for all disks except RL01s and RL02s (DL:) and RX01s (DX:).

CHAPTER 11

COMMAND LINE INTERPRETERS

11.1 INITIAL CONCEPTS

A command line interpreter, or CLI, is a task that services unsolicited commands meant for the operating system or an application. These commands are not prompted for by any task nor are they given to a task by a read operation on a terminal.

The best-known CLI on RSX-11M/M-PLUS systems is MCR, which controls the operating system. MCR, like any other CLI, receives command lines that originate as unsolicited input typed on a terminal or as issued by a task using an Executive directive. Since all lines typed on a terminal are seen by the CLI (except those used to satisfy a task's read requests), the CLI task is, in effect, a complete user interface. This control over the user at the terminal is absolute. There is no way to bypass a CLI to send commands to the system or application.

A CLI is also an efficient means of interfacing an application command processor to any number of terminals, since no read QIO or character AST processing is involved. The input from the terminal simply goes to the CLI by default for processing.

This chapter introduces the concepts of CLIs on RSX-11M/M-PLUS systems. The chapter provides enough information to allow a programmer to write application-specific CLIs. A brief description of three DIGITAL CLIs supplied with the system illustrates these concepts. These three system CLIs do quite different things with the command lines they process, but the basic goal of each is the same: to provide a specific environment for the terminal user. This basic goal also applies to any application-specific CLIs, which, of course, may perform completely different command processing functions. A simple sample CLI - written in both FORTRAN and MACRO-11 - is included in this chapter. The example is also included on the distribution kit in either LB:[1,2] with the HELP files or in the new-user account [200,1] as TMCLI.MAC and TMCLI.FTN. See your system manager for more information. The example illustrates the CLI interface and should clarify the basic operations of a CLI.

NOTE

This chapter describes features available on RSX-11M and RSX-11M-PLUS. RSX-11S systems include Basic MCR, but do not support the CLI features discussed in this chapter.

11.1.1 MCR - The Traditional RSX-11M/M-PLUS CLI

The most important CLI on RSX-11M/M-PLUS systems is MCR. MCR receives commands from the terminal and executes them to control the system. MCR is a set of privileged tasks. These tasks call Executive routines and interact with the system's data base. Therefore, MCR is tied very closely to the Executive. This bond is necessary because of MCR's function in system control and has nothing to do with MCR's role as a CLI.

Most other CLIs interact with an application running on the system, rather than with the system itself. Therefore the CLI designer need not have any knowledge of RSX-11M/M-PLUS internals. In fact, the resulting CLI need not even be privileged. The only similarity between MCR and other CLIs is that any CLI controls the terminals associated with it just as MCR does. Every terminal on an RSX-11M/M-PLUS system is set to some CLI. Users can do nothing on their terminals that their CLI does not permit. The CLI controls the use of the terminal.

NOTE

While MCR is a CLI, it is a special CLI. MCR does not use the CLI interface directives because it must work on systems that do not support these directives. Therefore, MCR is not a pattern for other RSX-11M/M-PLUS CLIs.

11.1.2 DCL - DIGITAL Command Language

The DIGITAL Command Language (DCL) interpreter is a CLI that does not actually execute any commands. DCL provides an interface to operating system features by translating the commands it receives in DCL syntax to the equivalent MCR commands. DCL then passes the translated command line to the appropriate MCR task or utility for execution. The CLI interface hides this operation from the user at the terminal, who does not need to know about MCR or the utilities. DCL is oriented towards operations, such as COPY or LINK, rather than towards tasks like PIP and TKB. DCL presents a totally different external interface to RSX-11M/M-PLUS, one that is easier to learn and use than MCR. This makes it possible for less technical users to do work on RSX-11M/M-PLUS.

11.1.3 RMT - DECnet Remote Terminal Facility

The RMT task is a CLI that gathers commands from the terminals associated with it and sends them over a DECnet network to be executed on a remote RSX-11M/M-PLUS system. The RMT interface provides a mechanism for command handling with a minimum of overhead or effect on user tasks. RMT is different from DCL: RMT does not interpret the command strings it handles, while all DCL does is interpret strings. All the interpretation and execution of commands passed through RMT is done by the CLI that receives the commands on the remote system.

11.2 WRITING AN APPLICATION CLI

A CLI is an RSX-11M/M-PLUS task like any other. It need not be privileged, unless the application requires it. Any alternate CLI is expected to interact with the operating system in a certain way, however. This is necessary to synchronize with the operating system in command handling and prompting. This interaction is achieved through a set of Executive directives for CLIs.

11.2.1 Executive Directives for CLIs

The most important directive is GCCI$ (Get Command for Command Interpreter). GCCI$ must be used by every CLI, no matter what the CLI does. GCCI$ supplies a command to the CLI from a queue maintained by the system. It also controls the internal status bits necessary for correct synchronization of prompting and general command handling in a manner transparent to the CLI task.

Other CLI directives include:

1. RPOI$ (Request and Pass Offspring Information) and SDRP$ (Send, Request, and Pass), which can be used to pass the responsibility for handing a command off to another task so the CLI is free to handle another command.

2. SCAA$ (Specify Command Arrival AST), which specifies an AST in the CLI whenever a command arrives for processing by the CLI.

3. GCII$ (Get Command Interpreter Information), which returns information about a terminal or CLI to the task.

See the RSX-11M/M-PLUS Executive Reference Manual for a complete description of these directives.

11.2.2 Basic Interaction with the Operating System

Before a CLI can process commands, it must be made known to the system. This is done by installing the CLI task with the /CLI=YES switch and then initializing the CLI with the CLI /INIT command. This process creates an internal data structure that allows the system to recognize the existence of the CLI. See the RSX-11M/M-PLUS MCR Operations Manual for a description of the INSTALL command and the CLI commands.

A CLI's operation starts when a command is entered into the system. The command may originate as a command line typed on a terminal or from an Executive directive issued by a task. The source of the command is generally irrelevant to the CLI. The command is processed first by the system command dispatcher (the MCR... task). This task puts the command into a FIFO queue of commands to be retrieved by the CLI. The CLI associated with the terminal from which the command originated is then requested to run (or unstopped if it is already active). The CLI gets the command from the system by issuing the GCCI$ directive. This directive removes a command from the CLI's queue and copies it into a buffer in the CLI's task space. GCCI$ can also supply information about the source terminal to the CLI.

Once the CLI has the command, it is free to process it in any manner it chooses. Generally this involves executing the command, or passing the command to another task for execution. If the CLI is going to pass the command to another (non-CLI) task, it should use the RPOI$ or SDRP$ directive to do this, assuming the CLI has no further interest in the command. These directives propagate the information the system needs for controlling, prompting and getting exit status back to a parent task, if any. The CLI's obligations to the command are complete once the command has been passed off. There is no reason for the CLI to get involved again. This is the recommended procedure for handling commands, as it avoids the wait for the preceding command to complete execution. DCL works in this way: translating a command into a different format and dispatching it to another task for execution.

CLIs that process commands themselves should be very careful about handling any possible delays in execution. This is especially true if the CLI is to do I/O to a terminal; this means any terminal I/O, including prompts for input. Otherwise, it would be very easy for one user to suspend command handling by not responding to a prompt or by typing CTRL/S.

Once a CLI has completed processing a command, it must reissue the GCCI$ directive to get any other command which may have been queued to it. If the CLI wants to stop or exit when no commands are queued to it, it should do this by issuing the GCCI$ directive with the appropriate no-command action specified. This prevents a command queued between the return from the GCCI$ and the issuance of an EXIT or STOP directive from being lost, or not processed until the next command is queued.

11.2.3 Task Names for CLIs

CLIs can be installed with task names in the form ...xxx (commonly called a prototype task name) or with names in some other legal form, such as xxxxxx (commonly called a conventional task name).

If the CLI has a prototype task name, the system creates a separate copy of the CLI each time a command is entered on a terminal set to that CLI. The task is named xxxtnn, following the same general rule as any other task installed in the system with a prototype task name. Each copy of the CLI runs with the same TI: and UIC as that of the terminal the command came from. These tasks should exit when there is no work left for them to do.

If the CLI has a conventional task name, all commands from all terminals associated with the CLI are queued to a single copy of the task. This task runs with a TI: of CO: (console output pseudo device). CLIs with conventional task names must specify the TI: and UIC fields in any RPOI$ or SPWN$ directives they issue, since the the TI: and UIC will not default correctly. Since there is only one copy of these tasks in the system, CLIs with conventional task names can simply stop rather than exiting when there are no commands queued for them. This may be necessary if the CLI must maintain information from one command to the next.

DCL is installed with a prototype task name, ...DCL. RMT is installed as RMT..., with a conventional task name. The difference reflects the use and nature of each CLI.

DCL must be installed with a task name of ...DCL because most DCL commands issue prompts for input. No command processing can be done by DCL while it is waiting for the user to respond to a prompt. If there were only one copy of DCL on the system, command processing for all DCL terminals would stop while a prompt request to any one terminal is outstanding. This is clearly not an acceptable situation. Giving each terminal its own copy of DCL allows independent command processing for each terminal. To avoid filling the system with copies of DCL, the individual copies of the task exit when no commands are left for them. They can do this since there is no need to maintain any context between commands.

On the other hand, RMT is installed as RMT... because there can only be one copy of RMT in the system. This is because RMT provides a focal point with which the DECnet command handling software running on the remote system can communicate. Furthermore, it is not practical to have RMT exit when no commands are outstanding since it is necessary to maintain context between commands. If each user got a separate copy of RMT, the system would be cluttered with copies of RMT that did not exit.

11.2.4 MCR Control Commands

The MCR CLI command is the means by which the system has knowledge of CLIs. The CLI command has switches which perform such functions as making a CLI known to the system, eliminating it from the system, turning on and off its recognition of commands, and so forth. See the RSX-11M/M-PLUS MCR Operations Manual for a description of the CLI command.

11.2.5 System Messages to CLIs

Depending on the nature of the CLI, it may be necessary for it to receive messages notifying the CLI of the occurrence of certain events that could effect it. If the /MESSAGE subkeyword to the CLI /INIT command is specified, it can receive messages from the system informing it of such events. These messages are returned by the GCCI$ directive just like commands. They can be differentiated from commands by the first word of the command buffer (G.CCDV) being cleared, instead of being set up with the name of the device from which the command originated. The code identifying which message is present is returned in the unit number byte (G.CCUN). The other fields have the same meaning for messages as they do for commands. The actual text field of the buffer may be empty if the message does not contain any ASCII text. See the RSX-11M/M-PLUS MCR Operations Manual for further information.

NOTE

The CLI command includes both a /MESSAGE subkeyword to the /INIT keyword and a /MESSAGE keyword. The /MESSAGE subkeyword specifies that the CLI being initialized can receive messages. The /MESSAGE keyword sends an ASCII message to a CLI that has been initialized with the /MESSAGE subkeyword. See the RSX-11M/M-PLUS MCR Operations Manual for further information.

The following sections describe the messages. Message codes 1 through 127. are reserved for use by DIGITAL. Users wishing to create their own messages to CLIs should use codes in the range of 128. through 255. The symbol shown for each message code is defined in the Executive definition library: LB:[1,1]EXELIB.OLB. Tasks may pick them up at build time by linking with that library.

11.2.5.1 **CLI Initialized into an Enabled Mode** - This message is sent to a CLI when it is initialized by a CLI /INIT command without specifying the /DISABLE switch.

Message Code: 1 (CM.INE)

Text: CLI name in RAD50, length = 4 bytes.

11.2.5.2 **CLI Initialized into a Disabled Mode** - This message is sent to a CLI when it is initialized by a CLI /INIT command with the /DISABLE switch.

Message Code: 2 (CM.IND)

Text: CLI name in RAD50, length = 4 bytes.

11.2.5.3 **CLI Enabled** - This message is sent to the CLI when it is enabled by a CLI /ENABLE command.

Message Code: 3 (CM.CEN)

Text: None.

11.2.5.4 **CLI Disabled** - This message is sent to the CLI when it is disabled by a CLI /DISABLE command.

Message Code: 4 (CM.CDS)

Text: None.

11.2.5.5 **CLI Eliminated** - This message is sent to a CLI when it has been eliminated by an explicit CLI /ELIMINATE command.

Message Code: 5 (CM.ELM)

Text: None.

11.2.5.6 **All CLIs Eliminated, and must Exit** - This message is sent to all CLIs on the system which were initialized with the /MESSAGE subkeyword to the CLI /INIT command when a CLI /ELIM=* command is issued. A CLI that receives this message should exit as soon as possible, since system shutdown or other emergency procedure is taking place.

Message Code: 6 (CM.EXT)

Text: None.

11.2.5.7 **Terminal Linked to CLI** - This message is sent to a CLI when a terminal becomes associated with it. This message could result from a SET command processed by MCR, or any other use of the SCLI$ directive.

Message Code: 7 (CM.LKT)

Text: Name (ASCII) and unit number (octal) of terminal, length = 4 bytes. (2 bytes each).

11.2.5.8 **Terminal Removed from CLI** - This message is sent to a terminal's former CLI when it is switched to another CLI.

Message Code: 8. (CM.RMT)

Text: Name (ASCII) and unit number (octal) of terminal, length = 4 bytes. (2 bytes each).

11.2.5.9 **General Message** - This message is the result of a the CLI /MESSAGE command. It contains whatever text the issuer of the command included in the string, plus the name and unit number of the terminal on which the command was typed.

Message Code: 9. (CM.MSG)

Text: Name (ASCII) and unit number (octal) of terminal where the CLI /MESSAGE command originated, followed by the text from the command string. The length is equal to the length of the string (possibly zero) plus the length of the terminal name and unit number (2 bytes each).

11.2.6 **Prompting**

There are three kinds of prompts issued by the system that affect CLIs:

1. CTRL/C prompts
2. Empty line prompts generated by solitary carriage returns
3. Task exit prompts

11.2.6.1 **CTRL/C Prompts** - This prompt is issued by the terminal driver when it receives a CTRL/C character. The string to be issued can be specified when the CLI is initialized. There is no way the CLI can be notified when this prompt is issued, however, since it is handled by the terminal driver. These prompts are of no other concern to CLIs.

11.2.6.2 **Empty-Line Prompts** - This prompt results from terminal input consisting solely of a carriage-return character, which is called a null command line. The standard system action is to have the MCR... task issue the default prompt, which can be specified when the CLI is initialized. The CLI can be initialized with the /NULL switch, which causes the system to send null command lines to the CLI like other commands, and not output anything. Any prompting or other action is therefore the CLI's responsibility.

11.2.6.3 **Task-Exit Prompts** - These prompts are issued when a task started by a command line exits. The standard system action is to have the MCR... task issue the same default prompt string as in the case of an empty line. If the CLI wishes to be informed of these occurrences, it can be initialized with the /PROMPT switch, which causes the system to send these task exit prompt requests to the CLI without terminal output. Any prompting or other action is then the CLI's responsibility.

On RSX-11M-PLUS systems, a CLI initialized to receive either empty-line or task-exit prompts has to issue an SNXC$ directive when its processing of the prompt is complete. SNXC$ tells the terminal driver that the current command is complete, and it can send the next one if the terminal is in serial processing mode. This directive can be issued if the terminal is not in serial mode without any adverse effects. See Section 11.2.6, Serial Command Processing, for further details.

11.2.7 System Handling of Continuation Lines

Continuation lines are handled by the RSX-11M-PLUS terminal driver before the command is passed to the CLI. Incomplete segments of command lines are detected by the presence of a hyphen as the character immediately preceding the terminator. Any hyphens are removed and all the pieces of the command put together before the CLI is aware of the command, so the CLI does not have to deal with continuation lines in any way. If the CLI does not want the system to interpret the presence of the hyphen as denoting a continuation line, it can be initialized with the /SNGL switch. This tells the system to accept single line commands only, and pass every line to the CLI as it is entered. No continuation line checks are made.

The DCL CLI on RSX-11M does its own handling of continuation lines. This is done within DCL itself, and is not available as a general system feature. This option is open to CLIs on RSX-11M systems, but is not necessary for CLIs on RSX-11M-PLUS as the system puts all the segments together before the command is sent to the CLI.

11.2.8 Interaction with Serial Command Processing (RSX-11M-PLUS only)

Serial command processing is a mode in which commands from a terminal are executed in sequential order, with a command not being started until the preceding command is finished. The default for command line processing on RSX-11M-PLUS systems is parallel processing, but a CLI may need to synchronize commands. Command synchronization is achieved by having the terminal driver store characters in the typeahead buffer if they are entered while a command is outstanding on that terminal. These characters are issued as the next command by a special call to the terminal driver at the completion of the processing of the current command. This command completion is normally signalled to the system by the exiting of the final task involved in the processing of the command.

If a task - either the CLI or a task chained to by RPOI$ or SDRP$ - exits at the completion of command execution and no prompt request is generated (meaning the command was terminated by an ESCAPE character), the terminal driver is called directly from the task exit code in the Executive. If a task exit prompt request is generated, the terminal driver has to be called at the completion of the prompt request processing. If the CLI is not receiving prompt requests, the terminal driver call is done automatically by the system, and the CLI need never get involved.

If the CLI has been initialized to receive prompt requests, the system passes the prompt request on to the CLI like any other command. It becomes the CLI's responsibility to write any prompt string to the terminal and call the terminal driver to initiate the next command. The CLI should therefore issue an SNXC$ directive at the completion of its processing of the prompt request.

If no task exit will occur to terminate the command execution, the system has no way of knowing when the command is complete. Therefore the SNXC$ directive must be used to mark the end of the command and cause the next one to be initiated. Issuing this directive causes a task exit prompt request to be generated if one would have been generated by a task exit. If no prompt request is generated, SNXC$ calls the terminal driver to send the next command to the system for processing. This directive can be issued by the CLI or the final task chained to by means of an RPOI$ or SDRP$ directive.

It is not necessary for applications using the SNXC$ directive to check if a terminal is actually in serial mode. If the terminal is not in serial mode, the terminal driver sends commands to the system as they are entered. The directive does not cause the terminal driver to be called since it is not necessary. The terminal driver still generates a task exit prompt request if one would have been generated when the task exited. The issuance of this directive clears the status bits that tell the system to generate a prompt request or call the terminal driver at task exit. Therefore a task exiting after issuing an SNXC$ directive does not cause two prompt requests to be issued.

See the RSX-11M/M-PLUS Executive Reference Manual for more information on the SNXC$ directive.

11.3 SIMPLE SAMPLE CLI

Here is a sample CLI, written in both MACRO-11 and FORTRAN. The same programs are included on your system kit as TMCLI.MAC or TMCLI.FTN. This CLI is as simple as possible to illustrate what is necessary to construct a CLI. All TMCLI does is receive commands and pass them to MCR for execution. A user of this CLI may not even be aware that it is there as there is no apparent difference from being an MCR terminal. The only additional capabilities TMCLI offers are two single-letter commands, T to type a file, or M to switch back to MCR mode. The comments in the listing suggest several simple modifications that can be made to illustrate the options available when designing or setting up a CLI.

11.3.1 MACRO-11 CLI

```
        .TITLE TMCLI - EXAMPLE CLI THAT IMPLEMENTS 'T' AND 'M' COMMAND
        .IDENT  /1.00/
;
; THIS TASK (TMCLI) ILLUSTRATES THE USE OF THE CLI CONTROL DIRECTIVES OF
; RSX-11M V4.0 AND RSX-11M-PLUS V2.0.
;
; WHEN SET UP AS A TERMINAL'S CLI, IT PROVIDES THE FOLLOWING COMMANDS:
;
;       T       ->      TYPE FILE
;       M       ->      SET TERMINAL TO MCR
;
; IF THE FIRST TWO LETTERS OF THE COMMAND ARE NOT 'T' FOLLOWED
; BY A SPACE, OR 'M' FOLLOWED BY A CARRIAGE RETURN,
; THE COMMAND IS PASSED TO MCR FOR EXECUTION. THEREFORE,
; USERS AT TERMINALS SET TO THIS CLI WILL SEE THE ENTIRE MCR COMMAND SET
; AS BEING PRESENT, PLUS THE 'T' AND 'M' COMMANDS. IF THE /DPR SUBKEYWORD
; IS INCLUDED ON THE CLI /INIT COMMAND, USERS WILL KNOW IMMEDIATELY IF
; THEIR TERMINAL IS SET TO THIS CLI BY THE DOUBLE RIGHT ANGLE BRACKET PROMPT.
;
; IF TMCLI IS INITIALIZED WITH THE /MESSAGE KEYWORD, TMCLI WILL
; GREET USERS WHEN THEIR TERMINAL IS SET TO THE CLI, AND SAY GOODBYE
; TO THEM WHEN THEY SWITCH AWAY.
;
; IN ORDER TO BRING THIS CLI UP ON THE SYSTEM, ISSUE THE FOLLOWING
; COMMAND SEQUENCE:
;
; >MAC TMCLI,TMCLI/-SP=TMCLI
; >TKB TMCLI,TMCLI/-SP=TMCLI,[1,1]EXELIB/LB
; >INS TMCLI/CLI=YES/TASK=...TMC
; >CLI /INIT=TMCLI/MESSAGE/DPR="<15><12>/>>/"/TASK=...TMC
;
;       TO SET TERMINALS TO THIS CLI, ISSUE THE FOLLOWING COMMAND
;       WHERE TTNN: IS REPLACED BY THE TERMINAL NUMBER OR TI:
;
; >SET /CLI=TTNN:TMCLI
;
; NOTE THAT THE PROMPT ISSUED IN RESPONSE TO A CONTROL/C DEFAULTS TO THE
; NAME OF THE CLI, WHICH IS 'TMCLI>'.
;
; TMCLI CAN BE ELIMINATED FROM THE SYSTEM BY THE FOLLOWING COMMANDS, ASSUMING
; NO TERMINALS ARE SET TO IT:
;
; >CLI /ELIM=TMCLI
; >REM ...TMC
;
```

```
; EXPERIMENT WITH THE OTHER SUBKEYWORDS OF THE CLI /INIT COMMAND, THE CLI
; /ENABLE AND CLI /DISABLE COMMANDS, AND WITH THE /MESSAGE SWITCH ON THE /INIT
; COMMAND TO OBSERVE THE EFFECTS.  ALSO TRY CHANGING THE /TASK=
; SWITCH ON THE INS AND CLI /INIT COMMAND TO '/TASK=TMCLI' AND NOTICE
; THAT ONLY ONE COPY OF THE TASK WILL EXIST FOR ALL TERMINALS, RATHER THAN
; ONE PER COMMAND. IF YOU DO THIS, YOU SHOULD ALSO CHANGE THE OPTION
; ON THE GCCI$ DIRECTIVE FROM GC.CEX TO GC.CST SO THE TASK WILL STAY ACTIVE
; BUT STOPPED WHEN NO COMMANDS ARE OUTSTANDING FOR IT.
;
        .MCALL  GCCI$,RPOI$,EXIT$S,QIOW$,ALUN$,DIR$
;
; DIRECTIVE DPBS
;
GCCI:   GCCI$   CMDBUF,CBLEN,IBUF,IBLEN,0,GC.CEX
RPOI:   RPOI$   MCR...
QIOW:   QIOW$   IO.WVB,5,1,,,,<,,40>
ALUN:   ALUN$   5

;
; BUFFER AREAS
;
CMDBUF: .BLKB   80.+G.CCBF      ;LENGTH = HEADER PLUS COMMAND IN BYTES
                                ;NOTE THAT ON M-PLUS THE MAXIMUM COMMAND
                                ;LENGTH IS 255. BYTES, SO THE LENGTH SHOULD
                                ;BE <255.+G.CCBF> OR CHARACTERS MAY BE LOST.

CBLEN=.-CMDBUF                  ;CALCULATE LENGTH
        .EVEN

IBUF:   .BLKB   G.CCCA+2        ;LONG ENOUGH TO CONTAIN ALL INFORMATION
IBLEN=.-IBUF                    ;CALCULATE LENGTH
        .EVEN

TCMD:   .ASCII  /PIP TI:=/      ;BASIC PIP COMMAND TO IMPLEMENT 'T' COMMAND
TCLEN=.-TCMD                    ;LENGTH OF COMMAND
BUF:    .BLKB   80.-TCLEN       ;BUFFER TO GENERATE COMMANDS IN
        .EVEN

MCR:    .ASCII  /M/<15>         ;STRING TO RECOGNIZE AS A SET TO MCR REQUEST
MCMD:   .ASCII  %SET /MCR=TI:%  ;SET TERMINAL TO MCR COMMAND
MCLEN=.-MCMD                    ;LENGTH OF COMMAND
        .EVEN

;
; TEXT TO BE DISPLAYED UPON RECEIPT OF NEW TERMINAL-LINKED OR
; TERMINAL-REMOVED MESSAGES FROM SYSTEM.
;
LKT:    .ASCII  <15><12>/WELCOME TO THE EXAMPLE CLI/<15><12>
LKTLEN=.-LKT
        .EVEN

RMT:    .ASCII  <15><12>/GOOD BYE FROM THE EXAMPLE CLI/<15><12>
RMTLEN=.-RMT
        .EVEN

;
; MESSAGE FOR CHARACTERS LOST DUE TO BUFFER BEING TOO SMALL
;
CHLOST: .ASCII  <15><12>/SORRY, PART OF YOUR COMMAND WAS LOST/<15><12>
CLLEN=.-CHLOST
```

```
;
; ILLEGAL COMMAND MESSAGE
;
ILL:    .ASCII  <15><12>/ILLEGAL COMMAND/<15><12>
ILLLEN=.-ILL
        .EVEN

;
; CLI CODE
;

        .ENABL  LSB
START:  DIR$    #GCCI           ;GET A COMMAND FROM SYSTEM
        BCS     100$            ;IF CS, DIRECTIVE ERROR
;
; IF THE GCCI$ DIRECTIVE HAD THE GC.CST OPTION SPECIFIED, WE MAY HAVE
; JUST BEEN UNSTOPPED, SO WE HAVE TO GET THE COMMAND.
;
        CMP     #IS.CLR,$DSW    ;WERE WE JUST UNSTOPPED?
        BEQ     START           ;IF EQ YES
        MOV     #CMDBUF,R0      ;POINT TO COMMAND BUFFER
        TST     (R0)            ;IS THIS A SYSTEM MESSAGE?
        BEQ     MSG             ;IF EQ YES
        CMPB    G.CCCT(R0),G.CCCL(R0) ;DID ENTIRE COMMAND FIT IN BUFFER
        BLO     150$            ;IF LO NO
        MOV     #RPOI,R5        ;POINT TO DPB OF RPOI$ DIRECTIVE
;
; AT THIS POINT, THE CLI HAS A COMMAND TO BE PROCESSED. THE CODE FROM
; THE 'START:' LABEL TO HERE SHOULD BE FAIRLY COMMON AMONG ALL CLIS.
; FROM THIS POINT ON, THE CODE WILL BE SPECIFIC TO THE PARTICULAR LANGUAGE
; THIS CLI IS IMPLEMENTING. COMMANDS CAN BE ADDED OR DELETED AT WILL
; BY MODIFYING CODE IN THIS SECTION. AS STATED ABOVE, THIS EXTREMELY
; SIMPLE CLI PROVIDES TWO COMMANDS, AND GIVES EVERYTHING ELSE TO MCR.
; THIS CREATES AN OPEN ENDED ENVIRONMENT. REJECTING UNRECOGNIZED COMMANDS
; INSTEAD OF PASSING THEM TO MCR WOULD LIMIT THE TERMINAL USER'S USE OF
; THE SYSTEM TO ONLY THOSE OPERATIONS ALLOWED BY THE CLI. THIS IS
; A SECURE WAY OF CONTROLLING SYSTEM USE, IN CASES WHEN IT IS NECESSARY.
; IN THIS EXAMPLE, CHANGING THE 'BNE 50$' BELOW, TO A 'BNE 60$'
; WOULD CREATE AN ENVIRONMENT WHERE THE ONLY OPERATION A USER COULD
; PERFORM WOULD BE TO TYPE FILES ON THE TERMINAL, OR SWITCH BACK TO MCR.
; ALL OF THE REST OF THE SYSTEM IS INACCESSIBLE. IF NO MEANS OF SWITCHING
; BACK TO MCR IS PROVIDED, IT IS IMPOSSIBLE FOR THE USER TO DO
; ANYTHING THE CLI DOES NOT EXPLICITLY LET HIM DO.
;

        CMP     #'T ,G.CCBF(R0) ;IS IT A 'T' COMMAND?
        BEQ     10$             ;IF EQ YES
        CMP     MCR,G.CCBF(R0)  ;IS IT AN 'M' COMMAND?

;
; CHANGE THE NEXT INSTRUCTION TO 'BNE 60$' TO ELIMINATE PASSING UNKNOWN
; COMMANDS TO MCR.
;
        BNE     50$             ;IF NE NO
```

```
;
; THE USER HAS ENTERED AN 'M' COMMAND, SO SET THE TERMINAL TO MCR.
;
        MOV     #MCMD,R.POBF(R5)  ;PUT ADDRESS OF CMD TO ISSUE IN DPB
        MOV     #MCLEN,R.POBL(R5) ;PUT LENGTH OF CMD IN DPB
        BR      ISSCMD            ;ISSUE COMMAND

;
; THE USER HAS ENTERED A 'T' COMMAND SO CONSTRUCT THE NECESSARY PIP
; COMMAND AND SEND IT TO MCR.
;

10$:    MOV     R0,R1             ;COPY ADDRESS OF BUFFER WITH COMMAND
        ADD     #G.CCBF+1,R1      ;POINT TO SPACE AFTER THE T
        MOV     #BUF,R2           ;POINT TO BUFFER THAT WILL CONTAIN CMD
        MOV     #TCLEN,R4         ;GET LENGTH OF INITIAL PART OF COMMAND
20$:    MOVB    (R1)+,R3          ;GET NEXT CHARACTER
        CMPB    #15,R3            ;IS IT THE END OF THE COMMAND
        BEQ     30$               ;IF EQ YES
        CMPB    #40,R3            ;IS IT A SPACE
        BEQ     20$               ;IF EQ YES, SKIP IT
        MOVB    R3,(R2)+          ;PUT CHARACTER IN OUTPUT BUFFER
        INC     R4                ;COUNT CHARACTER
        BR      20$               ;GET NEXT CHARACTER
30$:    MOV     #TCMD,R.POBF(R5);PUT ADDRESS OF CMD TO ISSUE IN DPB
        MOV     R4,R.POBL(R5)     ;INSERT LENGTH OF COMMAND
;
; THIS COMMAND WILL BE PASSED TO THE MAIN COMMAND DISPATCHER (MCR...),
; WHICH WILL START A COPY OF PIP AND GIVE IT THE COMMAND. THE RPOI
; DIRECTIVE COULD BE USED TO GIVE THE COMMAND DIRECTLY TO ...PIP AND
; GIVE THE RESULTING TASK A DIFFERENT NAME. IN THIS EXAMPLE,
; WE COULD CALL THE RESULTING PIP TASK 'TYPTNN' WHERE THE LAST
; THREE CHARACTERS (TNN) ARE REPLACED BY A DEVICE NAME CHARACTER AND
; UNIT NUMBER. FOR EXAMPLE, IF THE 'T' COMMAND WAS TYPED ON TT10:
; THE RESULTING TASK NAME COULD BE MADE TO BE 'TYPT10'. THIS HAS THE
; ADVANTAGE OF ALLOWING THE TASK TO BE ABORTED BY AN 'ABO TYP' COMMAND,
; INSTEAD OF 'ABO PIP', THUS THE USER WOULD NOT HAVE TO KNOW THAT THE
; 'T' COMMAND HE ISSUED IS ACTUALLY BEING SERVICED BY THE PIP TASK. THIS
; IS EXACTLY WHAT THE DCL CLI DOES. IT IS NOT DONE HERE
; TO SIMPLIFY THE EXAMPLE. PICKING UP THE TERMINAL NAME AND UNIT
; NUMBER, CONVERTING IT TO RAD50, AND SUPPLYING IT IN THE RPOI DIRECTIVE
; ALONG WITH A DIRECT REQUEST OF PIP IS LEFT AS AN EXERCISE TO THE USER.
; (BE CAREFUL ON RSX-11M-PLUS SYSTEMS WITH MORE THAN 64. TERMINALS).
; NOTE THAT REQUESTING PIP DIRECTLY AVOIDS A SUBSTANTIAL AMOUNT OF SYSTEM
; OVERHEAD.
;
        BR      ISSCMD            ;EXECUTE COMMAND

;
; COMMAND IS NOT 'T' OR 'M', SO GIVE IT TO MCR AS IS.
;
50$:    MOV     #CMDBUF+G.CCBF,R.POBF(R5) ;PUT ADDRESS OF STRING IN DPB
        MOVB    G.CCCT(R0),R.POBL(R5) ;PUT LENGTH IN RPOI DPB

;
; ISSUE RPOI$ DIRECTIVE TO EXECUTE COMMAND
;
ISSCMD: MOV     (R0),R.PODV(R5) ;PUT NAME OF ISSUING DEVICE IN RPOI
        MOVB    G.CCUN(R0),R.POUN(R5) ;COPY UNIT NUMBER
        MOVB    IBUF+G.CCPU,R.POUM(R5)  ;PROPAGATE UIC OF TERMINAL
        MOVB    IBUF+G.CCPU+1,R.POUG(R5)
        MOV     IBUF+G.CCOA,R.POOA(R5)  ;COPY ID PARENT IN CASE WE WERE SPAWNED
        DIR$    #RPOI             ;PASS COMMAND TO EXECUTOR TASK
        BCC     START             ;IF CC, SUCCESS
        BR      110$              ;ERROR
```

```
;
; CONTROL IS TRANSFERRED HERE IF UNKNOWN COMMANDS ARE TO BE REJECTED
; INSTEAD OF BEING PASSED TO MCR.
;
60$:    MOV     #ILL,R1         ;POINT TO MESSAGE TO BE PRINTED
        MOV     #ILLLEN,R2      ;GET LENGTH OF MESSAGE
        CALL    ISSMSG          ;ISSUE MESSAGE
        BR      START           ;TRY FOR NEXT COMMAND

;
; ERROR HANDLING CODE
;

100$:   CLR     R0              ;FATAL ERROR ON GCCI$
        BR      150$
110$:   MOV     #1,R0           ;FATAL ERROR ON RPOI$
120$:   IOT                     ;FATAL

150$:   MOV     #CHLOST,R1      ;POINT TO MESSAGE TO BE PRINTED
        MOV     #CLLEN,R2       ;GET LENGTH OF MESSAGE
        CALL    ISSMSG          ;ISSUE MESSAGE
        BR      START
        .DSABL  LSB

;
; A SYSTEM MESSAGE WAS RECEIVED INSTEAD OF A COMMAND.
;

MSG:    CMPB    #CM.LKT,G.CCUN(R0) ;IS IT A NEW TERMINAL LINKED TO US?
        BEQ     10$             ;IF EQ YES
        CMPB    #CM.RMT,G.CCUN(R0) ;IS IT A TERMINAL BEING REMOVED
        BEQ     20$             ;IF EQ YES
        CMPB    #CM.EXT,G.CCUN(R0) ;IS IT AN ELIMINATE ALL PACKET
        BEQ     50$             ;IF EQ YES
        CMPB    #CM.ELM,G.CCUN(R0) ;IS IT A CLI ELIMINATION MESSAGE
        BEQ     50$             ;IF EQ YES
        BR      START           ;IGNORE ALL OTHER MESSAGES
;
; ISSUE WELCOME MESSAGE
;
10$:    MOV     #LKT,R1         ;POINT TO MESSAGE TEXT
        MOV     #LKTLEN,R2      ;GET LENGTH
        ADD     #G.CCBF,R0      ;POINT TO TERMINAL THAT WAS SET
        BR      30$             ;

;
; ISSUE GOODBYE MESSAGE
;
20$:    MOV     #RMT,R1         ;POINT TO MESSAGE TEXT
        MOV     #RMTLEN,R2      ;GET LENGTH
        ADD     #G.CCBF,R0      ;POINT TO TERMINAL NAME
30$:    CALL    ISSMSG          ;ISSUE MESSAGE
        JMP     START           ;GET NEXT COMMAND
;
; WHEN AN ELIMINATE ALL MESSAGE IS RECEIVED, EXIT IMMEDIATELY.
;
50$:    EXIT$S
```

```
;
; ROUTINE TO PRINT TO TERMINAL
;
;
; INPUTS:
;
;       R0=ADDRESS OF NAME OF TERMINAL TO PRINT TO
;       R1=ADDRESS OF MESSAGE TO PRINT
;       R2=LENGTH OF MESSAGE
;
; OUTPUTS:
;
;       NONE
;
; THIS ROUTINE PRINTS THE SPECIFIED MESSAGE ON THE SPECIFIED TERMINAL.
; ALL ERRORS THAT MIGHT RESULT ARE IGNORED. IN GENERAL, CLIS SHOULD
; DO ASYNCHRONOUS I/O WITH COMPLETION ASTS, AND PROVIDE A TIME-OUT
; FACILITY TO KILL THE I/O IF IT HASN'T COMPLETED IN A CERTAIN
; AMOUNT OF TIME. THIS PREVENTS TERMINAL I/O FROM HOLDING UP
; COMMAND PROCESSING. THIS IS NOT DONE HERE FOR SIMPLICITY.
;

ISSMSG: MOV     (R0),ALUN+A.LUNA  ;SET UP DEVICE NAME
        MOVB    2(R0),ALUN+A.LUNU ;AND UNIT NUMBER
        CLRB    ALUN+A.LUNU+1     ;CLEAR HIGH BYTE
        DIR$    #ALUN             ;ASSIGN LUN 5 TO THE SPECIFIED DEVICE
        BCS     10$               ;IF CS, SKIP I/O
        MOV     R1,QIOW+Q.IOPL    ;SET ADDRESS OF MESSAGE
        MOV     R2,QIOW+Q.IOPL+2  ;INSERT LENGTH OF MESSAGE
        DIR$    #QIOW             ;DO I/O AND WAIT FOR IT
10$:    RETURN

        .END    START
```

11.3.2 FORTRAN CLI

```
C
C THIS TASK (TMCLI) ILLUSTRATES THE USE OF THE CLI CONTROL DIRECTIVES OF
C RSX-11M V4.0 AND RSX-11M-PLUS V2.0.
C
C WHEN SET UP AS A TERMINAL'S CLI, IT PROVIDES THE FOLLOWING COMMANDS:
C
C       T       ->      TYPE FILE
C       M       ->      SET TERMINAL TO MCR
C
C IF THE FIRST TWO LETTERS OF THE COMMAND ARE NOT 'T' FOLLOWED
C BY A SPACE, OR 'M' FOLLOWED BY A CARRIAGE RETURN,
C THE COMMAND IS PASSED TO MCR FOR EXECUTION. THEREFORE,
C USERS AT TERMINALS SET TO THIS CLI WILL SEE THE ENTIRE MCR COMMAND SET
C AS BEING PRESENT, PLUS THE 'T' AND 'M' COMMANDS. IF THE /DPR SUBKEYWORD
C IS INCLUDED ON THE CLI /INIT COMMAND, USERS WILL KNOW IMMEDIATELY IF
C THEIR TERMINAL IS SET TO THIS CLI BY THE DOUBLE RIGHT ANGLE BRACKET PROMPT.
C
```

```
C IF TMCLI IS INITIALIZED WITH THE /MESSAGE KEYWORD, TMCLI WILL
C GREET USERS WHEN THEIR TERMINAL IS SET TO THE CLI, AND SAY GOODBYE
C TO THEM WHEN THEY SWITCH AWAY.
C
C IN ORDER TO BRING THIS CLI UP ON THE SYSTEM, ISSUE THE FOLLOWING
C COMMAND SEQUENCE:
C
C >FOR TMCLI,TMCLI/-SP=TMCLI
C >TKB TMCLI/FP,TMCLI/-SP=TMCLI,[1,1]FOROTS/LB
C >INS TMCLI/CLI=YES/TASK=...TMC
C >CLI /INIT=TMCLI/MESSAGE/DPR="<15><12>/>>/"/TASK=...TMC
C
C IF USING FORTRAN IV-PLUS INSTEAD OF FORTRAN IV, THE FIRST TWO
C COMMANDS ABOVE WOULD BE:
C
C >F4P TMCLI,TMCLI/-SP=TMCLI,FTN
C >TKB TMCLI/FP,TMCLI/-SP=TMCLI,[1,1]F4POTS/LB
C
C TO SET TERMINALS TO THIS CLI, ISSUE THE FOLLOWING COMMAND
C WHERE TTNN: IS REPLACED BY THE TERMINAL NUMBER OR TI:
C
C >SET /CLI=TTNN:TMCLI
C
C NOTE THAT THE PROMPT ISSUED IN RESPONSE TO A CONTROL/C DEFAULTS TO THE
C NAME OF THE CLI, WHICH IS 'TMCLI>'.
C
C TMCLI CAN BE ELIMINATED FROM THE SYSTEM BY THE FOLLOWING COMMANDS, ASSUMING
C NO TERMINALS ARE SET TO IT:
C
C >CLI /ELIM=TMCLI
C >REM ...TMC
C
C EXPERIMENT WITH THE OTHER SUBKEYWORDS OF THE CLI /INIT COMMAND, THE CLI
C /ENABLE AND CLI /DISABLE COMMANDS, AND WITH THE /MESSAGE SWITCH ON THE /INIT
C COMMAND TO OBSERVE THE EFFECTS. ALSO TRY CHANGING THE /TASK=
C SWITCH ON THE INS AND CLI /INIT COMMAND TO "/TASK=TMCLI" AND NOTICE
C THAT ONLY ONE COPY OF THE TASK WILL EXIST FOR ALL TERMINALS, RATHER THAN
C ONE PER COMMAND. IF YOU DO THIS, YOU SHOULD ALSO CHANGE THE OPTION
C ON THE GTCMCI DIRECTIVE FROM GC.CEX TO GC.CST SO THE TASK WILL STAY ACTIVE
C BUT STOPPED WHEN NO COMMANDS ARE OUTSTANDING FOR IT.
C
C
C BUFFER AREAS
C
C    COMMAND BUFFER - THIS BUFFER IS USED IN THE GTCMCI DIRECTIVE.  IT CON-
C                     TAINS INFORMATION ON THE COMMAND RECEIVED (8 BYTES)
C                     AND THE COMMAND TEXT.
        BYTE CMDBUF(264)
        INTEGER*2 CBLEN          ! NOTE THAT ON M+ THE MAXIMUM COMMAND
        DATA CBLEN/263/          ! LENGTH IS 255.
        INTEGER*2 GCCDV, GCCCL   ! ASCII DEV NAM, # CHARS IN CMD LINE
        BYTE GCCCT, GCCUN, GCCBF(255) ! # OF CHARS, TERMINAL UNIT #, CMD
        EQUIVALENCE (GCCDV,CMDBUF), (GCCCL,CMDBUF(5)), (GCCCT,CMDBUF(4)),
        1           (GCCBF,CMDBUF(9)), (GCCUN,CMDBUF(3))
C
```

```
C     INFORMATION BUFFER - ALSO RETURNED IN THE GTCMCI DIRECTIVE.
C
         BYTE IBUF(14)
         INTEGER*2 GCCPU
         DATA IBLEN/14/,GCCPU/9/ ! LENGTH OF IBUF, LOGIN UIC OFFSET
         INTEGER*4 GCCOA         ! ADDR OF OFFSPRING CONTROL BLOCK FROM PARENT
         EQUIVALENCE (GCCOA,IBUF(7))
C
C     BUFFER IN WHICH TO GENERATE COMMANDS
C
         BYTE BUF(72)            ! BUFFER IN WHICH TO GENERATE COMMANDS
         REAL*8 TCMD(2)          ! FOR CONVENIENCE IN ASSIGNMENT STATEMENTS
         EQUIVALENCE (BUF,TCMD)
C
C     MCR TASK NAME IN RAD50
C
         INTEGER*4 IMCR          ! MCR TASK NAME IN RAD50
         DATA IMCR/6RMCR.../
C
C DIRECTIVE SYMBOLIC VALUE DEFINITIONS
C
         INTEGER*2 GCCCS, GCCEX, GCCST, GCCND
         DATA GCCCS, GCCEX, GCCST, GCCND /0, 1, 2, 128/  ! INCP VALUES IN GTCMCI
         DATA ISCLR /0/          ! IS.CLR RETURN CODE
         INTEGER*2 CMELM, CMEXT, CMLKT, CMRMT  ! TERMINAL CODES FOR SYSTEM MSGS
         DATA CMELM, CMEXT, CMLKT, CMRMT /5, 6, 7, 8/
C
C CLI CODE
C
C TO FORCE THE CLI TO EXIT INSTEAD OF RETURNING, THE GC.CEX OPTION IS SPECI-
C FIED IN THE GTCMCI DIRECTIVE BELOW.  IF THE CLI IS INSTALLED SO THAT ONLY
C ONE COPY OF THE TASK WILL EXIST FOR ALL TERMINALS, CHANGE THE VARIABLE
C GCCEX BELOW TO GCCST.  THIS SPECIFIES THE GC.CST OPTION SO THAT THE TASK
C WILL STAY ACTIVE BUT STOPPED WHEN NO COMMANDS ARE OUTSTANDING FOR IT.
C
   1     CALL GTCMCI(CMDBUF,CBLEN,IBUF,IBLEN,0,GCCEX,IDS) ! GET CMD FOR CLI
         IF (IDS .LT. 0) GO TO 100
C
C IF THE GTCMCI DIRECTIVE HAD THE GC.CST OPTION SPECIFIED, WE MAY HAVE
C JUST BEEN UNSTOPPED, SO WE HAVE TO GET THE COMMAND.
C
         IF (IDS .EQ. ISCLR) GO TO 1      ! WERE WE JUST UNSTOPPED?

         IF (GCCDV .EQ. 0) GO TO 200      ! IS THIS A SYSTEM MESSAGE?

         IGCCUN = GCCUN                   ! CONVERT BYTE TO INTEGER (UNIT #)
         CALL ASNLUN(5,GCCDV,IGCCUN,IDS) ! ASSIGN TERMINAL TO LUN 5
         IGCCCT=GCCCT                     ! CONVERT BYTE TO INTEGER
         IF (IGCCCT .GE. 0) GO TO 5
         IGCCCT = 256 + IGCCCT            ! CONVERT NEGATIVE TO UNSIGNED
   5     IF (IGCCCT .LT. GCCCL) GOTO 150 ! DID ENTIRE COMMAND FIT IN BUFFER?
C
C AT THIS POINT, THE CLI HAS A COMMAND TO BE PROCESSED. THE CODE FROM
C THE LABEL 1 TO HERE SHOULD BE FAIRLY COMMON AMONG ALL CLIS.
C FROM THIS POINT ON, THE CODE WILL BE SPECIFIC TO THE PARTICULAR LANGUAGE
C THIS CLI IS IMPLEMENTING. COMMANDS CAN BE ADDED OR DELETED AT WILL
```

```
C BY MODIFYING CODE IN THIS SECTION, AS STATED ABOVE, THIS EXTREMELY
C SIMPLE CLI PROVIDES TWO COMMANDS, AND GIVES EVERYTHING ELSE TO MCR.
C THIS CREATES AN OPEN-ENDED ENVIRONMENT. REJECTING UNRECOGNIZED COMMANDS
C INSTEAD OF PASSING THEM TO MCR WOULD LIMIT THE TERMINAL USER'S USE OF
C THE SYSTEM TO ONLY THOSE OPERATIONS ALLOWED BY THE CLI. THIS IS
C A SECURE WAY OF CONTROLLING SYSTEM USE, IN CASES WHEN IT IS NECESSARY.
C IN THIS EXAMPLE, CHANGING THE "GO TO 50" BELOW, TO A "GO TO 60"
C WOULD CREATE AN ENVIRONMENT WHERE THE ONLY OPERATION A USER COULD
C PERFORM WOULD BE TO TYPE FILES ON THE TERMINAL, OR SWITCH BACK TO MCR.
C ALL OF THE REST OF THE SYSTEM IS INACCESSIBLE. IF NO MEANS OF SWITCHING
C BACK TO MCR IS PROVIDED, IT IS IMPOSSIBLE FOR THE USER TO DO
C ANYTHING THE CLI DOES NOT EXPLICITLY LET HIM DO.
C
C IS IT A "T" COMMAND?
C
      IF ((GCCBF(1) .EQ. 'T') .AND. (GCCBF(2) .EQ. ' ')) GO TO 10
C
C CHANGE THE NEXT IF STATEMENT TO "GO TO 60" TO ELIMINATE PASSING UNKNOWN
C COMMANDS TO MCR.
C
C IS IT ALSO NOT AN "M" COMMAND?
C
      IF ((GCCBF(1) .NE. 'M') .OR. (GCCBF(2) .NE. 13)) GO TO 50 ! M<CR>
C
C THE USER HAS ENTERED AN "M" COMMAND, SO SET THE TERMINAL TO MCR.
C
      TCMD(1) = 'SET /MCR'
      TCMD(2) = '=TI:     '     ! SET /MCR=TI:
      IGRP = IBUF(GCCPU+1)      ! (CONVERT BYTE TO INTEGER) GROUP
      IMEM = IBUF(GCCPU)        ! MEMBER PART OF UIC
C
C ISSUE RPOI$ DIRECTIVE TO EXECUTE COMMAND
C
      CALL RPOI(IMCR,IGRP,IMEM,,TCMD,12,,GCCDV,IGCCUN,,GCCOA,IDS)
      IF (IDS .LE. 0) GO TO 110 ! ERROR?
      GO TO 1                   ! BACK TO START
C
C THE USER HAS ENTERED A "T" COMMAND SO CONSTRUCT THE NECESSARY PIP
C COMMAND AND SEND IT TO MCR
C
 10   TCMD(1) = 'PIP TI:='      ! PUT BEGINNING OF PIP CMD IN BUFFER
      I2 = 8                    ! POINTER INTO TARGET BUFFER
      DO 20, I1 = 10, CBLEN     ! BEGIN AT SPACE AFTER THE T IN CMDBUF
        IF (CMDBUF(I1) .EQ. 13) GO TO 30 ! END OF COMMAND? (<CR>)
        IF (CMDBUF(I1) .EQ. ' ') GO TO 20 ! SKIP ANY SPACES
        I2 = I2 + 1             ! COUNT CHARACTER
        BUF(I2) = CMDBUF(I1)    ! COPY INTO OUTPUT BUFFER
 20   CONTINUE
 30   IGRP = IBUF(GCCPU+1)      ! (CONVERT BYTE TO INTEGER) GROUP
      IMEM = IBUF(GCCPU)        ! MEMBER PART OF UIC
C
C ISSUE RPOI$ DIRECTIVE TO EXECUTE COMMAND
C
      CALL RPOI(IMCR,IGRP,IMEM,,TCMD,I2,,GCCDV,IGCCUN,,GCCOA,IDS)
      IF (IDS .LE. 0) GO TO 110 ! ERROR?
      GO TO 1                   ! BACK TO START
```

```
C
C IN THE ABOVE SECTION OF CODE, THE RPOI DIRECTIVE WAS USED TO PASS
C THE PIP COMMAND TO THE MAIN COMMAND DISPATCHER (MCR...),
C WHICH WILL START A COPY OF PIP AND GIVE IT THE COMMAND. THE RPOI
C DIRECTIVE COULD BE USED TO GIVE THE COMMAND DIRECTLY TO ...PIP AND
C GIVE THE RESULTING TASK A DIFFERENT NAME. IN THIS EXAMPLE,
C WE COULD CALL THE RESULTING PIP TASK 'TYPTNN' WHERE THE LAST
C THREE CHARACTERS (TNN) ARE REPLACED BY A DEVICE NAME CHARACTER AND
C UNIT NUMBER. FOR EXAMPLE, IF THE 'T' COMMAND WAS TYPED ON TT10:
C THE RESULTING TASK NAME COULD BE MADE TO BE 'TYPT10'. THIS HAS THE
C ADVANTAGE OF ALLOWING THE TASK TO BE ABORTED BY AN 'ABO TYP' COMMAND,
C INSTEAD OF 'ABO PIP'; THUS THE USER WOULD NOT HAVE TO KNOW THAT THE
C 'T' COMMAND HE ISSUED IS ACTUALLY BEING SERVICED BY THE PIP TASK. THIS
C IS EXACTLY WHAT THE DCL CLI DOES. IT IS NOT DONE HERE IN ORDER
C TO SIMPLIFY THE EXAMPLE. PICKING UP THE TERMINAL NAME AND UNIT
C NUMBER, CONVERTING IT TO RAD50, AND SUPPLYING IT IN THE RPOI DIRECTIVE
C ALONG WITH A DIRECT REQUEST OF PIP IS LEFT AS AN EXERCISE TO THE USER.
C (BE CAREFUL ON RSX-11M-PLUS SYSTEMS WITH MORE THAN 64. TERMINALS).
C NOTE THAT REQUESTING PIP DIRECTLY AVOIDS A SUBSTANTIAL AMOUNT OF SYSTEM
C OVERHEAD.
C
C
C
C COMMAND IS NOT 'T' OR 'M', SO GIVE IT TO MCR AS IS.
C
  50    IGRP = IBUF(GCCPU+1)     ! (CONVERT BYTE TO INTEGER) GROUP
        IMEM = IBUF(GCCPU)       ! MEMBER PART OF UIC
        IGCCCT = GCCCT           ! # OF CHARS IN COMMAND
C
C ISSUE RPOI$ DIRECTIVE TO EXECUTE COMMAND
C
        CALL RPOI(IMCR,IGRP,IMEM,,GCCBF,IGCCCT,,GCCDV,IGCCUN,,GCCOA,IDS)
        IF (IDS .LE. 0) GO TO 110 ! ERROR?
        GO TO 1                  ! BACK TO START
C
C CONTROL IS TRANSFERRED HERE IF UNKNOWN COMMANDS ARE TO BE REJECTED
C INSTEAD OF BEING PASSED TO MCR.
C
  60    WRITE (5,62)
  62    FORMAT('0ILLEGAL COMMAND')
        GO TO 1                  ! BACK TO START
C
C ERROR HANDLING CODE
C
 100    WRITE (5,102) IDS        ! ERROR IN GTCMCI
 102    FORMAT('0FATAL ERROR IN GTCMCI DIRECTIVE.  DSW=',I6)
        GO TO 250                ! EXIT
C
 110    WRITE (5,112) IDS        ! ERROR IN RPOI
 112    FORMAT('0FATAL ERROR IN RPOI DIRECTIVE.  DSW=',I6)
        GO TO 250                ! EXIT
C
 150    WRITE (5,152)            ! ENTIRE COMMAND DID NOT FIT IN BUFFER
 152    FORMAT('0SORRY, PART OF YOUR COMMAND WAS LOST')
        GO TO 1                  ! BACK TO START
```

```
C
C A SYSTEM MESSAGE WAS RECEIVED INSTEAD OF A COMMAND
C
 200    CONTINUE
        IF (GCCUN .EQ. CMLKT) GO TO 210 ! IS IT A NEW TERMINAL LINKED TO US?
        IF (GCCUN .EQ. CMRMT) GO TO 220 ! IS IT A TERMINAL BEING REMOVED?
        IF (GCCUN .EQ. CMEXT) GO TO 250 ! IS IT AN ELIMINATE ALL PACKET?
        IF (GCCUN .EQ. CMELM) GO TO 250 ! IS IT A CLI ELIMINATION MESSAGE?
        GO TO 1                 ! IGNORE ALL OTHER MESSAGES
C
C ISSUE WELCOME MESSAGE
C
 210    IGCCUN = CMDBUF(11)     ! CONVERT BYTE TERMINAL UNIT # TO INTEGER
        CALL ASNLUN(5,GCCBF,IGCCUN,IDS) ! ASSIGN LUN 5 TO SPECIFIED DEVICE
        WRITE (5,212)
 212    FORMAT('0WELCOME TO THE EXAMPLE CLI')
        GO TO 1                 ! BACK TO START
C
C ISSUE GOODBYE MESSAGE
C
 220    IGCCUN = CMDBUF(11)     ! CONVERT BYTE TERMINAL UNIT # TO INTEGER
        CALL ASNLUN(5,GCCBF,IGCCUN,IDS) ! ASSIGN LUN 5 TO SPECIFIED DEVICE
        WRITE (5,222)
 222    FORMAT('0GOOD BYE FROM THE EXAMPLE CLI')
        GO TO 1                 ! BACK TO START
C
C WHEN AN ELIMINATE ALL MESSAGE IS RECEIVED, EXIT IMMEDIATELY
C
 250    CONTINUE
        CALL EXIT
        END
```

CHAPTER 12

THE DCL TASK

The DIGITAL Command Language (DCL) is a general purpose command language syntax implemented on several DIGITAL operating systems.

12.1 INTRODUCTION

DCL on RSX-11M/M-PLUS systems is a task that receives DCL commands by means of the generalized CLI command dispatcher, translates the DCL commands to equivalent MCR commands, and sends the MCR commands to the MCR command line interpreter for interpretation and execution. The DCL task, in conjunction with the CLI dispatcher, provides DCL support on RSX-11M/M-PLUS operating systems.

This chapter explains the ways in which DCL can be installed on your system and documents the means by which the DCL task parses and translates commands. In addition, this chapter documents the Macro MetaLanguage, a set of MACRO-11 macros used to define DCL. An experienced programmer can add commands to DCL or remove them, or alter DCL by changing parts of it, such as altering defaults or eliminating qualifiers.

For information on DCL syntax and functions, see the RSX-11M/M-PLUS Command Language Manual. For information on MCR syntax and functions, see the RSX-11M/M-PLUS MCR Operations Manual. For information on the CLI command dispatcher, and on alternate CLIs, see Chapter 11, Command Line Interpreters, in this manual. For information on the Executive directives used to implement alternate CLIs, see the RSX-11M/M-PLUS Executive Reference Manual.

The examples of source code in this chapter illustrate features of DCL and the DCL task, but they are often changed from the actual source code to make the example clearer. See the DCL source code in [23,10] for exact information. The command tables for the ASSIGN command are included in this chapter (Section 12.11.6) as a general example of DCL coding.

12.2 INSTALLING AND BUILDING DCL

The DCL task is best installed as an alternate CLI, but it can also be run as a task or installed as an external MCR task. External MCR tasks have task names in the form ...abc. Furthermore, you can rebuild DCL either to eliminate direct access to MCR from DCL, or to allow all commands not in DCL syntax to fall through DCL for handling by MCR. Your choice on these options depends on the nature of your users and your installation.

12.2.1 Conventional Installation of DCL as a CLI

Installing DCL as an alternate CLI is the cleanest use of DCL. This allows you to classify users by CLI in the system account file. Users can log in using either HELLO or LOGIN. The HELLO task checks the account file for the CLI and sets the user's terminal to the designated CLI.

On RSX-11M systems, DCL and alternate CLI support require a mapped system with multiuser protection. If you choose DCL at system generation, you are also choosing alternate CLI support, as DCL is an alternate CLI. In addition, choosing DCL support also gives you support for the following directives:

- Stop-bit
- Install-request-remove
- Parent-offspring tasking with chaining
- AST
- Specify request exit AST
- Get task parameters
- Get command for command interpreter
- Request and pass offspring information
- Specify command arrival AST
- Get command interpreter information

The last four directives listed are those specifically added for alternate CLI support. See Chapter 11, "Command Line Interpreters," and the RSX-11M/M-PLUS Executive Reference Manual.

During system generation on RSX-11M-PLUS, you are asked if you want alternate CLI support or DCL support. The other (non-CLI) directives listed above are supported on all RSX-11M-PLUS systems with the full-function Executive.

Installing DCL as an alternate CLI on either RSX-11M or RSX-11M-PLUS, requires only two commands:

```
MCR>INSTALL $DCL.TSK/CLI=YES
MCR>CLI /INIT=DCL
```

When you choose DCL support during system generation, the INSTALL command is issued in the SYSVMR portion of system generation and the CLI /INIT command must be included in the system STARTUP.CMD file.

12.2.2 Alternative Ways to Use DCL

If, for some reason, you choose not to have alternate CLI support, you can still use many of the features of DCL by taking advantage of alternative ways of installing DCL. You can run DCL as a task, install DCL as an external MCR task under the name ...DCL, or install DCL as the system catchall task under the name ...CA.

If DCL is not a CLI, you cannot log in with LOGIN, you cannot change your defaults with SET DEFAULT, breakthrough writes do not break through, and spawned tasks are not renamed. That is, a DIRECTORY command spawns a task named PIPTnn, not DIRTnn.

12.2.2.1 **Run DCL as a Task** - If you wish, you can simply run DCL as a task. If you place the task image file in the system directory on RSX-11M or the library directory on RSX-11M-PLUS, users can run DCL with the command:

```
>RUN $DCL
```

DCL runs from your terminal as a task. The terminal remains set to MCR. All DCL commands work in the normal way, but a carriage return on a line by itself causes the task to exit.

12.2.2.2 **Install as ...DCL** - If you install the DCL task as an external MCR task under the name ...DCL, users can issue DCL commands from an MCR terminal simply by preceding the command with DCL, as in the following example:

```
MCR>DCL SHOW USERS
```

12.2.2.3 **Install as ...CA.** - If you install the DCL task under the name ...CA., all commands not recognized by MCR are handled by DCL running as the catchall task. This means that DCL commands such as MOUNT and SET that are identical with MCR commands in the first three characters must be in MCR syntax. These commands in DCL syntax are recognized by MCR, but only as errors. To avoid infinite loops, the DCL installed as MCR's catchall must not be built with MCR as its catchall.

12.2.3 The MCR Command

The MCR command is included in the default DCL task. This command permits users to issue a one-line MCR command without leaving DCL. The MCR command is not affected by SET DEBUG.

Some installations may prefer not to permit DCL users any access to MCR. If you wish to restrict access to MCR, you must modify DCLBLD.CMD following the instructions contained therein and then rebuild DCL.

12.2.4 Allowing Unrecognized Commands to Fall Through to MCR

The default DCL task returns an error for any command not recognized as a DCL command. You may prefer that DCL allow any unrecognized commands to fall through to MCR for another chance at execution. This permits users direct access to MCR tasks and utilities without using the MCR command or the RUN command.

If you wish to allow commands to fall through DCL to MCR, modify DCLBLD.CMD following the instructions contained therein and then rebuild DCL.

12.2.5 Flying Installs

Tasks need not be installed for DCL to access them. Comments in DCLBLD.CMD explain how to enable flying installation of tasks. Again, modify DCLBLD.CMD and rebuild DCL.

12.2.6 Task-Building DCL

All of the changes to DCL described in this chapter require that the DCL task be rebuilt. This task build actually requires two passes by the Task Builder before it is complete. The first pass creates a symbol table (.STB file) that is used by TKB on the second pass to build the DCL task.

The file [1,24]DCLBLD.CMD is called by SYSGEN when you request DCL support. You should read this file before attempting to task-build DCL.

The overlay structure of the DCL task forces a complex task build. There is more detail about how the DCL task works in the next section, but here is an outline of the situation. Each DCL command is defined in a parser table, or PT, which defines the DCL syntax of the command. The PT is paired with a translator table, or TT, which defines the MCR equivalents for the DCL command. The addresses of the PT and TT are included in a master command table (MCT) that lists all DCL commands. When DCL parses a command, it seeks the address of that command in the MCT and then goes to that address for translation of the command to MCR at the PT/TT. The parser tables and translator tables are overlaid on the master command table. This means that at the time of the first task build, the master command table entries are undefined symbols. The following summarizes the process.

The first task build uses a command in the following form:

```
>TKB ,,DCL.STB=DCL
```

This results in one or more undefined symbols for each command, but the PT/TT addresses for each command are in the .STB file.

The second task build uses a command in the following form:

```
>TKB DCL,DCL=DCL,DCL.STB
```

The master command table symbols are defined by means of the .STB and the addresses in the PT/TT are doubly defined, once in the .STB and once in their overlays. The two definitions are identical.

In fact, this process is accomplished through two .ODL files. See DCLBLD.CMD for more information.

12.2.7 How the DCL Task Works

There are three processes involved in interpreting DCL commands -- the receive process, the parser/translator process, and the send process. The receive process gets the command line from the system. The table-driven parser/translator process translates the command line to an equivalent MCR command line. The send process sends the equivalent MCR command line to the system for further interpretation and execution.

The receive process involves two tasks: the CLI dispatcher task (MCR...) and the DCL task (...DCL). The CLI dispatcher task gets the command line from the terminal driver, queues it to a CLI queue, and activates the DCL task (if not already active). The DCL task activated is named DCLTnn, where nn is the number of the issuing terminal. The DCL task (being nonprivileged) issues the Get Command for Command Interpreter (GCCI$) directive to remove (or receive) the command line from the CLI queue.

The parser/translator process involves two subprocesses: the parser process and the translator process. The parser process checks syntax to ensure that the received command line is a valid DCL command line. The parser constructs an intermediate form (/FORM) to pass data about the command line to the translator. The translator process uses this data to construct an equivalent MCR command line from the valid DCL command line.

The send process dispatches the equivalent MCR command line to MCR or MCR utility tasks by means of the Request and Pass Offspring Information (RPOI$) directive. The RPOI$ directive is functionally similar to the SPWN$ directive. There are two significant differences from SPWN$: RPOI$ allows the spawned task to be named (for example, spawn PIP and name it COPTnn), and RPOI$ does not create OCBs (Offspring Control Blocks).

The DCL task needs the task-naming ability to track the association between the DCL command name and the MCR or utility task that executes the command. And, instead of creating an OCB, RPOI$ takes the OCB attached to the DCL task and attaches this OCB to the spawned task. This enables the spawned task to communicate exit status directly to the parent task, DCL. In this way, the DCL task can spawn a task without having to wait for the task to exit.

The remainder of this document describes how to construct the DCL command tables, which drive the parser/translator process. Basically, the tables contain, for each DCL command, the syntax of the DCL command, the syntax of the equivalent MCR command (that is, the syntax of the MCR command that the DCL command translates to), and the relation (or mapping) between the DCL syntax and the MCR syntax. MACRO-11 macros and rules define the tables. The macros and rules form a language designed to allow the user to express the tables.

The macros and rules are called the Macro MetaLanguage (MML). Here is a list of some of the features of the language and how they are employed in the DCL task:

- MML is modeled after a Backus-Naur Form (BNF) metalanguage. A BNF language specifies which sequences of symbols constitute a syntactically valid statement in a given language. In this case, MML is used to describe the proper syntax of a DCL command and the equivalent syntax of an MCR command.

- MML allows the specification of a prompt string for each syntax element. DCL prompts for more input, with the specified string, if a test for end-of-line is successful.

- MML interfaces with the RSX-11M/M-PLUS HELP facility. The user requests help by typing a "?" character in reply to a prompt.

- MML allows the specification of an error message string for each syntax element. If DCL detects an error while processing a syntax element and that syntax element has a specified error message, DCL issues the error message and aborts the parse.

- MML allows the specification of contradictory bit masks for each syntax element. DCL uses the bit masks to detect contradictory syntax.

- MML allows the use of globally defined syntax elements, such as filespecs, device specs, numerals, letters, alphanumerics and so forth.

- MML allows MACRO-11 Assembly Language subroutines to be mixed with MML statements in the event that a command cannot be completely defined using MML.

- MML allows one DCL command to translate to more than one MCR command. For instance, the DCL SET DEFAULT command can generate both an MCR ASN command and an MCR SET /UIC command.

- MML supports floating qualifiers. A floating qualifier can be specified anywhere in the command line.

The following section describes the DCL command tables. The DCL command tables define all DCL commands and are written in MML. See Sections 12.6 through 12.10 for more information on MML.

12.3 DCL COMMAND TABLES

The parser/translator process of DCL is table-driven. A master command table (MCT) contains an entry for each DCL command. Each entry in the MCT contains the name (verb) of the command, the address of a parser table (PT) for the command, the address of a translator table (TT) for the command, and, optionally, the name of the overlay containing the PT and the TT. Thus, there is a PT and a TT for each DCL command. Each PT/TT contains the information that the parser/translator process needs to translate a DCL command line to an equivalent MCR command line. For ease in adding and modifying commands, MCT and each PT/TT are kept in separate files.

The file for the MCT is called COMMAND.MAC. The files for the PT/TTs are named after the DCL command, with the type .MAC. For example, the PT/TT for the DCL COPY command is in a file named COPY.MAC.

The DCL task uses MCT to recognize an arbitrary command and to locate the PT/TT for that command. MCT consists of calls to the COMMAND macro. There is a COMMAND macro for each DCL command. Arguments to a COMMAND macro specify the name (or verb) of the command, the address of the PT for the command, the address of the TT for the command, and the overlay name for the command.

The PT/TTs drive the parser/translator process. The parser/translator process ensures that the syntax of the DCL command line is correct and creates the /FORM, which is the parser subprocess, and subsequently contructs equivalent MCR command lines, which is the translator subprocess. The PT drives the parser subprocess and the TT drives the translator subprocess. The PT is an MML definition of the syntax or structure of the DCL command and the TT is an MML definition of the syntax or structure of the corresponding MCR command.

Table 12-1 shows the structure of the DCL command table files.

The PT defines the syntax for a DCL command line. The TT defines the syntax for the corresponding MCR command line. For the translator to know which particular MCR command line to contruct from a particular DCL command line, the PT and the TT must express a relation between the DCL syntax and the MCR syntax.

Table 12-1
DCL Command Tables

COMMAND.MAC	ABORT.MAC	UNLOCK.MAC
Master Command Table COMMAND ABORT ABOPT ABOTT COMMAND ASSIGN ASSPT ASSTT . . . COMMAND UNLOCK UNLPT UNLTT	ABOPT:: Parser Table ABOTT:: Translator Table	UNLPT:: Parser Table UNLTT:: Translator Table

The PT/TT expresses this relation by mapping elements from DCL syntax to elements in MCR syntax. For example, the DCL qualifier "/GLOBAL" maps to the MCR switch "/GBL" (that is, the switch "/GBL" is output when the qualifier "/GLOBAL" is input).

The parser passes the mapping information to the translator by means of a dynamic data structure called the Intermediate Form (IFORM). The parser constructs the IFORM and the translator examines the IFORM. The IFORM contains the essential information that the translator needs to contruct the equivalent MCR command line. Both the PT and the TT define the mapping from DCL to MCR syntax.

The Macro MetaLanguage (MML) defines the PT and the TT. MML is a set of MACRO-11 macros and rules implementing a language in which a user can define DCL commands. With MML, a user can express the information in the PT and the TT in a BNF-type metalanguage.

The following section describes the structure of the file that contains the MCT and of the files that contain the PT/TTs. It also describes how these files assemble, overlay, and task build into the DCL task. Section 12.4.1.1 describes how to create the PTs and TTs.

12.4 MODIFYING DCL COMMAND TABLES

A DCL command consists of an entry in the master command table (MCT), a parser table (PT), and a translator table (TT). The PT defines the syntax of the DCL command. The TT defines the syntax of the corresponding MCR command. The entry in the MCT connects the PT and the TT to the DCL task. Thus, to add a new command to the DCL task the user must construct a PT and a TT and make an entry in the MCT.

12.4.1 Adding Commands

Adding a command to the DCL task involves:

1. Creating the PT and the TT
2. Creating the file that contains the PT and the TT
3. Making an entry in the MCT

4. Making an entry in the overlay description file for the DCL task

5. Incorporating the new PT/TT file and the modified MCT file into the DCL task

12.4.1.1 **The PT/TT File** - The file that contains the PT and the TT is named after the verb of the command. The file type is .MAC (for example, the PT and the TT for the COPY command are in a file named COPY.MAC). The PT/TT file must specify the MACRO-11 assembly directives .TITLE, .PSECT, and .END. The title of the file (specified as an argument to the .TITLE directive) must also be the same as the verb of the command. Normally, the program-section name is .CDEF, which stands for command definition. If, however, MACRO-11 subroutines are interspersed with MML notation by means of the $JSR predefined syntax element, the subroutines must be in the blank program section. (See Section 12.11 for details on $JSR.)

In addition to these directives, the PT/TT file must contain a (MACRO-11) global label on the first MML macro of the PT and a global label on the first MML macro of the TT. By convention, the label on the PT is the same as the verb of the DCL command, and the label on the TT is the same as the verb on the corresponding MCR command. The following shows the structure of the file that contains the PT and the TT for the DCL ASSIGN command.

ASSIGN.MAC

```
        .TITLE  ASSIGN
        .PSECT  .CDEF

ASSIGN::

        Parser Table

ASN::

        Translator Table

        .END
```

The two global labels in the PT/TT file connect the PT and the TT to the DCL task. This is accomplished by making an entry in the MCT.

12.4.1.2 **The MCT Entry** - The file COMMAND.MAC contains the master command table (MCT). The MCT consists of a list of (MACRO-11) COMMAND macrocalls. Each COMMAND macrocall identifies a DCL command. The list of COMMAND macrocalls is ordered alphabetically by the name or verb of the command. The COMMAND macro takes five arguments. The first four are mandatory and the fifth is optional. The format for the COMMAND macro is as follows:

```
COMMAND   name, min, ptaddr, ttaddr, [ovrnam]
```

where

name — Name or verb of the command

min — Minimum number of characters necessary to make the name unique

ptaddr Label of the PT for the command

ttaddr Label of the TT for the command

[ovrnam] Three-character name of the overlay for the PT and TT

The fifth argument to the COMMAND macro (ovrnam) is optional. The PT and TT must, however, be in an overlay whether or not you specify this argument. The default overlay name is the command verb, assuming the PT and TT are in the same module. If they are in separate modules, the PT overlay is named for the verb by default, and the TT is named for the verb preceded by a "T" (for example, TSET.MAC).

The MCT is in two parts. Part 1 of the MCT includes all DCL command verbs in alphabetical order, as described in this section. Part 1 is searched with a binary search algorithm. Part 2 of the MCT makes possible the brief forms of commonly used DCL commands. If the search of Part 1 for a command verb fails -- if a line is too short, for instance -- a search of Part 2 is initiated. Part 2 is not necessarily in alphabetical order as it is searched from the top down.

12.4.1.3 **The Overlay Entry** - Each PT/TT must reside in an overlay. Use one of the following structures to overlay a PT/TT:

1. One overlay for two or more PT/TTs
2. One overlay for one PT/TT
3. Two overlays for one PT/TT
4. More than two overlays for one PT/TT

Specify an overlay by making an entry in the overlay description files -- DCLBLD.ODL and DCDBLD.ODL -- and connect the new overlay entry to the root segment overlay entry. For the first type of overlay structure previously noted (one overlay for two or more PT/TTs), the format for the overlay entry is as follows:

```
        .NAME   ovrnam
label:  .FCTR   ovrnam-DCLO/LB:cmdnam1:cmdnam2...:cmdnamn-L
```

where

label Connects the entry to the root segment

ovrnam Three-character name of the overlay

DCLO Object library containing the modules in the overlays

cmdnam1 Title of the file containing the first PT/TT

cmdnam2 Title of the file containing the second PT/TT

.
.
.

cmdnamn Title of the file containing the last PT/TT

For each entry in the MCT whose command tables (PT/TTs) are in an overlay that has two or more PT/TTs, you must include the overlay name, ovrnam.

For the second overlay structure previously noted (one PT/TT per overlay), the entry has the following format:

```
label:  .FCTR DCLO/LB:cmdnam-L
```

where

label Connects the entry to the root segment

cmdnam Title of the file that contains the PT/TT

This structure, one overlay and one PT/TT, does not specify the overlay name. In this case, the DCL task uses the name or verb of the command as the overlay name since the Task Builder default for the overlay name is the title of the file. For this reason, the title of the file containing the PT/TT must be the same as the name or verb of the command.

For the third overlay structure (two overlays and one PT/TT), the entry has the following format:

```
label:  .FCTR (DCLO/LB:cmdnam-L,DCLO/LB:tcmdnam-L)
```

where

label Connects the entry to the root segment

cmdnam Title of the file that contains the PT

tcmdnam Title of the file that contains the TT

Note that this overlay structure, two overlays and one PT/TT, requires that the PT and TT reside in separate files. The separate files have the same format as a single file that contains the PT/TT except that there is only one global label in each file. By convention, the file containing the PT is named for the verb of the command and the file containing the TT is named for the verb prefixed by "T". The titles of the files follow the same convention. For example the file containing the PT for the DCL SET command is named SET.MAC with title SET, and the file containing the TT for the DCL SET command is named TSET.MAC with title TSET.

For the fourth overlay structure (more than two overlays and one PT/TT), the entry has the following format:

```
label:  .FCTR
        (DCLO/LB:cmdnam-L,DCLO/LB:tcmdnam-L,DCLO/LB:ovrnam1-L)
```

where

label Connects the entry to the root segment

cmdnam Title of the file that contains the PT

tcmdnam Title of the file that contains the TT

ovrnam1 Title of a file containing more parts of the PT/TT

Basically, the overlay structure of more than two overlays and a single PT/TT allows a PT or a TT to invoke additional overlays containing additional portions of the PT or TT. Since the PT and TT are read-only data, the additional overlay segments are loaded on top of the overlay that is invoking them. On return from the invoked

overlay, the DCL task reloads the previous overlay. Control returns to the point in the previous overlay whence the additional overlay was invoked.

Use the $CALLOV predefined syntax element to invoke additional overlays from a PT or a TT. (See Section 12.11, for details on the $CALLOV predefined syntax element.) To specify additional overlays, repeat the format above. That is, for each additional overlay, specify an additional argument to the .FCTR directive using the format: DCLO/LB:ovrnam-L.

Next, connect the new overlay entry to the root segment of the DCL task by adding the label of the entry to the list of labels for other entries. The list of labels for other entries is found in the overlay description file after the label OLAY. By convention, enter the labels in alphabetical order. The following shows the list of overlay entry labels:

```
OLAY:   .FCTR (l1,l2,l3,l4,l5,...,ln)
```

where

l1,...,ln Labels of overlay entries

Note that since there are two overlay description files for building the DCL task (DCLBLD.ODL and DCDBLD.ODL), you must make new overlay entries in both files. The format is the same for both overlay description files. For details on how to build the DCL task, see Section 12.2. For details on overlay structures, see the RSX-11M/M-PLUS Task Builder Manual.

The last step required to add a command to the DCL task is to incorporate all created or modified files into the DCL task.

12.4.1.4 **Incorporating the PT/TT and MCT** - The last step in adding new commands to DCL -- assembly and task building -- is accomplished through the indirect command file [23,24]DCL.CMD, which is on the same disk as the DCL sources. This file prompts for the name of a file containing the names of all modules you have changed or created. The input file consists of one module name per line, with no file type or version number. DCL.CMD then assembles the modules, updates the object libraries, and task-builds the new version of DCL.

Invoke DCL.CMD as follows:

```
>@DCL
*Do you want expanded comments;  [Y/N]:
```

Answer Y to this question. The comments in DCL.CMD direct the remainder of the process.

If all source files assemble correctly, DCL.CMD then builds the DCL task. See Section 12.2 for information on how to determine if the DCL task was built correctly and for information on how to install DCL as a CLI in a multi-CLI environment.

The DCL.CMD command file is designed to run under MCR with MAC, LBR, PIP, and TKB installed. DCL.CMD is designed to run using one UFD, the UFD that you invoke the command file from. This means that the following files must be in this UFD:

- Specified source files

- Library files for DCL: DCLO.OLB and DCLR.OLB, the overlay and root, respectively,
- Prefix files for DCL: DCLMAC.MAC and COMMAC.MAC
- Task-build command file: DCLBLD.CMD
- Overlay description files: DCLBLD.ODL and DCDBLD.ODL

Additionally, RSXMC.MAC must be in LB:[11,10]. Although DCL is nonprivileged, DCL needs this file to determine, at assembly time, whether your system is RSX-11M or RSX-11M-PLUS.

12.4.2 Deleting Commands

To delete a command from DCL, you must break the link between the MCT and the PT/TT. To do this, delete the appropriate COMMAND macro entry in the MCT, which is in the file COMMAND.MAC. Then incorporate the modified COMMAND.MAC file into the DCL task by following the procedure outlined in the preceding section. You can also delete the entry in the overlay description files and the object modules for the PT/TT in the DCLO.OLB library, but this is optional.

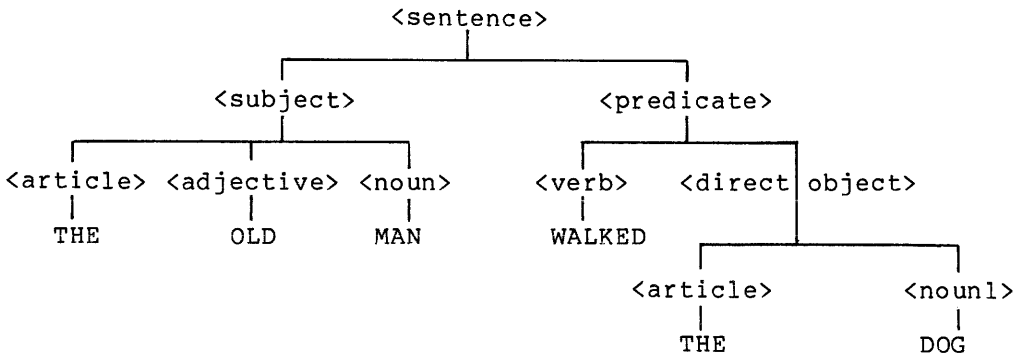
12.4.3 Modifying Commands

Modifying an existing DCL command involves modifying the PT/TT files to reflect the desired change and incorporating the modified files into the DCL task, as described in Section 12.4.1.

The user defines DCL PTs and TTs using MML. The following section describes a metalanguage in general terms, using a simple English sentence as the example. Following the introduction of these general concepts in Section 12.5 is a more detailed exposition of the Macro MetaLanguage (MML) itself in Section 12.6. For instance, the general introduction in Section 12.5 includes a description of ANDing and ORing in metalanguages, while Section 12.6 describes the AND and OR macros of the Macro MetaLanguage.

12.5 A GENERAL DESCRIPTION OF A METALANGUAGE

Consider a sentence in the English language. A syntax tree defines the structure or syntax of a sentence as shown below:



Note that the sentence "The old man walked the dog" is one of many sentences that could fit the above structure (syntax). The following description of the syntax tree is a metalanguage definition of the sentence:

```
sentence        ::= <subject> <predicate>

subject         ::= <article> <adjective> <noun>

predicate       ::= <verb> <direct object>

direct object   ::= <article> <noun1>

article         ::= THE

adjective       ::= OLD

noun            ::= MAN

verb            ::= WALKED

noun1           ::= DOG
```

Note the equivalence of the metalanguage definition and the syntax tree. They describe the same thing. Note also that the metalanguage definition uses the symbols "<", ">", and "::=".

The symbols "<" and ">" denote category references. These symbols enclose the name of the category. Categories define classes of objects. For example, the category <subject> defines English words that fit the structure of an <article> followed by an <adjective> followed by a <noun>.

The symbol "::=" means "is defined as" and denotes a production rule.

In a production rule, you specify the category name to the left of the "::=" symbol and the definition for the category to the right of the "::=" symbol. A category reference means there is a production rule for the category that requires that the right part of the production rule be substitued for the name of the category. For example, here is the production rule for <subject>:

```
subject ::= <article> <adjective> <noun>
```

that references the category <adjective>. The production rule for the category <adjective> is:

```
adjective ::= OLD
```

Thus, you can rewrite the production rule for <subject> as follows:

```
subject ::= <article> OLD <noun>
```

Note that you can also substitute further for references to the categories <article> and <noun> to produce the production rule:

```
subject ::= THE OLD MAN
```

You cannot substitute for the words: THE, OLD, and MAN. These words are literal strings. For simplicity, literal strings are called literals, and category references are called nonliterals. A category definition (the right part of a production rule) consists of literals and nonliterals, or both, concatenated by implied-ANDs.

The implied-AND has no symbol in the metalanguage (thus the term implied). Use the implied-AND to form category definitions by concatenating arbitrary numbers of syntax elements. A syntax element is either a literal or a nonliteral. For example, in the category definition for <subject> are three nonliterals -- <article>, <adjective>, and <noun>. An implied-AND concatenates these nonliterals.

Two additional notations in the metalanguage are:

Exclusive-OR indicator |

Optional syntax indicators []

Use the exclusive-OR indicator "|" to show that a category can be defined in several ways. For example, here is the production rule for the category <adjective>.

```
adjective ::= OLD | YOUNG | MIDDLE-AGED
```

When substituting for a nonliteral with more than one definition (separated by "|"), use only one of the definitions in the substitution. For example, substituting for the nonliteral <adjective> defined in this way in the production rule for <subject> yields one of the following three production rules:

```
subject ::= THE OLD MAN

   or

subject ::= THE YOUNG MAN

   or

subject ::= THE MIDDLE-AGED MAN
```

The square brackets, "[" and "]", enclose optional syntax. For example, the following production rule defines the category <subject>.

```
subject ::= <article> [ <adjective> ] <noun>
```

The square brackets around the nonliteral <adjective> indicate that <adjective> is optional syntax. This would allow, by means of substitution, rewriting the production rule for <subject> to either:

```
subject ::= <article> OLD <noun>

   or

subject ::= <article> <noun>
```

In addition to denoting optional syntax, the square brackets also indicate iteration. That is, syntax denoted as optional can also be used repeatedly to define the syntax. For example, if the following defines <adjective>:

```
adjective ::= MEAN | OLD | FAT
```

then you could rewrite the production rule for <subject> (if <subject> is defined with "[" and "]" as above) to:

```
subject ::= <article> MEAN OLD FAT <noun>
```

Note, the repeated substitution for the nonliteral <adjective>.

Thus, you can use the symbols, rules, and logic described above as a language (that is, as a metalanguage) to define the structure of other languages. The structure or syntax of a language is called the grammar of that language. The grammar is used to determine mechanically whether an arbitrary sentence is a sentence in the language.

Our interest here is in the grammars for DCL and MCR, the RSX-11M/M-PLUS command languages. Sentences in these two grammars are called commands. Here is a metalanguage definition for the grammar of the DCL ASSIGN command:

```
assign ::= ASSIGN [ / <qual> ] <device> <device>

qual ::= LOCAL | LOGIN | GLOBAL | TERMINAL:  <device>
```

Here is a metalanguage definition for the grammar of the MCR ASN command:

```
asn ::= ASN <device>=<device> [ / <qual> ]

qual ::= LOGIN | GBL | TERM= <device>
```

The following section describes the Macro Metalanguage (MML). MML implements the logic of the metalanguage above. MML defines the grammars for the DCL and MCR command languages. Additionally, MML expresses the mapping from DCL grammar to semantically equivalent MCR grammar. MML provides a more efficient means of generating (by means of the MACRO-11 assembler) the data structures that drive the parser/translator process than the metalanguage notation.

12.6 THE MACRO METALANGUAGE

The Macro MetaLanguage (MML) is a set of MACRO-11 macros and rules. The following describes the correspondence of the metalanguage notation presented in the preceding section to MML notation.

The general metalanguage presented in the preceding section supports two logical operators:

implied-AND and exclusive-OR

The Macro MetaLanguage has two MACRO-11 macros to correspond to these logical operators:

AND and OR macros

The general metalanguage presented in the previous section supports two types of syntax elements:

literals and nonliterals

In MML, literals and nonliterals are presented as arguments to the AND and the OR macros.

In MML, literals are distinguished from nonliterals by making the first character a quote character (for example, 'ASSIGN). An optional argument that you can specify with a literal is the abbreviation count. The abbreviation count is a numeral that indicates the minimum number of characters that the parser needs to match for a successful parse. For example you could rewrite

```
'ASSIGN -- to -- <'ASSIGN,4>
```

to indicate that the literal "ASSIGN" can be abbreviated to "ASSI".

NOTE

> The use of the symbols "<" and ">" indicates (to MACRO-11) that the argument "'ASSIGN,4" is one argument (not two: ASSIGN and 4). Do not confuse these symbols with the identical metalanguage symbols used to identify nonliterals.

In the general metalanguage presented in the previous section, a nonliteral is a reference to a category. You define a category with the "::=" symbol by specifying the name of the category to the left of the symbol and the definition for the category to the right of the symbol.

In MML, you define a category by placing a MACRO-11 label on either the AND or the OR macro. The label is the name of the category, and the arguments to the macro are the definition for the category. Thus, in MML, a nonliteral is the label (of an AND or OR macro) that you use as an argument to an AND or OR macro.

For example, the metalanguage notation for the category <qual> is:

```
qual ::= GLOBAL | LOCAL | TERMINAL:  <device>
```

The equivalent MML notation is:

```
QUAL:  OR 'GLOBAL 'LOCAL TERM

TERM:  AND 'TERMINAL:  $DEV
```

There are two categories -- QUAL and TERM. There are three category definitions for QUAL: 'GLOBAL, 'LOCAL, or TERM. 'GLOBAL and 'LOCAL are literals and TERM is a nonliteral. The category definition -- 'TERMINAL: $DEV -- defines the category TERM, which consists of a literal 'TERMINAL and a predefined nonliteral, $DEV. (See Section 12.11 for a description of $DEV).

Note that you must express a category definition like 'TERMINAL: $DEV on a separate line (using the AND macro) because an argument to either the AND or the OR macro can only represent one syntax element -- either a literal or a nonliteral.

Use the MML notation "OPT=T" (optional=true) to indicate that syntax elements are optional. This notation corresponds to the metalanguage notation "[" and "]". The optional syntax notation "OPT=T" is an attribute of syntax elements (literals and nonliterals). You represent literals and nonliterals in MML notation as arguments to AND and OR macros. Therefore, you specify the optional syntax indicator "OPT=T" as part of the arguments.

To specify "OPT=T" as an attribute of a syntax element, enclose the syntax element and the "OPT=T" notation in angle brackets. For example, to specify that the terminal device specification ($DEV) in the previous example is an optional syntax element (nonliteral), rewrite the category TERM as follows:

```
TERM:  AND 'TERMINAL:  <$DEV OPT=T>
```

Use the "OPTR=T" attribute to indicate that a syntax element is optional and to indicate that the parser/translator can process the syntax element repeatedly.

The notation OPT means a syntax element can occur, at most, one time. The notation OPTR means a syntax element can occur many times. In either case, the element can occur zero times.

For a complete description, see Section 12.10.

To specify an arbitrary number of arguments for the AND and OR macros, use the OR., AND., and the END macros. Use the AND. and OR. macros as continuation markers for the AND and OR macros respectively. That is, if there is not enough space on the line for all the arguments to either the AND or OR macros, place additional arguments on the next line. To do this, place the AND. or the OR. macro on the next line followed by the continuation of the specification of the arguments. For example, you could rewrite the category for <qual> to:

```
QUAL:   OR 'GLOBAL 'LOCAL
        OR. TERM
        END
```

Use the END macro to indicate the end of the specification of the arguments. Note, to allow this continuation ability (without any additional overhead), you need the END macro whether or not you use the continuation ability. For example, you would need the END macro in the following category for <qual>:

```
QUAL:   OR 'GLOBAL 'LOCAL TERM
        END
```

Thus, you can use MML like the metalanguage of the general description in Section 12.5 to define the syntax of a DCL command and the syntax of the corresponding MCR command. The parser table (PT) defines the syntax of the DCL command and the translator table (TT) defines the syntax of the corresponding MCR command. The following section describes how the PT and TT drive the parser/translator process (or how the parser and the translator interpret the PT and the TT).

12.7 INTERPRETING DCL COMMAND TABLES

A parser table (PT) and translator table (TT) drive the parser/translator process. There is a PT/TT for each DCL command. The PT defines the DCL syntax and the TT defines the corresponding MCR syntax. The parser/translator process locates the PT and the TT for a particular DCL command line by searching the master command table (MCT) for the command entry. The parser/translator extracts from the command entry the address of the first AND or OR macro of the PT and the address of the first AND or OR macro of the TT.

The parser starts processing at the address of the first macro in the PT. If the processing of this macro succeeds, then the parser process succeeds. A successful parser process means that, at this point, the input command line is a syntactically correct DCL command line (as defined by the PT) and the intermediate form (IFORM) has been constructed. The IFORM is a condensed representation of the input command line. If the processing of this macro fails, then the parser process fails.

The parser processes the AND macro by processing (left-to-right) each argument (syntax element) of the AND macro until either all arguments process successfully or an argument is encountered that does not process successfully. If all arguments process successfully, then the processing of the AND macro succeeds; otherwise, the processing of the AND macro fails.

The parser processes the OR macro by processing (left-to-right) each argument (syntax element) of the OR macro until either the parser successfully processes one syntax element or the parser fails to process any of the syntax elements. If the parser successfully processes one syntax element, then the processing of the OR macro succeeds; otherwise, the processing of the OR macro fails.

The translator starts processing at the address of the first macro of the TT. If the processing of this macro succeeds, then the translation succeeds and an equivalent MCR command line (as defined by the TT) has been constructed from the IFORM. If the processing of this macro fails, then the translator process fails. The processing of the AND and OR macros by the translator is the same as the processing of the AND and OR macros by the parser (as described in the previous section).

Arguments to AND and OR macros can be literals or nonliterals. Nonliterals are handled in the same way by both the parser and the translator. Literals, however, are handled differently by the parser and the translator.

Here is a description of how nonliterals are handled by both the parser and the translator.

In MML, when you use the label of an AND or OR macro as the argument to an AND or OR macro it is called a nonliteral; thus, a nonliteral is the address of another AND or OR macro. The processing of a nonliteral is the processing of the other AND or OR macro that the nonliteral addresses. That is, when the parser or translator encounters a nonliteral (as an argument to an AND or OR macro), the parser or translator transfers parsing or translation control to the macro that the nonliteral addresses. The addressed macro is either an AND or OR macro and the parser/translator processes this macro like all AND or OR macros (as described in the previous section). If the processing of this macro succeeds, then the processing of the nonliteral succeeds; otherwise the processing of the nonliteral fails.

Here is a description of the different handling of literals by the parser and the translator.

A string of ASCII characters used as an argument to an AND or OR macro is called a literal. The string typically is a keyword in the DCL or MCR command grammar. The parser and translator process literals differently.

In parsing a literal, the parser attempts to match the literal to the input DCL command line. If the input command line contains the same ASCII string as the literal, then the match is successful and the processing of the literal succeeds; otherwise, nothing happens.

However, in translating a literal, the translator copies the literal from the TT to the MCR command line buffer. If you do not associate attributes with a literal in a TT, then the processing of the literal always succeeds. (Writing a literal to the MCR command line buffer always succeeds.) A literal in a TT can fail only if you have associated one of the following attributes with the literal:

MAP, CBM, BCT, BST

With these attributes associated with a literal, the translator processes the attributes before the literal is written to the MCR command line buffer.

The MAP attribute instructs the translator to check for a mapping number in the IFORM. If the number is not there, the translator does not write the literal to the MCR command line buffer and the processing of the literal fails.

The CBM, BCT, and BST attributes instruct the translator to check the status of certain bits in its internal data structures. If the status of the bits is not correct, the translator does not write the literal to the MCR command line buffer and the processing of the literal fails. See Section 12.10 for more information on these attributes).

In summary, the parser table (PT) drives the parser and the translator table (TT) drives the translator. The parser/translator succeeds only if the processing of the first AND or OR macro of the PT/TT succeeds. The processing of an AND or OR macro succeeds only when the processing of the arguments to the macros succeed. Arguments can be either literals or nonliterals. A nonliteral instructs the parser/translator to process (recursively) another AND or OR macro and succeeds only if the subsequent AND or OR macro succeeds. A literal instructs the parser to ensure the input command line matches the literal and succeeds only if the match is successful. A literal instructs the translator to output the literal (possibly conditionally) to the MCR command line and the translator succeeds only if either the literal is not conditionalized or the conditionals are satisfied.

The following is an example of a parser table (PT) for the DCL ASSIGN command.

MML NOTATION for PARSER TABLE

```
ASSIGN:: AND      'ASSIGN <QUAL OPTR=T> $DEV $DEV
         END
QUAL:    AND      '/ QUALI
         END
QUALI:   OR       <'LOCAL,4>
         OR.      <'LOGIN,4>
         OR.      <'GLOBAL,2>
         OR.      TERM
         END
TERM:    AND      'TERMINAL: <'TERMINAL,4>':
         END
```

The following is an example of a translator table (TT) for the MCR ASN command.

```
                MML NOTATION for TRANSLATOR TABLE


ASN::   AND     'ASN $DEV '= $DEV <LAUQ OPTR=T>
        END
LAUQ:   OR      '/LOGIN
        OR.     '/GBL
        OR.     MRET
        END
MRET:   AND     '/TERM= $DEV
        END
```

These examples show the first two steps you must follow in creating the tables for the DCL ASSIGN command. The last step is to express the mapping of the syntax elements in the PT to the syntax elements in the TT. This mapping provides the information that the parser/translator process needs to construct a particular MCR ASN command line given a particular DCL ASSIGN command line. The following section describes how to express this mapping.

12.8 MAPPING DCL TO MCR

The mapping of a given DCL command line to the equivalent MCR command line is accomplished in the IFORM. The parser processes mapping information and records it in the IFORM. The translator searches the IFORM for mapping information and then generates the MCR command line.

Mapping information is expressed in MML with the "MAP=n" attribute. MAP takes one or two values. The first value is called the primary mapping number. The second value is called the secondary mapping number. If you need only the primary mapping number, specify it in the following form:

```
MAP=n
```

If you need to specify both mapping numbers, which is allowed only in a TT, or if you need to specify only the second number, then enclose the numbers in angle brackets and separate them with a comma in the following form:

```
MAP=<n,n> or MAP=<,n>
```

To associate the MAP attribute with a syntax element, enclose the syntax element and the MAP attribute in angle brackets as follows:

```
<syntax-element    MAP=n>
```

To map a DCL syntax element in a PT to an MCR syntax element in the corresponding TT, use MAP with the same mapping number for both syntax elements.

For example, in the ASSIGN command, the DCL qualifier "/GLOBAL" maps to the MCR qualifier "/GBL". That is, when the DCL qualifier "/GLOBAL" is input (as part of the DCL command line), the MCR switch "/GBL" is output (as part of the MCR command line).

The following expresses this mapping in MML notation and illustrates the use of the primary MAP number.

```
QUAL:   OR <'/GLOBAL MAP=1> ...
        END

LAUQ:   OR <'/GBL MAP=1> ...
        END
```

MML macros at QUAL: define DCL syntax. MML macros at LAUQ: (QUAL spelled backwards) define the corresponding MCR syntax. When the parser reaches QUAL:, it tries to match the literal "/GLOBAL" to the input (DCL) command line. If the match is successful, then the parser queues the MAP number "P1" (the "P" indicates a primary mapping number) to the IFORM queue. When the translator reaches LAUQ:, it searches the IFORM queue for the MAP number "P1". If the translator finds the MAP number "P1", it outputs to the (forming) MCR command line the literal "/GBL". If the translator cannot find the MAP number in the IFORM queue, it does not use the syntax element (associated with the MAP number) in the translation.

The MAP attribute in a PT can associate either a primary or a secondary mapping number with a syntax element, but not both.

The MAP attribute in a TT can associate either or both mapping numbers with a syntax element.

Associating a primary mapping number with a syntax element in the PT (by means of the MAP attribute) instructs the parser to queue the primary mapping number to the IFORM queue.

Associating a secondary mapping number with a syntax element in the PT instructs the parser to queue the secondary mapping number to the IFORM queue.

In both cases the parser queues a mapping number only if the parser processes the associated syntax element successfully.

The MAP attribute in a TT instructs the translator to search the IFORM queue for a mapping number. When found, the translator marks the mapping number used and processes the associated syntax element.

The search for a primary mapping number differs from the search for secondary mapping numbers. The translator searches the entire IFORM queue from head to tail to find primary mapping numbers. The translator limits its search for secondary mapping numbers, however. The translator starts its search for secondary mapping numbers at the queue element containing the last used primary mapping number and stops the search at the next queue element containing a primary mapping number.

Specifying both a primary and a secondary mapping number in the TT (for example, MAP=<1,1>) instructs the translator to search the entire IFORM queue for the specified primary mapping number such that the specified secondary mapping number follows without any intervening primary mapping numbers. The secondary mapping number qualifies the primary mapping number. The primary mapping number is the object of the search.

If the translator finds a mapping number in the IFORM queue, the translator processes the associated syntax element. If the translator fails to find the mapping number, the processing of the associated syntax element fails. When the translator finds a mapping number and successfully processes the associated syntax element, the translator marks as used the queue element containing the found mapping number. A used mapping number can no longer be found. This prevents infinite loops in the TT without requiring the specification of termination conditions.

A MAP attribute can be specified in both the PT and the TT without associating it with a syntax element as follows:

```
<MAP=n>
```

If you specify this attribute as an argument to an AND or OR macro in a PT, the parser assumes a null syntax element. A null syntax element always succeeds.

If you specify a null syntax element in a TT, the translator (after finding the queue element containing the mapping number) checks to see if a character buffer is attached to the queue element. If a character buffer is attached, then the translator copies the contents of the buffer to the MCR command line buffer. If a character buffer is not attached, then the translator (like the parser) assumes a null syntax element (which always succeeds).

The parser attaches the character buffer to the queue element. The COP attribute instructs the parser to create the character buffer and copy the text parsed by the associated syntax element to the character buffer (See Section 12.10 for details on the COP attribute).

Thus, mapping starts in the parser process when the parser queues mapping numbers to the IFORM queue as it successfully processes syntax elements that have associated MAP attributes.

Mapping is completed in the translator process, when the translator detects mapping numbers in the IFORM queue and translates the syntax elements. Note that the translator does not output a syntax element that has a MAP attribute associated with it unless (1) the translator first finds the mapping number in the /FORM queue, and (2) the mapping number has not been previously used.

Here is an example of a complete command table (with DCL-MCR mapping) for the DCL ASSIGN command.

```
                    ASSIGN MML NOTATION FOR PARSER TABLE

ASSIGN:: AND      'ASSIGN <QUAL OPTR=T> <$DEV MAP=1> <$DEV MAP=2>
         END
QUAL:    AND      '/ QUALI
         END
QUALI:   OR       <'LOCAL,3>
         OR.      <'LOGIN,3 MAP=3>
         OR.      <'GLOBAL,2 MAP=4>
         OR.      TERM
         END
TERM:    AND      <'TERMINAL,4>' <DEV MAP=5>
         END
```

ASN MML NOTATION FOR TRANSLATOR TABLE

```
ASN::    AND     'ASN <MAP=1> '= <MAP=2> <LAUQ OPTR=T>
         END
LAUQ:    OR      <'/LOGIN MAP=3>
         OR.     <'/GBL MAP=4>
         OR.     MRET
         END
MRET:    AND     '/TERM= <MAP=5>
         END
```

For a complete description of the MAP attribute, see Section 12.10. For information on related attributes see the discussions of:

DMAP, NO, COP, OCOP, TST, and DIR

The parser records mapping information; the translator detects the mapping information. The following section presents IFORM, the dynamic data structure that passes mapping information from the parser to the translator.

12.9 THE INTERMEDIATE FORM (IFORM)

The IFORM contains the essential information (extracted from the input DCL command line) that the translator needs to construct the MCR command line. The IFORM consists of three data structures: the queue, the character buffers, and the general bit-mask words.

NOTE

To display the IFORM for a given DCL command, use SET DEBUG/FULL.

The parser creates the IFORM queue when it successfully processes syntax elements with MAP attributes. Each time the parser successfully processes a MAP attribute, it queues the mapping number to the IFORM queue.

The parser creates a character buffer when a syntax element has both MAP and COP attributes. The parser attaches the character buffer to the queue entry that contains the mapping number. The character buffer contains the text parsed during the processing of the associated syntax element.

A syntax element with a COP attribute and no MAP attribute instructs the parser to copy the text parsed to the last queued mapping number (creating and attaching the character buffer if neccessary).

The general bit-mask words are two words used as a set of 32 flags. The parser/translator process attaches no meaning to the flags, but MML provides a set of attributes to set flags, clear them, test if the flags are set, and test if the flags are clear. You can use the flags to check for contradictory syntax. In addition, use the flags to pass information from the parser to the translator. For example, set a flag during the parser process if a DCL qualifier is seen and detect the set flag during the translator process, generating the equivalent MCR switch.

The translator searches the IFORM queue for mapping numbers. If the translator finds the mapping numbers, it processes the syntax elements. If the translator cannot find the mapping numbers, then the syntax elements fail.

If a MAP attribute is specified without an associated syntax element as follows:

```
<MAP=n>
```

the translator first finds the queue element containing the mapping number and then examines the queue element for a character buffer attached by means of the COP attribute. If the translator finds an attached character buffer, it copies the contents of the buffer to the MCR command line buffer. If the translator cannot find an attached buffer, it assumes a null syntax element (which always succeeds).

Use the following attributes (in the PT) to direct the parser in the creation of the IFORM:

MAP=n	MAP=<,n>	(map)
DMAP=n	DMAP=<,n>	(map unless duplicate exists)
COP=T	OCOP=T	(copy, or copy optional input)
BS=n	BS=<,n> BS=<n,n>	(set bit in mask word)
BC=n	BC=<,n> BC=<n,n>	(clear bit in mask word)
NO=T		(repeats for negative form of syntax element)

Use the following attributes (in the TT) to direct the translator in the examination of the IFORM:

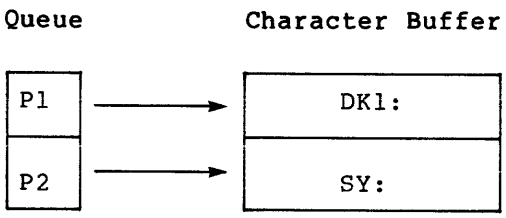
MAP=n	MAP=<,n> MAP=<n,n>	(unmap)
BST=n	BST=<,n> BST=<n,n>	(test for bit set in mask word)
BCT=n	BCT=<,n> BCT=<n,n>	(test for bit clear in mask word)
TST=T		(test for mapping number, do not mark as used)
DIR=T		(search IFORM in reverse direction)

For details on these attributes, see Section 12.10.

For example, if the input command line is:

```
ASSIGN DK1:  SY:
```

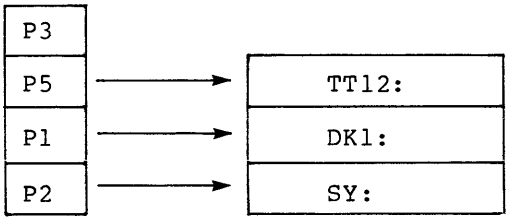
then the IFORM would be:



If the input command line is:

```
ASSIGN/LOGIN/T:TT12:  DK1:  SY:
```

then the IFORM would be:



12.10 ATTRIBUTES OF SYNTAX ELEMENTS

Attributes associate characteristics with literal and nonliteral syntax elements. The parser/translator processes attributes either before or after it processes the associated syntax elements (depending on the attribute). The processing of attributes can influence the processing of the associated syntax elements.

Here is the format for specifying attributes as part of an argument to an AND or OR macro:

```
<'literal  KEY=value KEY1=value ...>

   or

<label     KEY=value KEY1=value ...>
```

where

literal	Literal (ASCII) string
label	Nonliteral (address of an AND or OR macro)
KEY	Attribute keyword identifying the attribute
value	Value assigned to the attribute

Note the use of the angle brackets "<" and ">". Use these MACRO-11 symbols to indicate that everything enclosed is considered as one argument (when expanding macros). Do not confuse these MACRO-11 angle brackets with the angle brackets used to indicate nonliterals in metalanguage notation.

12.10.1 AND

AND allows you to extend a syntax element by specifying a nonliteral that is ANDed to the syntax element. The primary function of AND is to allow OR logic to be mixed with AND logic. Assign a nonliteral to the AND attribute as follows:

```
AND=label
```

where

label	Nonliteral address of an AND or OR macro

AND instructs the parser/translator to process the AND or OR macro that the label (the nonliteral) addresses in addition to processing the syntax element with which the AND attribute is associated.

For example, in the following MML definition for QUAL, the second syntax element defines the structure "FILE:<filespec>":

```
QUAL:   OR      'BRIEF
        OR.     <'FILE AND=FNAME>
        OR.     'FULL
        END

FNAME:  AND     ': $FILE
        END
```

There are three syntax elements in this structure:

FILE

and

:

and

<filespec>

The literal ":" and the nonliteral <filespec> connect to the literal "FILE" by means of the AND attribute. The parser/translator processes all three elements as one element to the OR macro by definition of the AND attribute.

12.10.2 BC and BS

BC (Bit Clear) and BS (Bit Set) operate on two global bit-mask words.

The BC attribute instructs the parser/translator to clear bits in these two global words.

The BS attribute instructs the parser/translator to set bits in these two global words.

Use the BCT and BST attributes to test these bits. See next section.

The format for these attributes is as follows:

BC=n or BC=<,n1> or BC=<n,n1>

BS=n or BS=<,n1> or BS=<n,n1>

where

n Bit mask specifying bits in the first global word

n1 Bit mask specifying bits in the second global word

Specify a bit mask by ORing single bit specifications with the MACRO-11 symbol "!". The symbols B1, B2,...B32 are predefined to correspond to single bit specifications for each bit in the two global words.

For example, BS=B1!B2!B5 means to set bits 1, 2, and 5 of the first global word. See the next section for an example of these attributes in use.

12.10.3 BCT and BST

BCT (Bit Clear Test) and BST (Bit Set Test) test the status of bits in the two global bit-mask words. See the previous section for information on how these bits are set or cleared.

The BCT attribute instructs the parser/translator to test whether bits are cleared.

The BST attribute instructs the parser/translator to test whether bits are set.

The parser/translator does not clear the two global bit-mask words between the parse and the translation. You can use these words as part of the IFORM to pass information from a PT to a TT.

The format for these attributes is as follows:

```
BCT=n    or    BCT=<,n1>    or    BCT=<n,n1>

BST=n    or    BST=<,n1>    or    BST=<n,n1>
```

where

n — Bit mask specifying bits in the first global word

n1 — Bit mask specifying bits in the second global word

For example, BST=<,B19!B31> means to test whether bits 19 and 31 of the second global bit-mask word are set.

The following example illustrates the use of BS, BC, BST, and BCT:

```
COPY:   AND     FILE <FILE OPTR=T> QUA=QUAL
        END

FILE:   OR      <$NFIL          BS=B1   BCT=B2!B3>
        OR.     $FILE
        END

QUAL:   OR      <'INDEX         BS=B2   BCT=B1!B3>
        OR.     <'SEQUENTIAL    BS=B3   BCT=B1!B2>
        END
```

In this example, INDEX and SEQUENTIAL are qualifiers, $NFIL is a predefined syntax element that is a file specification that has a node name prefixing it, and $FILE is a predefined syntax element that is a conventional file specification (without a node name). The example specifies that the qualifiers INDEX and SEQUENTIAL are contradictory, that INDEX and $NFIL are contradictory, that SEQUENTIAL and $NFIL are contradictory, but that duplicates of each are not contradictory. The BS attribute instructs the parser to set the specified bits if the processing of the associated syntax elements is successful. The BCT instructs the parser to test whether the specified bits are clear.

If the tested bits are set, the parser does not process the associated syntax elements. Thus, bits are set to remember which syntax elements process successfully, and bits are tested to determine if a contradiction would result from the processing of syntax elements.

The main difference between the BC, BCT, BS, and BST attributes and CBM (described in the next section) is that CBM always prevents duplicate syntax as well as contradictory syntax, but CBM cannot pass information from the parser to the translator.

12.10.4 CBM, CCBM

Use CBM (Contradictory Bit Mask) to prevent the parser or translator from allowing contradictory syntax. To indicate that two or more syntax elements cannot be specified in the same command line, specify CBM with each syntax element. Assign a bit mask as the value to CBM. Specify the same bit in each bit mask associated with each contradictory syntax element.

CBM conditionalizes the processing of the associated syntax element. The parser uses the specified bit mask to set bits in two global words. If the parser detects that any bits specified in the bit mask are already set in the two global words, then the parser assumes a contradiction and the syntax element fails. This also means CBM does not allows duplicate syntax elements.

Also note that the parser/translator clears the two global words between the parse and the translation. You cannot use CBM to pass information from a PT to a TT.

The format of the CBM attribute is as follows:

```
CBM=n or CBM=<,n1> or CBM=<n,n1>
```

where

n Bit mask specifying bits in the first global word

n1 Bit mask specifying bits in the second global word

Specify a bit mask by ORing single bit specifications with the MACRO-11 symbol "!". The symbols B1, B2,..., B32 are predefined to correspond to single bit specification. For example, CBM=B1!B2!B5 specifies bits 1, 2, and 5 of the first global word, and CBM=<,B19!B31> specifies bits 19 and 31 of the second global word.

The following example illustrates the use of the CBM attribute:

```
QUAL:   OR      <'LOCAL     CBM=B1!B2>
        OR.     <'LOGIN     CBM=B2>
        OR.     <'GLOBAL    CBM=B1>
        END
```

The example specifies that the literals "LOCAL" and "GLOBAL" are contradictory, and that the literals "LOCAL" and "LOGIN" are also contradictory, but that the literals "LOGIN" and "GLOBAL" are not. The example also specifies that each of these elements can be used only once.

CCBM (Clear Contradictory Bit Mask) clears the bits in the two global words. The format for specifying CCBM is identical to the format that you use to specify the CBM attribute.

12.10.5 COP, OCOP

COP (COPy) is legal only in a PT and only with a literal syntax element. COP passes a literal from the input DCL command line to the IFORM. The translator can then use the literal in the MCR command line. The parser copies the literal to a character buffer and attaches it to the mapping number last queued. If you specify a MAP attribute with the COP attribute, the mapping number (assigned to the MAP attribute) is the mapping number last queued.

The format of the COP attribute is as follows:

```
COP=T
```

For example, the following code defines a terminal specification:

```
TERM:   OR <'TT COP=T AND=N>
        OR. <'HT COP=T AND=N>
        OR. <'VT COP=T AND=N>
        OR. <'TI COP=T AND=N>
        END
N:      AND <$ONUM,0,3> <': OCOP=T>
```

This code parses and copies a terminal name as TT, VT, HT, or TI followed by an octal number as terminal names. (See Section 12.11.1 for a description of $ONUM, which defines an octal number.) The OCOP attribute assures the presence of the terminating colon by copying it into a character buffer whether or not the user inputs one.

To instruct the translator to copy the character buffer to the MCR command line, specify the MAP attribute as follows, without specifying an associated syntax element:

```
<MAP=n>
```

When the translator processes a MAP attribute that has no associated syntax element, the translator examines the mapping number queue element for an attached character buffer. If a character buffer is found, the translator copies the contents of the character buffer to the forming MCR command line buffer. If a character buffer is not found, the translator assumes a null syntax element (which always processes successfully).

OCOP (Optional COPy) is identical to COP except for one significant difference. OCOP instructs the parser to copy the literal to the IFORM whether or not the processing of the literal succeeds. That is, the literal is optional and always succeeds if you associate OCOP with it.

The following illustrates the use of the OCOP attribute:

```
DEV:    AND     <$ALF,2,2> <$NUM,0,3> <':  OCOP=T>
```

The MML definition defines a device specification such that the colon on the end of the specification is optional. The translator copies the colon to the output MCR command line whether or not the parser found a colon in the input DCL command line.

12.10.6 DIR

DIR (DIRection) is legal only in the TT and only when you specify it with a MAP attribute. DIR instructs the translator to search the IFORM queue for the specified mapping number in reverse order. The normal direction of the search is from the head of the queue to the tail of the queue. DIR instructs the translator to search the IFORM queue from the tail to the head of the IFORM queue. The format for DIR is as follows:

```
DIR=R
```

The following example of a TT fragment illustrates the use of the DIR attribute:

```
        .
        .
        .
FILE:   AND     <MAP=1 DIR=R <MORE OPT=T>
        END

MORE:   AND     <<',>> FILE
        END
```

This example (recursively) defines the structure of a file specification list, where the file specifications are mapped to mapping number 1. Due to the associated DIR attribute, the translator searches for the mapping numbers in reverse order; thus, the list of file specifications is output to the MCR command line in reverse order.

NOTE

The double angle brackets are necessary in cases where the contents of the brackets include a MACRO-11 separator character (in this case, the comma) that is to be treated as a literal by MML.

12.10.7 ERR

ERR (ERRor) allows you to associate error messages with syntax elements. If the parser or the translator fails to process a syntax element, and the syntax element has ERR specified, then the parser or translator prints the error message specified and aborts the parse or translation.

The format for ERR is as follows:

```
        ERR='string

           or

        ERR=label

label:  ERROR </string/>
```

For example, to specify an error message for the device being assigned, use the following MML notation in the PT for the ASSIGN command:

```
AND     'ASSIGN <QUAL OPT=T> <$DEV ERR=<'Invalid device>>
```

If the parser cannot parse the input DCL command line text that corresponds to the device specification ($DEV), the parser prints the message "ASSIGN -- Invalid device", and aborts the parse.

12.10.8 HLP

Use HLP (HeLP) in a PT only to associate help text with syntax elements that have the PRO attribute (Section 12.10.12) specified. When the parser is prompting and a "?" character is input, the parser spawns a HELP command. The parser appends the text specified in the HLP attribute to the HELP command "HELP command" where "command" is the name of the DCL command being processed.

The format of the HLP attribute is as follows:

```
        HLP='string

           or

        HLP=label

label:  HELP </string/>
```

where

string	Appended to the HELP command
label	Label of a HELP macro

The parser appends the string to the command "HELP command", where "command" is the name of the current command being processed. For example, if you specify the PT for ASSIGN as:

```
AND     <QUAL OPTR=T> <$DEV PRO='Device?  HLP=<'Logical name>>
```

and the user types a "?" in response to the prompt "Device?", the parser forms and spawns the command "HELP ASSIGN LOGICAL NAME".

This requires a file named DCLASSIGN.HLP containing the necessary help text. For the format of the HELP file, see the HELP documentation in the RSX-11M/M-PLUS Command Language Manual or the RSX-11M/M-PLUS MCR Operations Manual.

12.10.9 MAP, DMAP

MAP is legal in both the PT and the TT. However, the parser and the translator interpret MAP differently. The following summarizes the types of legal MAP attributes in a parser table.

MAP=n	Instructs the parser to queue the specified primary mapping number n to the IFORM queue only if the associated syntax element is successfully parsed.

MAP=<,n1> Instructs the parser to queue the specified secondary mapping number n1 to the IFORM queue only if the associated syntax element is sucessfully parsed.

Note that specifying COP with MAP in a PT instructs the parser to create and attach a character buffer to the mapping number, and copy the text parsed by the associated syntax element to the attached character buffer. See Section 12.10.5, which describes COP, for additional details.

The following summarizes the types of legal MAP attributes in a translator table.

MAP=n Instructs the translator to search the IFORM queue from head to tail for the specified primary mapping number n. If found, the translator places the element in the MCR command line. If not found, nothing happens.

MAP=<,n1> Instructs the translator to search the IFORM queue for the secondary mapping number n1. Since the object of the search is a secondary (not primary) mapping number, the translator does not search the entire IFORM queue. The translator starts searching at the last used primary mapping number and stops the search at the next primary mapping number. If the secondary mapping number is found, then the associated syntax element is processed; otherwise, nothing happens.

MAP=<n,n1> Instructs the translator to search the IFORM queue from head to tail for the specified primary mapping number n. If found, the translator searches for the specified secondary mapping number n1. If found, the associated syntax element is processed. If not found, another occurrence of primary mapping number n is sought. If another instance of primary mapping number n is found, then the translator looks again for the specified secondary number n1. This search process continues until either all primary mapping numbers n have been found and none of them have a secondary mapping number n1 after them, or until a primary mapping number n is found that does have a secondary mapping number n1 after it.

If you specify a MAP attribute in a TT without specifying an associated syntax element, the translator examines the queue element containing the specified mapping number for an attached character buffer. If the translator finds an attached character buffer, the translator copies the contents of the buffer to the forming MCR command line buffer. If the translator does not find an attached character buffer, the translator assumes a null syntax element (which always processes successfully).

To allow additional flexibility in the search for secondary mapping numbers, the translator supports the following additional formats for specifying the MAP attribute in a TT.

MAP=<X,n1> Instructs the translator to search the IFORM queue from head to tail for the specified secondary mapping number n1. If found, the translator processes the associated syntax element. If not found, the associated syntax element fails to process. The keyword "X" identifies this MAP attribute as an unconditional search for a secondary mapping number.

MAP=<X1,n1> Instructs the translator to seach the IFORM queue for the secondary mapping number n1. The keyword "X1" instructs the translator to limit the search of the queue as follows. The search is started at the head of the queue and proceeds until the first primary mapping number is detected. If the secondary mapping number n1 is not found, then another search is conducted. The second search looks for the secondary mapping number n1 such that it follows a negative primary mapping number. If found, the translator processes the associated syntax element. If not found, nothing happens.

Note that you use the "X" MAP attribute to search for a secondary like a primary (that is, unconditionally) and the "X1" MAP attribute to search for floating command qualifiers. For information on floating command qualifiers see Section 12.10.13, which describes QUA.

When the translator finds a mapping number and processes the associated syntax element, the translator marks the mapping number as used. A used mapping number cannot be found again by the translator. Thus marking mapping numbers used prevents infinite loops in the TT without the need to specify termination conditions. To override the marking of mapping numbers, use TST. To reset all marked mapping numbers, use the $RESET predefined syntax element. (This is useful for generating multiple MCR commands by means of the $NEW predefined syntax element.)

DMAP (duplicate map) is identical to MAP in most ways. However, it is only allowed in a PT. DMAP instructs the parser not to queue the mapping number to the IFORM queue if the same mapping number is already present in the IFORM queue. If the parser does not queue the mapping number, then the syntax element that MAP is associated with fails to process. Thus, the DMAP attribute can prevent duplicate syntax.

For additional information on the use of the MAP attribute, see the Section 12.8, Mapping DCL to MCR.

The following example shows the use of primary mapping numbers with predefined syntax elements.

DCL filespecs, device specifications, numerals, and so forth, map to themselves. That is, when a filespec, device specification, numeral, and so forth, is input, the same filespec, device specification, numeral, and so forth, is output. Here is an example of how a terminal device specication maps to itself.

Parser Table

```
TERM:  AND 'TERMINAL:  <$DEV,MAP=2>
       END
```

Translator Table

```
MRET:  AND 'TERM= <MAP=2>
       END
```

The parser table specifies at label TERM: If the input command line matches a device specification ($DEV), then queue mapping number P2 to the IFORM queue, attach a character buffer to this queue element, and copy the device specification to the attached character buffer. The translator table specifies at label MRET: If P2 is in the IFORM queue, then output the contents of any attached buffer to the forming MCR command line.

The following example illustrates the use of secondary mappings.

When translating the DCL MACRO command to the MCR MAC command, it is nessessary to determine which (if any) input filespec has the qualifier /LIST on it. That is, the translator must be able to determine (by means of information in the TT) that the filespec "C" in the DCL command line "MACRO A,B,C/LIST,D" has the /LIST qualifier on it. The IFORM must map the DCL filespec that has the /LIST qualifier on it to the right position in the MCR command line.

The mapping can more easily be understood from the viewpoint of the translator. The translator generates the MCR MAC command from left to right. When the translator reaches the position in the output MCR command line where the list file must be placed, the translator looks in the IFORM queue for the mapping number that represents the filespec with the /LIST qualifier on it. Since the filespec and qualifier are two distinct syntax elements, there are two mapping numbers in the IFORM queue to represent them. Also, since the IFORM queue is a linear data stucture (and not a tree), a means of associating the /LIST qualifier with the filespec is needed.

To express an association between elements in the IFORM queue, use primary and secondary mapping numbers. Secondary mapping numbers (such as S1) have meaning only as related to primary mapping numbers (such as P1).

The following MML notation expresses this mapping and illustrates the use of primary and secondary mappings.

Parser Table

```
        AND     'MACRO FILES
        END
FILES:  AND     <$FILE MAP=1> <'/LIST OPT=T MAP=<,1>> <MORE OPT=T>
        END
MORE:   AND     <<',>> FILES
        END
```

Translator Table

```
        AND     'MAC <OBJ OPT=T> <LST OPT=T> '= SELIF
        END
LST:    AND     <<',>> <MAP=<1,1>>
        END
        SELIF:
                .
                .
                .
```

It is possible when parsing (due to the recursive definition of FILES) to queue to the IFORM queue identical map numbers ("P1") that represent the input filespecs. The only means of distinguishing the "P1" map numbers is by their left-to-right positioning. For the translator to find the "P1" map number that represents the filespec with the /LIST on it, the translator has to find the "P1" map number that is followed by the "S1" map number such that there are no intervening "Pn" map numbers. In MML, to instruct the translator to perform such a search, use the argument "<MAP=<1,1>>", as specified at label LST.

Remember that if a reference is made to a secondary map number without specifying a primary map number, then the translator looks for a secondary map number that is associated with the last used primary map number.

12.10.10 NO

NO is allowed only in a PT. NO instructs the parser to process either a literal "NO" or a literal "-" before it processes the associated literal. The NO attribute is legal only when associated with a literal syntax element. The format for the NO attribute is:

```
NO=T
```

If you specify a MAP with a NO, the parser queues two different mapping numbers. If the parser does not process either the "NO" or the "-", the parser queues the mapping number assigned to MAP. If the parser does process either the "NO" or the "-", the parser increments the mapping number by one before it queues the mapping number.

The following illustrates the use of NO:

```
OR      <'LIST MAP=10.  NO=T>
```

In this example, the parser attempts to parse either NOLIST, -LIST, or LIST. If LIST is parsed, the parser adds the mapping number 10 to the IFORM queue. If either NOLIST or -LIST is parsed, the parser queues the mapping number 11.

12.10.11 OPT, OPTR

Use OPT (OPTional) to indicate that a syntax element is not required. You can use OPT in both the PT and the TT. The parser and the translator interpret OPT identically.

The format for OPT is as follows:

```
OPT=T
```

For example, to specify that qualifiers (label QUAL) are optional syntax in the ASSIGN command, use the following MML notation:

```
AND     'ASSIGN <QUAL OPT=T> ...
```

The parser goes to label QUAL and processes the syntax defined at label QUAL. If the parser cannot process a syntax element in an AND macro, the failure implies the failure of the AND macro. The OPT=T attribute tells the parser to ignore the failure and continue processing the AND macro with the next (left-to-right) argument to the AND macro.

Use OPTR (OPTional Repeat) in the same way to indicate that a syntax element is not required. In addition, use OPTR to indicate that the syntax element can be used repeatedly in the parse/translation until a failure results. In other words, the presence of OPTR indicates that the syntax element can be used to parse or translate syntax from zero to an infinite number of times.

The format for OPTR is as follows:

```
OPTR=T
```

For example, to specify that qualifiers (label QUAL) are optional syntax in the ASSIGN command and that the qualifiers can be used repeatedly in the input command line, use the following MML notation:

```
AND     'ASSIGN <QUAL OPTR=T> ...
```

The parser goes to label QUAL and processes the syntax defined there. If a syntax element in an AND macro fails, the failure cannot be used in the parse, implies the failure of the AND macro. The OPTR=T attribute tells the parser to ignore failure and to continue processing the AND macro with the next (left-to-right) argument to the AND macro. However, if and while the syntax defined at label QUAL matches the input command, the parser uses it repeatedly.

If the OPT and AND attributes are used in the same syntax element, the OPT modifies the AND. That is, the OPT does not modify the entire syntax element.

For example, here is a fragment of code based on the LINK command translator table. The TT specifies a TKB switch that can be specified with or without arguments attached.

```
                .
                .
                .
        OR.     <'/PR MAP=1 AND=PRIV OPT=T>
        END

                .
                .
                .
PRIV:   AND     ': <MAP=3>
        END
```

In the example, the optional argument to the /PR switch is attached to mapping number 3. The OPT applies to AND=PRIV and not to the whole syntax element. If the OPT applied to the entire syntax element, then the entire OR macro, of which this fragment is a part, would always succeed.

12.10.12 PRO

Use PRO (PROmpt) in the PT to associate prompt strings with syntax elements. If the parser detects end-of-line while processing a syntax element that has the PRO attribute specified, the parser prints the prompt string specified, and suspends itself to wait for more input. The parser appends any input received to the input DCL command line. PRO is legal only in the PT. The translator does not prompt.

The format of PRO is as follows:

```
        PRO='string

           or

        PRO=label

label:  PROMPT </string/>
```

where

string ASCII string to be used as the prompt string

label (MACRO-11) label of a (MACRO-11) PROMPT macro

Thus, specify the prompt string directly after the "=" symbol, or as the argument to the PROMPT macro. The PROMPT macro allows global prompt strings, which conserve space.

The following example illustrates the use of PRO:

```
AND <QUAL OPTR=T> <$DEV PRO='Device?> <$DEV PRO=PRODEV>

PRODEV:  PROMPT </Logical device?  />
```

The command requires two device specifications ($DEV). If the parser expects the first $DEV but reaches the end of the line, then the parser prompts:

Device?

If the parser expects the second $DEV but reaches the end of the line, then the parser prompts:

Logical device?

12.10.13 QUA

Use QUA (QUAlifier) to specify qualifiers. You can specify floating or positional qualifiers. Floating qualifiers are qualifiers that are legal anywhere in the input DCL command line. Positional qualifiers are qualifiers that are legal only at specific positions in the input DCL command line or output MCR command line.

The positional form of the QUA attribute is legal in both a PT and a TT. The floating form is legal only in a PT.

The format for QUA is as follows:

```
QUA=label
```

where

label Nonliteral (address of an AND or OR macro)

When the QUA attribute is used in a PT and the parser encounters a slash (/) in an appropriate location, it attempts to match the text following the slash with one of the elements of the AND or OR macro at the specified label. If an end-of-line is encountered after the slash, the parser issues a prompt. If an illegal qualifier is given, then an error message is printed and the parse is aborted. The list of qualifiers is repeatedly scanned until all qualifiers in the input command line have been parsed.

To specify a positional QUA attribute, use QUA as part of a syntax element. For instance:

```
        AND     <'ASSIGN,3 QUA=QUAL> $DEV $DEV
        END

QUAL:   OR      <'GLOBAL,1>
        OR.     <'LOCAL,3>
        OR.     <'LOGIN,3>
        OR.     TERM
        END
```

In this case, any qualifiers (such as /GLOBAL) must immediately follow "ASSIGN" in the command line. (See the following example for more details on how the QUA attribute works.)

To specify floating qualifiers use, QUA as a separate argument to an AND macro, not as part of another argument. The parser attempts to match the qualifiers defined at the label before and after it processes each argument of the AND macro. That is, the qualifiers float before and after each syntax element defined by the AND macro. Furthermore, the parser attempts to parse the floating qualifiers after any prompts. If QUA is being used to specify floating qualifiers, it must be an argument to an AND macro, rather than an AND. (extension) macro. The floating-qualifier form of QUA is legal only in a PT.

The following illustrates the floating-qualifier form of QUA:

```
            AND     'ASSIGN, 3 $DEV $DEV QUA=QUAL
            END

     QUAL:  OR      <'LOCAL,3>
            OR.     <'LOGIN,3>
            OR.     <'GLOBAL,1>
            OR.     TERM
            END
```

The label QUAL defines the allowable qualifiers. The parser checks for the qualifiers before, between, and after the $DEV syntax elements. Before the parser transfers parsing control to the OR macro at label QUAL, the parser attempts to parse the slash "/" character. Only if the parser successfully parses the slash, does it attempt to parse the qualifiers defined at label QUAL. If the parser detects the end of the input DCL command line after it parses a slash character, the parser prompts for more input. If a slash is detected and the qualifiers defined at label QUAL fail to process, the parser issues an error message and aborts the parse. The parser repeatedly attempts to parse the qualifiers until it no longer can parse a slash character.

Thus, using QUA in the previous example is equivalent to the following:

```
            AND     <QUAL OPTR=T> $DEV <QUAL OPTR=T> $DEV <QUAL OPTR=T>
            END

     QUAL:  AND     '/ <QUAL1 PRO='Qualifier? ERR=<'Illegal qualifier>>

     QUAL1: OR      <'LOCAL,2>
            OR.     <'LOGIN,2>
            OR.     <'GLOBAL,1>
            END
```

The following illustrates the use of the QUA attribute in a TT.

```
     ASN::  AND <<'ASN >> <MAP=1> '= <MAP=2 QUA=LAUQ>
            END

     LAUQ:  OR <'GBL     MAP=4>
            OR. <'LOGIN  MAP=3>
            OR. MRET
            END
```

The label LAUQ defines the MCR switches equivalent to the DCL qualifiers. The translator processes QUA just as does the parser. However, instead of parsing the input DCL command line for qualifiers, the translator uses the qualifiers to construct the equivalent MCR command line. Remember that QUA automatically includes slashes before each switch.

12.10.14 TST

TST is legal only in a TT and only when specified with MAP. TST instructs the translator not to mark a mapping element used in a translation as having been used.

The format of the TST attribute is as follows:

```
TST=T
```

The following illustrates the use of the TST attribute:

```
MCRCOP: OR      <PIP    MAP=1 TST=T>
        OR.     <PIP    MAP=2 TST=T>
        OR.     <NFT    MAP=3>
        END
```

In the previous example, the OR macros instruct the translator to process the nonliteral PIP if either mapping numbers 1 or 2 are in the IFORM queue, or to process the nonliteral NFT if mapping number 3 is in the IFORM queue. The TST attribute associated with the syntax element PIP instructs the translator not to mark mapping numbers 2 or 3 as having been used. This allows the syntax defined at label PIP to use these mapping numbers also.

12.11 PREDEFINED SYNTAX ELEMENTS

Predefined syntax elements (PSEs) are syntax elements defined for use by all DCL command tables. PSE names, by convention, are distinguished from local category names by having a "$" as their first character.

NOTE

If the syntax of a command does not match that of a predefined syntax element, then the parse fails. Several PSEs have two forms, one ending with an "E" and one not, such as $NOD and $NODE, $FIL and $FILE, $DEV and $DEVE, and $UIC and $UICE. The form ending in "E" produces an error message before aborting the parse. The shorter form does not. Use the shorter form when defining optional syntax.

The following list briefly defines the PSEs. Note that some PSEs require arguments. Specify the required arguments in the order shown and enclose the PSE and the arguments in angle brackets to identify the PSE and arguments as one argument to the MACRO-11 assembler.

12.11.1 Basic Data Types

The following list of PSEs define basic data types in MML. You can use them only in a PT. The parser copies to an IFORM character buffer the text in the input DCL command line that fits the structure defined by the PSE. The buffer is attached to the last queued mapping number and can be accessed by the translator for use in constructing the equivalent MCR command line.

$NUM,A1,A2	Defines the structure or syntax of a numeric string (0 through 9). The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string. Additionally, the PSE allows an optional decimal point on the end of the string. The decimal point is not copied to the IFORM character buffer.
$DNUM,A1,A2	Defines the structure or syntax of a decimal numeric string (0 through 9). The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string. Additionally, the PSE allows an optional decimal point on the end of the string. A decimal point is copied to the IFORM character buffer whether or not it is specified in the input DCL command line.
$ONUM,A1,A2	Defines the structure or syntax of an octal numeric string (0 through 7). The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string.
$WONUM,A1,A2	Defines the structure or syntax of $ONUM or the wildcard character "*". The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string.
$DONUM,A1,A2[.]	Defines the structure or syntax of a decimal or octal numeric string (0 through 9). The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string. Additionally, the PSE allows an optional decimal point on the end of the string. A decimal point is copied to the IFORM character buffer if either the input DCL command line or the parsed string contains the characters 8 or 9.
$WDONUM,A1,A2	Defines the structure or syntax of $DONUM or a "*" character. The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string.
$BNUM,A1,A2	Defines the structure or syntax of a binary numeric string (0 through 1). The arguments A1 and A2 designate the minimum and maximum length, respectively, for the numeric string.
$ALF,A1,A2	Defines the structure or syntax of an alphabetic string (A through Z). The arguments A1 and A2 designate the minimum and maximum length for the alphabetic string.
$ALFN,A1,A2	Defines the structure or syntax of an alphanumeric string (0 through 9 and/or A through Z). The arguments A1 and A2 designate the minimum and maximum length for the alphanumeric string.
$WALFN,A1,A2	Defines the structure or syntax of $ALFN with the additional characters: "*" and "%". These characters are used by some RSX-11M/M-PLUS utilities as wildcard characters. The arguments A1 and A2 designate the minimum and maximum length for the alphanumeric string.

$R50,A1,A2 — Defines the structure or syntax of a Radix-50 string (0 through 9, A through Z, "$", and "."). The arguments A1 and A2 designate the minimum and maximum length for the Radix-50 string.

12.11.2 Common Command Elements

The following PSEs define parts of the DCL command grammar. You can use them only in a PT. The text (except for $DATE) parsed from the input DCL command line that fits the structure defined by the PSE is copied to an IFORM character buffer that is attached to the last queued mapping number.

$FIL — Defines the structure or syntax of a file specification. Each element explicitly included in the filespec sets a bit in the global bit-mask words. These are:

device B28	filetype B31
UIC B29	version B32
filename B30	

These bits are also set by the other $FIL-based predefined syntax elements.

$FILE — Defines the structure or syntax of $FIL. Additionally, if the parser fails to process $FILE, the parser issues an error message and aborts the parse.

$WFIL — Defines the structure or syntax of $FIL. Additionally, the wildcard characters: "*" and "%" are allowed as part of the file specification.

$WFILE — Defines the structure or syntax of $WFIL. Additionally, if the parser fails to process $WFILE, the parser issues an error message and aborts the parse.

$NFIL — Defines the structure or syntax of $WFIL if preceded by a DECnet node specification.

$NFILE — Defines the structure or syntax of $NFIL. Additionally, if the parser fails to process $NFILE, the parser issues an error message and aborts the parse.

$IFIL — Defines the structure or syntax of $FIL. Additionally, the indirect file indicator "@" must prefix the file specification.

$IFILE — Defines the structure or syntax of $IFIL. Additionally, if the parser fails to process $IFILE, the parser issues an error message and aborts the parse.

$NOD — Defines the structure or syntax of a DECnet node name specification.

$NODE — Defines the structure or syntax of $NOD. Additionally, if the parser fails to process $NODE, the parser issues an error message and aborts the parse.

$DEV	Defines the structure or syntax of a device specification.
$DEVE	Defines the structure or syntax of $DEV. Additionally, if the parser fails to process $DEVE, the parser issues an error message and aborts the parse.
$ODEV	Defines the structure or syntax of $DEV. Additionally, the character ":" is optional syntax.
$ODEVE	Defines the structure or syntax of $ODEV. Additionally, if the parser fails to process $ODEVE, the parser issues an error message and aborts the parse.
$UIC	Defines the structure or syntax of a UFD specification.
$UICE	Defines the structure or syntax of $UIC. Additionally, if the parser fails to process $UICE, the parser issues an error message and aborts the parse.
$WUIC	Defines the structure or syntax of $UIC. Additionally, the wildcard character "*" is allowed as part of the UIC specification.
$WUICE	Defines the structure or syntax of $WUIC. Additionally, if the parser fails to process $WUICE, the parser issues an error message and aborts the parse.
$OUIC	Defines the structure or syntax of $UIC. Additionally, the characters "[", "]", and "," are optional syntax.
$OUICE	Defines the structure or syntax of $OUIC. Additionally, if the parser fails to process $OUICE, the parser issues an error message and aborts the parse.
$DATE	Defines the structure or syntax of a date specification. A date specification can take one of two forms: mm/dd/yy or dd-mmm-yy. $DATE produces the following data structures in the IFORM queue: the day is attached to secondary mapping number 210, the month is attached to secondary mapping number 211, and the year is attached to secondary mapping number 212. The output format is always dd-mmm-yy (with the first three characters of the month name spelled out) even if the slash format (with the number of the month) is used.
$DATE1	For commands that require two dates (for example, SINCE and THROUGH), a second PSE for dates, $DATE1, uses secondary mapping numbers 213, 214, and 215 for the day, month, and year respectively. $DATE1 is otherwise identical to $DATE.
$TIME	Defines the structure or syntax of a time specification. The time specification is of the form hh:mm[:ss].

12.11.3 Special Parser Elements

You can specify the following PSEs only in a PT.

$ALL,label — Defines the syntax or structure of an arbitrary string of characters. $ALL matches all characters in the command line (left-to-right) until any one of a set of specified terminating characters is detected. The set of terminating characters are specified with the TERMINATORS macro. The label is a label on the TERMINATORS macro. The text parsed from the input DCL command line that fits the structure defined by $ALL is copied to an IFORM character buffer that is attached to the last queued mapping number.

Here is an example of $ALL with the TERMINATORS macro:

```
                .
                .
                .
        AND. <$ALL,EXCEPT>
                .
                .
                .
EXCEPT: TERMINATORS '" ') '] $EOL
                .
                .
                .
```

This example allows the "," the ")", and the "]" as terminators, in addition to the $EOL predefined syntax element, discussed next.

$EOL — $EOL is used to parse end-of-line characters. For instance, in the previous example, a carriage-return or escape would also be acceptable as a terminator.

$COP,'lit — $COP does not define syntax. $COP instructs the parser to copy the specified literal, 'lit, unconditionally to an IFORM character buffer. The IFORM character buffer is always attached to the last queued mapping number.

$DSB — $DSB (disable skip blanks) does not define syntax. $DSB instructs the parser not to skip blanks. Normally, blanks are ignored. $DSB and $ESB must be in the same syntax element, such as AND or OR, but need not be on the same line.

$ESB — $ESB (enable skip blanks) does not define syntax. $ESB restores the default state, which is to ignore blanks.

For example, a filespec cannot include blanks. The following could define a filespec:

```
$DSB $DEV $UIC <$ALFN,0,9> '. <$ALFN,0,3> '; <$ONUM,0,3> $ESB
```

(The actual filespec definition is more complex. This is only an example.)

$DDR — $DDR does not define syntax. $DDR instructs the parser to disable delimiter recognition. Delimiter recognition is enabled by default and instructs the parser to ensure that all literal syntax elements have delimiting characters terminating them. $DDR must be used on the same line with $EDR. See the next example.

$EDR — Enable delimiter recognition. This is the default case.

For example, the following line allows you to attach the first one to three characters to one mapping number while the remainder are attached to another.

```
COM1:  AND $DDR <$R50,1,3 MAP=1> <$R50,0,7 MAP=2> $EDR
       END
```

Notice that no delimiting character distinguishes the two strings, and that the second string may be null.

$SEP — $SEP defines the colon (:) or equals (=). These are used as separators between qualifiers and arguments.

$OSEP — $OSEP defines an optional separator, that is colon, equals, or null.

$CSEP — $CSEP is the same as $SEP, but additionally copies a colon to the IFORM.

$COSEP — $COSEP is the same as $OSEP, but additionally copies a colon to the IFORM whether anything is input or not.

$FSEP — $FSEP defines the comma (,) and plus (+) as filespec separators.

$TERM — $TERM defines a terminal specification and copies it to the IFORM.

$PROC — $PROC defines a processor name for the Queue Manager as either a device name followed by a colon or six Radix-50 characters followed by a colon.

$PRONC — $PRONC defines a processor name for the Queue Manager as a device name followed by a colon or six Radix-50 characters not followed by a colon.

12.11.4 Special Translator Elements

You can specify the following PSEs only in a TT.

$NEW — Used to indicate the start of an additional MCR command line. $NEW instructs the dispatcher subprocess of the DCL task that there are multiple MCR commands that must be dispatched serially.

$RESET — Used to reset the IFORM queue. Resetting the IFORM queue allows all used mapping numbers to be reused. $RESET is designed to be used with $NEW to define multiple MCR commands.

$FNAM — Used to specify that only the file name in a file specification is to be copied to the MCR command line buffer as a result of unmapping a mapped file specification.

$SY — Converts the assigned device for SY: to ASCII and appends it to the MCR command line; for example, DB2:.

$TI — Converts the assigned terminal device for TI: to ASCII and appends it to the MCR command line; for example, TT15:.

$CUIC — Converts the current UIC for the user to ASCII and appends it to the MCR command line; for example, [301,113].

$LUIC — Converts the login UIC for the user to ASCII and appends it to the MCR command line.

12.11.5 Program Control Elements

You can specify the following PSEs in either a PT or a TT.

$JSR,A — A special category used to gain control of the CPU. The argument A must be the label of a MACRO-11 subroutine. To indicate failure, the MACRO-11 subroutine should (on return) set the carry bit to indicate failure and clear it to indicate success. Remember, the routine must be in the blank program section.

$TST — $TST is used to test if a mapping number is in the IFORM queue. A MAP attribute specifies the mapping number. $TST succeeds only if the mapping number is in the IFORM queue; for example, <$TST MAP=1>.

$TSTNOT — $TSTNOT is used to test if a mapping number is not in the IFORM queue. A MAP attribute specifies the mapping number. $TSTNOT succeeds only if the mapping number is not in the IFORM queue. The syntax is identical to $TST.

$LOAD,name — $LOAD is used to load an overlay from a parser or translator table. The argument "name" specifies the name of the overlay as specified in the overlay description file.

$CALLOV,name — $CALLOV is used to load and activate part of either a parser or translator table. The starting address within the overlay must be contained in the first word of the overlay. Use .WORD to specify the address. On return, the previous overlay is loaded. The argument "name" specifies the name of the overlay as specified in the overlay description file.

$NULL — $NULL is used to execute attributes without specifying a syntax element. $NULL succeeds if the attributes succeed.

$AB — $AB (angle bracket) is equivalent to the literal "<" and is used to pass the "<" as an argument to MACRO-11.

$BA — $BA is equivalent to the literal ">" and is used to pass the ">" as an argument to MACRO-11.

12.11.6 Example from DCL Command Tables

```
                .TITLE  ASSIGN
                .IDENT  /00/
                .PSECT  .CDEF
                .ENABL  LC
;
; THIS  SOFTWARE  IS FURNISHED UNDER A LICENSE AND MAY
; BE USED OR COPIED ONLY IN ACCORDANCE WITH THE  TERMS
; OF SUCH LICENSE.
;
; COPYRIGHT (c) 1981 BY DIGITAL EQUIPMENT CORPORATION.
; ALL RIGHTS RESERVED.
;
; AUTHOR:       J.M. SALMAN
;
; DATE:         07-OCT-80
;
; MODIFIED:     DATE    ABSTRACT
;
; FUNCTION:
;
;       TRANSLATE THE DCL COMMANDS 'ASSIGN', 'ASSIGN/QUEUE', 'ASSIGN/REDIRECT',
;       'ASSIGN/TASK' TO THE SEMANTICALLY EQUIVALENT MCR COMMANDS 'ASN', 'QUE',
;       'RED', 'REA'.
;
;       THE FOLLOWING TABLE DEFINES THE MAPPING OF THE DCL TO MCR SYNTAX:
;
;       DCL SYNTAX ELEMENT      MCR SYNTAX ELEMENT      MAPPING LETTER/NUMBER
;       ------------------      ------------------      ---------------------
;
;       <queuename>             <queuename>             P1
;       <processorname>         <processorname>         P2
;
;       <olddevice>             <olddevice>             P3
;       <newdevice>             <newdevice>             P4
;
;       <taskname>              <taskname>              P5
;       <device>                <device>                P6
;       <lun>                   <lun>                   P7
;
;       <device>                <device>                P8.
;       <logicaldevice>         <logicaldevice>         P9.
;       /GLOBAL                 /GBL                    P10.
;       /LOGIN                  /LOGIN                  P11.
;       /TERMINAL=<termdev>     /TERM=<termdev>         P12.
;
        .PAGE
ASSIGN::
        OR      <QUEUE BS=B1> <REDIR BS=B2> <TASK BS=B3> <LOGDEV BS=B4>
        END

QUEUE:  AND     '/ <'QUEUE,1> <$PRONC MAP=1 PRO=$QNMSG ERR=$BQMSG>
        AND.    <$PROC MAP=2 PRO=$PRMSG ERR=$IPMSG>
        END

REDIR:  AND     '/ <'REDIRECT,1> <$ODEVE MAP=3 PRO=$FRMSG>
        AND.    <$ODEVE MAP=4 PRO=$TOMSG>
        END

TASK:   AND     '/ <'TASK,2> $OSEP <$R50,1,6 MAP=5 PRO=$TKMSG ERR=$ITMSG>
        AND.    <$ODEVE MAP=6 PRO=$DVMSG>
        AND.    <$DNUM,1,5 MAP=7 PRO=$LUMSG ERR=$NEMSG>
        END
```

```
                .TITLE  ASSIGN
                .IDENT  /00/
```

THE DCL TASK

```
LOGDEV: AND     <$ODEVE MAP=8. PRO=$DVMSG> <$ODEVE MAP=9. PRO=$LDMSG> QUA=QUAL
        END
QUAL:   OR      <'GLOBAL,1 MAP=10. CBM=B1!B2>
        OR.     <'LOCAL,3 CBM=B1>
        OR.     <'LOGIN,3 MAP=11. CBM=B1>
        OR.     <'SYSTEM,1 MAP=10. CBM=B1!B2>
        OR.     <$TERM MAP=12. CBM=B2>
        END
        .PAGE
MCRASN::
        OR      <QUE BST=B1> <RED BST=B2> <REA BST=B3> <ASN BST=B4>
        END

QUE:    AND     <<'QUE >> <MAP=2> '/AS: <MAP=1>
        END

RED:    AND     <<'RED >> <MAP=4> '= <MAP=3>
        END

REA:    AND     <<'REA >> <MAP=5> <<' >> <MAP=7> <<' >> <MAP=6>
        END

ASN:    AND     <<'ASN >> <MAP=8.> '= <MAP=9.> <QUA=LAUQ>
        END
LAUQ:   OR      <'GBL MAP=10.>
        OR.     <'LOGIN MAP=11.>
        OR.     <'TERM= AND=VTERM>
        END
VTERM:  AND     <MAP=12.>
        END
        .ENABLE LC
$BQMSG: ERROR   </Illegal queue name/>
$DVMSG: PROMPT  </Device? />
$FRMSG: PROMPT  </From? />
$IPMSG: ERROR   </Illegal processor name/>
$ITMSG: ERROR   </Illegal task name/>
$LDMSG: PROMPT  </Logical device? />
$LUMSG: PROMPT  </Logical unit? />
$PRMSG: PROMPT  </Processor? />
$QNMSG: PROMPT  </Queue? />
$TKMSG: PROMPT  </Task? />
$TOMSG: PROMPT  </To? />
        .DSABLE LC
        .END
```

CHAPTER 13

POOL MONITORING SUPPORT

Pool monitoring support controls the use of the system's dynamic storage region (pool). This support monitors pool levels, restricts use, and notifies you when pool is near depletion. This support consists of two parts: the RSX-11M/M-PLUS Executive pool monitor code, and the privileged Pool Monitor Task (PMT).

Pool monitoring support is not available on unmapped systems.

13.1 HOW YOU GET POOL MONITORING SUPPORT

On RSX-11M systems, you receive pool monitoring support when you select the pool monitoring support option in system generation (SYSGEN). If you select the option, you receive the Executive pool monitor code and SYSGEN assembles and task-builds PMT. If you do not select the option, you do not receive any pool monitoring support.

On RSX-11M-PLUS systems, pool monitoring support, including the Executive pool monitor code and PMT, is included by default. SYSGEN establishes the support for you. You need not select any option for it.

One of PMT's features is that it responds to extreme fragmentation. Due to the seriousness of this pool problem, PMT requires your interaction to aid in system recovery. Therefore, if you select pool monitoring support, you must be prepared to respond to PMT's prompts (see Section 13.3.4).

13.2 THE EXECUTIVE POOL MONITOR CODE

The pool monitor code within the Executive monitors the amount of free pool and detects major pool events. When a major pool event occurs, the Executive notifies PMT of the event and of the current condition of pool.

13.2.1 Pool Events Affecting the Executive

The Executive pool monitor code responds to three major pool events. These events are classified as either low or high pool events. The occurrence of these events directly affects the condition of pool, creating either a high or low pool condition. The Executive responds by notifying PMT of the pool event and the resultant pool condition and also activates PMT. Any further response to the pool condition is left to PMT.

13.2.1.1 **High Pool Event** - The following is the possible high pool event:

- The attainment of a high pool condition from a low pool condition.

This event occurs when a system that has a low pool condition relinquishes enough pool to cause the amount of free pool to rise above the high limit.

The high pool limit is set using the MCR and VMR SET PLCTL command. See Section 13.4 for further discussion of the high limit.

13.2.1.2 **Low Pool Event** - The following are the possible low pool events:

- The degradation of pool from a high pool condition to a low pool condition
- A pool allocation failure

The degradation of pool occurs when the total amount of free pool has fallen below the low limit. A pool allocation failure occurs when the largest pool fragment is too small for the process requesting it. A pool allocation failure is not to be confused with future discussions in this chapter of general pool fragmentation and extreme pool fragmentation. Pool allocation failures and pool fragmentation are, in the context of this chapter, separate pool problems and are discussed separately. See Section 13.3.1 for additional information on pool fragmentation.

The low pool limit is set using the MCR and VMR SET /PLCTL command. See Section 13.4 for further discussion of the low limit.

13.2.1.3 **Pool Monitoring Interface** - The mechanism used by the Executive to detect pool events is called the pool monitor interface. The details of how this interface works are beyond the scope of this manual. However, it is fully described in the source code for the Executive module CORAL.MAC. CORAL.MAC is in [11,10] on your kit. In summary, the interface:

- Provides the Executive with the ability to declare information to a pool monitor task.
- Provides the Executive with the capability to receive a response or acknowledgement from a pool monitor task. (An acknowledgement can be used by the Executive to control further information declarations.)

13.2.2 Pool Conditions

The condition of pool is directly related to major pool events and also to the acceptable high and low pool limits you specify. (You set and manipulate these limits using either the MCR or VMR SET /PLCTL command described in Section 13.4). When a major pool event occurs and causes the size of free pool to cross the low or high limit, a new pool condition exists. The possible pool conditions are low pool and high pool.

A high pool condition exists when the amount of free pool is sufficient to support current and less demanding system workloads.

A low pool condition exists when the amount of free pool is nearing the point where system performance will seriously degrade if the system's workload increases.

Major pool events cause either a high or low pool condition to exist. In turn, a high or low pool condition, combined with a specific pool fragmentation level, creates either a high or low pool state. Pool states are discussed in Section 13.3.

13.3 THE POOL MONITOR TASK

The Pool Monitor Task (PMT) is assembled, task-built, and installed during the SYSGEN procedure. Whenever the resultant system is booted, PMT is activated by the Executive.

(If you do not wish to use the DIGITAL-supplied Pool Monitor Task, you do have the option to write and run your own pool monitor task. If you wish to write your own pool monitor task, you should first be familiar with the Executive pool monitor interface. The interface is documented in the Executive module [11,10]CORAL.MAC. A task installed on a system that supports pool monitoring and also has the task name "PMT..." is considered to be a "pool monitor task".)

PMT's first responsibility is to monitor pool conditions. The conditions that PMT responds to include those detected and reported by the Executive (see Section 13.2). PMT's second responsibility is to monitor pool fragmentation. Through this monitoring, PMT can respond to another major pool event and also react to extreme pool fragmentation. The other major pool event (not reported by the Executive) occurs when PMT detects that the largest free pool fragment is less than the minimum required fragment size. (See Section 13.3.4 for further discussion of extreme fragmentation.)

Once it has been activated by the Executive, PMT basically operates in a cycle. Whenever it receives the Executive's notification of a pool condition, or at least at regular time intervals, PMT determines the size of the largest pool fragment. It then compares this size to the minimum required fragment size. Based on the Executive's report on the current pool condition and what PMT detects when monitoring fragmentation, PMT determines what the current state of pool is and responds to it accordingly.

The minimum required fragment size is set using the MCR and VMR SET /PLCTL command. See Section 13.4 for further discussion of the minimum required fragment size.

13.3.1 Conditions Affecting PMT

PMT responds to pool fragmentation and to pool conditions; however, the response varies depending upon the current level of pool

fragmentation. The two levels that influence PMT's response are as follows:

- If the largest fragment is equal to or greater than the minimum required fragment size, PMT then acts only on the condition of pool as reported by the Executive.
- If the largest fragment becomes smaller than the minimum required fragment size, PMT considers this to be a low pool event. If the system is not already in a low pool state, this event causes a transition to that state. PMT then responds accordingly.

Although extreme fragmentation is another pool event that PMT responds to, it is different from those previously described. It is defined and documented in Section 13.3.4.

PMT uses the information it has concerning the current level of fragmentation and also the information provided by the Executive to determine the overall state of pool. Following is a discussion of PMT's response to specific pool events and conditions.

13.3.1.1 **Pool Level and Fragment Size Are Satisfactory** - The Executive notifies PMT that the pool level is above the high limit and PMT detects that fragment size is greater than the minimum required fragment size.

PMT concludes that a high pool state exists and executes actions appropriate for a high pool state (see Section 13.3.2).

13.3.1.2 **Low Pool Condition** - The Executive notifies PMT that a low pool condition is achieved.

PMT determines whether the change in pool reported by the Executive is temporary. The change is temporary if, by the time PMT responds to the Executive's notification, enough free pool has accumulated to push the total above the high limit. If the change is temporary, PMT ignores the notification and continues to execute high pool state actions. If the change is not temporary, PMT concludes that a low pool state exists and executes actions appropriate for a low pool state (see Section 13.3.3).

13.3.1.3 **High Pool Condition** - The Executive notifies PMT that a high pool condition is achieved.

PMT determines whether the change in pool reported by the Executive is temporary. The change is temporary if, by the time PMT responds to the Executive's notification, the total amount of free pool has moved below the high limit. If the change is temporary, PMT ignores the notification and continues to execute low pool state actions. If the change is not temporary, PMT initiates actions appropriate for a high pool state (see Section 13.3.2).

13.3.1.4 **Largest Pool Fragment Is Too Small** - PMT detects that the largest pool fragment is smaller than the minimum required fragment size.

PMT concludes that a low pool state exists and executes actions for a low pool state. Note that this low pool state overrides any pool conditions reported by the Executive. This pool event will cause the system to be in a low pool state even if the Executive reports a high pool condition.

13.3.1.5 **Largest Pool Fragment Becomes Larger** - PMT detects that the size of the largest pool fragment has become larger than the minimum required fragment size.

PMT decides that the current pool state must be based upon the latest pool condition reported by the Executive. If the Executive reported a high pool condition, PMT decides that a high pool state exists. If the Executive reported a low pool condition, PMT decides that a low pool state exists. In either situation, PMT takes the appropriate action for that state.

13.3.2 PMT Default Actions for a High Pool State

When PMT concludes that a high pool state exists, it executes the following actions:

- Sets its internal cycle interval to one minute
- Enables checkpointing for itself
- Removes any pool use controls

PMT then stops itself so that its impact on system resources is minimized. PMT unstops itself when the internal cycle interval has elapsed.

13.3.3 PMT Default Actions for a Low Pool State

When PMT concludes that a low pool state exists, it executes the following actions:

- Disables checkpointing for itself.
- Sets its internal cycle interval to 20 seconds.
- Establishes pool use controls as follows:
  a. Prevents nonprivileged users from logging on.
  b. Suppresses INSTALL/RUN/REMOVE sequences on nonprivileged terminals.
  c. Delays nonprivileged task loading.

NOTE

Nonprivileged tasks with a task priority less than the specified base priority cannot compete for memory resources until a high pool state exists (see Section 13.4.3). This does not mean that

such tasks cannot be requested; they will simply remain blocked from execution until PMT concludes that a high pool state exists. All other tasks, including any tasks already active at the onset of the low pool state, are not affected by this control.

d. Sends a warning message to all logged on terminals (or on-line terminals on systems without multiuser-protection). This message cannot be suppressed by the MCR SET /NOBRO command or the DCL SET TERM /NOBROADCAST command.

e. Sends a warning message to and displays pool statistics at the console terminal.

13.3.3.1 **PMT Output to Terminals During Low Pool State** - When PMT determines that a low pool state exists, PMT sends the following warning message to logged on terminals (or on-line terminals on systems without multiuser protection):

```
08-AUG-81 13:35:10 -- Low pool -- Please exit active tasks - ZEPHYR
```

The effectiveness of this message depends on whether users exit from their active tasks. If they choose to ignore this message, a low pool state continues to exist.

ZEPHYR is the system name selected during SYSGEN or the DECnet node name (if DECnet is running on your system).

PMT sends this message at 20-second intervals until it concludes that a high pool state exists.

13.3.3.2 **PMT Output to the Console Terminal During Low Pool State** - In a low pool state, PMT also sends a warning message to and displays pool statistics at the console terminal. The following is an annotated example of PMT output to the console terminal:

```
08-AUG-81 13:35:10 -- Warning -- Pool is critically low (1)
Total free pool = 1116. bytes (2)
Largest fragment = 112. bytes (3)
```

In this example, PMT:

(1) Sends a warning message

(2) Displays the amount of free pool existing at the onset of the low pool state

(3) Displays the size of the largest free fragment of pool existing at the onset of the low pool state

PMT broadcasts this message at 20-second time intervals until it concludes that a high pool state exists.

13.3.4 PMT Default Actions for Extreme Fragmentation

Extreme fragmentation exists when the largest pool fragment is less than 84(10) bytes. In this case, pool is exhausted and it is not possible to free any pool by unsolicited terminal input, for example, issuing an ABORT command. At this point, PMT takes immediate action to prevent total pool depletion by executing the following steps:

- Enters kernel mode, which prevents any other tasks from executing but does allow the system to respond to device interrupts.
- Prevents the invocation of TKTN. (Invoking TKTN at this time depletes the available pool.) It does this by clearing TKTN's special TCB pointer that is stored within the Executive.
- Sends a warning message to the console terminal.
- Prompts you to abort one or more of the tasks displayed. (You must be at the console to free pool in response to the PMT prompts.) After you select a task to abort, PMT returns to user mode until the abort operation is finished.
- Displays a message at the terminal running the task and the console terminal if the task is successfully aborted.
- Displays (upon request) an updated list of abortable tasks at the console terminal.

(On RSX-11M systems 84(10) bytes is the size of a CLI command buffer.)

13.3.4.1 **Abortable Tasks** - The abortable task list includes only those tasks that PMT considers eligible to abort. A task is abortable if it meets the following requirements:

- It must be active.
- It must be resident in memory.
- It cannot be privileged and running from the console terminal.
- It cannot be a Command Line Interpreter (CLI) task.
- It cannot be an Ancillary Control Processor (ACP) task.
- It cannot be the task loader.
- It cannot have any internal conditions that prevent it from exiting upon an abort request (for example, it is already aborted).

The ability to abort tasks included in this list is a powerful feature and one that is potentially harmful to your system. You should abort tasks that consume pool and are relatively expendable. This action should release pool resources for use by the remaining active tasks and allow the system to continue operation, rather than deadlock over pool resources.

It is important that you use discretion when selecting tasks to abort. Aborting certain tasks could render all or part of the application unusable. You can abort certain tasks, for example, PIP or MAC, and experience relatively few side effects. As an added measure of protection for your system, PMT does not, by default, include in the abortable task list the privileged tasks running from the console terminal. Therefore, if it is possible, you should run the privileged tasks that you do not want considered abortable (by PMT) from the console terminal.

13.3.4.2 **PMT Messages** - When extreme pool fragmentation occurs, PMT displays a message in the following format and a list of abortable tasks at the console terminal:

```
dd-mmm-yy hh:mm:ss -- Warning -- Free pool exhausted
Abortable tasks in memory:

tttttt   P   I/O=xxx.    ttnn:
```

where:

tttttt	The name of the abortable task
P	The indicator that the task is privileged (a blank means that it is nonprivileged)
xxx	The total outstanding I/O count for the task
ttnn	The terminal where the task is running from

PMT lists the tasks in order of decreasing priority but does not display the priority of individual tasks.

PMT then prompts you to select one of the tasks from the list as follows:

```
Enter a task to abort, or press RETURN to exit:
```

If the task you specify is not installed or not on the list of abortable tasks, PMT displays one of the following messages:

```
Task not installed, try again:

Task not aborted, try again:
```

PMT displays the list of abortable tasks if you press LINE FEED in response to either prompt. If you press RETURN, PMT relinquishes control of the system, by returning to user mode, for at least seven seconds, giving the system an opportunity to recover from the low pool state unassisted.

If PMT is not able to abort the task you specify and the task is listed as abortable, PMT displays the following message (this only occurs if the task exits before PMT is able to abort it):

```
Task not aborted, try again:
```

If PMT successfully aborts the task, it sends the following message to the console terminal:

```
"tttttt" aborted
```

Also, if the task was not running from the console terminal, PMT displays the following messages at the terminal where the task is running from:

```
08-AUG-81 13:35:10 -- Warning -- Free pool exhausted

    "tttttt" aborted
```

PMT then sends the following message to the console terminal:

```
Press LINE FEED for an updated task list,
enter a task to abort, or press RETURN to exit:
```

13.3.4.3 **Annotated Example of PMT Output for Extreme Fragmentation** - An annotated example of PMT output for extreme fragmentation is shown in Example 13-1. The numbers in the example correspond to the numbered items in the list that follows Example 13-1.

Example 13-1 PMT Output for Extreme Fragmentation

```
8-AUG-81 13:42:23 -- Warning -- Free pool exhausted  (1)

Abortable tasks in memory:  (2)

RMDEMO    P   I/O=  0.   TT0:
COT...    P   I/O=  0.   TT0:
AT.T30    P   I/O=  1.   TT30:
EDIT14        I/O=  1.   TT14:
MAIT20        I/O=  1.   TT20:
TEST          I/O=  0.   CO0:
PIPT21        I/O=  1.   TT21:
MACT47        I/O=  1.   TT47:

Enter a task to abort, or press RETURN to exit: PIPT21  (3)

    "PIPT21" aborted  (4)

Press LINE FEED for an updated task list,
enter a task to abort, or press RETURN to exit: LF  (5)

Abortable tasks in memory:  (6)

RMDEMO    P   I/O=  0.   TT0:
COT...    P   I/O=  0.   TT0:
AT.T30    P   I/O=  1.   TT30:
EDIT14        I/O=  1.   TT14:
MAIT20        I/O=  1.   TT20:
TEST          I/O=  0.   CO0:
MACT47        I/O=  1.   TT47:

Enter a task to abort, or press RETURN to exit: EDT  (7)

Task not installed, try again: EDIT14  (8)

    "EDIT14" aborted  (9)

Press LINE FEED for an updated task list,
enter a task to abort, or press RETURN to EXIT: RET  (10)
```

In this example, PMT:

❶ Sends a warning message to logged on terminals (or on-line terminals if the system does not support multiuser protection).

❷ Displays the list of abortable tasks at the console terminal.

❸ Prompts for a task to abort (PIPT21 is entered).

❹ Aborts "PIPT21".

❺ Prompts for an updated task list, a task to abort, or an exit. (A LINE FEED is entered for an updated task list.)

❻ Displays the list of abortable tasks.

❼ Prompts for a task to abort (EDT is entered).

❽ Indicates that a task with the name "EDT" is not installed, and requests that you try again (EDIT14 is entered).

❾ Aborts "EDIT14".

❿ Prompts for an updated task list, a task to abort, or an exit. (The RETURN key is pressed for an exit.)

13.3.5 PMT Task-Build Options

You can modify PMT's response to pool conditions by editing PMT's task-build file PMTBLD.CMD and re-task-building PMT. If you wish to modify PMT, you must:

1. Remove PMT (the task name is PMT...) by using VMR
2. Edit the task-build file
3. Re-task-build PMT
4. Install the modified task image by using VMR

When the system is rebooted, the new PMT task will begin executing. If, however, you want the modified task image to run immediately, you must first abort PMT before you remove and reinstall it using MCR or DCL.

The PMT features that you can modify are:

- Timing control
- Actions
- Pool use controls

The specifics of modifying PMTBLD.CMD are documented in the file itself. Following is a discussion of the PMT features that you can modify.

13.3.5.1 **Controlling PMT Timing** - PMT basically operates in a cycle. It becomes unstopped at regular intervals to monitor fragmentation and then stops itself. When the Executive interrupts the cycle to report a new pool event, PMT resets the cycle to begin at the point at which it was interrupted.

PMT's cycle can have one of two possible time lengths. The length depends on the current state of pool. If a low pool state exists, the cycle is 20 seconds (plus the time PMT requires to execute all low pool state actions). If a high pool state exists, the cycle is 60 seconds (plus the time PMT requires to execute all high pool state actions).

You can modify PMT's cycle; however, the only modification allowed is to lengthen the high pool state cycle. The 60-second value is not only the default value but also the minimum value that you can specify. If you lengthen the cycle, PMT may not be as effective at monitoring pool fragmentation problems. You can partially compensate for this, by using the SET /PLCTL command (see Section 13.4.2) to increase the minimum size of the largest pool fragment parameter. This ensures that PMT will detect imminent pool fragmentation problems earlier.

13.3.5.2 **Controlling PMT Action** - You can modify some of the actions that PMT executes during the cycle. The actions are as follows:

- Sending warning messages to logged on terminals (or on-line terminals on systems that do not support multiuser protection)
- Forcibly checkpointing stopped tasks in memory
- Aborting certain privileged tasks

When a low pool state exists, PMT sends warning messages to logged on terminals (or on-line terminals on systems that do not support multiuser protection). You can suppress this action by modifying PMTBLD.CMD and re-task-building PMT.

When a low pool state exists, PMT requests the Executive to forcibly checkpoint all memory-resident stopped tasks without outstanding nonbuffered I/O. These tasks include CLI tasks, ACP tasks, and tasks with outstanding buffered I/O, only if they are stopped during a low pool state. When the Executive checkpoints a task, it deallocates the task header, thereby reducing fragmentation.

You can modify PMTBLD.CMD to enable such checkpointing for a high pool state as well. This reduces the effect of long-term fragmentation, making it more likely that a long-lived data structure will be allocated to a lower address within pool. This clusters the long-lived data structures together, leaving larger blocks of free pool available for more dynamic types of allocation. Although this can increase the effective use of available pool space, it also causes a noticeable increase in disk I/O as tasks are checkpointed. CLI tasks, ACP tasks, and tasks with outstanding buffered I/O are exempt from checkpointing while a high pool state exists.

The remaining option concerns PMT's list of abortable tasks. During extreme fragmentation, PMT displays a list of abortable tasks and prompts for tasks to abort. By default, PMT does not list or abort privileged tasks running from the console terminal. You can override this default by modifying PMTBLD.CMD. There are, however, exceptions to this option. These exceptions, CLI tasks, ACP tasks, and the task loader, are never listed by PMT as abortable tasks.

13.3.5.3 **Controlling Pool Use** - When a low pool state is in effect, PMT executes the default pool use controls described in Section 13.3.3. You can suppress any of these controls by modifying PMTBLD.CMD and re-task-building PMT.

13.3.6 Aborting PMT

If you wish to terminate PMT, you can easily abort it. You must do one of the following to abort PMT:

- Issue an ABORT command from a privileged terminal
- Issue an abort directive from a privileged task

When you select either of these methods, PMT performs the following actions before it exits:

- Restores TKTN's special TCB pointer if PMT previously cleared the pointer and TKTN is still installed (see Section 13.3.4). This restores normal TKTN operations.
- Prevents the Executive's pool monitor code from requesting PMT after PMT exits. This is done by clearing the special TCB pointer for PMT... as set by MCR INSTALL, DCL INSTALL, and VMR INSTALL commands.
- Cancels any outstanding pool use controls that it may have imposed because of a low pool state.
- Displays the following message at the console terminal:

```
Remove and reinstall PMT to restore pool monitoring

    "PMT..." aborted
```

  This indicates that PMT's special TCB pointer has been cleared so that the Executive can no longer request PMT to run. To restore the pointer, you must remove and reinstall PMT using MCR or DCL.

13.4 THE SET /PLCTL COMMAND

The SET /PLCTL command sets the pool limit parameters used by the Executive pool monitor code and PMT. The syntax, parameters, and parameter limits for the command are discussed in Chapter 10 and also in the RSX-11M/M-PLUS MCR Operations Manual. Basic guidelines for setting the four command parameters are discussed in the following subsections.

During a low pool state, you can compare the values you set and the information that is output to the console terminal (see Section 13.3.3.2) to determine whether the parameters require further modification.

13.4.1 Low and High Pool Limits

The first two parameters of the SET /PLCTL command establish the low and high pool limits. The Executive pool monitor code uses these limits to determine when major pool events occur.

In general, you should set the low limit value to the least desirable pool level. This is a limit where your system's workload cannot increase without the risk of a pool resource deadlock. The appropriate value for this limit varies with each system. You can determine the best value by experimenting with various limits. The low limit defaults to 600(10) bytes.

Set the high limit value to reflect the type of use your system receives. The high limit parameter essentially controls the point where a low pool state is transformed into a high pool state. The high limit defaults to 1600(10) bytes.

If pool use fluctuates considerably, a limit set relatively higher than the low limit will tend to reduce the number of pool state transitions. If pool use is typically constant, a high limit set close to the low limit will quickly show critical changes in the pool state. Since pool needs and use vary, you should experiment with this parameter to determine the optimum value.

13.4.2 Pool Fragment Size

This parameter establishes the minimum size of the largest pool fragment. PMT uses this limit to determine when pool fragmentation problems exist. When the largest pool fragment is the minimum size or greater, a high pool state exists -- as far as fragmentation is concerned. A low pool state exists when the largest pool fragment is smaller than the minimum size. The lowest permissible size of the largest pool fragment is 200(10) bytes by default. You should experiment with this parameter to judge the optimum value.

Fragmentation is one of the more frequent and less desirable pool problems. Thus, the value that you specify for this parameter can have an immediate impact on successful pool monitoring.

13.4.3 Base Priority for Nonprivileged Tasks

This parameter establishes the base priority for nonprivileged tasks. The optimum value of this parameter is application specific, but defaults to 51(10). During a low pool state, PMT uses this parameter to restrict newly requested nonprivileged tasks from competing for memory resources (see Section 13.3.3). Only those tasks with a priority less than the specified base priority are so restricted.

CHAPTER 14

THE SHUFFLER

The Shuffler (SHF) is a privileged task designed to compact space in system-controlled partitions when a memory allocation failure occurs. The Shuffler operates only within system-controlled partitions and is available for the RSX-11M/M-PLUS operating systems, but is not available for unmapped systems or for RSX-11S.

This chapter describes Shuffler interaction with the operating system. Briefly, it describes system-controlled partitions. In more detail, it describes Shuffler structure, ways to monitor the Shuffler, and Shuffler algorithms for the operating system.

14.1 EXECUTIVE AND MEMORY USAGE

Memory is the processor storage medium in which the Executive and tasks reside when executing. The Executive is the kernel of the operating system that interfaces the hardware system with other components of the operating system and with the user.

The Executive brings a task into memory for execution and places the task in competition with other active tasks. When tasks are brought into memory, they are placed in a section of memory called a partition. Partitions are contiguous areas of memory in which executable tasks run; however, a common partition may be defined that contains only data or code.

RSX-11M-PLUS supports only system-controlled partitions.

RSX-11M supports three types of partitions:

- System-controlled partitions
- User-controlled partitions
- Common partitions

User-controlled partitions and system-controlled partitions are used to load waiting tasks. User-controlled partitions accommodate only one task at a time. If you want to run multiple tasks within a user-controlled partition, you must define multiple subpartitions.

System-controlled partitions accommodate as many tasks as possible at any one time. The Executive allocates available space in system-controlled partitions and creates the dynamic subpartition for each task.

On RSX-11M-PLUS, tasks, device drivers, dynamic common regions, and static common regions reside in subpartitions of the system-controlled partition. Static common regions are resident libraries and resident commons and both may be shared by tasks for more efficient use of physical memory. Resident libraries cortain code, and resident commons contain data.

On RSX-11M, static common regions must be loaded into common partitions. A common partition is a dedicated area of memory that may be shared by tasks. Tasks, device drivers, and dynamic common regions created by memory management directives must be loaded into subpartitions of the system-controlled partition.

14.1.1 System-Controlled Partitions

Only mapped systems support system-controlled partitions. A mapped system is a system with memory management. A task in a mapped system can run in any system-controlled partition large enough to contain it, and the Executive is responsible for allocating space in that partition. Refer to the RSX-11M/M-PLUS MCR Operations Reference Manual for detailed information on system-controlled partitions and memory management.

The Executive allocates dynamic memory space for waiting tasks in the following manner:

1. Uses a priority level of 1 through 250 to determine a waiting task's use of a system-controlled partition. A higher number indicates a higher priority.

2. Begins at the base of the partition and searches for a free, contiguous area large enough for the waiting task.

3. Creates a dynamic subpartition within the system-controlled partition.

4. Eliminates the subpartition to free memory when the task has finished execution.

In some instances, a waiting task cannot be loaded into a system-controlled partition because there is not enough memory space available. If there is not enough space to load a waiting task, the Executive uses checkpointing to make memory space available to the waiting task.

To make room for a waiting task, the Executive searches the system-controlled partitions for one or more tasks with a lower priority than that of the waiting task. The Executive then checkpoints (moves) the task of lower priority from memory to disk to make room for the waiting task of higher priority. The checkpointed task is immediately placed in competition with other tasks waiting for residency. When the checkpointed task is reloaded, it continues processing from the point where it was interrupted.

On RSX-11M-PLUS, the Executive keeps track of outstanding I/O on a partition-by-partition basis. Thus, the Executive may checkpoint subpartitions with dynamic common regions, static common regions, and task regions if there is no I/O pending.

On RSX-11M, the Executive keeps track of outstanding I/O on a task-by-task basis. The Executive cannot easily determine to which partition outstanding I/O is directed. Therefore, RSX-11M can only checkpoint subpartitions with task regions. Common partitions and dynamic common regions created by memory management directives cannot be checkpointed.

14.1.2 Partition Fragmentation

Checkpointing does not always create enough memory space to load a waiting task. This is a result of partition fragmentation. Fragmentation in system-controlled partitions occurs when tasks of various sizes are entering and exiting memory, or when, for example, device drivers are loaded in the middle of a system-controlled partition. Device drivers, which cannot be checkpointed, often create space between subpartitions and below subpartitions called fragments.

Tasks with nonbuffered, outstanding I/O cause fragmentation of system-controlled partitions. Such tasks cannot be checkpointed. The Executive must wait for the task's I/O count to reach zero before the task can be checkpointed. While the Executive waits for the task's I/O count to reach zero, the task is blocked from execution to prevent the task from issuing more I/O requests that could delay the loading of the waiting task. The following lists the serious causes of fragmentation in system-controlled partitions:

1. Device drivers loaded in the middle of a system-controlled partition
2. Subpartitions that are connected to the IP11 driver
3. Subpartitions that are connected to an interrupt vector by means of a CINT$ directive
4. Subpartitions with nonbuffered, long-outstanding I/O, and whose I/O count has not yet dropped to zero
5. Regions that have been fixed by the Executive for a memory parity error
6. Dynamic regions created by PLAS directives (RSX-11M only)

If, for any reason, including fragmentation, the Executive is unable to load a waiting task into a subpartition, the Executive requests the Shuffler. The Shuffler moves subpartitions in system-controlled partitions in an attempt to create the free space necessary to load a waiting task. Thus, the Executive requests the Shuffler only when all other attempts to load a waiting task fail.

14.2 SHUFFLER PERFORMANCE

The Shuffler is approximately 700(10) words and can be installed only on a mapped system. The Shuffler is installed during system generation by a VMR command and by default is built for the partition SYSPAR, which can also be used by the MCR dispatcher and TKTN. Refer to the RSX-11M or RSX-11M-PLUS System Generation and Installation Guide for more information on installing the Shuffler.

The overhead incurred for the Shuffler task is found in the Shuffler, not in the Executive. The Shuffler responds well on a system if there are fewer tasks loaded in memory and if the Shuffler does not execute frequently. If you have a heavily loaded system, the Shuffler may execute more than you want because of repeated memory allocation failures. If the Shuffler executes frequently on your system, better performance is obtained when the Shuffler is fixed in its own partition where it does not have to compete with MCR and TKTN for memory.

The word, $SHFTM, in the Executive's system common area limits the number of times the Shuffler may be requested within a time interval. $SHFTM heightens the Shuffler's effectiveness on your system. Each time the Executive requests the Shuffler, this timer is checked. If the timer registers zero, the Shuffler may be requested. If the timer registers a positive number, the request for the Shuffler is ignored. The amount of time (clock ticks) that must elapse before the Shuffler is requested again is stored in the system common area, $SHFCT. When the Shuffler is successfully requested, $SHFTM is initialized for the new interval with the value stored in $SHFCT.

If you are a privileged user, you may change $SHFCT. To change $SHFCT, you need a copy of RSX11.MAP, the Executive map produced during system generation. Using the map, you must determine the address of the global symbol $SHFCT. You can change the contents of the system $SHFCT with the MCR OPEN command. The Shuffler is requested infrequently if the value is set too high. If the value is set too low (zero), system response may decrease because the Shuffler is being requested too frequently.

14.2.1 Monitoring The Shuffler

Several memory allocation failures during peak usage periods result in the repeated request of the Shuffler. The repeated execution of the Shuffler can produce a noticeable decrease in response time. You may monitor the Shuffler's effect on your system by using RMD. RMD is a privileged task that displays information concerning task activity on the operating system. Refer to Chapter 6 for a detailed explanation of RMD.

If you want to monitor Shuffler activity with RMD, you must dedicate a video terminal on which to run the RMD display. The key features to monitor on the memory display page are as follows:

1. The currently executing task; *IDLE* indicates that no task is executing

2. The number of active tasks in memory and the amount of memory they occupy, plus the number of active tasks checkpointed (on disk) and the total memory they would require.

These features show task activity. If the Shuffler is active, tasks are moving down in the partition, and the currently executing task field registers the Shuffler as the active task.

Occasionally, the Shuffler registers as the active task, but tasks cannot be seen moving down in the partition. To verify that tasks are not moving when the Shuffler is executing, check the number of active tasks checkpointed (on disk) and the number of tasks waiting for memory. If this count remains the same over a period of time, the Shuffler is executing but cannot create enough space for waiting tasks. Thus, the Shuffler is using too much of the system's resources. If the system does not recover, it might be better to abort the Shuffler from your system.

If the Shuffler task continues to use too much of the system's resources, you should remove the Shuffler from your system with the MCR REMOVE command.

14.3 THE SHUFFLER ALGORITHMS

When a memory allocation failure occurs, the Executive requests the Shuffler. The Shuffler searches memory for a system-controlled partition that has at least one task waiting for space. When the Shuffler locates such a system-controlled partition, it attempts to move (shuffle) subpartitions in the memory partition to create the continuous area necessary to load waiting tasks.

On RSX-11M, the Shuffler can only shuffle task subpartitions. Because RSX-11M does not maintain an outstanding I/O count on a partition-by-partition basis, the Shuffler cannot determine whether outstanding I/O is pending to a common partition or a dynamic common region. Thus, the Shuffler does not shuffle common partitions or subpartitions with dynamic common regions created by memory management directives.

On RSX-11M-PLUS, the Shuffler does shuffle common regions because an outstanding I/O count is maintained on a partition-by-partition basis. Thus subpartitions containing static common regions or dynamic common regions can be shuffled when there is no I/O pending.

The Shuffler uses two algorithms to move subpartitions in memory. Each algorithm executes several steps repeatedly until the partition reaches a stable state.

14.3.1 The First Pass Algorithm

In the first pass, the Shuffler attempts to move each subpartition to the base of the partition. When subpartitions are moved to the base of the partition, free space is created at the top of the partition so that a waiting task can enter memory if there is enough space. Shuffling occurs in the following steps:

1. The Shuffler requests the Executive to unconditionally checkpoint all stopped checkpointable subpartitions and checkpointable subpartitions that have been blocked with a CLI command.

2. The Shuffler attempts to move each subpartition to the base of the partition and blocks each executable task from further execution. If a subpartition has outstanding I/O, the Shuffler waits for the task's nonbuffered, outstanding I/O count to reach zero. If the I/O count does not reach zero in approximately .5 seconds, the Shuffler marks the subpartition as having outstanding I/O, and restarts the first pass algorithm. The Executive clears the subpartition's I/O indicator when the task's I/O count becomes zero.

3. The Shuffler moves each task to the base of the partition. If the subpartition's I/O count reaches zero within 30 seconds.

4. Each subpartition is unblocked to allow further execution after shuffling is complete, and the free space, now above the subpartition, is deallocated.

Be aware that a fixed task can be shuffled. If you do not want to have a fixed task shuffled, you must fix this task in a user-controlled partition or in its own dedicated system-controlled partition.

If fragmentation has not occurred at the end of the first pass, the Shuffler has successfully merged the free space in the partition into one continuous area at the top of the system-controlled partition. A task may be loaded into memory if there is enough available space.

Memory space may not be available if there are several tasks already loaded in memory, or if fragmentation is dividing the system-controlled partition into two or more partition sections.

A partition section is a subset of the area in a system-controlled partition. Each partition section is flanked on at least one side by a subpartition that cannot be shuffled. For example, if a device driver is loaded in the middle of a system-controlled partition, the system-controlled partition is divided into two partition sections because a device driver is a subpartition that cannot be moved.

If during the first pass algorithm, the Shuffler encounters a subpartition that cannot be moved, the Shuffler begins to shuffle subpartitions that can be moved to the base of the partition section that bounds it. The Shuffler cannot shuffle subpartitions around a subpartition that cannot be moved. Therefore, if fragmentation has occurred during the first pass, the system-controlled partition is divided into two or more partition sections, and the free space in the partition may no longer be merged at the top of the partition, but rather at the top of the partition section.

The Shuffler initializes the second pass algorithm if enough memory space is not available for waiting tasks, or if fragmentation has divided the system-controlled partition into several partition sections. Little memory space or partition sections prevent the Shuffler from accumulating all the free space at the top of the partition. The free space in each partition section is separate and cannot be merged with the free space in another partition section.

14.3.2 Second Pass Algorithm

The second pass algorithm is executed once for each partition section within a system-controlled partition. This pass attempts to create a reverse priority list of subpartitions that can be checkpointed by the waiting task. During the second pass, the Shuffler creates a list in ascending priority order of the subpartitions in each partition section. The Shuffler then compares the size of the waiting task with the sum of free space in the partition section plus the total of one or more lower priority, checkpointable tasks. If the waiting task is smaller, the Shuffler requests the Executive to checkpoint the lower priority task(s).

Checkpointing the lower priority task(s) may create new space, but the new space may not comprise a continuous area in memory. The Executive may begin the first pass alogorithm to accumulate the free space of the partition section. The free space is then allocated to the waiting task. As long as there are subpartitions that cannot be moved in a system-controlled partition, memory is still divided into two or more partition sections.

If the Shuffler completes the second pass without finding room for a waiting task, the Shuffler searches for the next system-controlled partition with waiting tasks. If there are no other system-controlled partitions with tasks waiting for memory space, the Shuffler exits.

CHAPTER 15

RSX-11M-PLUS SYSTEM RECONFIGURATION

A reconfigurable system is a system in which you can add or remove system resources such as memory and devices. Reconfiguration is a means of physically and logically connecting and disconnecting various resources. This chapter discusses the purpose and application of reconfiguration, and describes the components and commands of the reconfiguration services.

System reconfiguration is available only on RSX-11M-PLUS systems. Some of the information in this chapter, however, applies only to multiprocessor systems. Blue ink is used to distinguish this information.

15.1 THE RECONFIGURATION SERVICES

The reconfiguration services allow you to reconfigure an RSX-11M-PLUS system to bypass faulty hardware elements and to isolate the system from the effects of these elements.

With the reconfiguration services, you can define a set of hardware resources that are accessible from the on-line system and also remove devices from the pool of resources allocated to the on-line system. For example, after booting the system, you can place a failed disk drive off line and then use another drive that is already on line or place one on line to take over for the disabled unit.

15.1.1 Reconfiguration Components (CON,RD:,HRC)[1]

You reconfigure system resources by using together a command interface (CON), a loadable driver (RD:), and a privileged reconfiguration task (HRC). HRC and CON must have enough checkpoint space allocated for them both to be resident in memory at the same time. If the operating system cannot load both tasks at the same time, CON commands will fail.

CON, as the command interface, receives and parses the commands that you issue to reconfigure the current system. CON then creates and submits QIO packets to the reconfiguration driver (RD:).

---

1. DIGITAL will accept SPRs regarding the interface but reserves the right to treat such SPRs as informational only.

RD:, the reconfiguration driver, serves as an interface between a user task (either CON or a user application task) and the HRC task; it is the gateway between tasks that generate reconfiguration requests and the actual implementation of those requests. RD: accepts QIO packets from any task and places these packets into the receive queue of HRC.[1] Before passing the QIO packets to HRC, RD: performs privilege and address boundary checks on them. The checks ensure that the packets are using valid addresses and legal function codes, and, for some functions, that RD: is attached to them.

HRC, the privileged reconfiguration task, performs the actual reconfiguration operations for the system. To process reconfiguration requests, HRC calls Executive routines, which then call the associated device drivers. HRC services the requests for configuration information by passing the desired information directly to the user buffer.

After HRC services the request, a status is returned (by means of I/O completion) to the requesting task. The status return was implemented in part to allow tasks other than CON to interface with HRC.

The data base for each system device contains status words that reflect device state. When you change a system device state, HRC modifies the appropriate data base status word to either grant or inhibit device access.

15.1.2 Device States

The RSX-11M-PLUS system reconfiguration services control the configuration state (on line or off line) of the hardware devices, and therefore control device access. A system device can assume one of four states:

- On line - device on line, access granted
- Off line - device off line, access denied
- Marked for on line - device will be brought on line when access path exists, access denied
- Marked for off line - indeterminate state, access denied

In the on-line state, a device is logically on line for access. This state indicates that an access path exists between the device and an on-line processor and that a call (if any) to the driver's controller or unit status change entry points was successful. RSX-11M-PLUS grants access to on-line devices only. To access an on-line disk or tape volume and perform virtual I/O on it (for example, reads or

---

1. Note that programs written to use the current version of RD: may have to be modified in the next release. You should centralize all code concerning RD: so that modifications can be made easily.

Because of the likely changes, RD: has not been documented. For some information, check the reconfiguration sources located in UFD [27,10] of the distribution kit (except the RL02 kit, which does not contain any sources). Important modules for the interface are CNHRC, HROOT, and HRSUB. (See also the macro OLRDF$ in [1,1]EXEMC.MLB for a description of volume access rights and mounted volumes.)

writes to files), the volume must be mounted. (See the RSX-11M/M-PLUS MCR Operations Manual or the RSX-11M/M-PLUS Command Language Manual for a description of volume access rights and mounted volumes.)

When a device is in the off-line state, the system denies access to the device. Attempts to access an off-line device result in an I/O error (IE.OFL). (See the IAS/RSX-11 I/O Operations Reference Manual for lists and explanations of error codes.) Taking a controller off line also takes all of its associated units off line, which places the units in the marked for on-line state.

The marked for on-line state indicates that a request to bring a device or controller on line has been made, but part of the access path to the device or controller is off line. Access attempts yield the error code IE.OFL. HRC places a device marked for on line logically on line when it brings the entire access path on line.

The marked for off-line state indicates that the system encountered an unexpected error in attempting to place a specific device logically on line. You must issue an explicit off-line command before attempting or retrying an on-line operation.

15.1.3 Placing Devices Logically On Line and Off Line

To place a device logically on line, HRC first verifies that the device driver is loaded (for loadable drivers only) and then tests whether the controller can access the device (the controller must be on line). If the controller can access the device (an access path exists), HRC modifies the appropriate status word to reflect that the device is on line. If the controller or device cannot be accessed, HRC marks the device for on line and then places the device on line when the access path is brought on line.

Before placing a device off line, HRC attempts to protect against inadvertent loss or possible corruption of data by verifying that the state change will not interfere with active I/O. HRC considers a device active and rejects off-line commands if the device has context. The reconfiguration services define device context as a condition that indicates (or permits) I/O activity on the device. The following is a list of conditions that define device context:

1. A task is attached to the device (for example, the print despooler attaches the line printer).
2. The device (or an associated device) is a logged-in terminal.
3. The device (or an associated device) contains a mounted volume.
4. For the DT07 bus switch, a port is logically connected to a processor (linked).
5. For memory boxes, the memory within the box is logically on line (multiprocessor systems only).

You cannot take a controller off line if any of the units on the controller have context.

For dual-port disks (RK06/07, RM02/03/05/80/RP07, and RP04/05/06), which have each port connected to a different controller, you can place off line one path with context as long as the alternate path is on line. (See the description of the OFFLINE command for more information.)

15.2 INITIATING THE RECONFIGURATION SERVICES

You reconfigure an RSX-11M-PLUS system interactively from a terminal or by means of indirect command files. The reconfiguration services act as an interface between your terminal and the RSX-11M-PLUS system.

To issue a reconfiguration command, type a command line in response to the default prompt (>) or to the explicit command interface prompt (CON>). A command line consists of three components: a command name, any required parameters, and a line terminator. (Note that there is no DCL command to access CON.)

15.2.1 Invoking CON

It is not necessary to use the entire command name when issuing a command line; CON requires only the first three characters of the name. However, to make the intent of the command input clear to someone reading a printed copy, you can use the entire command name. For example, you can issue the ONLINE command as follows:

```
CON>ONL dev-spec RET
```

or

```
CON>ONLINE dev-spec RET
```

dev-spec

The target device.

RET

The RETURN key (the line terminator).

A space or tab signals the end of the command name. Therefore, no spaces or tabs are permitted within the name itself. To use the reconfiguration services, first invoke the command interface CON and then issue the commands. There are two ways to invoke CON interactively:

1. Use the 1-line format to invoke CON, execute a single command, and return control to MCR. For example:

   ```
   MCR>CON command-line RET
   ```

2. Use the direct task format to invoke CON and obtain a task prompt (CON>). For example:

   ```
   MCR>CON RET
   CON>
   ```

   Type CTRL/Z to exit from the direct task format.

You can also enter commands through indirect command files. An indirect command file that contains reconfiguration commands does not require terminal interaction when the file is executed. Thus, indirect command files are a convenient means of creating configuration command sequences that create different configurations or restore previous configurations.

For example:

```
MCR>CON @CONFIG RET
```

This command line invokes CON, executes the reconfiguration commands contained in the indirect command file CONFIG.CMD, and returns control to MCR.

The command line

```
CON>@CONFIG RET
```

executes the commands in the file CONFIG.CMD and returns control to CON.

Indirect command files can be nested to a maximum depth of three.

15.3 CON COMMAND SUMMARY

Table 15-1 summarizes the reconfiguration commands and provides a compact overview of the commands. Detailed command descriptions appear in alphabetical order in Section 15.4.

Table 15-1
CON Command Summary

BUILD	Creates a command sequence in an internal buffer that, if executed, duplicates the current system configuration.
CLEAR	Erases the command sequence created by BUILD (but not the file created by LIST) that is stored in the internal buffer by CON.
DISPLAY	Displays the configuration and status of the hardware devices in the current system.
ESTATUS	Emits the current status of a specific device.
HELP	Displays the help text for CON.
IDENT	Displays the current versions of CON and HRC and the date and time the two tasks were built.
LINK	Logically connects a port of a switched bus run to a processor.
LIST	Displays the results from a BUILD command or puts the command sequence into a specified file. A file type of .CMD is suggested because that is the default type for an indirect command file.

(continued on next page)

Table 15-1 (Cont.)
CON Command Summary

OFFLINE	Removes a device without context from the active set of devices in the current configuration.
OFFLINE MEMORY	(Multiprocessor systems only.) Removes the memory box with the highest addresses from the system. Note that you must first ensure that nothing is using the memory in the memory box.
ONLINE	Attempts to place a device logically on line.
ONLINE MEMORY	(Multiprocessor systems only.) Adds a memory box to the on-line configuration. Note that you must expand the partition structure to use the additional memory.
SET	Alters the CSR or interrupt address values of a device and, on multiprocessor systems, activates or inhibits the sanity timer and its alarm.
SWITCH	(Multiprocessor systems only.) Logically disconnects a port of a switched bus run from a processor and then connects the port to another processor. The command is equivalent to a LINK command followed by an UNLINK command.
UNLINK	Logically disconnects a port of a switched bus run from its respective processor.

CON also accepts two switches, /HE and /NOMSG. These switches are used alone in the CON command line. Their functions are as follows:

/HE	Displays the help text for CON.
/NOMSG	Suppresses all information displays and error messages returned by CON. You must exit from and then reenter CON to have the messages be displayed again.

15.4 CON COMMAND DESCRIPTIONS

This section describes each of the reconfiguration commands. Note that some of the commands must be issued from a privileged terminal.

BUILD

The BUILD command creates a sequence of commands that, when executed, duplicates the current system configuration. The sequence of reconfiguration commands resulting from a BUILD command is held in a buffer in memory. When this sequence of commands executes, the reconfiguration commands restore the system to the exact state existing when the BUILD command was issued.

Format

```
CON>BUI[LD]
```

Example

```
CON>BUILD
```

This command creates a sequence of commands that duplicates the current system configuration.

CLEAR

CLEAR

The CLEAR command erases from the internal buffer the current sequence of commands created by the BUILD command. The command does not delete the file created by the LIST command (described later in this section).

Format

```
CON>CLEAR
```

Example

```
CON>BUILD
CON>CLEAR
```

The CLEAR command removes from the internal buffer the command sequence resulting from the previous BUILD command.

DIS

DISPLAY

The DISPLAY command provides current system configuration information about the device units, device controllers, processors, memory boxes, and memory box controllers.

Format

```
CON>DIS[PLAY] [keyword(s)] [FOR string]
```

Keywords

UNI[TS]

Displays every device unit in the current configuration. The display equates each device unit with its respective controller.

CON[TROLLERS]

Displays every device controller in the current configuration.

FUL[L]

Displays the status flags for every device unit and device controller in the current configuration. The status flags indicate the following:

Accpath	The device has an on-line access path.
Context	For a tape or disk (mountable device), the device contains a mounted volume. For a terminal, a user is logged in. Other devices are attached.
Driver	For a device unit, the driver is loaded. For a device controller, a driver is available.
Massbus	The device is connected to an RH MASSBUS controller.
Multiport	The device is dual-ported (a dual-access device) between two controllers (for example, a dual-ported RP06 drive).
Offline	The device is off line.
Online	The device is on line (or marked for on line).
Prv-Diag	(Used with the OFFLINE and ONLINE keywords.) The device is marked for on line or off line, but nothing can happen until the respective controller is brought on line or taken off line.
Unknown	The CSR assignment for the device is 160000 and thus cannot be accessed.

DIS (Cont.)

ALL

Same display as for the FULL keyword with the addition of interrupt vector addresses and CSR assignments for device controllers.

ATT[RIBUTES]

Displays interrupt vector addresses and CSR assignments for device controllers.

The DISPLAY command allows more than one keyword per command line. However, some combinations (such as DISPLAY UNITS FULL FOR DB) are meaningless. The DISPLAY command ignores an inappropriate keyword.

If the command omits a keyword, the display contains the logical names and status flags for every device unit and device controller in the current configuration.

FOR string

An optional parameter that allows a selective display of a particular device or device group. When you use this parameter, CON displays the requested information for the devices specified in the string.

You can also use one or two wildcard characters (*) in the string. You use one wildcard to find out which devices are connected to the controller you have specified after the wildcard. You use two wildcards to find out which devices have the attribute you have specified after the wildcards (either ONLINE or OFFLINE).

For example:

```
CON>DIS FULL FOR RH
```

This command displays all of the RH controllers.

```
CON>DIS FULL FOR **ONL
```

This command checks to see which devices have the ONLINE attribute (port-specific status is not displayed).

Examples

```
CON>DISPLAY FULL FOR DB
DB0:    Offline,Multiport,Massbus,Accpath,Driver
 PORT DB0A RHB0:        Online
 PORT DB0B RHC0:        Online,Current,Accpath
DB1:    Offline,Multiport,Massbus,Accpath,Driver
 PORT DB1A RHB1:        Online
 PORT DB1B RHC1:        Online,Current,Accpath
```

This command displays information about all of the DB:-type devices configured into the system.

DIS (Cont.)

```
CON>DISPLAY UNITS
DB0:    RHB0:,RHC0:
DB1:    RHB1:,RHC1:
DR0:    RHA0:
DR1:    RHA1:
MM0:    RHD0_0:
MM1:    RHD1_1:
CO0:
TT0:    YLA0:
TT1:    YZA0:
TT2:    YZA1:
TT3:    YZA2:
VT0:
DK0:    DKA0:
DK1:    DKA1:
DL0:    DLA0:
LP0:    LPA0:
LP1:    LPB0:
LP2:    LPC0:
NL0:
DM0:    DMA0:
DM1:    DMA1:
```

This command displays all of the device units and controllers configured into the system.

ESTAT

ESTATUS

The ESTATUS command causes CON to exit with its exit status word reflecting the current status of a specified device. Used with the Indirect Command Processor (Indirect; see the RSX-11M/M-PLUS MCR Operations Manual), the command returns a 16-bit word describing device status. You can use the status word to obtain information about the configuration. The meanings of the bits in the word are given in Table 15-2. Note that the meanings for units and controllers are different from those for ports.

Table 15-2
Exit Status Values for CON ESTATUS

If the status is for a unit or controller:

Bit No.	Bit Value	Description	Display
0 1 2	1 2 4	Exit status value: 0 = Warning 1 = Success 2 = Error 4 = Severe error	
6	100	This is a subcontroller device.	
7	200	This device is a controller.	
8	400	0-device is on line, 1-device is off line.	Online Offline
9	1000	Device is restricted to privileged and diagnostic functions.	Prv_Diag
10	2000	This is a multiport device.	Multiport
11	4000	This is a MASSBUS device.	Massbus
12	10000	This device's CSR is 160000 (unknown).	Unknown
13	20000	This device has an on-line access path.	Accpath
14	40000	This device has context.	Context
15	100000	A driver is loaded for this device.	Driver

If the request is for a port:

Bit No.	Bit Value	Description	Display
0 1 2	1 2 4	Exit status value (see above)	
8	400	0-port on line, 1-port off line.	Online Offline
10	2000	This is the current port.	Current
13	20000	There is an on-line access path to this port.	Accpath
14	40000	This port or the device it services has context.	Context

ESTAT (Cont.)

Example

```
>@TI:                          ! Invoke Indirect.
AT.>.ENABLE SUBSTITUTION       ! Enable substitution mode.
AT.>CON ESTAT DB1:             ! Type CON command line.
>CON ESTAT DB1:               ! System displays command line.
AT.>;'<EXSTAT>'                ! Request contents of <EXSTAT> to be displayed.
>;126401                       ! System displays current <EXSTAT> contents.
AT.>^Z                         ! Exit Indirect
>@ <EOF>
>
```

Using Indirect interactively, the CON command line emits the current status of DB1:, which is stored in the Indirect special symbol <EXSTAT>.

The number 126401 has the following meaning:

```
100000    Device's driver loaded.
 20000    Device has on-line access path.
  6000    Device is both multiport and MASSBUS.
   400    Device is off line.
     1    Exit status value = Success.
```

HE

HELP

The HELP command displays the help text for CON.

Format

```
CON>HE[LP]
```

Example

```
CON>HELP

CON -- Configuration Control Command Task
Implemented commands:

ONLINE <device_list>
ONLINE MEMORY <memory_box>
ONLINE ALL
OFFLINE <device_list>
OFFLINE MEMORY <memory_box>
OFFLINE ALL
SET <controller_sp

(User typed CTRL/O to suppress further output.)
```

IDENT

IDENT

The IDENT command displays the current version numbers of the CON and HRC tasks. The date and time the tasks were task-built are also displayed.

Format

```
CON>IDENT
```

Example

```
CON>IDENT

CON -- CONFIGURE Version 4.00
Task built: 1-DEC-81 12:52:43.5

HRC -- HRC Version 004.000
Task built: 15-DEC-81 23:59:10.9
```

LIN

LINK

The LINK command logically connects a port of a switched bus run to a processor and thus permits access to the devices on the switched segment of the bus run.

To logically connect a switched bus run to the processor's UNIBUS, the devices on the switched segment must all be off line or marked for on line. You cannot dynamically link switched bus runs.

Format

CON>LINK UBx TO CPx

UBx

The target-switched bus run. Switched bus runs are named using the designation UB followed by a letter from the DIGITAL hardware alphabet as follows: UBE, UBF, UBH, UBJ, UBK, UBL, UBM, UBN, UBP, UBR, UBS, and UBT.

CPx

The target processor.

Example

```
CON>LINK UBE TO CPA
```

This command logically connects the first switched bus run (UBE) to processor A (CPA).

LIST

LIST

The LIST command displays the current command sequence or writes the sequence to a file. You can use this command to keep a list of the commands that will duplicate the current configuration. If the buffer in which the current command sequence is stored (see the description of the BUILD command) is empty, CON returns the following message:

```
CON -- Command list is empty. Nothing to print
```

If you specify a .CMD file type, you can execute the commands from the indirect command file (that is, BUILD, LIST filename.CMD, @filename) and then return to CON.

Format

```
CON>LIST [filespec]
```

filespec

Any legal RSX-11M-PLUS file specification in the form ddnn:[ggg,mmm]filename.type;version.

Example

```
CON>BUILD
CON>LIST
ONLINE  RHA
ONLINE  RHB
ONLINE  RHC
OFFLINE DMA
OFFLINE YLA
ONLINE  YZA
ONLINE  DKA
ONLINE  DLA
ONLINE  LPA
ONLINE  DB0:
ONLINE  DB1:
ONLINE  DR0:
ONLINE  DR1:
ONLINE  MM0:
ONLINE  MM1:
ONLINE  CO0:
ONLINE  TT0:
ONLINE  TT1:
ONLINE  TT2:
ONLINE  TT3:
ONLINE  TT4:
ONLINE  TT5:
ONLINE  TT6:
ONLINE  TT7:
ONLINE  TT10:
ONLINE  DK0:
ONLINE  DK1:
ONLINE  DL0:
ONLINE  LP0:
ONLINE  NL0:
ONLINE  DM0:
ONLINE  DM1:
```

OFF

OFFLINE

The OFFLINE command changes the state of a quiet device to off-line, thus removing the device from the active set of resources in the current system configuration. This prevents further device access. To take a device off line, it must not have context (see Section 15.1.3), and HRC must be able, in 1000(10) tries or less, to catch the device when there is no I/O activity in progress.

The transition to the off-line state for a specific device controller is completed only if all associated devices are also off line. If an associated device is not off line, HRC rejects the OFFLINE command because placing the device off line could interfere with active I/O.

Issue the appropriate MCR command(s) to remove device context. This ensures that the device does not have context and can be placed off line.

If a task has attached the device unit, the task must be detached (or request to be aborted) before the unit can be placed off line.

For example, if you issue an OFFLINE command for a line printer controller (such as OFF LPB), and a connected unit (such as LP1:) has context (in this case, LP1: is a spooled device and attached by the Queue Manager), CON rejects the command and issues the following error message:

```
**Unable to take unit with context offline
```

If the device is marked for on line (for example, a device unit with its controller off line), the command changes the device state to off-line. Subsequently, when the controller is brought on line, the system does not activate the device.

If the device is marked for off line, the command changes the device state to off-line and thus enables a future on-line attempt.

Individual ports of dual-port devices can be taken off line separately. You may want to do this if the port does not work correctly (for example, you cannot access the device through the port) or if you want to increase system throughput. Figure 15-1 is an example of the way throughput could be increased:

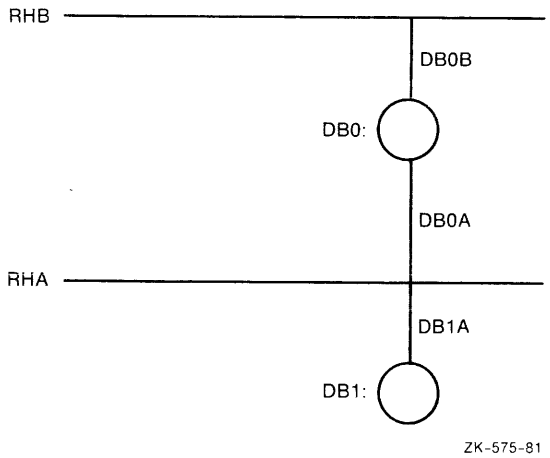


Figure 15-1 Sample Configuration for Increasing System Throughput

OFF (Cont.)

With this configuration, it is probable that the system will use the DB0A port. Because activity on DB1: must use the DB1A port, it has to wait until DB0A is free. If you put only ports DB0B and DB1A on line, each DB disk has its own controller and does not have to wait for another port to be free.

The OFFLINE command also accepts an optional keyword, ALL, which places off line all devices except the system disk and the terminal that issued the command. (The OFFLINE ALL command is useful prior to a system save.) Before issuing the OFFLINE ALL command, you must first dismount all volumes (tapes and disks) except the system disk and log out all terminals except the issuing terminal.

Formats

```
CON>OFF[LINE] device-spec1[,device-spec2...,device-specn]

CON>OFF[LINE] ALL
```

Note that device-spec also includes processors (CPx).

Requirements

1. Only a privileged user can modify the system configuration.
2. No tasks are attached to the device.
3. No volumes are mounted on the device.
4. If the device is a terminal, no users are logged in on it.

Example

```
CON>OFFLINE LPA,LP0:
CON>OFF YZA,YZB,YZC,YZD      ! Terminals assumed not logged in
```

The command sequence logically places the devices off line, which allows the bus run UBE to be placed off line.

OFF MEM

OFFLINE MEMCRY

(Multiprocessor systems only.) The OFFLINE MEMCRY command removes a memory box from the system configuration. Because memory must be contiguous, the target box must be the memory box in the system configuration with the highest addresses (the last memory box added to the system). If the target box is not the last box in the memory configuration, CON rejects the command and issues the message:

```
**HRC... Can't take box offline. Not last box in memory
```

If a partition maps to a memory box targeted for off-line status, CON rejects the command and issues the following message:

```
**HRC... Can't take box offline. Partition overmaps box
```

Therefore, you must contract the partition structure before removing the memory box. The MCR SET /TOP command can adjust the partition structure up or down (see the RSX-11M/M-PLUS MCR Operations Manual for details), but it is your responsibility to ensure that nothing is using the memory in the memory box. The system is protected so that you cannot bring the memory box off line if anything is using it.

Format

```
CON>OFF[LINE] MEM[ORY] memory-box
```

Example

```
CON>OFF MEM MK2:
```

This command removes memory box MK2: from the system configuration.

ONL

ONLINE

The ONLINE command changes the state of a device to on line and enables device access.

The transition to the on-line state is completed only if an access path exists between the device and the processor, and if the device driver can successfully initialize the device.

If the device lacks an access path, the ONLINE command changes device state to marked for on-line; the device is activated when an on-line access path becomes available (when the controller is brought on line).

The ONLINE command also accepts an optional keyword, ALL, which places logically on line every device controller and device unit known to the system.

Note that the data base values for interrupt vector and CSR assignments have no effect on the system until the device is brought on line.

Formats

```
CON>ONL[INE] device-specl[,device-spec2...,device-specn]

CON>ONL[INE] ALL
```

Requirements

1. Only a privileged user can modify the system configuration.

2. The specified device must be physically present in the target system and included in the system at system generation.

3. If the specified device is serviced by a loadable driver, the driver and device data base must be loaded.

The ONLINE command activates a device unit only if the associated controller is on line. The command first tests the unit to determine whether it actually exists on the controller and then performs drive sizing to determine device unit type. If the unit exists, it is now ready for user access. If the associated controller is off line, the command marks the unit for on-line status; it is presence-tested and sized when the access path is completed and, if present, activated.

To activate a device controller, the ONLINE command tests the controller's status register (CSR) to determine whether it is physically present. If the controller is present, then all of the associated units that are marked for on-line status are presence-tested, sized, and brought on line. Off-line units remain in the relative off-line state regardless of any ONLINE command performed on the associated controller.

To activate a processor on multiprocessor systems, the ONLINE command issues a boot request to the target processor. If the target processor ignores the boot request or cannot boot successfully, CON issues the following message (after a timeout of 5 seconds):

```
**Command execution error. Device offline
```

ONL (Cont.)

You can also bring on line individually the ports of a dual-port device, thus increasing system throughput because you do not have to wait for both ports if one does not work.

For dual-port terminals, you need a Status Control Block (SCB) for each terminal to bring the ports of dual-port terminals on line individually and to switch them individually. To switch the ports, the software will bring the controller on line, but the terminals must be switched manually using the CS11 switch on each terminal. If you have only one SCB for each controller, all of the terminals on a controller have to be switched over to the same controller.

Examples

The following command sequence (used on a multiprocessor system) assumes that the UBE bus run has been switched from CPA to CPB. The commands place the software bus devices on line for access:

```
CON>ONLINE LPA,LP0:              ! On line line printer and controller
CON>ONL YZA,YZB,YZC,YZD          ! On line terminal interfaces
CON>ONL TT2:,TT3:,TT4:,TT5:      ! On line terminals
CON>ONL TT6:,TT7:,TT10:,TT11:
```

The following command places all devices on the system logically on line:

```
CON>ONL ALL
```

This command places on line every device controller and device unit (except CPUs) known to the system (that is, all controllers and devices included at system generation). If a device unit or device controller lacks an access path, the command marks these for on-line status.

Note that placing all devices on line takes several seconds on a large system and that the command does not generate any error messages saying that a device may not be on line.

ONL MEM

ONLINE MEMORY

(Multiprocessor systems only.) The ONLINE MEMORY command adds a memory box to the system configuration. To place the additional memory on line, you must first place the appropriate memory box port controllers and the memory box itself logically on line. The box you specify will have the highest memory addresses.

To use the additional memory, you must expand the partition structure with the MCR SET /TOP command (see the RSX-11M/M-PLUS MCR Operations Manual).

Format

```
CON>ONLINE MEMORY memory-box
```

Example

```
CON>ONLINE MEMORY MK2:
```

This command adds the memory box MK2: to the system configuration.

SET

SET

The SET command changes the value of the CSR or the interrupt vector address for an off-line device. The values are stored in the device driver's data base.

SET cannot verify the CSR or interrupt vector address that you specify. If you specify an incorrect CSR address, the system will not be able to access the device when you issue the ONLINE command for the device. If you specify an incorrect interrupt vector address, the system will send interrupts to nonexistent interrupt vectors when you place the device logically on line; the device cannot be accessed. Also, if you specify an interrupt vector that is in use, you will get an error when you attempt to place the controller on line. (See the RSX-11M-PLUS Guide to Writing an I/O Driver for descriptions of the CSR and vector assignment errors.)

On multiprocessor systems, the SET command also activates the sanity timer and enables an alarm if the timer expires.

Format

CON>SET controller option=value

controller

The device controller (for example, RHA or DMA).

option

One of the following:

CSR	Alters the value of the controller's CSR address stored in the driver data base.
VEC	Alters the value of the controller's interrupt vector address stored in the driver data base.
TIMER	(Multiprocessor systems only.) Activates or inhibits the sanity timer (controller is RHx).
ALARM	(Multiprocessor systems only.) Activates or inhibits an alarm upon sanity timer timeout (controller is RHx).

value

If the SET option is CSR, the value must be between 160000(8) and 177777(8). Note that if you specify a CSR address of 160000 or less, the device is "unknown" and cannot be placed on line. This allows you to use the CON ONLINE ALL command without bringing every device on line when the current hardware configuration differs from the generated system. For example, if the target system has a DMC11 with a CSR address of 160100, but the generated system has a DZ11 at that address, CON ONL ALL attempts to bring the DMC11 on line as a DZ11. Note that the attempt may crash the system.

SET (Cont.)

If the SET option is VEC, the value must be less than 774(8).

On multiprocessor systems, if the SET option is TIMER or ALARM, the value is either ON (to activate) or OFF (to inhibit).

Requirements

The device must be off line and the driver must be loaded.

Examples

```
CON>SET LPA VEC=160
```

This command changes the interrupt vector address for the first line printer controller to 160(8).

```
CON>SET LPA CSR=177600
```

This command changes the CSR address for the first line printer controller to 177600(8).

SWI

SWITCH

(Multiprocessor systems only.) The SWITCH command logically inverts the processor association of a switched bus run; that is, it performs an unlink operation followed by a link operation. The command uses a special hardware feature of the DT07 to guarantee that the switched bus is linked to the target processor after the unlink operation. Thus, the SWITCH command ensures that the switched bus will be logically connected to the desired processor.

For example, in a 3-processor system, a properly configured switched bus can be connected to any one of three processors. An UNLINK command permits any processor to then link the switched bus. However, the SWITCH command specifies that only a specific processor can link the switched bus run following the unlink operation. If CPA and CPB are running as a dual-processor system and CPC is a single-processor system, a switch from CPA to CPB occurs even if CPC is trying to gain control of the bus.

CON rejects the command if a device attached to the switched bus has context (that is, the device is processing or allowing processing of I/O).

Format

```
CON>SWITCH UBx TO CPx
```

UBx

The target-switched bus run. Switched bus runs are named using the designation UB followed by a letter from the DIGITAL hardware alphabet as follows: UBE, UBF, UBH, UBJ, UBK, UBL, UBM, UBN, UBP, UBR, UBS, and UBT.

CPx

The target processor.

Example

```
CON>OFF LPA,LP0:                ! Off line printer and controller
CON>OFF YZA,YZB,YZC,YZD         ! Off line terminal interfaces
CON>OFF TT2:,TT3:,TT4:,TT5:     ! Off line all terminals
CON>OFF TT6:,TT7:,TT10:,TT11:
CON>SWITCH UBE TO CPB           ! Terminals assumed not logged in
CON>ONL LPA,LP0:                ! On line line printer and controller
CON>ONL YZA,YZB,YZC,YZD         ! On line terminal interfaces
CON>ONL TT2:,TT3:,TT4:,TT5:,    ! On line terminals
CON>ONL TT6:,TT7:,TT10:,TT11:   ! (Devices now connected to CPB)
```

This command sequence logically disconnects the first switched bus run (UBE) from its respective processor and then logically connects UBE to processor B.

UNL

UNLINK

The UNLINK command disconnects a port of a switched bus run from its respective processor. This command reverses the effect of the LINK command.

CON rejects the command if any device attached to the switched bus has context (that is, the device is processing or allowing processing of I/O).

Format

```
CON>UNLINK UBx
```

UBx

The switched bus.

Example

```
CON>UNL UBE      ! Terminals assumed not logged in
```

This command disconnects the switched bus port of UBE from its respective processor.

15.5 SYSTEM FAULTS

System faults cause an RSX-11M-PLUS system to stop processing an application properly. A fault could be a user error, a system software failure, an application software error, or a hardware fault. User errors (for example, accidentally removing a disk from the system) are obvious because the fault usually occurs immediately after the user action. System software failures usually occur when a system bug is encountered while you are debugging the application software. Application software errors are application-specific and not discussed in this manual. Hardware faults -- their detection, isolation, and recovery from -- are discussed in the remainder of this chapter.

15.5.1 Hardware Faults

Hardware faults can be placed in the following categories:

- Media failures
- Transient failures
- Intermittent failures
- Soft errors
- Hard errors

The following sections describe each of these types of faults.

15.5.1.1 **Media Failures** - Media failures occur when I/O operations fail because portions of the medium (a disk or tape) are bad. For both disks and tapes, when a media failure is detected, read operations are retried.

Before disks are initialized, you should check them for faulty sectors with the BAD utility. BAD marks the bad sectors, which prevents the file system from using them. (See the RSX-11M/M-PLUS Utilities Manual for more information.)

Occasionally, sectors on the media become bad, causing I/O operations to those sectors to fail. The error logging system can be used to detect these failures. (See the RSX-11M/M-PLUS Error Logging Manual for more information.) You should probably replace faulty media if they have critical data recorded on them or if a large number of errors occur on them.

On devices supported by DUDRV, bad blocks may be replaced by the Bad Block Replacement Control Task (RCT). See Chapter 5 for more information on this process.

15.5.1.2 **Transient Failures** - Transient failures cannot be repeated and are caused by electrical factors such as static from clothing or carpeting. If the failure occurs on an I/O device, the I/O operation is retried automatically and the application software never sees the failure. (The Error Logger records the failure, however.) If the failure occurs in memory or in a central processor, the system may crash and have to be rebooted.

15.5.1.3 **Intermittent Failures** - Intermittent failures are caused by irregular conditions within the system and usually happen during periods of high system or device activity. The failures usually occur where recorded data is corrupted enough so that it cannot be read. The data becomes corrupted when the device on which it is recorded is not working properly, usually because of a temporary power shortage.

An example of an intermittent failure is a disk device detecting a power failure, which causes the disk's volume valid bit to be reset. Dismounting and remounting the disk clears this condition and may allow I/O operations to proceed.

If the sources of intermittent failures are not obvious or directly reproducible, try to recreate the conditions that caused the failure by using the I/O Exerciser (IOX). IOX simulates high system activity, which could cause the failure to recur. (See Chapter 4 for information on IOX).

Intermittent failures that result in device errors appear in the error log.

15.5.1.4 **Soft Errors** - Soft errors are device errors that do not prevent an I/O operation from completing successfully. The device driver retries the operation and, depending on the capabilities of the device, performs ECC correction or disk-head offset positioning.

Soft errors usually indicate poor media or a misaligned device. For example, disk heads that are not aligned properly cause disk errors. These errors can be corrected by head offset positioning.

If several soft errors occur on a device, you should find the source of the errors and correct it through field service maintenance.

Soft errors are recorded by the Error Logger.

15.5.1.5 **Hard Errors** - Hard errors are solid device failures that prevent a device from working properly. The errors make the device unusable, but they are also the easiest to diagnose and correct.

The Error Logger records failed operations on the disks and tapes, but that information is usually unnecessary since the failures are obvious.

You can also use IOX to test the capabilities of many peripheral devices to determine if they are failing.

15.5.2 Recoverable and Nonrecoverable Errors

Soft errors and many media errors are recoverable errors. They are recorded by the Error Logger but do not impact system operation.

Hard errors and some media errors are nonrecoverable errors. They usually result in recurring device errors and failure of the application software. (The application can survive one failure on each device type before it will stop processing completely.) However, the operating system itself often survives the failure because all it requires to continue running is the system disk, the central processor(s), and memory.

15.6 FAULT ISOLATION

Once the software is stable, the availability of a system depends on the reliability of individual devices and the time required to restore processing. RSX-11M-PLUS systems can, in some cases, work around equipment failures in individual devices, but you must eventually isolate a faulty device, correct the reason for the failure, and restore it to the system. Depending on the severity and location of the error, you may have to restart the application and/or the system.

Failures often exhibit symptoms that allow you to detect a failing device before it can disrupt system operation. Because of this, you should regularly monitor the condition of all RSX-11M-PLUS peripherals and their ability to respond correctly to I/O requests by periodically examining the error log reports on all devices and occasionally running IOX on infrequently used devices. The Error Logger gathers information concerning all types of errors as they occur and generates comprehensive error reports. IOX tests the validity of mass storage devices. The tests exercise combinations of devices under high I/O activity.

Both the Error Logger and IOX provide error reports that contain a description of device fault conditions. Use the reports to determine whether the device should remain in operation. If the fault conditions are transient (see Section 15.5.1.2) and the device is capable of correctly performing its I/O functions, the device can remain in operation. However, you should closely monitor the device and schedule it for future maintenance as required.

15.7 SYSTEM RECOVERY

The procedure you use to diagnose and recover from a system fault depends on the primary use of your system. If it is mainly used for application development, then you would want to use the Error Logger or IOX to determine the cause of a failure to prevent future failures. If your system is used for process control and is running an application that requires enhanced availability, utilities such as the Executive Debugging Tool (XDT) are necessary only as much as they can be used to isolate and correct failed components. Your main objective is to bring the system back up and restart the application, with only limited diagnostics for future references.

To restart the system and the application, first bring up the minimum system required for running RSX-11M-PLUS. Then gradually bring devices on line until you have the minimum system required for running the application. If any of the devices are faulty, you can take them back out of the system and bring others on line until you have the required configuration.

The method for recovering from system faults also depends on the type of fault. Transient errors could cause the system to crash, which would require rebooting. Hard errors require the device not to be used and possibly to be reconfigured out of the system so that diagnostics for it can be run. Intermittent errors, because their source is not so obvious, could require you to begin with a minimum configuration and then gradually add devices until you find the one that caused the error. If the minimum configuration has the problem, you will have to switch to another one. Another possible way to recover from intermittent failures is to turn the error into a hard error by imposing an appropriate system load using IOX. By recreating the error, you can discover its source and, therefore, how to correct it.

The following sections describe how to recover from different kinds of hardware failures.

15.7.1 Device Unit and Controller Failures

Most failures in an operating system result from disk, tape, or other such device failures. Either the units themselves fail or the respective controller fails.

The time needed to reconfigure around these failures depends entirely on the application. If the application can notify an operator of the failure, thus allowing the operator to remove and inspect the media, mount it on another drive, and reintegrate it into the system, then the time to reconfigure is short. If, on the other hand, the operator must find a way to halt the application, move the media, and restart, then the time to reconfigure increases.

On RSX-11M-PLUS systems, TKTN prints a message at the console terminal whenever an unrecoverable hardware error occurs. For example:

```
*** DB1: -- Unrecoverable hardware device error
```

These messages can alert an operator to an impending failure and allow time to issue the commands to remove the failing unit from the system.

15.7.2 Processor Failures

Although it occurs rarely in comparison with device unit and controller failures, a processor failure also halts system operation.

Note that, on multiprocessor systems, a crash or trap or other failure may occur on a processor other than the faulty one.

If a processor fails, the system usually exhibits one of the following symptoms:

- The system traps to XDT or crashes.
- The sanity timer expires for one or more processors (multiprocessor systems only).
- The console lights on nonremote diagnostic consoles show that the system is halted in the Executive.
- The system halts with no printouts or visual indicators.

On multiprocessor systems, regardless of the cause of a processor failure, the recovery technique is straightforward and direct: you must isolate and exclude the disabled processor and reboot the system. The memory configuration need not change unless you wish to perform stand-alone diagnostics (see Section 15.8) on the failed processor.

15.7.3 MK11 Memory Box Failures

The two most common causes of memory failures are uncorrectable double-bit errors (parity errors) and loss of memory power.

If a user-state task encounters an uncorrectable memory parity error, the system aborts the task and locks that portion of memory. The Fixer task (FXR) detects the parity errors and creates a distinct segment in the task's region using the memory in which the errors were detected. This segment is separate from the rest of the region and is not used by other tasks. FXR is completely transparent to both users and tasks.

If the Executive or a system-state task encounters an uncorrectable parity error, the system prints the following message and then halts:

```
***EXEC PARITY ERROR STOP
```

Also, the sanity timers on multiprocessing systems may expire.

Regardless of system state (user or kernel), a double-bit error causes the memory box hardware to light the UNCOR ERROR light(s) on the respective memory box port controllers.

When an uncorrectable error halts the system, you must manually exclude that memory box and reboot. First, switch all ports of the faulty box off line. If necessary, use the thumbwheel switches to properly address the remaining memory box(es). (The thumbwheel switches are used when the failed box is box 0, which is the boot box, or when you want to disable a memory box by changing the starting addresses of the remaining boxes.) Then, halt the processor, initialize it, and reboot the system.

When loss of power causes the memory failure, the system halts. The system exhibits one or more of the following symptoms:

1. No lights or indicators are lit - usually indicates that the outside power supply has failed.

2. No MEM POWER READY lights are lit on memory box - indicates that the memory box's power supply has failed or has been powered down (check the power switch on the Port A controller to the box).

3. No MEM POWER READY lights are lit on one port of all memory boxes - indicates that the CPU's power supply has failed or the CPU has been powered down. Note that the power supply failure could have come through an expander box or through the far side of a switched bus currently switched to the CPU.

In all of the above cases, if a CPU or a memory box has been switched off but then switched back on before the memory batteries discharge, the system usually recovers without rebooting.

Otherwise, to recover from a memory box power failure, first manually switch all ports of the failed box off line. If necessary, use the thumbwheel switches to properly address the remaining memory box(es). Then, halt and initialize the processor(s) and reboot the system.

15.8 STAND-ALONE MAINTENANCE FOR MULTIPROCCESSOR SYSTEMS

On multiprocessing systems, when a device fails, it is often necessary to run diagnostic tasks before servicing the device. However, the diagnostic tasks require a processor for execution. Thus, to service most devices, you must remove a processor from the system.

The following list gives the requirements for testing and servicing peripheral devices:

1. A processor must be available to execute the diagnostics for the peripheral.

2. Memory must be available for the diagnostics.

3. A mechanism to load the diagnostic task into memory must be available.

The processor that executes the diagnostics must be connected to the UNIBUS on which the device to be tested resides. This implies that other peripherals on that bus run will not be available to the remainder of the system. Therefore, when determining the initial system configuration, you must ensure that duplicate devices are located on distinct bus runs. If the device to be tested is on a switched bus run, the switched bus can be connected to any processor to which it has access.

To make memory available for diagnostic tasks, the usual procedure is to logically remove a memory box from the running system and use the box for diagnostics only. This memory box is then switched manually so that the processor to be used for diagnostics views the box with a starting address of 0.

For loading a diagnostic into memory, the RSX-11M-PLUS multiprocessor system provides the Diagnostic Loader task (DLD). DLD loads a diagnostic program into a partition in the memory box of the diagnostic processor (the processor to which the failed device is attached). Diagnostic programs are system image files that you use to test the failed devices. The partition is created and specified by you and is mapped by program logical address space (PLAS) directives. By default, DLD clears the partition to the end after loading the program. The diagnostic is then run stand-alone.

The steps for using DLD are as follows:

1. Use FLX to transfer the diagnostic program from the diagnostic tape or disk to a Files-11 device attached to the diagnostic processor. (See the RSX-11M/M-PLUS Utilities Manual for information on FLX.) Because the tape or disk is in DOS-11 format, use the FLX Image Mode switch (/IM) to convert the file (the diagnostic program) to Files-11 format. The file is in formatted binary format and has a file type of .BIC or .BIN. (Note that binary formatted files are not patchable.)

2. Reconfigure your system so that you have a system with two CPUs and two memory boxes, and each box has 256K words of memory. The configuration would be as follows:

```
BOX 0   PORT A  ONLINE, START=000
        PORT B  OFFLINE

BOX 1   PORT A  OFFLINE
        PORT B  ONLINE, START=000

IIST    Off line both sides.
```

3. Boot RSX-11M-PLUS on CPA.

4. Set BOX 0 PORT A to ONLINE, START=010

5. Bring Box 1 on line. This memory box will contain the diagnostic partition. For example:

```
CON>ONL MKE,MK2:
```

6. Use the CON DISPLAY command to find out the base address and size of the memory box. For example:

```
CON>DISP FULL ATT FOR MK2:
```

   The base address of the partition will be the base address of the diagnostic program.

7. Create the diagnostic partition in the memory box by using the command SET /PAR and specifying type DIAG. This partition will be devoted to diagnostic functions and should begin at the location in memory that the diagnostic processor will see as location 0 or the bottom of memory. For example:

```
>SET /PAR=JIMK:20000:1777:DIAG
```

This command creates a partition named JIMK with a base address of 2000000 and consisting of 1777(10) 32-word blocks. The other processors see the memory box used for diagnostics as starting at location 2000000, but the processor running the diagnostic will see the box as starting at location 0.

Other partition types used with the SET /PAR command ensure that the requested memory is within the system image area (less than $SYSIZ), but DIAG ensures only that the memory physically exists. For the other processor to run concurrently with the diagnostic processor, the partition must be larger than $SYSIZ.

8. Use DLD to convert the binary formatted file into a bootable system image file (patchable with ZAP; see the RSX-11M/M-PLUS Utilites Manual) and then to load it into the partition. The DLD command line has the following format:

```
DLD>[partition][/HE]=filename[.typ][/qualifier]
```

partition

The partition created in the diagnostic processor's memory box into which the diagnostic program will be loaded. You must specify a partition name if you do not use the /NL switch (see below). DLD produces a "partition not found" error message otherwise. If you specify both a partition name and the /NL switch, the switch overrides the name.

/HE

Provides a brief description of DLD's use and format.

typ

The file type for the diagnostic program. File type can be .BIC, .BIN, or .SYS.

If the file type is .BIC or .BIN, DLD converts the file into a system image file. If you specify either file type, but DLD cannot find the file, it searches for the other file type before giving up. If you do not specify any file type, DLD first looks for a file with the .BIC file type and then for one with the .BIN file type.

If the file type is .SYS, DLD does not have to convert the file; it is ready to be loaded into the memory box.

/qualifier

One of the following:

/NL Inhibits loading of the diagnostic program (DLD only creates the system image file).

/NC — Inhibits clearing (zeroing) of any remaining memory in the partition after the diagnostic program has been loaded. The default is to clear the remaining memory.

/OFF:n — Offsets the starting location of the load by n times 32(10) words. The area between the beginning of the partition and the beginning of the diagnostic is untouched.

/LI — Lists information about the partition and the file: the partition name (if you have specified one), the base and size of the partition, and the new name of the system image file (if converted).

Specifying this switch gives you the transfer address (usually 200 or 1000) of the diagnostic program. The transfer address is needed to initiate the diagnostic.

/LW — Accounts for the DOS-11 link word that may exist in bytes 0 and 1 of every input block.

Example

```
DLD>JIMK=TEST.BIN/LI
PARTITION NAME:  JIMK
PARTITION BASE:  20000    SIZE:  1777 (32. WORD BLOCKS)
FILE NAME:  TEST.SYS;1
TRANSFER ADDRESS:  200
```

This command sequence converts the binary formatted diagnostic file TEST.BIN into the system image file TEST.SYS. DLD also lists the transfer address of the file and other information about the file and the partition.

9. DLD loads the program into a region that it creates in the diagnostic partition. The region is called DIAG and is the same size as the partition. After the program is loaded, DLD clears any remaining memory from the partition (unless otherwise specified).

10. To run the diagnostic program, halt the processor (in this case, CPB), toggle in the transfer address of the program (from using the /LI switch), and then push the START button to begin the diagnostics.

11. Use the MCR command line

```
>REMOVE DIAG/REG
```

to remove the diagnostic partition from the memory box.

15.9 ERROR MESSAGES

The CON and HRC error messages consist of one or more lines of information in the following format:

```
CON -- error message
(detail of error message)
```

The first line is the general error message. The second and any succeeding lines define the error message.

In addition to the description, the error messages may include Directive Status Word error codes and I/O error codes. For explanations of these codes, refer to the IAS/RSX I/O Operations Reference Manual.

A list of the CON and HRC error messages and a brief explanation of each follows.

15.9.1 CON Error Messages

CON -- Bad indirect command file specification

Explanation: The name of an indirect command file (@filename) is syntactically incorrect.

CON -- Command execution error

Explanation: CON cannot process the specified command line. (More specific information is also given.)

CON -- Command intent inconsistent with device state

Explanation: An attempt was made to place on line a device that is in the marked for off-line state.

CON -- Command list is empty. Nothing to print

Explanation: The CON LIST command was issued when CON's internal buffer was empty.

CON -- Command syntax error

Explanation: The command line does not conform to command syntax rules.

The command is displayed with a question mark (?) following the last recognized keyword or parameter typed by the user. Check the command description for correct format and reenter the command line.

CON -- CONFIGURE Version nnn

Explanation: This message is returned by the CON IDENT command.

CON -- CON is unable to do an extend task operation to obtain pool for a read of the system configuration. Please ensure that a checkpoint file has been allocated and reissue the CON command

Explanation: Self-explanatory.

CON -- File open error/FILE=filename/No such file

Explanation: An invalid file specification was used with the LIST command. Check the device name and UFD and reenter the command line.

CON -- HRC has timed out during initialization

Explanation: This message could indicate an error in the I/O data structure. Please verify that your device data structures are correct.

CON -- HRC version nnn

Explanation: This message is returned by the CON IDENT command.

CON -- Illegal command

Explanation: The command is not a valid CON command or a valid command was used illegally.

CON -- Illegal keyword

Explanation: The keyword is not valid for the specified CON command or a valid keyword was used illegally.

CON -- Initialization error

Explanation: CON was unable to properly initialize itself or its links with the RD: driver and the HRC... task. Verify that the driver is loaded and that HRC... is installed.

CON -- Initialization request of HRC... has failed. Please ensure that HRC... is installed.

Explanation: CON is unable to start HRC running.

CON -- Input error on input command file
FILE=filename

Explanation: The file you have specified exists, but CON could not read it.

CON -- Internal consistency error

Explanation: An error internal to either CON or HRC has occurred. This is most often an internal addressing trap or a malfunctioning driver. Submit a Software Performance Report (SPR) if the problem seems to be with DIGITAL-standard software.

CON -- Internal core pool exhausted

Explanation: CON has run out of internal pool space or was invoked without checkpoint space first being allocated. If out of pool space, check the /MAXEXT setting. (See the description of the SET command in the RSX-11M/M-PLUS MCR Operations Manual.)

CON -- Maximum indirect command file depth exceeded

Explanation: You have specified that CON use an indirect command file that exceeds CON's maximum nesting depth of three.

CON -- No device name matches select string

Explanation: You have specified with the DISPLAY command a logical device name that does not exist in the system.

CON -- No HRC version information available

Explanation: This message is returned by the CON IDENT command if HRC is not installed.

CON -- No reconfiguration driver is available
Output to HRC is suppressed

Explanation: The reconfiguration driver (RD:) is not loaded in the system.

CON -- Open error on input command file
FILE=filename

Explanation: The input file you have specified does not exist.

CON -- Unable to assign command LUN

Explanation: CON was unable to assign its command input LUN to TI:.

CON -- Unknown command error

Explanation: For undetermined reasons, CON cannot read the command.

CON -- Unknown device or invalid device specification

Explanation: The device specified for a command other than the DISPLAY command is not known to the system. Consult the system configuration for the proper name of the target device.

CON -- You must be privileged to modify the system configuration

Explanation: The command specified in the command line must be issued from a privileged terminal.

15.9.2 HRC Error Messages

The following messages are related to HRC but also come from CON.

Attempt to quiet unit for controller failed

Explanation: When a controller with units on line is taken off line, the on-line units are quieted and set to the marked for on-line state. HRC attempts to access the unit when it does not have context and can be taken off line. HRC makes 1000(10) attempts to quiet the device. If the device has not been successfully accessed after the 1000 attempts, CON displays this message.

Attribute format error

Explanation: HRC received a read or write attributes QIO in which the desired attributes were either incorrectly specified or inappropriate for the specified device.

CSR for controller not present in I/O page

Explanation: An attempt was made to bring on line a controller that does not exist in the current hardware configuration.

Device already linked

Explanation: A LINK command was issued for a bus run that is already linked to a CPU.

Device not linked

Explanation: An UNLINK command was issued for a bus run that is not currently linked to a CPU.

Device is unknown in this configuration

Explanation: HRC has received a request to bring on line a device that is unknown because it had been previously assigned a CSR address of 160000 or less.

Executive or driver status change error

Explanation: During a controller on-line or off-line operation, either the executive controller status change routines ($ONKRB/$OFKRB) or the driver's controller status change routine (xxKRB) were unable to change the state of the controller.

HRC... can't take box offline. Not last box in memory

Explanation: An attempt was made to take off line a memory box that is not at the end of the current physical memory configuration.

HRC... can't take box offline. Partition overmaps box

Explanation: An attempt was made to take a memory box off line, but the memory is currently in use for a partition. The partition must either be removed with SET /NOPAR or reduced in size with SET /TOP (see the RSX-11M/M-PLUS MCR Operations Manual).

HRC... detected I/O data base consistency error

Explanation: During the course of taking a device off line, HRC detected an error in the structure of the system device data base.

HRC... internal addressing error

Explanation: An odd address trap (or other software trap) has occurred within HRC. This may be due to a bug in HRC or an inconsistency in the system's I/O data base.

HRC... internal tables insufficient for this system

Explanation: HRC creates a list of units on the stack when taking a controller off line. This message is produced if HRC encounters a controller with too many units to create a list. The maximum number of units for each controller is defined by the symbol .MXCTL in HRPRE and is currently set to 256(10).

HRC... invalid device specification

Explanation: HRC received a request to bring memory on line, but the specified device was not a memory box.

HRC... request format error

Explanation: HRC received a QIO in which the format of the parameters was incorrect. If the issuing task for the request was CON, this error implies a software fault.

HRC... unable to access device to size drive

Explanation: When certain types of device units are brought on line, HRC must "size" the drive (for example, to determine if a DB:-type device is an RP05 or an RP06). This message is issued if the sizing process fails for any reason (for example, the drive is not powered up when the CON ONL command is issued).

HRC... unknown error from online/offline call

Explanation: HRC received from the driver on-line/off-line call an unsuccessful status that is not one of the normally expected return codes.

Invalid device descriptor

Explanation: HRC received a request to take a port off line, but the port description was invalid.

Memory box parameter error

Explanation: An attempt has been made to bring on line a set of memory boxes whose current physical configuration does not allow the specified command to be executed.

Offline controller failure

Explanation: HRC attempted to take a controller off line, but the controller did not properly change state. However, no explicit error was recognized.

Offline unit failure

Explanation: HRC attempted to take a unit off line, but the unit did not properly change state. However, no explicit error was recognized.

Parameter error

Explanation: HRC received a request to link a bus run, but the device specified was not a bus switch. If the requesting task was CON, this probably implies a software fault.

Syntax error

Explanation: HRC received a command to link a bus run to a device that is not a CPU.

Timeout on unit quieting operation

Explanation: HRC was unsuccessful in attempting to "catch" a unit in its quiet state to take it off line after 1000(10) attempts.

Unable to access bus run

Explanation: HRC was unable to create or remove the vectors for a device because the device's bus run was inaccessible. Because HRC checks bus run status and uses the marked for on-line state for devices on off-line bus runs, this message probably implies a software fault.

Unable to switch unit from the current controller

Explanation: This error is displayed for two reasons:

1. HRC attempted to take off line the controller for a port to which the bus switch is currently switched.

2. HRC attempted to take off line a controller (with units) whose access path cannot be changed.

Unable to take unit with context offline

Explanation: An attempt was made to take one of the following units off line:

- A mounted Files-11 device
- An attached device
- A logged-in terminal
- A bus switch whose bus run is currently linked
- A memory box unit whose memory is currently on line

CHAPTER 16

RSX-11M-PLUS I/O QUEUE OPTIMIZATION

I/O Queue Optimization is used to improve the throughput of the disk subsystem used in your RSX-11M-PLUS system. Disk throughput is improved by making the most effective use of the list of I/O requests in the queue for a particular disk device. The I/O queue for the device is scanned by I/O Queue Optimization and after each request is examined, the best request is dequeued and passed to the I/O driver for processing. How Queue Optimization determines which I/O request is best, depends on which method of I/O Queue Optimization is used. There are three methods of I/O Queue Optimization available (you can also choose not to have I/O Queue Optimization):

Nearest Cylinder
Elevator
Cylinder Scan

The method that serves you best depends on your processing environment. All three methods attempt to minimize head-seek time, which decreases the seek-time component of the file service time.

Since I/O Queue Optimization works by optimizing requests in the queue, optimization happens only when I/O requests are present in the queue. If the system loading is such that there are few, if any, requests in the queue, little optimization is achieved.

Without I/O Queue Optimization, the operating system groups the I/O requests in the queue, by priority, on a first-in first-out basis. The highest priority requests appear first in the queue and are processed in sequence. With I/O Queue Optimization, the I/O requests within priority groups are examined and the request having the appropriate disk address is chosen as the next I/O operation. The highest priority requests are still serviced first. However, throughput is increased by advantageous reordering of requests within priority.

In some cases, I/O Queue Optimization may provide smoother disk operation by reducing erratic head movement.

16.1 INITIATING OPTIMIZATION

The MCR SET /OPT command is used to initiate I/O Queue Optimization. The parameters in this command are used to specify the disk device, the method of optimization, and a fairness count limit for passed over I/O requests. These parameters are explained in detail in following paragraphs.

The SET command is only for privileged users. Nonprivileged users can use this command only to display the optimization status of devices in the system.

Format:

```
SET /[NO]OPT[=ddn:opttype:fairnesscount.]
```

Parameters

ddn

The disk device requiring optimization:

DBn	RP04,RP05,RP06 Pack Disks
DRn	RM02,RM03,RM05,RM80,RP07 Pack Disks
DMn	RK06 and RK07 Cartridge Disks
DLn	RL01 and RL02 Cartridge Disks

opttype

The type of optimization needed:

NEAR[EST]	Initiates Nearest Cylinder method; next request picked is the closest to current request.
ELEV[ATOR]	Initiates Elevator method; requests are processed first in one direction along the disk and then in the reverse direction.
CSCAN	Initiates Cylinder Scan method; requests are processed in only one direction along the disk (lowest to highest).

The default is NEAREST.

fairnesscount

A limit on the number of times the most unsatisfactory I/O request is passed over (see Section 16.2.1).

The default is 10.

16.1.1 Displaying Optimization Status

You can display the optimization status for all devices in the system that are currently using I/O Queue Optimization by entering:

```
>SET /OPT
```

This displays status in the form:

```
OPT=ddn:opttype:fairnesscount
```

For example:

```
OPT=DB0:NEAR:10.
OPT=DB1:ELEV:5.
OPT=DB2:CSCAN:8.
```

You can also display which devices are not using I/O Queue Optimization by entering:

```
>SET /NOOPT
```

This displays status in the form:

```
NOOPT=ddn:
```

For example:

```
NOOPT=DR1:
```

Nonprivileged as well as privileged users can display status.

16.2 SELECTING THE TYPE OF OPTIMIZATION

Selecting the correct optimization method is dependent upon the I/O processing environment in which your application is running, the physical location of data on the disk drive, and how often tasks access certain data areas.

A general guideline is to try the default first. (This initiates the Nearest Cylinder method with a fairness count limit of 10.) Then compare the difference in throughput by timing the application runtime against a suitable clock. For a closer look at disk activity and throughput, use Resource Accounting (see Chapter 17). Resource Accounting provides this data under the heading DEVICE STATISTICS in the output from the Resource Accounting transaction file.

The default (NEAREST and a fairness count of 10) is an average setting and, in most cases, should improve I/O throughput. After a closer observation and as you become more familiar with the I/O load and data organization on the disk in your application, you can make adjustments. You can change the fairness count limit or use one of the other methods of I/O Queue Optimization.

16.2.1 Adjusting the Fairness Count Limit

When using I/O Queue Optimization, you should watch for increased waiting time. I/O requests to some data areas may have slower than normal access times, because Optimization categorizes them as unsatisfactory. When this happens, Optimization passes over these requests for a better choice, keeping the unsatisfactory requests in the queue longer. This may cause a decrease in performance for some tasks even though overall system performance has improved.

To counteract increased waiting time and acquire a better balance, you can adjust the fairness count limit. This puts a limit on how many times an I/O request can be passed over. Optimization increments the fairness count for the first I/O request in the queue whenever that request in the queue is passed over. When the fairness count for an I/O request reaches the limit indicated with the SET command, the request is dequeued and the fairness count is cleared.

Decreasing the fairness count limit reduces waiting time but decreases the overall throughput initially gained with the higher count. If you find you must decrease the fairness count limit, try another optimization method. It may be more applicable to your situation.

16.2.2 Nearest Cylinder

The Nearest Cylinder method processes the I/O request that is closest to the current cylinder. In general, this method is effective for both long and short I/O queues. There may be an increase in waiting time, however, and this may be unacceptable in an interactive environment. To remedy this, reduce the fairness count limit, which reduces the throughput, or try one of the other optimization methods. (Elevator may be the next best choice.)

16.2.3 Elevator

The Elevator method processes I/O requests like an elevator. It processes requests as it moves in one direction along the disk until it reaches the last request in that direction. Elevator then changes direction and processes requests in that direction. This method is effective if there are many I/O requests outstanding. Also, waiting time tends to be less with the Elevator method than with the Nearest Cylinder method.

The Elevator method favors the center of the disk. Requests are done as they are passed, and the center of the disk tends to be passed over twice as often as the periphery. So a task that has a number of requests located in the center of the disk tends to have faster I/O than a task with requests at the periphery.

16.2.4 Cylinder Scan

The Cylinder Scan optimization method operates like the Elevator method except the Cylinder Scan method processes I/O requests only while passing in one direction along the disk. The direction is lowest cylinder number to highest cylinder number. This method is most effective if each task tends to do I/O to a localized area on the disk. It results in a more even distribution of I/O among tasks.

16.3 ERROR MESSAGES

SET -- Privileged command

Nonprivileged users can only display status.

SET -- Feature not supported

User tried to initiate Optimization for a disk device that does not support Optimization.

SET -- Device not in system

User tried to initiate Optimization for a disk device that is not mounted in the system.

SET -- Illegal keyword value

User specified a fairness count higher than 128(10).

SET -- Invalid keyword

User tried to use an invalid keyword.

SET -- Device offline or privileged diagnostic

User tried to initiate Optimization for an off-line device.

CHAPTER 17

RSX-11M-PLUS RESOURCE ACCOUNTING

17.1 INTRODUCTION

RSX-11M-PLUS Resource Accounting is a system generation option that provides a transaction file of system usage information. Accounting gathers data for both the user and the system. The data allows the billing of individual users for the resources used, as well as the measuring of overall system usage.

Some of the system usage information gathered by Resource Accounting and saved in the transaction file is:

User Data

- Session identification, terminal, and account number
- Billing stop date and time
- CPU usage time
- Tasks active at time of logoff or crash
- Number of tasks run
- Number of directives and QIOs issued
- Status at logoff or crash

Task Data

- Session identification, terminal, and account number
- Disk overlay loads
- Number of times checkpointed
- Number of times context has been loaded
- Highest running priority
- Task begin time and end time
- CPU usage time
- Number of directives and QIOs issued
- Status at logoff or crash

System Data

- Accounting start date and time
- Accounting stop date and time (0 if system crashes)
- Shutdown code
- Transaction file ID, sequence number, and device
- Last scan date, time, and scan rate in seconds
- Total CPU time and total CPU zero intervals
- Total number of tasks executed
- Total number of logons
- Number of current users
- Number of checkpoints
- Number of Shuffler runs
- Number of directives issued
- Number of QIOs issued

Logon Data or Invalid Logon Data

- User name, ID, terminal, and account number
- Logon UIC, date, and time
- Illegal password

Device Allocation, Deallocation Data, or Dismount Data

- Session ID, terminal, and account number
- Allocation or deallocation date, time, and device

Device Mount

- Session ID, terminal, and account number
- Mount date, time, and device
- Volume label for Files-11 mounted device
- Type of mount (foreign, shared, and so on)
- Owner UIC
- Volume protection code
- Name of ACP for device

Print Job Data

- Session ID, terminal, and account number
- Print job date and time

- Job name and page count
- Number of files printed, printer device, and forms number
- Job priority

Card Reader Job Data

- Session ID, terminal, and account number
- Batch or print job name
- Number of cards read
- Reader device name and number

System Time Change

- Old Time
- New Time

Device Usage Data

- I/O counts
- Error counts
- Tuning information

There are several ways to access the system usage information contained in the transaction file. They are:

- The SHOW ACCOUNTING commands (see Section 17.3):
  - SHOW ACCOUNTING/INFORMATION displays system usage information about specific terminals
  - SHOW ACCOUNTING/TRANSACTION_FILE generates a formatted version of the transaction file
- A report program written by DATATRIEVE-11 (see Section 17.4)
- User-written analysis programs (see Section 17.5)

Resource Accounting gathers system usage information by collecting the data in account blocks in secondary pool and writing each account block as a record in the transaction file. These account blocks are:

- Task Account Block (TAB)
- User Account Block (UAB)
- System Account Block (SAB)
- Transaction account blocks

For a description of account blocks, their structure, and how Resource Accounting uses them, see Section 17.6.

The following sections describe the Resource Accounting commands. Parameters are separated by spaces, you may abbreviate, and you are prompted for missing parameters. All commands, except SHOW ACCOUNTING/INFORMATION, are privileged. (Slashes in SET and SHOW commands are optional.)

The Resource Accounting commands are:

START/ACCOUNTING	Starts up the Accounting subsystem
STOP/ACCOUNTING	Shuts down the Accounting subsystem
SET ACCOUNTING	Changes parameters in a started Accounting subsystem
SHOW ACCOUNTING/INFORMATION	Displays Accounting information about a specific terminal
SHOW ACCOUNTING/TRANSACTION_FILE	Converts the transaction file to a readable format so it can be displayed on a terminal or put in a file
SHOW ACCOUNTING/DATATRIEVE	Converts a transaction file to a file that is readable by DATATRIEVE-11.

START/ACCOUNTING

17.2 CONTROL COMMANDS

17.2.1 START/ACCOUNTING Command

The START/ACCOUNTING command starts up the Accounting subsystem. The SYSLOG and ...ACC tasks must be installed before you issue the START/ACCOUNTING command. If you do not supply the optional parameters, they assume defaults.

Format

START/ACCOUNTING [parameter1][parametern]

Parameter

FILE:filespec

This filespec is the name of the transaction file that Accounting creates to store the data. You can name the file with this parameter or accept the default, LB:[1,6]ACNTRN.SYS.

EXTEND_SIZE:value

This value is the size of the initial allocation for the transaction file and for each extension when it is needed. The value can be any reasonable amount. If there is low Accounting activity, a small extension size is adequate because few extensions are required and overhead is low. If Accounting activity is high, a larger extension size lowers the overhead if disk space is available for the larger extension.

The default value is 10(10) blocks.

POOL_RESERVE:decimal value

This value is the number of blocks of secondary pool that Accounting must leave free after each allocation. If Accounting cannot leave this amount free, the allocation will fail and prevent the swamping of secondary pool.

The default is a quarter of the total secondary pool size. Use of the default is recommended.

SYSTEM_STATISTICS:yes/no

This parameter specifies to Accounting whether or not to accumulate system-wide statistics. System-wide statistics are the contents of the accumulation fields in the System Account Block. These fields are:

B.CPU Total CPU time used

B.DIR Total directive count

START/ACCOUNTING (Cont.)

B.QIO Total QIO$ count

B.TAS Total task count

The default is yes.

STATISTICS_SCAN[:value]

This value specifies a scan rate for the collection of device statistics. These device statistics are mainly used to measure seek-optimization parameters. They also provide excellent information on disk activity and throughput. The device statistics are collected either once or periodically. To indicate a one-time only collection, no value is specified. To collect statistics periodically, the value specifies the rate in minutes (nM) or seconds (nS). The default value is one hour if SYSTEM_STATISTICS:YES is specified.

If STATISTICS_SCAN is specified, SYSTEM_STATISTICS must be specified. This restriction does not apply to a one-time only request. If a value of 0 is specified or SYSTEM_STATISTICS:NO is specified, this feature is turned off. The example below shows the type of statistics collected in a transaction file.

SCAN_RATE:decimal value

This value is the time interval between the periods when Accounting writes the SAB and active UABs to the scan file called LB:[1,6]SYSSCAN.TMP for protection against loss in event of a system crash. The value can be supplied as "nM" or "n" for n minutes, or "nS" for seconds. If you specify 0, no scanning is done; therefore, if the system crashes, the active UABs and the SAB are not recorded in the transaction file and the data is lost.

The default value is five minutes.

If Accounting finds a scan file when you start, it outputs a crash transaction and copies the data in the scan file to the new transaction file. A copy of this crash transaction followed by the transaction file is shown as the output of the SHOW ACCOUNTING/TRANSACTION_FILE command in Section 17.3.2. Accounting copies all the old data before it writes the startup transaction and before it processes any new data.

When you stop Accounting, the scan file is deleted.

CRASH_REASON:yes/no

If you enter "yes" and Accounting is started after a crash that occurred when scanning was active, Accounting prompts you to enter a reason for the crash. The reason can be as many as 60 characters in length and is stored in the Crash Recovery

START/ACCOUNTING (Cont.)

Transaction Block. Accounting prompts for "reason for crash" when Accounting is restarted. If the system did not crash, no prompt occurs.

If you enter "no", a prompt never occurs; the default is no.

TASK:yes/no

If you enter "yes," task accounting is initiated. Task accounting gathers data on individual tasks running in the system. You can run task accounting when the system is running at peak load times and the task accounting data can be analyzed on a per task basis to identify heavy resource users.

If you enter "no," task accounting is stopped. The default is no.

Task accounting should only be specified when you absolutely need task accounting data. The reason is that task accounting requires substantial disk space because the transaction file can become quite large.

START/ACCOUNTING (Cont.)

Example

The type of device statistics collected by Resource Accounting is shown in the following output from a transaction file:

```
DEVICE STATISTICS
      TIME OF DEVICE STATISTICS = 15-DEC-81 16:27:23    DEVICE = DR1:
      IO COUNT = 0.             WORDS TRANSFERRED COUNT = 0.
      SOFT ERROR LIMIT = 8.           SOFT ERROR COUNT = 0.
      HARD ERROR LIMIT = 5.           HARD ERROR COUNT = 0.
      CYLINDER CROSSED COUNT = 0.     CURRENT FAIRNESS COUNT = 0.
      FAIRNESS COUNT LIMIT = 10.


DEVICE STATISTICS
      TIME OF DEVICE STATISTICS = 15-DEC-81 16:27:23    DEVICE = DR2:
      IO COUNT = 13832.         WORDS TRANSFERRED COUNT = 9002804.
      SOFT ERROR LIMIT = 8.           SOFT ERROR COUNT = 0.
      HARD ERROR LIMIT = 5.           HARD ERROR COUNT = 0.
      CYLINDER CROSSED COUNT = 1053   CURRENT FAIRNESS COUNT = 0.
      FAIRNESS COUNT LIMIT = 10.


DEVICE STATISTICS
      TIME OF DEVICE STATISTICS = 15-DEC-81 16:27:24    DEVICE = DR3:
      IO COUNT = 68490.         WORDS TRANSFERRED COUNT = 80640306.
      SOFT ERROR LIMIT = 8.           SOFT ERROR COUNT = 0.
      HARD ERROR LIMIT = 5.           HARD ERROR COUNT = 0.
      CYLINDER CROSSED COUNT = 313.   CURRENT FAIRNESS COUNT = 7.
      FAIRNESS COUNT LIMIT = 10.
```

Some definitions are:

IO COUNT is the number of I/O requests.

WORDS TRANSFERRED COUNT is the number of words transferred.

SOFT ERROR is a recovered error.

HARD ERROR is a nonrecovered error. (See the RSX-11M/M-PLUS Error Logging Reference Manual for more information about hard and soft errors.)

CYLINDER CROSSED COUNT is how many cylinders were crossed while accessing data. This data can be analyzed for I/O Queue Optimization purposes.

FAIRNESS COUNT is a Queue Optimization parameter that indicates how many times an I/O request was passed over (see Chapter 16).

SET/ACCOUNTING

17.2.2 SET ACCOUNTING Command

This command allows you to change the value of parameters specified when Resource Accounting was started with the START/ACCOUNTING command or previously modified with a SET command. (A transaction is written containing the old value and the new.)

The following options can be changed while Resource Accounting is running.

- The file transactions are written to
- The size of the extensions
- The scan rate for the scan file
- The scan rate for collecting device statistics
- The state of task accounting (on or off)

Format

SET ACCOUNTING/[parameter1][parametern]

Parameter

FILE[:filespec]

Changes the file transactions are written to. Defaults to the current filespec, if not specified.

EXTEND_SIZE:value

This value changes the size of the transaction file specified in the START/ACCOUNTING EXTEND_SIZE parameter.

STATISTICS_SCAN:value

This value changes the scan rate for collecting device statistics. A value of 0 stops this option.

SCAN_RATE:value

This value changes the scan rate for the scan file specified in the SCAN_RATE parameter. A value of 0 stops this option.

SET/ACCOUNTING (Cont.)

TASK:yes/no

If you enter "yes," task accounting is started.

When specified, task accounting creates task account blocks for all active tasks in the system. All data, for each task, is gathered starting at this time.

If you enter "no," task accounting is stopped.

When specified, task accounting on a per task basis is terminated and the data for all active tasks is written to the transaction file. The only exception is tasks that have a CPU time limit specified, since the data is necessary for the time limit to be checked. The data for these tasks is written to the transaction file when the task exits.

STOP/ACCOUNTING

17.2.3 STOP/ACCOUNTING Command

This command shuts down the Resource Accounting subsystem. The parameter is provided to indicate why the shutdown is occurring and must be entered with the command. Accounting puts the reason for stopping into the System Account Block.

Formats

STOP/ACCOUNTING reason

STOP/ACCOUNTING CLEAN_UP

Reason

MAINTENANCE

You can use this reason when you stop Accounting for system maintenance.

REBOOT

You can use this reason if you must reboot the system.

SCHEDULED_SHUTDOWN

You can use this reason when the system is being shut down or used for other purposes where Accounting is not applicable.

SHUTUP

The SHUTUP program uses this reason as a default because it has no means of determining a shutdown reason.

OTHER

You can use this reason when any of the above reasons are not adequate.

The STOP/ACCOUNTING CLEAN_UP command has a special use. It must be used if the SYSLOG task aborts or Accounting encounters a fatal error. This command cleans up all the data structures to allow Accounting to restart successfully. Accounting accepts this command only if it is necessary (after an Accounting failure or abort).

When you use this command, the system assumes all Accounting data in memory to be invalid and deallocates the data back to secondary pool without writing it to the transaction file. Note that if the Accounting data structures are corrupted, the **system may crash** while it attempts to deallocate secondary pool.

You can restart Accounting normally after you issue this command.

SHOW ACCOUNTING/INFORMATION

17.3 SHOW COMMANDS

17.3.1 SHOW ACCOUNTING/INFORMATION Command

This command displays on the terminal from which the command was issued the Accounting information for the specified terminal. Nonprivileged users can receive only their Accounting data. A privileged user can obtain any Accounting data.

Format

SHOW ACCOUNTING/INFORMATION[parameter]

Parameter

TTn:

If you specify TTn:, this command displays the Accounting data for the specified terminal. A privileged user has access to the Accounting data of any terminal on the system. A nonprivileged user can only get his own data. The default terminal is TI:.

If you enter no terminal name, the Accounting data of the terminal issuing the command is displayed.

CO:

If you specify CO:, the Accounting data of the system tasks running is displayed on the terminal.

SYS

If you specify SYS, the current system totals are displayed on the terminal.

TASK=taskname

Shows Accounting data for the specified task if the system is maintaining task accounting.

SHOW ACCOUNTING/TRANSACTION_FILE

17.3.2 SHOW ACCOUNTING/TRANSACTION_FILE Command

The SHOW ACCOUNTING/TRANSACTION_FILE command converts the transaction file into a readable format. It either displays this file on a terminal or writes it into another file. The SHOW ACCOUNTING/-TRANSACTION_FILE command uses the current transaction file as the default file. (When Accounting is not running, there is no default file.)

Format

SHOW ACCOUNTING/TRANSACTION_FILE[:inputfilespec] outputfilespec

Parameter

inputfilespec

The input file is the transaction file to be read. If you do not specify an input file, the current transaction file is used.

outputfilespec

The output file can be either a terminal (specified as TTn: or TI:) or a file of your choice.

Notes

UIC [126,10] on the distribution kit includes the source code for the SHOW ACCOUNTING/TRANSACTION_FILE command to serve as an example for how to read the file. Note that the record size in the file is equal to the transaction length. The only exception to this is the Accounting startup transaction that only occurs once and is 512(10) bytes long. The startup transaction consists of a header only, but its size is increased to 512(10) bytes to force a file extension. See Section 17.6 or [126,10] ACNDF.DOC on the distribution kit for the format of this file.

Example

The output of the SHOW ACCOUNTING/TRANSACTION_FILE command is a formatted display of the transaction file. The default file is the transaction file named in the START/ACCOUNTING command. This example of output of a SHOW ACCOUNTING/TRANSACTION_FILE command is keyed to the following explanatory comments:

```
ACCOUNTING DATA - CURRENT FILE                    11-DEC-81 16:56:04

SYSTEM CRASH RECORD
      TIME OF LAST SCAN = 7-DEC-81 10:48:17     SCAN RATE (SEC) = 300. (2)
      RESTART TIME = 7-DEC-81 19:58:15
      REASON = (1)
```

SHOW ACCOUNTING/TRANSACTION_FILE (Cont.)

```
USER - ALPHA     P ❸
      SESSION ID = F1143        TI: = TT40:     ACCOUNT = 1,
      LOGON UIC = [7,111]       LOGGED ON = 7-DEC-81 08:58:53
      LOGGED OFF = 00-000-00 00:00:00     BILLING STOPPED = 7-DEC-81 10:48:17
      CPU = 4461.          TASKS ACTIVE = 1.          TASKS RUN = 95.
      QIOS = 6215.                      DIRECTIVES = 10240.
STATUS      ACT   CRH

                                              .
                                              .
                                              .

USER - SYSTEM TASKS
      SESSION ID = $SY0              TI: = CO:       ACCOUNT = 0.
      LOGON UIC = [0,0]       LOGGED ON = 6-DEC-81 15:48:16
      LOGGED OFF = 00-00-00 00:00:00  BILLING STOPPED = 7-DEC-81 10:48:17
      CPU = 103866.        TASKS ACTIVE = 9.         TASKS RUN = 601.
      QIOS = 128967.             DIRECTIVES = 667250.
STATUS      ACT   CRH


TOTAL SYSTEM STATISTICS ❹
      ACCOUNTING STARTED = 6-DEC-81 15:48:16
      ACCOUNTING STOPPED = 00-000-00 00:00:00     SHUTDOWN CODE = 0
      TRANS FILE ID = 1422        SEQ NUM = 1           DEVICE = DB1:
      TIME OF LAST SCAN = 7-DEC-81 10:48:17     SCAN RATE (SEC) = 300.
      TOTAL CPU = 441264.        ZERO CPU INTERVALS = 374665.
      TOTAL TASKS = 3236.          TOTAL LOGONS = 60.
      CURRENT USERS = 27.        CHECKPOINTS = 23.
      SHF RUNS = 0.             DIRECTIVES = 2373564.       QIOS = 556015.

ACCOUNTING STARTUP
      ACCOUNTING STARTED = 7-DEC-81 19:58:16


LOGIN - BETA           B
      SESSION ID = RXP1         TI: = TT26:     ACCOUNT = 0,
      LOGON UIC = [7,102]       TIME = 7-DEC-81 19:58:35


LOGIN - ALPHA          P
      SESSION ID = F112         TI: = TT55:     ACCOUNT = 1,
      LOGON UIC = [7,111]       TIME = 7-DEC-81 20:05:44


DEVICE MOUNT
      SESSION ID = F112         TI: = TT55:     ACCOUNT = 1,
      TIME = 7-DEC-81 20:06:37      DEVICE = DR1:      VOL ID = WELCOMEBACK
      OWNER UIC = [0,0]         ACP NAME =F11ACP      VOL PROT MASK = 0


USER - ALPHA           P
      SESSION ID = F112         TI: = TT55:     ACCOUNT = 1,
      LOGON UIC = [7,111]       LOGGED ON = 7-DEC-81 20:05:44
      LOGGED OFF = 7-DEC-81 20:06:43      BILLING STOPPED = 7-DEC-81 20:06:43
      CPU = 141.           TASKS ACTIVE = 0.          TASKS RUN = 17,
      DIRECTIVES = 472.    QIOS = 166.
```

SHOW ACCOUNTING/TRANSACTION_FILE (Cont.)

```
LOGIN - OMEGA          D
      SESSION ID = RXP3          TI: = TT37:     ACCOUNT = 4.
      LOGON UIC = [7,71]         TIME = 7-DEC-81 20:08:11


DEVICE ALLOCATION
      SESSION ID = RXP4          TI: = TT37:     ACCOUNT = 4.
      TIME = 7-DEC-81 20:09:12       DEVICE = DL0:


LOGIN - THETA          D
      SESSION ID = SPR5          TI: = TT55:     ACCOUNT = 0.
      LOGON UIC = [7,327]        TIME = 7-DEC-81 20:09:14


DEVICE MOUNT
      SESSION ID = RXP4          TI: = TT37:     ACCOUNT = 4.
      TIME = 7-DEC-81 20:09:15       DEVICE = DL0:      VOL ID =
      OWNER UIC = [7,107]        ACP NAME = TSTACP     VOL PROT MASK = 0


DEVICE DEALLOCATION
      SESSION ID = RXP4          TI: = TT37:     ACCOUNT = 4.
      TIME = 7-DEC-81 20:09:15       DEVICE = DL0:


USER - SYSTEM TASKS
      SESSION ID = $SY0          TI: = CO:       ACCOUNT = 0.
      LOGON UIC = [0,0]          LOGGED ON = 7-DEC-81 19:58:16
      LOGGED OFF = 00-000-00 00:00:00    BILLING STOPPED = 7-DEC-81 20:13:21
      CPU = 1603.          TASKS ACTIVE = 4.          TASKS RUN = 64.
      DIRECTIVES = 2986.


TOTAL SYSTEM STATISTICS
      ACCOUNTING STARTED = 7-DEC-81 19:58:16
      ACCOUNTING STOPPED = 7-DEC-81 20:13:21     SHUTDOWN CODE = 4
      TRANS FILE ID = 205          SEQ NUM = 20          DEVICE = DR0:
      STATISTICAL SCAN RATE (SEC.) = 3600.     FILE EXT. SIZE = 10.
      TIME OF LAST SCAN = 7-DEC-81 20:13:17     SCAN RATE (SEC) = 300.
      TOTAL CPU = 4260.          ZERO CPU INTERVALS = 3063.
      TOTAL TASKS = 243.           TOTAL LOGONS = 7.
      CURRENT USERS = 1.         CHECKPOINTS = 0.
      SHF RUNS = 0.              DIRECTIVES = 10241.          QIOS = 4946.
```

❶ The crash reason appears here if Accounting prompts you for a reason for the crash. The prompt occurs only if you answered yes to the CRASH_REASON parameter in the START/ACCOUNTING command.

❷ To minimize loss of data, Accounting writes the SABs and currently active UABs to a temporary file every n seconds or minutes unless you specify a zero scan rate in the START/ACCOUNTING command. The process of writing this file is called a scan. When you restart Accounting, it copies any existing temporary file of Accounting data into the Accounting file now starting. If the scan rate is zero, the SABs are memory-resident until you stop Accounting or the system crashes.

SHOW ACCOUNTING/TRANSACTION_FILE (Cont.)

Also, the currently active UABs are memory-resident until a user logs off. Therefore, if a crash occurs with the scan rate at zero, all the SAB and current UAB data is lost.

❸ The User Account Block (P. Alpha) shows a logged off date and time of zero because the system crashed. There may or may not be a "status" line as the last line of the User Account Block. The status line indicates the bit setting of the status mask byte (B.STM) in the UAB. Three of the bits have external importance and cause this line to appear if they are on. These bits are:

ACT (BS.ACT) Indicates that the UAB was written to the file while still active (all activity had not ceased). This can happen when UABs are copied from the scan file, or when all the data structures are written out because Accounting has stopped.

LGO (BS.LGO) Indicates that the user has logged off with tasks remaining active or with clock queue requests pending.

CRH (BS.CRH) Indicates that this UAB came from the scan file and was copied into the transaction file when Accounting was started after a system crash.

❹ The TOTAL SYSTEM STATISTICS block contains the shutdown code as a digit in the range 0 through 5. These digits have the following meanings:

0 Accounting not stopped due to crash

1 Maintenance

2 Reboot

3 Scheduled shutdown

4 Shutup

5 Other

The ZERO CPU INTERVALS field shows the possible error in the CPU time measurements. The error occurs because the system clock is usually run at 60 Hz and is slower than the context switch time of the Executive. Therefore, a task can sometimes execute for "zero" ticks, because the time needed for its execution is less than the resolution of the clock. Resource Accounting rounds each zero tick to one-half tick of CPU time and counts them. The ZERO CPU INTERVALS field contains the count of the number of zero ticks that have occurred.

17.4 WRITING AN ACCOUNTING REPORT PROGRAM USING DATATRIEVE-11

The data collected in the transaction file pertains to your computing environment. How you interpret and analyze this data depends on your objectives. For instance, are you interested in billing or system measurement? To obtain the Accounting data you are looking for, in the form you want, you will most likely write your own report program. This program accesses the transaction file, reads the required data fields, and writes a report containing the Resource Accounting data you are interested in.

One way to write this specialized report is to use DATATRIEVE-11. But before a DATATRIEVE-11 program can access the transaction file, the file must be converted to a DATATRIEVE-11 readable file. To convert the file, use the Resource Accounting SHOW ACCOUNTING/DATATRIEVE command (see Section 17.4.1). The format of the converted file is shown in LB:[126,24]ACNTRN.CMD along with an example of a simple report generator written in DATATRIEVE-11.

To obtain a report using the example report generator, you must:

1. Use the SHOW ACCOUNTING command (see Section 17.4.1) to convert the appropriate transaction file to a file that is readable by DATATRIEVE-11. Name the file ACCOUNT.DAT (this is the name used in ACNTRN.CMD, for example purposes only).

2. Copy the following files from the system disk to your directory:

   [1,2]QUERY.DIC

   [126,24]ACNTRN.CMD

After the above files are made available, use the following command line to generate a report:

```
>DTR @ACNTRN.CMD
```

This command line calls DATATRIEVE-11 and executes the DATATRIEVE-11 commands in the ACNTRN.CMD file. A brief description is then displayed on your terminal screen with a portion of the DATATRIEVE-11 commands that make up the report generator. After this you are prompted with:

```
Enter DEVICE OR FILE:
```

You then enter where you want the Resource Accounting information displayed. The information can be displayed on a device or directed to a file. For example, the device could be your terminal:

```
Enter DEVICE OR FILE:  TI:
```

If you wanted to put the information in a file named ACT.REP, the response would be:

```
Enter DEVICE OR FILE:  ACT.REP
```

While creating the report, you are prompted to respond with where the report should be displayed. These prompts appear at junctures in the report generating process where certain parts of the report can be either diplayed or directed to a file. For example, the first prompt you receive is at a point where the STARTUP TRANSACTION part of the report can be displayed. The next prompt is at a point where the

first part of USER ACCOUNT BLOCK TRANSACTION can be displayed. At each prompt, you can direct that part of the report to the same report file, a different report file, or to a device.

To stop the Resource Accounting report process before it is finished, you must abort DTR. If this is done, you must recopy [1,2] QUERY.DIC back into your area before using the example Resource Accounting report program.

17.4.1 SHOW ACCOUNTING/DATATRIEVE Command

This command allows you to write your own Resource Accounting report generator using DATATRIEVE-11. This command is used to convert an existing transaction file to a file that is readable by DATATRIEVE-11.

Format

SHOW ACCOUNTING/DATATRIEVE[:trnsfilespec] outfilespec

Parameter

trnsfilespec

Specifies the name of an existing Resource Accounting transaction file.

outfilespec

Specifies the name of the output file where the converted transaction file is to reside.

17.5 USER TASK INTERFACE FOR CREATING TRANSACTIONS

You can define and create your own transactions and write them into the transaction file by using specific entry points in the Executive Accounting module. The task that you write must be privileged. The privileged task must enter the system state (by means of $SWSTK) and call one of the following entry points in the Executive module ACSUB: $QTRAN, $QTRN1, or $QTRN2.

In general, before calling one of the three entry points, the task must have the address in register R0 of a defined area (packet) that is within the task. Also, the task must supply in the packet a value for B.LEN and B.TYP (see Section 17.6.2). The packet contains the header and the information that is to be written as a transaction. The action taken by the ACSUB code depends on the entry point called by the privileged task.

17.5.1 Calling $QTRAN

The $QTRAN routine automatically fills in the user-ID fields of the packet from those used by the TI: terminal of the calling task. In effect, this shows that the new transaction is coming from the same user who ran the task.

To call this entry point, the task must have the address of the transaction packet in register R0. The two fields B.LEN (containing the transaction length) and B.TYP (containing a transaction code from

128 through 255) must be filled in and contained within the packet. $QTRAN obtains the transaction header from the UAB of the privileged task, copies the header except for B.LEN and B.TYP into the packet, time-stamps the packet, and links the packet into a queue to be written into the transaction file by the SYSLOG task. $QTRAN does not alter the task's space, but copies the packet into secondary pool space before copying the header and queuing the packet into the SYSLOG queue.

Example 17-1 is an example of the code needed to call $QTRAN.

Example 17-1 Code That Calls $QTRAN

```
        .MCALL ACNDF$
        ACNDF$          ;DEFINE ACCOUNTING SYMBOLS

TRAN:   .BLKB   B.MLEN
                .
                .
                .
        This area contains code that
        describes the data to be
        written into the transaction
        file.
                .
                .
                .
        MOV     #TRAN,R0                ;GET PACKET ADDRESS
        MOVB    #BT.MOV,B.TYP(R0)       ;FILL IN TYPE
        MOVB    #B.MLEN,B.LEN(R0)       ;FILL IN LENGTH
        CALL    $SWSTK, .....           ;SWITCH TO SYSTEM STATE
        CALL    $QTRAN                  ;TO CALL QUEUE TRANSACTION EXEC ROUTINE
                .
                .
```

17.5.2 Calling $QTRN1

The $QTRN1 routine allows the user-ID fields to be obtained from a specified place, instead of assuming the current task.

To call this entry point, the task must have filled the B.LEN and B.TYP fields within the packet and must have the address of the transaction packet in register R0. In addition, register R1 must contain an APR value that points to a UAB or TAB that contains the user-ID data that should be copied into this transaction.

The intent is that the privileged task will write a transaction associated with another task. $QTRN1 time-stamps the packet and obtains the transaction header from the UAB of the other task. $QTRN1 then queues the packet in the SYSLOG queue to be written into the transaction file. $QTRN1 does not alter the packet within the task, but copies identifying information from the specified UAB into the secondary pool, then copies data into the packet, and queues the packet into the SYSLOG queue.

Example 17-2 is an example of the code needed to call $QTRN1.

Example 17-2 Code That Calls $QTRN1

```
                .
                .
        .MCALL ACNDF$
        ACNDF$                          ;DEFINE ACCOUNTING SYMBOLS
                .
                .
        This example assumes that R1 contains the pointer
        to the TCB of the task for which the transaction
        is to be posted.
                .
                .
                .
TRAN:   .BLKB B.MLEN                    ;START OF TRANSACTION
                .
                .
                .
        MOV     T.ACN(R1),R1            ;GET OTHER TASK'S UAB OR TAB ADDRESS
        BEQ     10$                     ;IF EQ NO ACCOUNTING DATA PRESENT
        MOV     #TRAN,R0                ;POINT TO TRANSACTION HERE
                .
                .
        MOVB    #BT.MOV,B.TYP(R0)       ;FILL IN TRANSACTION TYPE
        MOVB    #B.MLEN,B.LEN(R0)       ;FILL IN TRANSACTION LENGTH
        CALL    $SWSTK, .....           ;SWITCH TO SYSTEM STATE
                                        ;TO CALL EXECUTIVE ROUTINES
        CALL    $QTRN1
                .
                .
                .
```

17.5.3 Calling $QTRN2

The $QTRN2 routine assumes the calling task completely fills in the user-ID area of the transaction. The Executive queues the packet with no modification, except for a time stamp.

To call this entry point, the task must have the address of the transaction packet in register R0. $QTRN2 does not obtain a header from any other source. Therefore, except for the transaction time, the privileged task must have completely filled in the transaction header and the transaction data contained within the packet. $QTRN2 fills in the transaction time field in the transaction header area of the packet created by the user task and queues the complete packet to the SYSLOG queue to be written to the transaction file.

Example 17-3 is an example of the code needed to call $QTRN2.

Figure 17-1 summarily describes the privileged task interface just discussed. Section 17.6.2 contains a description of the header area fields.

Example 17-3 Code That Calls $QTRN2

```
        .MCALL ACNDF$
        ACNDF$                          ;DEFINE ACCOUNTING SYMBOLS
                        .
                        .
TRAN:   .BLKB B.MLEN
                        .
                        .
        This area contains the code that
        describes the header and data to
        be written into the transaction
        file.
                        .
                        .
                        .
        MOV     #TRAN,R0                ;POINT TO TRANSACTION
        MOVB    #BT.MOV,B.TYP(R0)       ;FILL IN TRANSACTION TYPE
        MOVB    #B.MLEN,B.LEN(R0)       ;FILL IN TRANSACTION LENGTH
        MOV     UID,B.UID(R0)           ;USER SESSION ID
        MOV     UID+2,B.UID+2(R0)       ;SECOND HALF
        MOV     ACN,B.ACN(R0)           ;ACCOUNT NUMBER
        MOV     TID,B.TID(R0)           ;TERMINAL ID
        CALL    $SWSTK, 10$             ;SWITCH TO SYSTEM STATE
        CALL    $QTRN2
        RETURN
                                                ;TO USER
      10$:
```

17.6 ACCOUNT BLOCKS

There are two types of Resource Accounting account blocks: the memory-resident and non-memory-resident account blocks. The memory-resident account blocks are the Task Account Block (TAB), the User Account Block (UAB), and the System Account Block (SAB). The Task Account Block (TAB) accumulates accounting data per task and stays in secondary pool as long as a task is active. The User Account Block (UAB) for each logged-on user accumulates the Accounting data and stays in secondary pool as long as the user is logged on. The system's UAB accumulates data for tasks running on terminal CO:. The SAB stays in secondary pool and accumulates system accounting data as long as Resource Accounting is running.

All other account blocks are non-memory-resident; that is, they are created and filled when any transaction occurs, copied by Accounting into secondary pool, and immediately written into the transaction file, and the space is deallocated back to secondary pool.

All of the blocks have a header area that contains the same fields and offsets. In addition, the TAB, the UAB, and the SAB have data accumulation field areas following the header. All data accumulation fields have the same offsets. The information specific to the TAB, UAB, and SAB follows the accumulation areas.

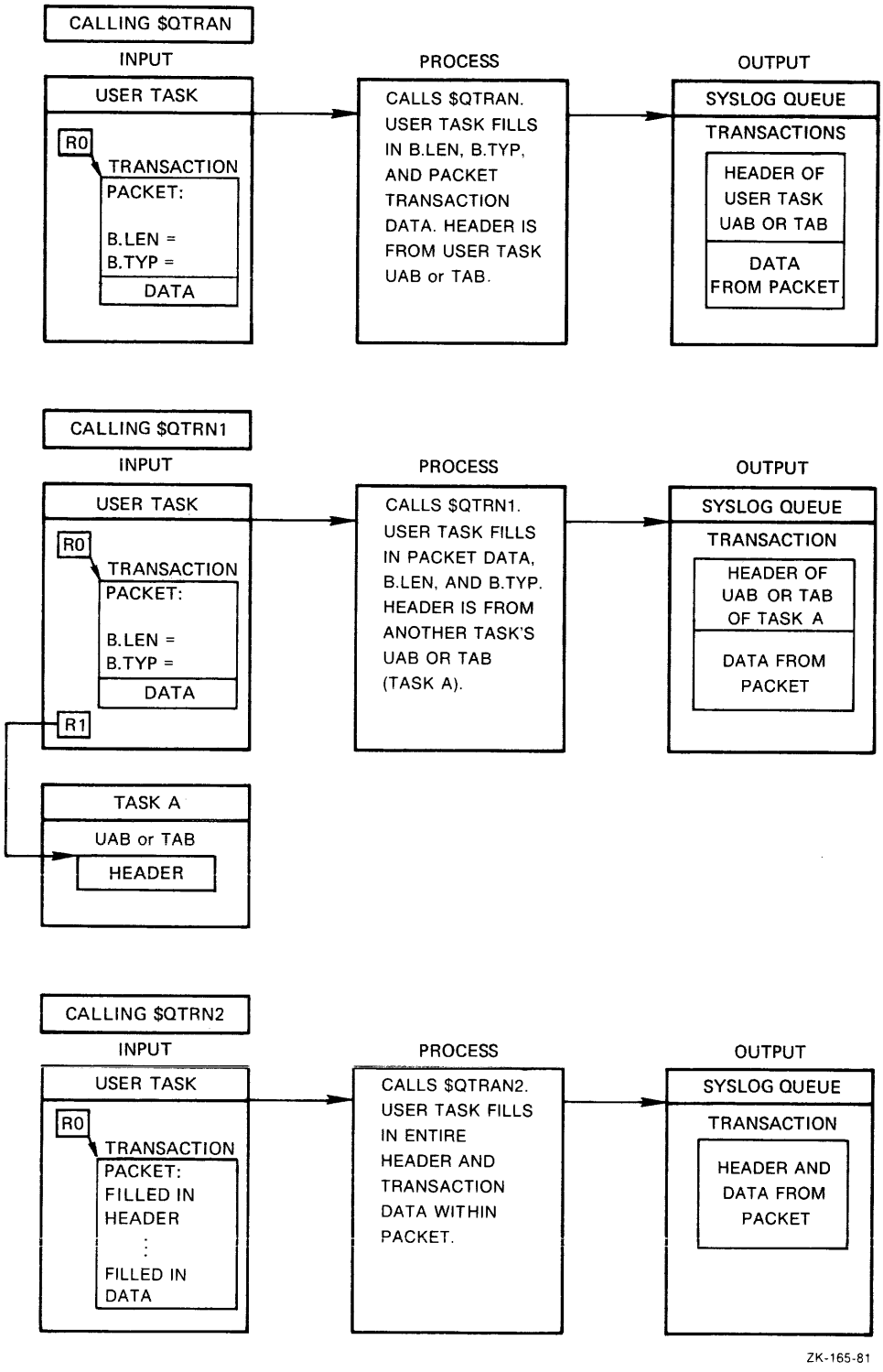


Figure 17-1 Privileged Task Interface Overview

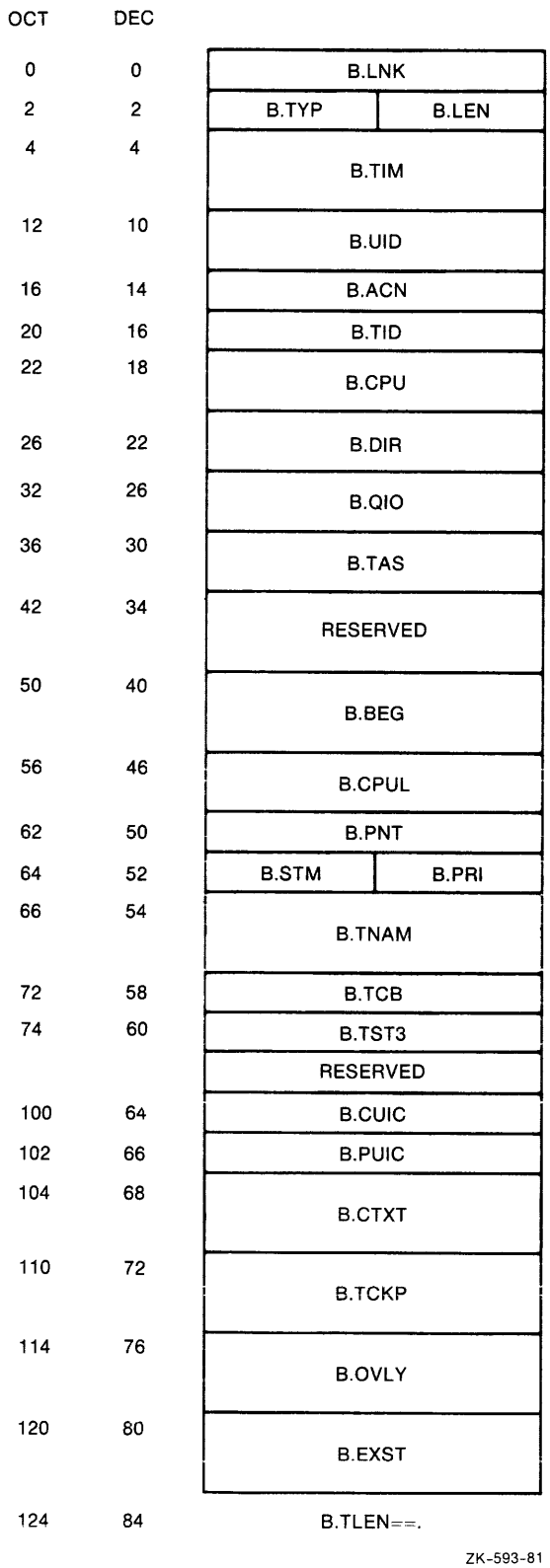
You can find the size and the description of the offsets in Section 17.6.2. The offsets are grouped logically; the header offsets are described in one group, the accumulation field offsets in another group, and so forth. The account and transaction block diagrams in Section 17.6.1 show how these groups fit together to make a functional account and transaction block. The account and transaction block diagrams are shown before the offset descriptions.

Check the [126,10]ACNDF.DOC file on the distribution kit for the correct numeric offsets if you plan to write a program that uses the offsets. In MACRO-11 programs, use the symbolic names defined by the ACNDF$ macro in [1,1]EXEMC.MLB. The names will not change even if their associated offsets change.

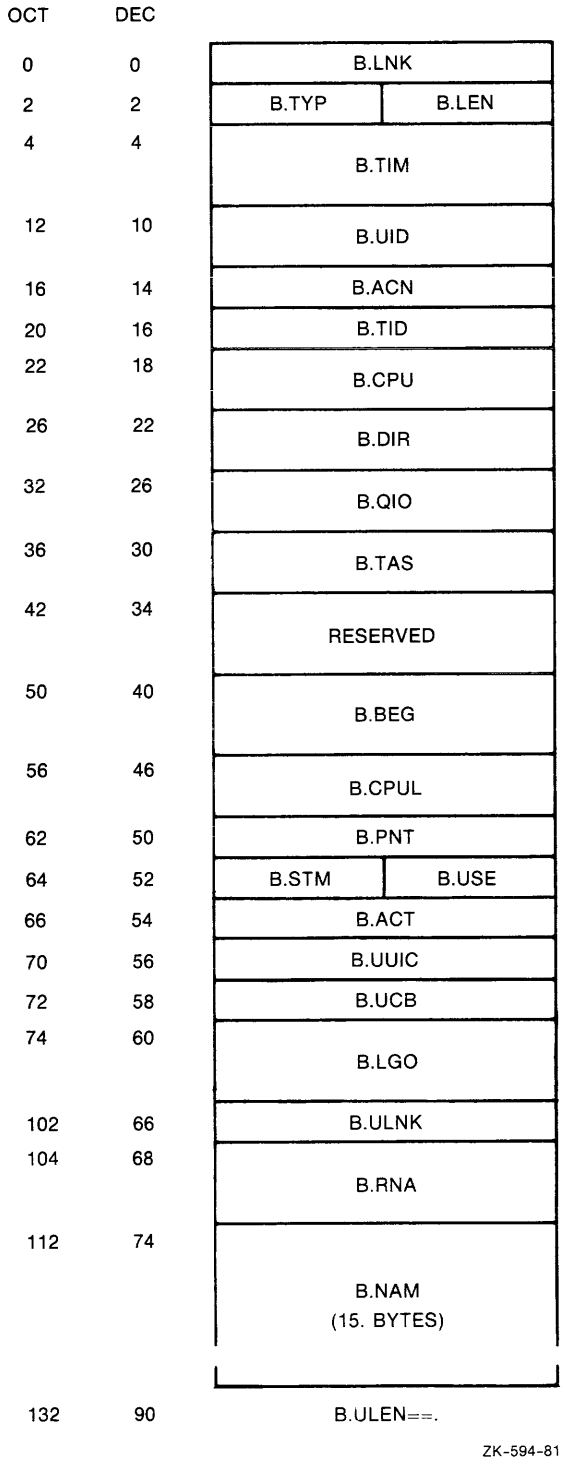
NOTE

The UAB, SAB, mount, and print despooler transaction blocks have changed for this version of RSX-11M-PLUS. An analysis program written for the last version will require modification if numeric field offsets were used instead of the symbolic offsets resolved from the ACNDF$ macro or EXELIB.OLB. A program can distinguish the two forms of transactions by the length code.

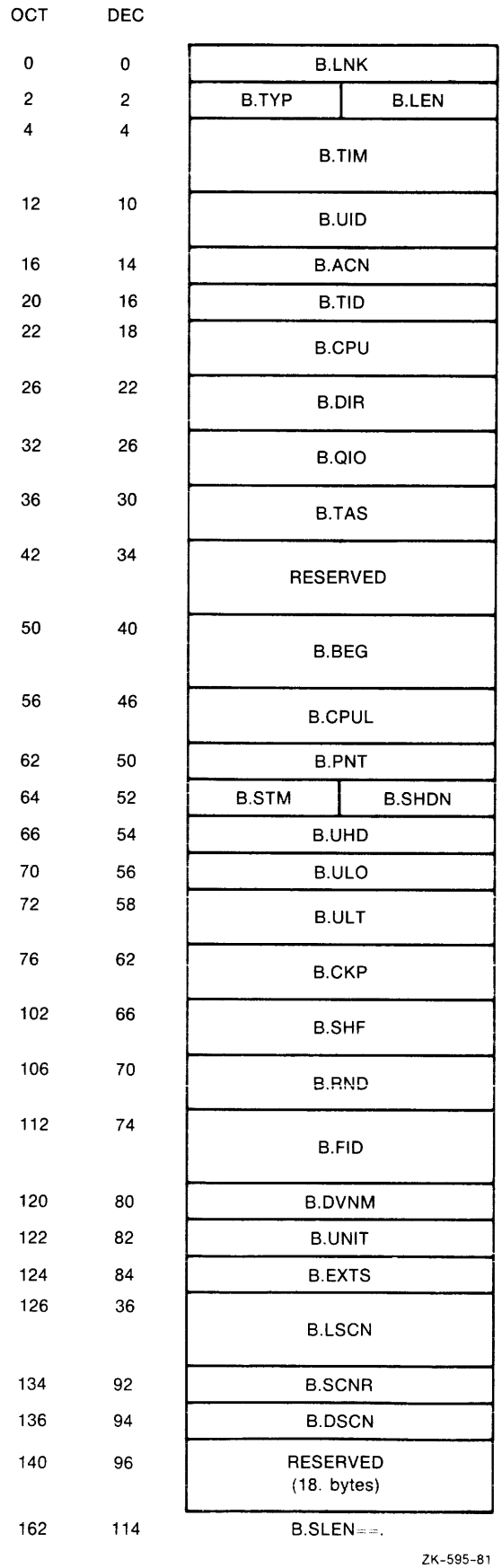
17.6.1 Account and Transaction Block Diagrams



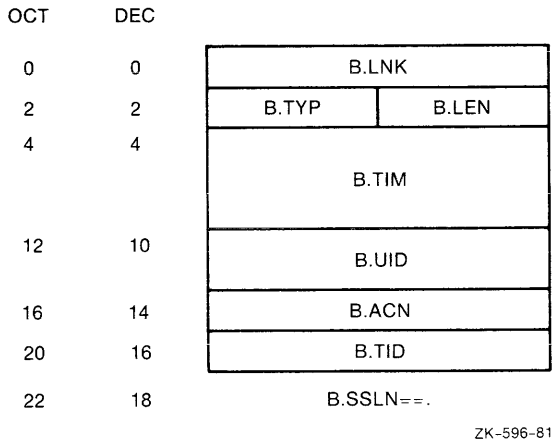
Task Account Block



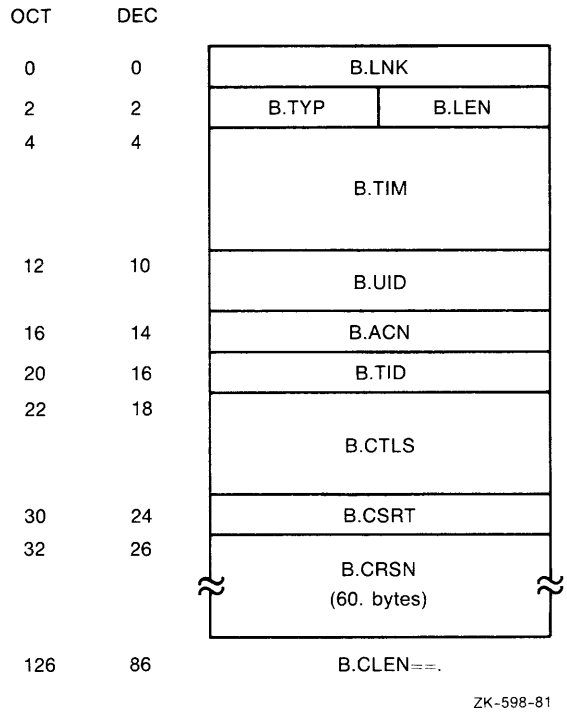
User Account Block



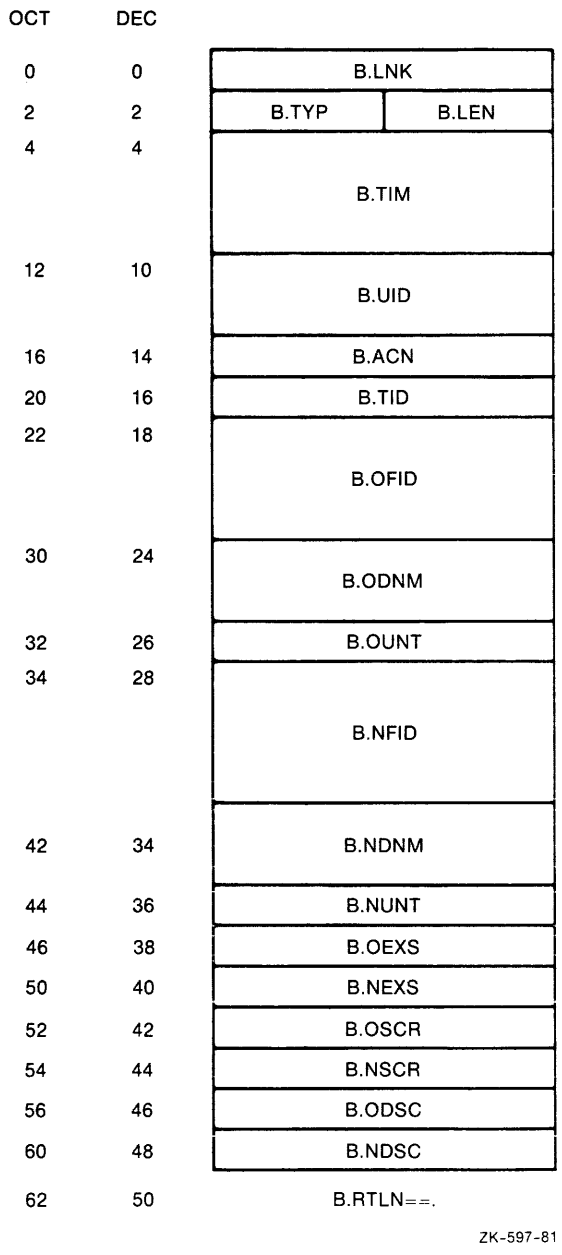
System Account Block



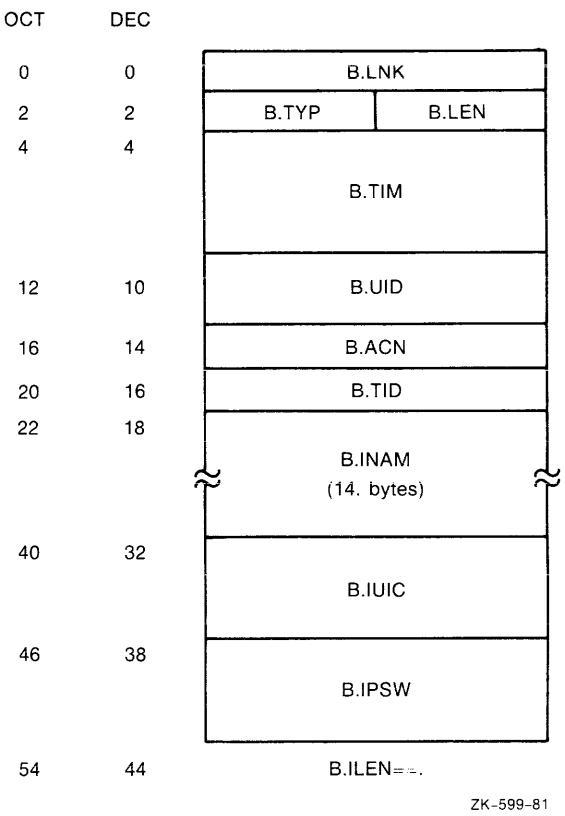
Accounting Startup Transaction Block



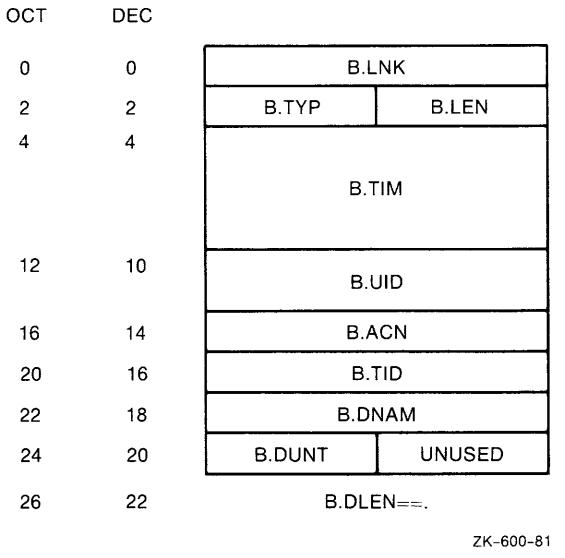
Crash Recovery Transaction Block



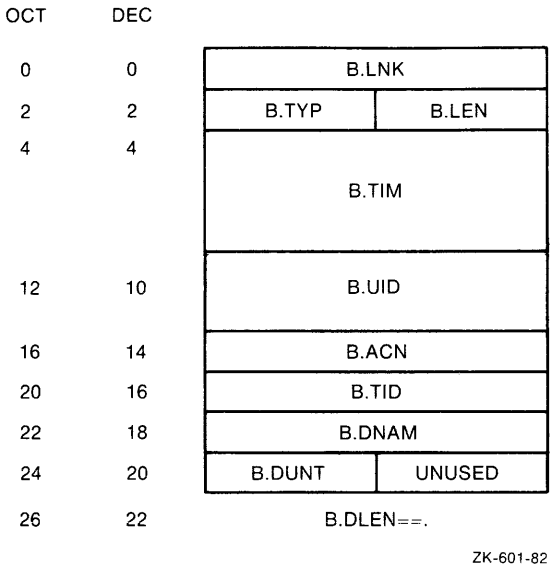
Set Accounting Transaction Block



Invalid Login Transaction Block

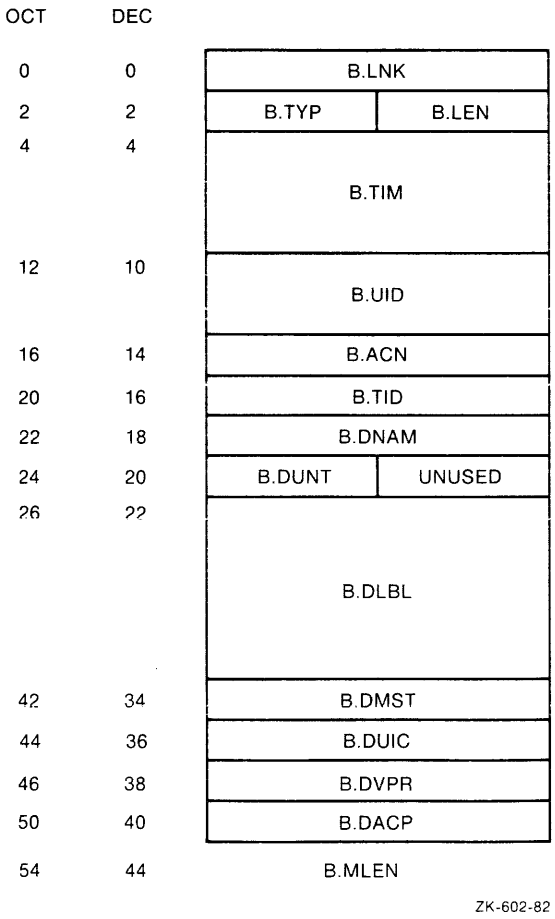


Allocate Device Transaction Block



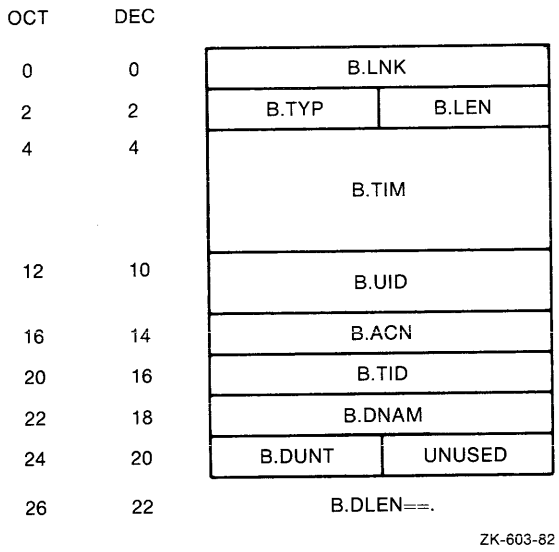
ZK-601-82

Deallocate Device Transaction Block

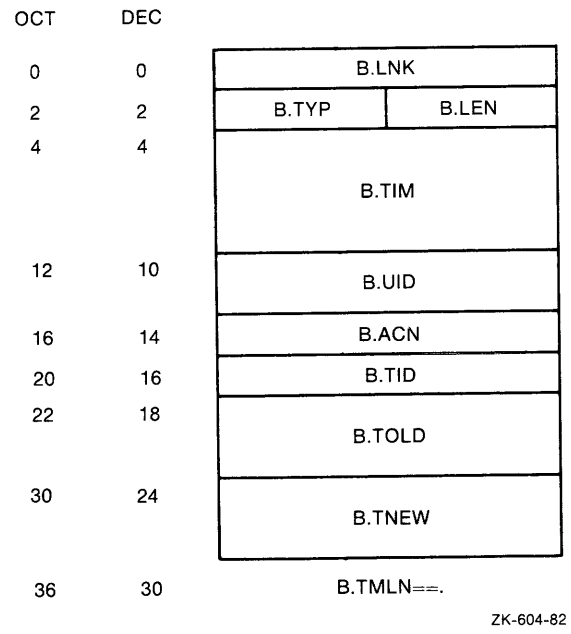


ZK-602-82

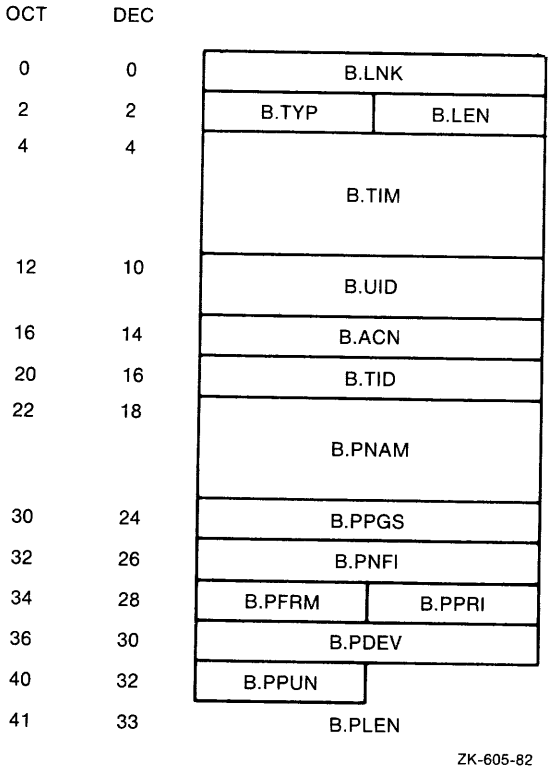
Mount Device Transaction Block



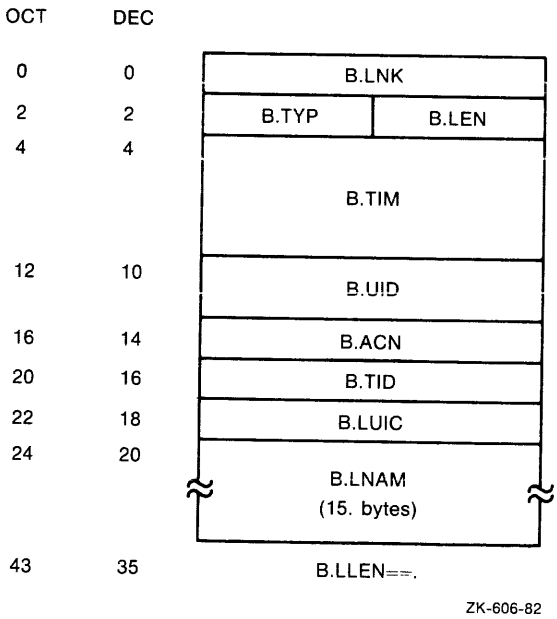
Dismount Device Transaction Block



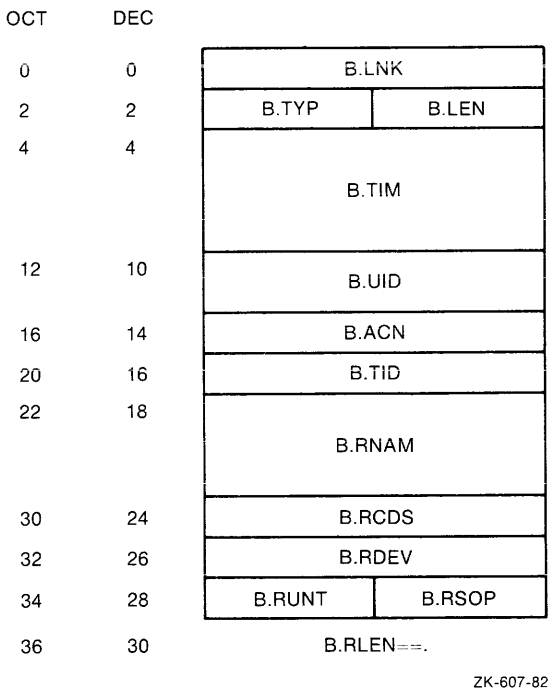
System Time Change Transaction Block



Print Despooler Transaction Block

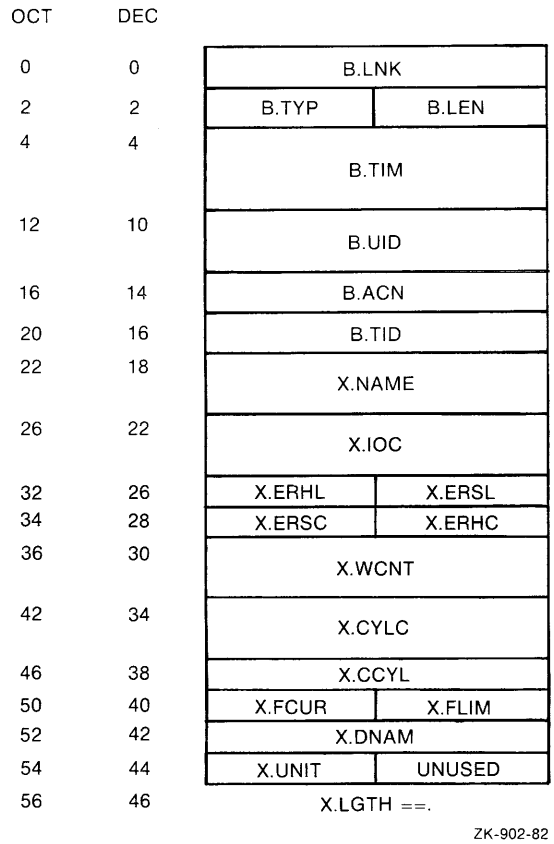


Login Transaction Block



ZK-607-82

Card Reader Spooling Transaction Block



ZK-902-82

Device Statistics Block

17.6.2 Account and Transaction Block Offset Definitions

The header area fields precede the accumulation fields for the TAB, UAB, and SAB. The header area fields also precede each transaction block.

The header provides user identification by two fields (B.UID and B.ACN) that appear in every transaction. These fields are the unique session identification and the account number. The unique session identification is two words long. The first word, the session identification, is three RAD50 characters supplied by the user when he creates the system account by using the ACNT program. The second word is a unique number assigned by the system at login. Thus, it is possible to match every transaction with the login session that produced it. The account number field is a binary word (entered through ACNT) that is copied into every transaction. Resource Accounting does not use either of these fields, except to ensure that they appear in every transaction.

Header Area Fields

Offset	Size or Value	Definition
		Start of header area:
B.LNK::	.BLKW 1	Link to next in SYSLOG queue
B.TYP::	.BLKB 1	Transaction type: 000 through 127 - reserved for DIGITAL use 128 through 255 - reserved for customer use
	BT.SAB==1	System Account Block (SAB)
	BT.UAB==2	User Account Block (UAB)
	BT.TAB==3	Task Account Block (TAB)
	BT.SS==11	Accounting startup transaction
	BT.INV==12	Invalid login transaction
	BT.TIM==13	System time change transaction
	BT.ALL==14	Allocate device transaction
	BT.DEA==15	Deallocate device transaction
	BT.MOU==16	Mount device transaction
	BT.DMO==17	Dismount device transaction
	BT.PRT==20	Print despooler transaction
		Reserved
		Reserved
	BT.LOG==23	Login transaction
	BT.CRH==24	Crash recovery transaction
	BT.DST==25	Device Statistics (UCB Extension)
	BT.RTP==26	Reset Transaction Parameters
	BT.INP==27	Card Reader Spooling Transaction
B.LEN::	.BLKB 1	Transaction length in bytes
B.TIM::	.BLKW 3	Ending time of transaction (supplied by $QTRAN,$QTRN1,$QTRN2)
B.HID==.		Start of header identification area
B.UID::	.BLKW 2	Unique session identification First word - RAD50 Second word - binary
B.ACN::	.BLKW 1	Account number
B.TID::	.BLKB 1	ASCII terminal type: V - virtual, T - real, B - batch, C - console
	.BLKB 1	Unit number
B.HEND==.		End of header identification area

Accumulation Fields for User Account Block, Task Account Block, and System Account Block

```
B.CPU::         .BLKW 2         Total CPU time used
B.DIR::         .BLKW 2         Total directive count
B.QIO::         .BLKW 2         Total QIO$ count
B.TAS::         .BLKW 2         Total task count (not used in TAB)
                .BLKW 3         Reserved
B.BEG::         .BLKW 3         Beginning/login time
B.CPUL::        .BLKW 2         CPU time limit in TAB or UAB
                                   (not used in SAB)
B.PNT::         .BLKW 1         Pointer to SAB if system statistics
                                 to be collected.  Equals 0 if no
                                 system statistics collection.  In TAB
                                 this points to UAB.  In UAB points to
                                 SAB if SYSTEM-STATISTICS:YES or zero.
                                 In SAB always zero.
B.STM::         .BLKB 1         Status mask
                BS.ACT==200      Control block active (UAB, TAB)
                BS.CRH==100      Record from TMP file after crash (UAB, SAB)
                BS.LGO==40       Logged off with outstanding
                                 activities (UAB only)
                BS.CO==40        Tasks TI: is CO: (TAB only)
                BS.TML==20       TAB exists only for time limit (TAB only)
                                 TABS with this bit set will not
                                 be written to the transaction bite.
                BS.ZER==10       Last CPU interval was zero length (TAB, UAB)
                BS.SCN==4        Transaction ready for write to scan (UAB,
                                 SAB) file
```

User Account Block

Note: User Account Blocks must end on a word boundary.

```
B.USE::         .BLKB 1         Use count
B.ACT::         .BLKW 1         Number of currently active tasks
B.UUIC::        .BLKW 1         Login UIC
B.UCB::         .BLKW 1         Pointer to UCB
B.LGO::         .BLKW 3         Logoff time
B.ULNK::        .BLKW 1         Link to next User Account Block
B.RNA::         .BLKW 3         Location in system account file -
                                 (offset, VBN-hi, VBN-lo)
                                 This will be inaccurate if the
                                 account file is changed (additions,
                                 deletions, sorts).
B.NAM::         .BLKB 14.       Last name of user
                .BLKB 1         First initial of user
                .BLKB 1         Unused byte
B.ULEN==.                       User Account Block Length
```

Task Account Block

NOTE: Task Account Blocks must end on a word boundary.

B.PRI::	.BLKB 1	Highest running priority
B.TNAM::	.BLKW 2	Task name
B.TCB::	.BLKW 1	TCB address
B.TST3::	.BLKW 1	T.ST3 from task's TCB
	.BLKW 1	Reserved for future status bits
B.CUIC::	.BLKW 1	Current UIC of task
B.PUIC::	.BLKW 1	Protection UIC of task
B.CTXT::	.BLKW 2	Number of context loads
B.TCKP::	.BLKW 2	Times task has been checkpointed
B.OVLY::	.BLKW 2	Number of disk overlay loads
B.EXST::	.BLKW 2	Exit status and abort code ❶
B.TLEN==.		TAB length

System Account Block

B.SHDN::	.BLKB 1	Accounting shutdown reason code 1 - Maintenance 2 - Reboot 3 - Scheduled shutdown 4 - Accounting shutdown by Shutup task 5 - Other
B.UHD::	.BLKW 1	User Account Block listhead
B.ULO::	.BLKW 1	Number of users currently logged on
B.ULT::	.BLKW 2	Total number of logons
B.CKP::	.BLKW 2	Total number of checkpoints
B.SHF::	.BLKW 2	Total number of shuffler runs
B.RND::	.BLKW 2	Number of CPU intervals rounded up to 1/2

---

❶ The abort codes are:

S.CACT=-4.	Task still active
S.CEXT=-2.	Task exited normally
S.COAD=0.	Odd address and traps to 4
S.CSGF=2.	Segment fault
S.CBPT=4.	Break point or trace trap
S.CIOT=6.	IOT instruction
S.CILI=8.	Illegal or reserved instruction
S.CEMT=10.	Non-RSX EMT instruction
S.CTRP=12.	TRAP instruction
S.CFLT=14.	PDP-11/40 floating point exception
S.CSST=16.	SST abort-bad stack
S.CAST=18.	AST abort-bad stack
S.CABO=20.	Abort via directive
S.CLRF=22.	Task load request failure
S.CCRF=24.	Task checkpoint read failure
S.IOMG=26.	Task exit with outstanding I/O
S.PRTY=28.	Task memory parity error
S.CPMD=30.	Task aborted with PMD request
S.CELV=32.	TI: virtual terminal was eliminated
S.CINS=34.	Task installed in 2 different systems
S.CAFF=36.	Task aborted due to bad affinity (required by runs are offline or not present)
S.CCSM=38.	Bad parameters or bad stack
S.COTL=40.	Task has run over its time limit

```
B.FID::         .BLKW 3         File-ID of transaction file
B.DVNM::        .BLKB 2         Device of transaction file
B.UNIT::        .BLKW 1         Unit of transaction file
B.EXTS::        .BLKW 1         Extend size for transaction file
B.LSCN::        .BLKW 3         Time of last scan
B.SCNR::        .BLKW 1         Scan rate in seconds
B.DSCN::        .BLKW 1         Device statistical scan rate (in
                                 seconds)
                .BLKW 9.        Reserved
B.SLEN==.                       System Account Block length
```

SYSLOG Startup Transaction Block

```
B.SSLN==.                       Transaction length
```

Set Accounting Transaction Block

```
B.OFID::        .BLKW 3         File-ID of old transaction file
B.ODNM::        .BLKB 2         Device of old transaction file
B.OUNT::        .BLKW 1         Unit of old transaction file
B.NFID::        .BLKW 3         File-ID of new transaction file
B.NDNM::        .BLKB 2         Device of new transaction file
B.NUNT::        .BLKW 1         Unit of new transaction file
B.OEXS::        .BLKW 1         Extend size for old transaction file
B.NEXS::        .BLKW 1         Extend size for new transaction file
B.OSCR::        .BLKW 1         Old scan rate in seconds
B.NSCR::        .BLKW 1         New scan rate in seconds
B.ODSC::        .BLKW 1         Old statistical scan rate
B.NDSC::        .BLKW 1         New statistical scan rate
B.RTLN==                        Transaction length
```

Crash Recovery Transaction Block

```
B.CTLS::        .BLKW 3         Time of last scan before crash
B.CSRT::        .BLKW 1         Scan rate before crash
B.CRSN::        .BLKB 60.       ASCII text explaining crash
B.CLEN==.                       Transaction length
```

Invalid Login Transaction Block

```
B.INAM::        .BLKB 14.       Name from login line
B.IUIC::        .BLKB 6.        UIC from login line
B.IPSW::        .BLKB 6.        Password from login line
B.ILEN==.                       Transaction length
```

Device Transaction for Disk Accounting
(ALLOCATE, DEALLOCATE, MOUNT, and DISMOUNT)

```
B.DNAM::        .BLKW 1         ASCII device name
B.DUNT::        .BLKB 1         Octal device unit number
B.DLEN==.                       Transaction length for ALLOCATE,
                                 DISMOUNT, and DEALLOCATE
```

Only Appear for MOUNT Transactions

```
                .BLKB 1         Unused byte
B.DLBL::        .BLKW 6         Volume label
B.DMST::        .BLKW 1         Mount Status Bits
B.DUIC::        .BLKW 1         Owner UIC
B.DVPR::        .BLKW 1         Volume protection code ❷
B.DACP::        .BLKW 2         Name of ACP for device
B.MLEN==.                       Length of MOUNT transactions
```

System Time Change Transaction Block

```
B.TOLD::        .BLKB 6         Old time (yr, mon, day, hr, min, sec)
B.TNEW::        .BLKB 6         New time (yr, mon, day, hr, min, sec)
B.TMLN==.                       Transaction length
```

Print Despooler Transaction Block

```
B.PNAM::        .BLKW 3         Print job name (RAD50)
B.PPGS::        .BLKW 1         Page count
B.PNFI::        .BLKW 1         Number of files printed
B.PFRM::        .BLKB 1         Form number
B.PPRI::        .BLKB 1         Print priority
B.PDEV::        .BLKW 1         Print device name (ASCII)
B.PPUN::        .BLKB 1         Unit number of print device
B.PLEN==.                       Transaction length
```

Card Reader Spooling Transaction Block

```
B.RNAM::        .BLKW 3         Batch or print job name
B.RCDS::        .BLKW 1         Number of cards read
B.RDEV::        .BLKW 1         Reader device name (ASCII)
B.RUNT::        .BLKB 1         Unit number of reader device
B.RSOP::        .BLKB 1         Submit or print (0=SUBMIT, 1=PRINT)
B.RLEN==.                       Transaction length
```

---

❷ The volume protection codes are:

Bits	Protection
0-3	system
4-7	owner
8-11	group
12-15	world

The bit settings within each protection group are:

Bit	Access denied (if bit set)
0	read
1	write
2	create
3	delete

Login Transaction Block

```
B.LUIC::        .BLKW 1         Login UIC
B.LNAM::        .BLKB 14.       User's last name;
                .BLKB 1         User's first initial
B.LLEN==.                       Transaction length
```

The following globals are also defined:

```
B.MAXL==128.                    Maximum transaction length (words)
B.MINL==9.                      Minimum transaction length (words)
```

Device Statistics Block

```
X.NAME::        .BLKW 2         Drive name in RAD50
X.IOC::         .BLKW 2         I/O Count
X.ERHL::        .BLKB 1         Hard error limit
X.ERSL::        .BLKB 1         Soft error limit
X.ERSC::        .BLKB 1         Soft error count
X.ERHC::        .BLKB 1         Hard error count
X.WCNT::        .BLKW 2         Words transferred count
X.CYLC::        .BLKW 2         Cylinders crossed count
X.CCYL::        .BLKW 1         Current cylinder
X.FCUR::        .BLKB 1         Current fairness count
X.FLIM::                        Fairness count limit
X.DSKD::        .BLKB 1         Disk direction (high bit 1=out)
X.DNAM::        .BLKB 1         Device name for accounting
X.UNIT::        .BLKB 1         Unit number for accounting
                .BLKB 1         Unused

X.LGTH==.                       Length of the UCB extension
```

17.6.3 Account Block Linkages

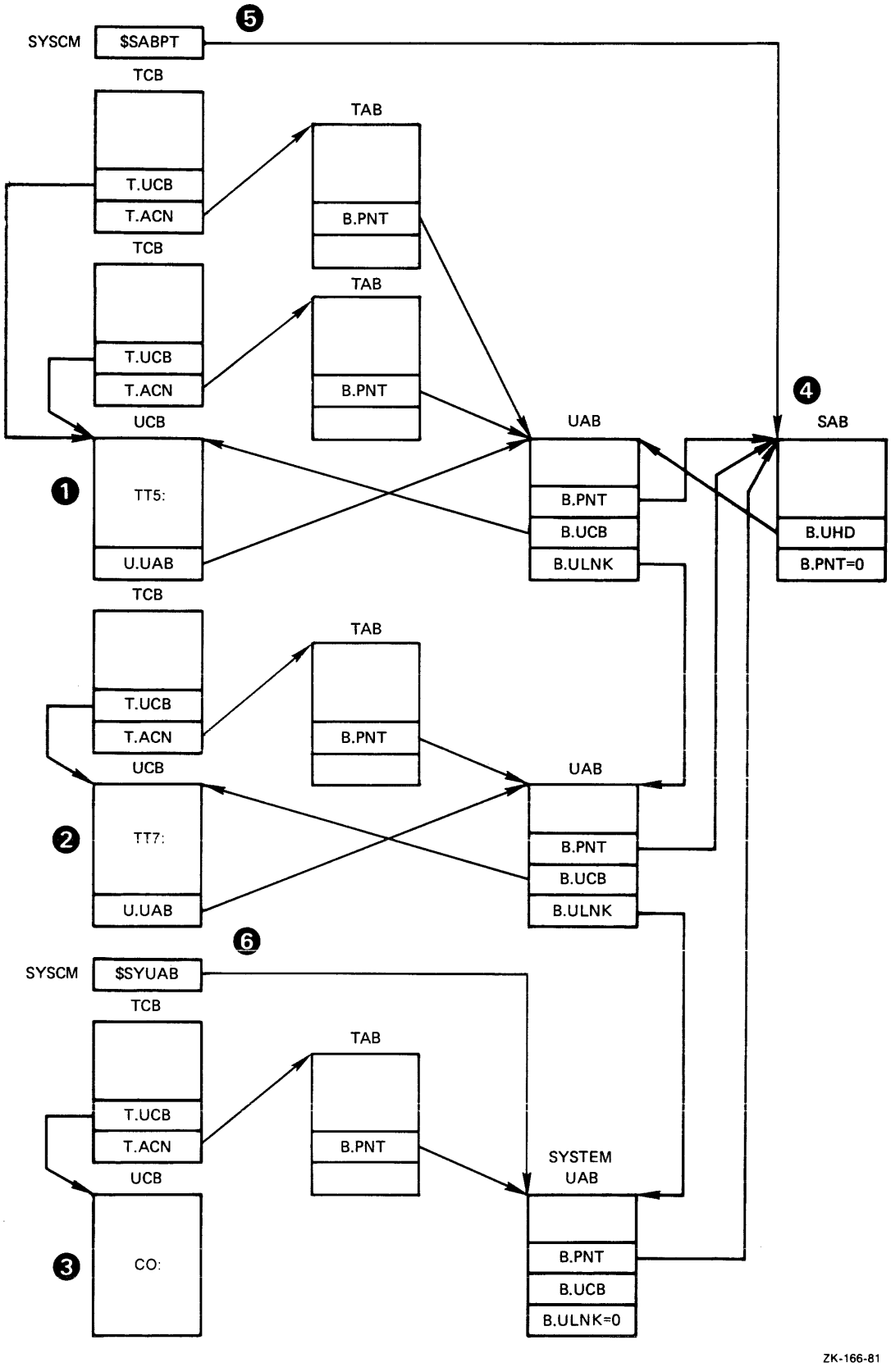


Figure 17-2 Account Block Linkages

❶ Terminal TT5:, which is indicated by the UCB, is that of the most recently logged-on user. The user of this terminal has started two tasks, which are indicated by the two Task Control Blocks (TCB) that contain pointers to the UCB (T.UCB) and each associated TAB (T.ACN). Each TAB then points to the terminal's UAB. The total usage of terminal TT5: is totaled in the associated UAB.

❷ Terminal TT7: is that of the next most recently logged-on user. The user has started a task indicated by the single TCB, which points to a TAB which points to a UAB. The total usage of terminal TT7: is totaled in the associated UAB.

❸ The CO: console terminal has associated with it a single TCB that indicates a system task is running. In effect, the system is also a user and has system usage accumulated at the task level in the TAB and totaled in the UAB associated with the CO: terminal.

❹ The SAB contains the usage totals for all the users including the system itself. The B.UHD pointer in the SAB points to the UAB of the most recently logged-in user. B.UHD and the B.ULNK words in each UAB are the linkages that are used when Accounting periodically writes the UABs to the scan file.

❺ The pointer to the SAB ($SABPT) is in the Executive module SYSCM.

❻ The pointer to the system UAB ($SYUAB) is in the Executive module SYSCM.

NOTE

Although all active UCBs point to the appropriate UABs using U.UAB, this is not true for CO:. U.UAB does not exist for CO:, but instead is reflected by location $SYUAB.

17.7 SYSTEM RESOURCES USED

The following sections discuss the amount of system space that Resource Accounting uses. Secondary pool usage varies from a certain minimum to an amount determined by the number of logged-in users and event-associated transactions.

17.7.1 Executive Space Requirements

Accounting support, which is a SYSGEN option, takes up approximately 532 words.

Space in system pool is required for one active task and one open file.

17.7.2 Secondary Pool Usage

Each logged-in user requires two blocks (32 words per block) of secondary pool for the User Account Block (UAB); that is, two blocks for each logged-on terminal or virtual terminal. Each task running in the system requires two blocks for the Task Account Block (TAB), if task accounting is enabled. Resource Accounting needs two blocks for the SAB and two blocks for the system UAB. Non-memory-resident events such as device mounts require one block for a brief period of time.

17.8 RESOURCE ACCOUNTING MESSAGES

Accounting messages are numbered from 1(10) through 79(10). However, not all of the numbers in this range are used. The number appears at the start of the Accounting message. The Accounting messages have the following format:

```
hh: mm: ss aaaTnn nn.  *aaaaaa* message...
```

where

hh: mm: ss

The time of the message occurrence is hh: mm: ss

aaaTnn

The task issuing the message and the terminal executing the task are displayed as aaaTnn. The first three letters of the task are indicated by aaa. Tnn is the terminal number of the terminal executing the task. SYSLOG appears if the message originated from that task.

nn.

The message number is nn. The message number enables you to find a description of the message in the following list of messages.

aaaaaa

The message severity code is aaaaaa and it has four forms. It may not be present; if it is not, the message is an informational message. The four forms of the severity code are:

DIAG - Information only

WARN - Possible error condition

ERROR - Command did not succeed

FATAL - Extreme error condition

The following is a list of the Accounting messages:

nn:nn:nn aaaTnn 1. Undefined message number nn. at PC aaaaaa

Explanation: The message number is a decimal number specified by nn. The location is specified by the address aaaaaa. This is an internal Accounting error.

User Action: Submit a Software Performance Report (SPR).

nn:nn:nn aaaTnn 2. Directive error. $DSW=nnn PC is just before aaaaaa

Explanation: The directive error code is an octal number specified by nnn. The PC contains an address just preceding the octal address aaaaaa. This message provides additional information to a message that will appear with it.

User Action: Look up the directive error code in the RSX-11M-PLUS Mini-Reference to obtain more information.

nn:nn:nn aaaTnn 4. FCS aaa ERROR CODE = nnn
FILE: DBn: [???,???]filename.filetype;v

Explanation: The word "I/O" or "DSW" is specified by aaa. The error code is specified by nnn. The UIC is inaccessible and appears as question marks.

User Action: An explanation of the error code can be found in the RSX-11M-PLUS Mini-Reference.

nn:nn:nn aaaTnn 5. *ERROR* Illegal filename

Explanation: The file name specified in the START/ACCOUNTING command is invalid.

User Action: Correct the file name and reenter the command.

nn:nn:nn aaaTnn 6. *FATAL* Can't open transaction file

Explanation: The transaction file could not be created for the specified reason. The specified reason is in the message that appears with this one.

User Action: Determine the cause of the error and correct it.

nn:nn:nn aaaTnn 7. *FATAL* Can't open transaction file (by id)

Explanation: After it was created, the transaction file became inaccessible for the specified reason. The specified reason is in the message that appears with this one.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 8. *FATAL* Can't write transaction file

Explanation: The transaction file could not be written for the specified reason. The specified reason is in the message that appears with this one. Accounting cannot recover from this error. Accounting data is lost because the file cannot be written.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 9. *ERROR* Can't close transaction file

Explanation: The transaction file could not be closed. This error cannot be recovered from. Accounting data is lost because the file cannot be closed.

User Action: If possible, determine why the file cannot be closed. Restart Accounting for a short period to see if the error recurs.

nn:nn:nn aaaTnn 10. *FATAL* Can't init new blocks

Explanation: The transaction file cannot be written when the file is being extended. Accounting cannot recover from this error. Accounting data is lost because the file cannot be written to be extended.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 11. *FATAL* Can't init new blocks. I/O error = nnn

Explanation: The transaction file cannot be written when the file is being extended. Accounting cannot recover from this error. Accounting data is lost because the file cannot be written to be extended.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

An explanation of the I/O error code can be found in the RSX-11M/M-PLUS Mini-Reference.

nn:nn:nn aaaTnn 12. *FATAL* Can't write attributes. I/O error = nnn

Explanation: The transaction file cannot be written when the file is being extended. Accounting cannot recover from this error. Accounting data is lost because the file cannot be written to be extended.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

An explanation of the I/O error code can be found in the RSX-11M/M-PLUS Mini-Reference.

nn:nn:nn aaaTnn 13. *FATAL* Can't write attributes

Explanation: The transaction file cannot be written when the file is being extended. Accounting cannot recover from this error. Accounting data is lost because the file cannot be written to be extended.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 14. *FATAL* Secondary pool allocation failure

Explanation: There is insufficient secondary pool remaining to start Accounting.

User Action: Use VMR (Chapter 10) to increase the secondary pool partition size, and reboot.

nn:nn:nn aaaTnn 15. *FATAL* Bad SYSLOG queue

Explanation: The SYSLOG task's input is not empty during Accounting startup.

User Action: Issue STOP/ACCOUNTING CLEAN_UP and restart Accounting.

nn:nn:nn aaaTnn 16. *ERROR* Accounting already active

Explanation: Accounting is already active.

User Action: None.

nn:nn:nn aaaTnn 17. *FATAL* Error spawning SYSLOG

Explanation: The SYSLOG task could not be started due to the specified reason in the message that appears with this one.

User Action: Determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 18. Accounting system not started

Explanation: Accounting was not started due to the specified reason in the message that appears with this one.

User Action: Determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 21. SYSLOG exit with status = n.

Explanation: The exit status of the SYSLOG task is represented by the decimal number n. See the accompanying SYSLOG message.

User Action: Correct the problem, if there is one, and try again.

nn:nn:nn aaaTnn 22. *ERROR* SYSLOG not installed

Explanation: The SYSLOG task was not installed when START/ACCOUNTING was specified.

User Action: Install SYSLOG and reissue the START/ACCOUNTING command.

nn:nn:nn aaaTnn 23. *ERROR* Error bit set. cleanup before restart.

Explanation: Accounting had an error and cannot be restarted until you issue the STOP/ACCOUNTING CLEAN_UP command.

User Action: Issue the STOP/ACCOUNTING CLEAN_UP command and restart Accounting.

nn:nn:nn aaaTnn 25. *ERROR* Accounting did not crash

Explanation: A STOP/ACCOUNTING CLEAN_UP command was issued unnecessarily.

User Action: None.

nn:nn:nn aaaTnn 27. *FATAL* System command handling error

Explanation: This is a Get MCR command line directive failure.

User Action: The ACC task was RUN, so no command line was queued. Access to this task is only legal with the specified Accounting commands.

nn:nn:nn aaaTnn 28. *ERROR* Privilege violation

Explanation: You did not have the privilege for the specified operation.

User Action: Nonprivileged users can see the Accounting data for their terminal only.

nn:nn:nn aaaTnn 29. *ERROR* Accounting not supported in system

Explanation: Accounting cannot be used unless it is selected as an option in SYSGEN.

User Action: Do another SYSGEN and select Accounting as an option.

nn:nn:nn aaaTnn 30. *ERROR* Illegal command or parameter

Explanation: The entered command is incorrect.

User Action: Enter the correct command.

nn:nn:nn aaaTnn 31. *ERROR* Command or parameter ambiguous

Explanation: Parameter was entered with too few letters to be uniquely identified.

User Action: Reenter the command with more letters in the parameter.

nn:nn:nn aaaTnn 32. *ERROR* Syntax error

Explanation: The entered command is incorrect.

User Action: Enter the correct command.

nn:nn:nn aaaTnn 33. *ERROR* Terminal I/O error

Explanation: A terminal I/O error occurred during a prompt or a read to the terminal.

User Action: Try the operation again.

nn:nn:nn aaaTnn 34. *ERROR* Repeated or contradictory parameter

Explanation: Illegal or duplicate parameter combination.

User Action: Enter the correct command parameter.

nn:nn:nn aaaTnn 35. *ERROR* Required parameter missing

Explanation: A parameter is missing in the command line.

User Action: Reenter the command with the required parameter.

nn:nn:nn aaaTnn 36. *ERROR* Value out of range

Explanation: The value entered was either too large or too small.

User Action: Reenter the command with the correct value.

nn:nn:nn aaaTnn 38. *ERROR* No accounting data present

Explanation: The terminal specified in the SHOW ACCOUNTING/-INFORMATION command either does not have any Accounting data or is not logged in.

User Action: Determine the correct terminal to use and reenter the command.

nn:nn:nn aaaTnn 39. *ERROR* Device specification error

Explanation: A device specification was entered incorrectly in the command line.

User Action: Determine the correct device and reenter the command.

nn:nn:nn aaaTnn 40. *FATAL* No startup transaction found

Explanation: The START/ACCOUNTING command and the SYSLOG task are closely related and depend upon each other for correct operation. Any attempt to start the SYSLOG Accounting task by any method other than the use of START/ACCOUNTING results in this message.

User Action: If START/ACCOUNTING and SYSLOG were used as intended and this error resulted, submit an SPR.

nn:nn:nn aaaTnn 41. *WARN* Bad transaction length.
Block(s) of secondary pool may be lost

Explanation: A transaction was probably too large or too small and is probably a user-designed transaction.

User Action: Correct the generation of the transaction. See B.MINL or B.MAXL in Section 17.6.2.

nn:nn:nn aaaTnn 42. *ERROR* UAB Listhead not zero on shutdown

Explanation: Internal consistency error.

User Action: Submit an SPR.

nn:nn:nn aaaTnn 43. *ERROR* Transactions follow SAB

Explanation: Internal consistency error.

User Action: Submit an SPR.

nn:nn:nn aaaTnn 44. *FATAL* UAB Not found in list

Explanation: Internal consistency error.

User Action: Submit an SPR.

nn:nn:nn aaaTnn 45. *FATAL* Exiting due to error

Explanation: SYSLOG exited for the specified reason. The specified reason is in the message that appears with this one.

User Action: Issue STOP/ACCOUNTING CLEAN_UP, determine the cause of the error, correct it, and restart Accounting.

nn:nn:nn aaaTnn 46. *FATAL* Transaction not in secondary pool

Explanation: Internal consistency error.

User Action: Submit an SPR.

nn:nn:nn aaaTnn 47. Exiting

Explanation: Normal SYSLOG exit due to STOP/ACCOUNTING command.

User Action: None.

nn:nn:nn aaaTnn 50. Accounting not running

Explanation: The command issued requires Accounting to be running.

User Action: Issue START/ACCOUNTING and reissue the command.

nn:nn:nn aaaTnn 51. *ERROR* Accounting had a fatal error, must be cleaned up

Explanation: Accounting failed.

User Action: Issue the STOP/ACCOUNTING CLEAN_UP command.

nn:nn:nn aaaTnn 52. *ERROR* SYSLOG Not active

Explanation: SYSLOG has unexpectedly exited.

User Action: Issue the STOP/ACCOUNTING CLEAN_UP command and restart Accounting.

nn:nn:nn aaaTnn 53. *ERROR* Cannot write crash transaction

Explanation: The crash transaction could not be written to the new transaction file for the reason specified in the accompanying message. The crash data is lost.

User Action: None.

nn:nn:nn aaaTnn 54. *ERROR* Error during scan file rename

Explanation: The specified error was encountered when dealing with the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 55. *ERROR* On scan file PUT$

Explanation: The specified error was encountered when dealing with the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 57. Please enter reason for the crash (maximum 60. characters)

Explanation: This is the prompt for the user-supplied crash reason because "yes" was answered as the CRASH_REASON parameter on the START/ACCOUNTING command.

User Action: Enter a crash reason.

nn:nn:nn aaaTnn 58. *ERROR* Terminal I/O Error

Explanation: A terminal I/O error occurred during the input of the crash reason. The crash information is lost.

User Action: None.

nn:nn:nn aaaTnn 60. *ERROR* IO Error

Explanation: The specified I/O error occurred during the SHOW ACCOUNTING TRANSACTION command.

User Action: Correct the I/O condition and try the command again.

nn:nn:nn aaaTnn 64. *ERROR* Illegal file name

Explanation: There is an incorrect file specification for the SHOW ACCOUNTING TRANSACTION command.

User Action: Correct the file specification and try the command again.

nn:nn:nn aaaTnn 65. *ERROR* Open error

Explanation: The specified error occurred during a file open procedure of the SHOW ACCOUNTING/TRANSACTION_FILE command.

User Action: Determine the cause of the error, correct it, and try the operation again.

nn:nn:nn aaaTnn 70. *ERROR* On scan file open

Explanation: The specified error was encountered when Accounting was processing the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 71. *ERROR* On scan file delete

Explanation: The specified error was encountered when Accounting was processing the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 72. *ERROR* On scan file close

Explanation: The specified error was encountered when Accounting was processing the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 73. *ERROR* On scan file open

Explanation: The specified error was encountered when processing the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 74. *ERROR* Copying scan file to trans file

Explanation: The specified error was encountered when processing the scan file. The crash information that would have appeared in the transaction file is lost.

User Action: None.

nn:nn:nn aaaTnn 75. *ERROR* On scan file close

Explanation: The specified error was encountered when processing the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 76. *ERROR* On scan file delete

Explanation: The specified error was encountered when processing the scan file.

User Action: Determine the cause of the problem and correct it.

nn:nn:nn aaaTnn 79. *ERROR* Task not in memory or not active

CHAPTER 18

RSX-11M-PLUS SHADOW RECORDING

This chapter describes Shadow Recording, how to prepare your system for it, and how to control it. Shadow Recording, which is a SYSGEN option, is an Executive feature of RSX-11M-PLUS that allows your system to back up all new data as it is written to a Files-11 disk.

Shadow Recording creates two identical sets of disks called a "shadowed" pair. More than one pair of disks may be shadowed, but shadowed disk pairs cannot overlap. The first disk of the pair, the primary disk, is the original disk that exists whether or not Shadow Recording is active. You can make any disk on your system, including the system disk, the primary disk of a shadowed pair. By using the procedure and the START command described in the following sections, the second disk of the pair, the secondary disk, becomes an exact copy of the primary disk.

The START command invokes a Shadow Recording task called SHAddnn, where ddnn is the device unit of the primary disk. This task is the "catchup" task that copies the primary disk to the secondary disk. Shadow Recording occurs while catchup is in progress. In other words, records are written to both the primary disk and secondary disk without interfering with the catchup process.

After you issue the START command, Shadow Recording provides a dynamic backup of all blocks as they are written to the primary disk. Shadow Recording has important uses in the following applications:

1. Duplicating important information to prevent inadvertent damage or loss. With Shadow Recording, critical data is duplicated on two disks. This redundancy prevents loss of data if a disk error occurs and therefore may provide quicker recovery time in critical applications. Also, downtime may be reduced because disk errors do not necessarily mean an application must be halted.

2. Providing on-line backup to give an "instant" duplicate disk where later backup time or resources are unavailable.

The Shadow Recording feature operates transparently because writing to and reading from the secondary disk is an Executive function. The Executive always writes the same data to the secondary disk that it writes to the primary disk when Shadow Recording is active. However, when a disk read occurs, the Executive reads the primary disk first. If a read error occurs on the primary disk, the Executive reads the secondary disk. The Executive displays all I/O errors occurring on a Shadow Recording disk pair on the operator's console.

You can use five commands to control Shadow Recording (as described in Section 18.2). The START command copies the primary disk onto any secondary disk of the same device type. After you make the bad block

information identical and then use the START command, the secondary disk becomes a logical duplicate of the primary disk. This means that all copied data blocks, bad blocks, headers, directories, and so forth occupy the same logical locations on the secondary disk as they do on the primary disk. Further action on your part is unnecessary unless you want to stop Shadow Recording.

18.1 PREPARING YOUR SYSTEM FOR SHADOW RECORDING

More than one pair of disks can be shadowed, but shadowed pairs cannot overlap. For example, if you make DB0: and DB1: a shadowed pair, you can make DB2: and DB3: a shadowed pair but not DB1: and DB2:. Additionally, the devices must be of the same type.

Before you can use a pair of disk drives for Shadow Recording, you must have available two identical disk packs. The two disk packs must have the same bad block data. You can make the bad block data on the two disks functionally equivalent by following the procedures discussed next.

In the following procedure the assumption is made that you will shadow record the system disk. However, the procedure is valid for any two devices of the same type. To get ready for Shadow Recording, you must perform the following actions for each shadowed pair:

1. Shadow Recording is a SYSGEN option. You must select Shadow Recording as an option during system generation before you can run Shadow Recording.

2. Determine which two device units you want to use for the shadowed pair. They must be the same type of device, for example, two RP06s or two RM02s. One of these two devices will be the device that will contain the primary disk that you want to be "shadowed." The other will contain the secondary disk.

   If for example, your system has four RP06s and you decide to shadow record the system device, DB0:, you can select one of the three remaining devices as the secondary device of the shadowed pair: DB1:, DB2:, or DB3:.

3. Next, gather a set (three or more) of disks that you can use for Shadow Recording. There are two principles involved with the selection of the set of disks for Shadow Recording:

   - The disks being used for Shadow Recording must be logically equivalent (have the same bad blocks). To make the disks logically equivalent, you select one disk of the set as the primary disk and make its bad block descriptor block a logical OR of all the bad blocks in the set of disks. Then, when you start Shadow Recording, the contents of the primary disk, including the bad block information, is copied to the secondary disk. Subsequently, the Executive writes the same data to the secondary disk that it writes to the primary disk. Because the bad block information on the primary disk is an OR of its own bad blocks and those of the secondary disk, the write to the secondary disk does not occur for a defective block.

     You can make the disks logically identical by making entries in the bad block descriptor block of the primary disk (see step 4).

- You can use another disk of the set as a spare, logically equivalent disk in case of disk errors. This is why you should have selected three or more disks for your set of Shadow Recording disks. Using one of these spare disks enables quick recovery from possible disk errors and decreases downtime in critical applications. Error recovery is discussed in Section 18.3.

4. Before you make entries in the bad block descriptor block of the primary disk, run the Bad Block Utility (BAD) with the /LI switch on all the disks in the set of Shadow Recording disks. BAD locates the bad blocks and lists them on your terminal. This list is the information you need to make the bad block descriptor block on the primary disk the logical OR of all the bad blocks.

   Now you can make entries in the bad block descriptor block of the primary disk by using the INITVOLUME command with the /BAD=[MAN] keyword as follows:

   ```
   >INI ddnn:/BAD=[MAN]
   ```

   After you enter the INITVOLUME command, it prints:

   ```
   INI>LBN(S)=
   ```

   In response to this prompt, you can then type from the terminal the list of bad blocks. The INITVOLUME command is described in the RSX-11M/M-PLUS MCR Operations Manual. Also, this bad block information will be used by subsequent INI, DSC, or BRU operations on this volume. The Disk Save and Compare Utility (DSC) and the Backup and Restore Utility (BRU) are described in the RSX-11M/M-PLUS Utilities Manual.

5. Copy the data from your system disk to your primary disk. This disk becomes the primary disk and your new system disk for Shadow Recording.

   For example, if you want to shadow record your system disk, copy it to the primary disk. After this is done, you will have two system disks with identical data, but they will not be identical physically because of the bad block locations. The new system disk becomes the primary disk for shadow recording.

6. Mount as foreign one of the previously selected disks on the secondary device. Use the following MOUNT command:

   ```
   >MOUNT ddnn:/FOR
   ```

Now you are ready to begin using Shadow Recording. Shadow Recording copies the primary disk to the secondary disk when you start Shadow Recording by using the START command. Thereafter, all write operations to the primary disk also are duplicated to the secondary disk.

Both disks have equivalent logical block numbers and, in the case of system disks, both disks are bootable. In addition, after the START command is issued, the secondary disk becomes a Files-11 format disk because the START command makes an exact copy of the primary disk on the secondary disk.

18.2 CONTROLLING SHADOW RECORDING

You can use five commands to control Shadow Recording. However, you can only enter these commands from a privileged terminal. The commands allow you to start, stop, abort, or continue Shadow Recording and to display all device pairs that are being "shadowed." The five commands are:

1. START
2. STOP
3. ABORT
4. CONTINUE
5. DISPLAY

You may enter these commands on one line, for example:

```
>SHAdow START
```

Or, you may enter the command on two lines, for example:

```
>SHA
SHA>START
```

The following sections describe the commands.

START

18.2.1 START Command

The START command:

- Verifies that the primary disk is a Files-11 device and mounted
- Verifies that the secondary disk is mounted as a foreign device
- Verifies that both the primary and secondary disks are identical disk types
- Sets up the Shadow Recording data structure (UMB), which starts the Shadow Recording task within the Executive
- Starts the copying of the primary disk to the secondary disk (catchup)

Format

START ddnn: TO ddxx:

Parameters

ddnn:

The primary Files-11 device that is to be shadowed on the secondary Files-11 device.

ddxx:

The secondary device that is to be the shadowed copy of the primary device.

If new data is written to the primary disk after Shadow Recording is initiated with the START command, Shadow Recording also writes the new data to the secondary disk. That is, when a write occurs on the primary disk, it also occurs on the secondary disk. However, the secondary disk can only be read from the last portion that has been copied. The information about the last block copied is in the UMB control block. A fully redundant secondary disk does not exist until catchup has completed.

STOP

18.2.2 STOP Command

The STOP command prevents Shadow Recording from continuing. However, during catchup, the STOP command cannot stop Shadow Recording. In fact, if you issue the STOP command during catchup, you receive an error message.

To stop Shadow Recording during catchup, use the ABORT command (see Section 18.2.3).

The absence of outstanding I/O does not necessarily mean that all file activity has stopped. You must make certain that all tasks that write records to the shadowed pair have stopped to ensure the integrity of the secondary data before you issue STOP or ABORT.

The STOP command:

- Verifies that the primary device is part of a shadowed pair
- Unlinks the UMB data structure, if no outstanding I/O exists, thereby stopping the Executive from Shadow Recording
- Marks the data structure for deletion if outstanding I/O does exist

Format

STOP ddnn:

Parameters

ddnn:

The device unit of the primary device

ABORT

18.2.3 ABORT Command

The ABORT command stops Shadow Recording even if catchup is in progress. The ABORT command:

- Verifies that the primary disk is part of a shadowed pair
- Aborts catchup if it is in progress and then executes a STOP command

Format

ABORT ddnn:

Parameters

ddnn:

The device unit of the primary device

CONTINUE

18.2.4 CONTINUE Command

The CONTINUE command allows you to restart Shadow Recording on the same two disks that you were using when the STOP command was issued. The CONTINUE command assumes that the two disks are physically alike, and it does not check the disks for equality. You must be sure that no writing occurred on the primary or secondary disks after you issued the STOP command.

The CONTINUE command:

- Assumes that the primary and secondary disks are identical
- Verifies that the primary disk is in Files-11 format and mounted properly
- Verifies that the secondary disk is mounted foreign
- Verifies that both disks are identical device types
- Sets up the Shadow Recording data structure, which starts Shadow Recording within the Executive

Format

```
CONTINUE ddnn: TO ddxx:
```

Parameters

ddnn:

The device unit of the primary device

ddxx:

The device unit of the secondary device

DISPLAY

18.2.5 DISPLAY Command

The DISPLAY command shows all shadowed pairs.

Format

```
DISPLAY
```

Entering the DISPLAY command causes a display of the shadowed device pairs in the following format:

```
UMB          PRIMARY          SECONDARY

xxxxxx       ddnn:            ddxx:
```

xxxxxx

The address of the UMB control block for Shadow Recording. This control block is in the system Executive pool space.

ddnn:

The device unit of the primary device

ddxx:

The device unit of the secondary device

18.3 BAD BLOCK HANDLING

Except for the START command, which copies the primary disk to the secondary disk, the Executive writes the data from a task buffer to the primary disk and then writes it from the task buffer to the secondary disk. If write errors occur on one of the disks, you may not know it until the Executive attempts to read the block in error.

To be notified of write errors as they occur, you can run Shadow Recording with write check for both disks of the shadowed pair. The MCR command:

```
>SET /WCHK=ddnn:
```

enables write check on the device that you specify in ddnn:. However, be aware that a write check of every write operation takes more time to execute than a write only.

When write check is enabled, the Shadow Recording program prints the error messages on the console terminal. These error messages consist of device and logical block numbers when available.

In the case of a read, the Executive reads from the primary disk first. If it encounters an error on the primary disk, it reads the same record from the secondary disk. It is extremely unlikely that both disks would develop an error in exactly the same place. However, if they do, a notification of read errors on the secondary disk means that the primary disk is also bad in the same location.

If read errors occur, several procedures can be followed as discussed next.

18.3.1 Errors on the Primary Disk

If errors occur when the Executive tries to read from the primary disk, the Executive then tries to read the same record from the secondary disk. You are notified of the errors by a message on the console and, at this point, you may decide to let your task continue. If you do decide to let your task continue, be aware that the shadowed pair are no longer alike.

If you decide not to let your task continue, stop the processing task (or tasks) and use the STOP command to stop Shadow Recording. The secondary disk contains all the data that was on the primary disk, including the bad block record. Use the secondary disk as the new primary disk. Designate a new device of the same type for the new secondary disk. Mount the new secondary disk on the secondary device as foreign.

After mounting the old secondary disk on the primary device as the new primary disk, issue the START command. The START command causes the copying of the primary disk to the secondary disk, including the bad block record.

18.3.2 Errors on the Secondary Disk

If errors occur on the secondary disk, the shadow of the primary data is no longer valid. Shadow Recording should be stopped, a new secondary disk should be mounted, and Shadow Recording restarted. The catchup task will then copy the primary disk to the new secondary disk.

18.3.3 Errors on the Primary and Secondary Disk

The simultaneous occurrence of bad data in the same blocks of two different disks is unlikely, and this kind of error is not recoverable. However, whether or not an error occurs at the same or different places on both disks, the only recourse is to stop Shadow Recording and your tasks. Then, select a new primary disk and new secondary disk, mark the bad block information on the new primary disk as before, and copy the old primary disk to the new primary disk.

If the error occurred on different blocks on both disks, the previously processed information can be salvaged by using the old primary and secondary as a shadowed pair input to the Backup and Restore Utility (BRU). BRU copies all the data that is on the input disk and writes it to the output disk. If you use a shadowed pair as input to BRU, only the good data is written to the output disk because of the alternative read that the Executive does on the secondary disk if primary disk errors occur.

Using BRU will not give you a new primary disk. You must create a new primary disk from the output of BRU as discussed in Section 18.1.

Information about how to use BRU is in the RSX-11M/M-PLUS Utilities Manual.

18.4 SHADOW RECORDING MESSAGES

```
hh:mm:ss -- Handler Error code-nn.  on ddnn:
            Shadow Pair:  ddnn:  ddxx:
            Task:  xxxxxx
            Logical Block:  nnn.
            Bytes in xfer:  nnn.
            File ID:  n,n
            File Name:  xxxxxx
            nn.  errors lost due to saturation.
```

Explanation:

hh:mm:ss

The time in hours, minutes, and seconds of the error

code-nn.

The I/O error code (decimal) for the error

I/O error codes are summarized in the RSX-11M-PLUS Mini Reference Manual.

File ID
File Name

The File ID and File name are printed only when they can be determined at the time of the error.

nn. errors lost

When an error occurs on a shadowed disk, the Executive notifies the shadow error reporting task that prints the above message on the console terminal. The Executive stores the information about the error in pool for the error reporting task to pick up and print.

If a number of errors occur rapidly, pool can fill up with this error information before the error reporting task can process it and there may be no pool left to run the system. Therefore, the Executive allows only a certain number of errors to be queued in pool at one time. If this limit has been reached and more errors occur, the Executive does not process them (it keeps track of how many it did not process). This message tells the user how many errors were ignored in this way. "Saturation" refers to pool being saturated with as many error messages as the Executive limit allows.

This message is only printed when errors have not been processed.

User Action: See Section 18.3, Bad Block Handling.

SHA -- Abort error

Explanation: This message occurs if the Shadow Recording ABORT command cannot abort Shadow Recording during catchup.

User Action: Wait until catchup terminates and then issue the STOP command.

SHA -- Bad indirect file specification

Explanation: The indirect command file name is incorrect.

User Action: Reissue the indirect command file specification.

SHA -- Bad volume type

Explanation: The disk is not in Files-11 format or the primary and secondary devices are different device types.

User Action: Check the command line or devices that you are trying to use.

SHA -- Catch-up completed

Explanation: This is an information message. The primary disk has been completely copied to the secondary disk.

User Action: None.

SHA -- Catch-up in progress

Explanation: You have tried to issue a STOP command during the catchup process.

User Action: Issue an ABORT command if catchup is still in progress.

SHA -- I/O error on logical block nnn.

Explanation: An I/O error occurred during catchup.

User Action: None. If the I/O errors are frequent, enter the ABORT command, change disks, and restart Shadow Recording.

SHA -- Illegal command

Explanation: The command that you entered is not a legal command.

User Action: Reenter the command.

SHA -- Input error on input command file
FILE=

Explanation: An error occurred while the indirect command file was being read.

User Action: Take appropriate action; for example, use a different indirect command file, or use a copy of the specified command file.

SHA -- Illegal parse data (internal error)

Explanation: This is an internal system error. The command was correctly entered.

User Action: Reenter the command.

SHA -- Invalid device

Explanation: The device is not a disk device.

User Action: Check the device specification and the device.

SHA -- Maximum indirect command file depth exceeded

Explanation: The indirect command file was nested to too many levels. Only three levels are allowed.

User Action: Check the indirect command file.

SHA -- No room available for UMB

Explanation: Not enough pool space exists to allocate space for the UMB control block for Shadow Recording.

User Action: Check your system. The system may be on the verge of running out of pool space.

SHA -- Not a shadowed pair

Explanation: When STOP or ABORT was issued, the wrong device was specified.

User Action: Reissue the command with the correct device unit.

SHA -- Open error on input command file
FILE=

Explanation: The input command file could not be found.

User Action: Check the command file and its location.

SHA -- Privileged command

Explanation: You are not using a privileged account or terminal.

User Action: Log in on a privileged account or terminal.

SHA -- Send/Receive error

Explanation: A RECEIVE Executive directive error occurred in Shadow Recording during catchup.

User Action: None. This is a system error.

SHA -- Shadow Recording not in system

Explanation: Shadow Recording is not part of your system.

User Action: Shadow Recording must be specified at system generation.

SHA -- Start-up error

Explanation: Shadow Recording cannot spawn the catchup task, so Shadow Recording exits.

User Action: Restart Shadow Recording.

SHA -- Syntax error

Explanation: The entered command has incorrect syntax.

User Action: Reenter the command.

SHA -- System error

Explanation: A system error occurred.

User Action: Determine the effects of the error if possible.

SHA -- Unknown command error

Explanation: Shadow Recording received a command line that it could not recognize.

User Action: Check the command that you entered or the indirect command file.

SHA -- Volume currently part of shadowed pair

Explanation: The specified device is already being shadow recorded.

User Action: Check the devices being shadowed with the DISPLAY command.

SHA -- Volume not mounted properly

Explanation: The primary device is not mounted Files-11 or the secondary device is not mounted foreign.

User Action: Check the devices and mount them properly.

CHAPTER 5

THE BAD BLOCK REPLACEMENT CONTROL TASK (RCT)

The Bad Block Replacement Control Task (RCT) handles bad block replacement and recovery on Mass Storage Control Protocol (MSCP) disks such as the RA80. Bad block handling on MSCP disks consists of four stages: detecting, notifying, replacing, and revectoring. The disk controller (UDA50) handles the detection of bad blocks and notification of the driver (DUDRV). The driver activates RCT. RCT performs all the bad block replacement functions that enable the controller to revector (redirect) I/O from the bad block to the replacement block.

RCT also performs replacement and recovery on MSCP disks that went off line during bad block replacement or before the contents of a write-back cache was copied to the disk.

5.1 BAD BLOCK REPLACEMENT FUNCTIONS

RCT performs the following bad block replacement functions:

- Stores data from the bad block
- Allocates a replacement block
- Updates data structures on the disk
- Initializes the replacement block

5.2 RECOVERY TECHNIQUES

RCT checks MSCP disks that have just come on line for two conditions: incomplete bad block replacement, and write-back cache corruption. If RCT determines that bad block replacement was partially completed when the disk went off line, RCT completes the bad block replacement process. If RCT determines that the write-back cache was not copied to the disk before the disk went off line, RCT software write-locks the disk so that the content of the write-back cache is preserved.

5.3 INSTALLING RCT

If you select a DU: device in SYSGEN, RCT... and DUDRV are both built. RCT... is installed and DUDRV is loaded in SYSVMR.CMD. You load or install and load RCT... and DUDRV only if:

- You have removed RCT... and/or unloaded DUDRV.
- You have edited SYSVMR.CMD to exclude one or both.

If you manually install RCT... and load DUDRV, you must install RCT... first. Otherwise, you receive a message at the console terminal, indicating that the Replacement Control Task is not installed.

To install RCT, use the following command line:

```
>INS $RCT/TASK=RCT...
```

5.4 DISK CONFIGURATION OF MSCP DISKS

Three address spaces on MSCP disks are relevant to RCT:

- Logical blocks (LBNs)
- Replacement blocks (RBNs)
- Revector control tables

5.4.1 Logical Blocks (LBNs)

Logical blocks (LBNs) are accessible to user tasks for transferring data between the disk and the operating system. When the controller encounters a bad LBN and notifies DUDRV, DUDRV activates RCT to replace the LBN.

5.4.2 Replacement Blocks (RBNs)

Replacement blocks (RBNs) are blocks reserved throughout the disk that RCT allocates as substitutes for bad LBNs. RBNs cannot be directly accessed by user tasks during normal data transfers.

5.4.3 Revector Control Tables

The revector control tables are accessible only by RCT (by way of DUDRV) and the device controller. Each table entry corresponds to an RBN according to its location in the table. An entry contains the LBN being replaced and a code describing the status of the RBN. There are five status codes: ALLOCATED, UNALLOCATED, UNUSABLE, PRIMARY, and SECONDARY.

For a given LBN, a primary RBN is in a predetermined location, such as the last sector of the same track as the LBN. For that LBN, all other RBNs are secondary.

Sector 0 of the revector control tables stores information in a flag word that enables RCT to determine if recovery techniques are required when a disk is brought on line.

5.5 BAD BLOCK HANDLING ON MSCP DISKS

Bad block handling on MSCP disks consists of four stages:

- Detection performed by the device controller, UDA50
- Notification performed by the controller which notifies the driver, DUDRV
- Replacement performed by RCT
- Revectoring performed dynamically by the device controller

5.5.1 Bad Block Detection

The UDA50 controller may detect a bad LBN while processing an I/O request from the driver.

5.5.2 Bad Block Notification

When the UDA50 controller detects a bad LBN, it notifies DUDRV by listing the bad LBN in an end packet. DUDRV stores this information in pool and activates RCT.

DUDRV provides RCT with the information necessary to perform bad block replacement:

- LBN
- I/O packet address
- Unit Control Block (UCB) address

5.5.3 Bad Block Replacement

RCT performs the following steps in bad block replacement:

1. Emulates an ATTACH to the disk, inhibiting all I/O other than I/O necessary for bad block replacement.
2. Reads the data from the bad LBN into a buffer in its data space.
3. Determines which RBN on the disk to use by applying a predetermined algorithm.
4. Issues QIOs to DUDRV that update the revector control tables (discussed in Section 5.4.3).

5. Updates the header and data of the bad LBN by issuing an IO.RPL to DUDRV. This I/O function code directs the controller to indicate in the header of the LBN that the LBN has been replaced either by a primary or secondary RBN and writes 128 copies of the address of the RBN in the data field of the LBN.

6. Initializes the RBN by writing the data read from the LBN back to the bad LBN. Since the bad LBN has been replaced by the RBN, the data is transparently written to the RBN by the controller.

   If an RBN is bad, RCT treats it as a bad LBN. RCT finds another RBN, updates the revector control tables, marks the old RBN as unusable, and initializes the new RBN by writing the stored data back to the bad LBN.

7. RCT detaches the unit, and normal I/O to the disk resumes.

5.5.4 Bad Block Revectoring

Revectoring is performed dynamically by the device controller. When the controller first accesses an LBN that has been replaced by RCT, the controller revectors (redirects) the I/O to the RBN that RCT designated as the replacement for the LBN. Thereafter, I/O to the bad LBN is transparently revectored to the RBN without DUDRV's activating RCT.

5.5.4.1 **Primary RBNs** - When the controller detects a bad LBN whose header indicates that it has been replaced by a primary RBN, the controller directly accesses the RBN.

5.5.4.2 **Secondary RBNs** - When the controller detects a bad LBN whose header indicates that it has been replaced by a secondary RBN, the controller must look up the address of the RBN before it can access the RBN. The controller finds the address of the secondary RBN in one of two ways:

1. Checking the data field of the bad LBN, which contains the 128 copies of the RBN

2. Checking the revector control tables for the entry containing the bad LBN

5.6 COMPLETING PARTIAL BAD BLOCK REPLACEMENTS

When a disk unit is brought on line, RCT checks that no bad block replacement was in progress when the disk went off line by checking the flag word in sector 0 of the revector control table. If replacement needs to be completed, RCT completes the bad block replacement.

5.7 PREVENTING LOSS OF WRITE-BACK CACHE DATA

RCT prevents loss of write-back cache data.

A disk with the write-back-cache-in-use flag set will be write-locked each time it is brought on line. To write-enable such a disk, use the /WRITE qualifier with the MCR MOUNT command each time you mount the disk. RCT checks the write-back-cache-in-use flag word in sector 0 of the revector control tables, software write-locks the disk if the flag is set, and writes the following TKTN message to the console output device (CO:):

ddnn: -- Write back caching data lost. Unit write locked

where ddnn: is the disk that RCT has write locked.

NOTE

Because RSX-11M and RSX-11M-PLUS operating systems do not support disks with write-back caching, this function is extremely rare.

RCT performs this function only under the following circumstances:

1. A disk with a write-back cache failed to copy the data from the write-back cache to the disk before going off line, while on another operating system (that supports write-back caching), and was then moved to an RSX-11M or RSX-11M-PLUS operating system.

2. The revector control tables have been altered such that the write-back-cache-in-use flag is set.

5.8 RCT AND OTHER RSX11M/M-PLUS UTILITIES

Utilities such as BAD, BRU, and INI treat MSCP devices as non-last-track devices. RCT does not replace the BAD utility. You must run BAD on MSCP devices to create a bad block descriptor file before you initialize the disk. You can use BAD to find bad blocks and list them in a bad block descriptor file. However, the bad block descriptor file will not contain any information about blocks detected by the device controller and replaced by RCT.

5.9 RCT AND THE ERROR LOGGER

DUDRV notifies the Error Logger with an error log packet whenever an error has been detected that requires bad block replacement. RCT also generates an error log packet indicating that a bad LBN has been successfully (or unsuccessfully) replaced by an RBN. RCT logs the packet under the name of the task that issued the QIO resulting in the detection of the bad block by the controller.

APPENDIX A

SERIAL DESPOOLER TASK

The Serial Despooler Task (PRT...) provides a means of eliminating contention for the system line printer. Rather than waiting for the line printer to become available, a task directs the output intended for the line printer to a disk file. The task issues a Send Data directive to the serial despooler, placing a data block that identifies the file to be spooled in the serial despooler queue. A Request directive is then issued by the task to activate the serial despooler, in case it is not already active. PRT handles FCS-created files, but RMS files can be read only if they are sequential. All files identified in the serial despooler queue are printed in first-in-first-out (FIFO) order.

You may wish to compare the capability of this task with the RSX-11M Queue Manager and its associated despooler. See Chapter 7 for more information about the Queue Manager.

A.1 RECEIVE QUEUE OPERATION

The standard method of placing a user file in the serial despooler receive queue (and requesting its execution) is via the PRINT$ macro call, which is described in the IAS/RSX-11 I/O Operations Reference Manual. Files are spooled in this same manner by the RSX-11 utilities that support the spool (/SP) option. Each entry in the serial despooler receive queue consists of a 13-word data block containing the file-related information illustrated in Figure A-1.

A.2 TEXT REQUIREMENTS

The serial despooler task prints ASCII text with a maximum line length of 132 bytes. It will properly handle files with all modes of FCS carriage control (that is, standard, embedded, and FORTRAN).

A.3 TASK-BUILD INFORMATION

The serial despooler task must be built during an RSX-11M system generation because the task image file (PRT.TSK) is not distributed on the standard release kits. Normally, the serial despooler is built to retain all files that have been spooled, but the serial despooler build file can be edited during system generation to enable the automatic delete feature. (Note also, that a nondeleting despooler can be made to delete, if the data block mentioned in Section B.1 and shown in Figure A-1 has a negative value in word 7.) When the serial despooler is built without automatic delete, spooled files are retained after printing. If the system has a deleting serial despooler, all spooled files are deleted after printing.

See the System Generation and Management Guide for detailed system generation information.

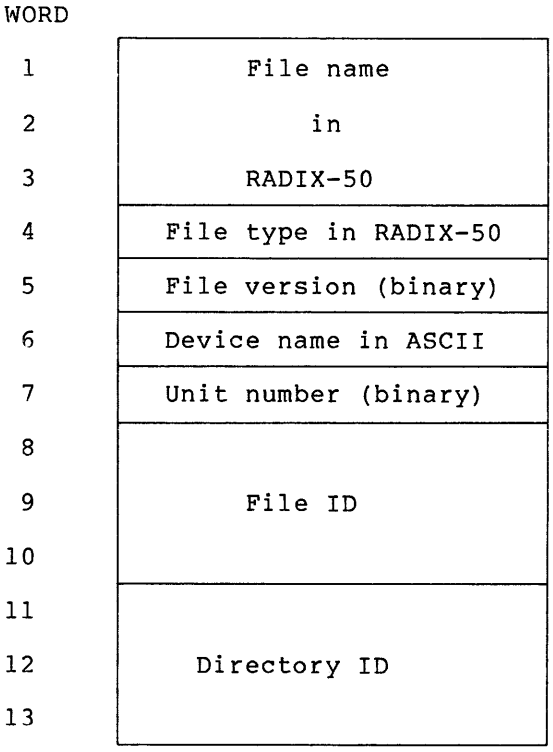


Figure A-1 PRT Send Data Buffer Format

The serial despooler is set to spool files to LP0:. It uses LUN3 as the spool device. The system manager can redirect print files to any record-oriented device (that is, a device similar to the printer) by reassigning LUN3. See the RSX-11M/M-PLUS MCR Operations Manual for information on the REASSIGN command.

A.4 PRT ERROR MESSAGES

All error messages issued by PRT are sent to the console terminal via pseudo-device CO:. The error messages have the following format:

```
PRT -- text
```

In all but the receive failure error, the messages supply information that identifies the sender task and the file in question. All PRT errors are fatal; upon error detection, printing of the input file is terminated, and a cleanup/restart procedure is entered.

In the case of the receive failure error, the sender and file information are unavailable. Furthermore, PRT does not attempt to dequeue additional spool requests because of the nature of this error condition. Instead, PRT exits causing its receive queue to be purged by the system.

RECEIVE FAILURE, d. -- TASK EXITING

Explanation: The Receive Data or Exit directive failed while attempting to obtain the next file specifier from the queue. The system error code (d.) is printed to identify the error.

NO DEVICE NAME - SENDER: task FILE: filename.typ;ver

Explanation: The dequeued print request did not contain a device name.

NO FILE ID - SENDER: task FILE: filename.typ;ver

Explanation: The dequeued print request did not contain a file ID.

OPEN FAILURE INPUT FILE - SENDER: task FILE: filename.typ;ver, d.

Explanation: The specified file could not be opened. One of the following conditions may exist:

1. The file is protected against access for read privileges.
2. A problem exists on the physical device (for example, device cycled down).
3. The volume is not mounted.
4. The specified file directory does not exist.
5. The named file does not exist in the specified directory.
6. The file is already deleted.

The system error code (d.) is printed to identify the failure.

ATTACH FAILURE - SENDER: task FILE: filename.typ;ver, d.

or

DETACH FAILURE - SENDER: task FILE: filename.typ;ver, d.

Explanation: The line printer could not be attached/detached (that is, the system does not contain a line printer). The system error code (d.) is printed to identify the error.

PRINT ERROR - SENDER: task FILE: filename.typ;ver, d.

Explanation: A Queue I/O request to the line printer has failed. The system error code (d.) is printed to identify the error.

I/O ERROR INPUT FILE - SENDER: task FILE: filename.typ;ver, d.

Explanation: An error was detected while reading the input file. One of the following conditions may exist:

1. A problem exists on the physical device (for example, device cycled down).
2. Length of the text line is greater than 132 bytes.
3. File is corrupted or the format is incorrect.

The system error code (d.) is printed to identify the error.

d. LINES WERE TRUNCATED

Explanation: While printing the file PRT truncated d. lines.

APPENDIX B

RSX-11M/M-PLUS UFD CONVENTIONS

The RSX-11M and RSX-11M-PLUS operating systems observe a set of User File Directory (UFD) conventions for files on disk. These conventions provide a consistent method of locating, allocating, and maintaining a file on disk.

A UFD is itself a file, named in the format gggmmm.DIR where g and m are octal numbers in the range 1 through 377 (0 is reserved). The numbers represent the file owner's group and member number respectively. For more information on file ownership and directories, see the RSX-11M/M-PLUS MCR Operations Manual and the RSX-11M/M-PLUS Command Language Manual.

The usage of certain UFDs is common to both operating systems; other UFDs are specific to the RSX-11M or RSX-11M-PLUS operating systems. The following sections describe group and member numbers and their use in identifying the contents of system UFDs. A section on file naming conventions is also included.

B.1 GROUP NUMBER

The group number identifies groups of directories. By convention, group numbers 1 through 200 are reserved by DIGITAL for system files and group numbers 201 through 377 are reserved for user files. The group numbers may vary between RSX-11M and RSX-11M-PLUS. Table B-1 lists the group numbers and their usage.

Table B-1
Group Number Identification

Group Number	Usage
11	Executive files
12	MCR files
13	FCP files (RSX-11M-PLUS)
14	RMD files
15	Privileged utility tasks
23	DCL files
25	Queue Manager files

(continued on next page)

Table B-1 (Cont.)
Group Number Identification

Group Number	Usage
27	On-line configuration (RSX-11M-PLUS)
45	K-Series files
50	FCS files
121	Line printer despooler (RSX-11M-PLUS)
125	Shadow Recording (RSX-11M-PLUS)
126	Accounting tasks (RSX-11M-PLUS)
200	SYSGEN Indirect command files

B.2 MEMBER NUMBER

The second number (member number) identifies the different file types within each group. Table B-2 lists the member numbers.

Table B-2
Member Number Identification

Member Number	File Types
10	Source modules
20	Object modules, assembly and task-build command files, and object module libraries for RSX-11M nonprivileged tasks and RSX-11M/M-PLUS build files
24	Object modules, assembly and task-build command files, and object module libraries for the RSX-11M/M-PLUS Executive and for privileged and nonprivileged tasks on RSX-11M-PLUS
30	RSX-11M unmapped system listing and map files for privileged and nonprivileged tasks.
34	RSX-11M-PLUS and RSX-11M mapped system listings and map files for privileged and nonprivileged tasks.
40	SLP correction files for files located under member number 10
50	RSX-11M unmapped system image and task image files

(continued on next page)

Table B-2 (Cont.)
Member Number Identification

Member Number	File Types
54	RSX-11M-PLUS and RSX-11M mapped system image and task image files
60	RSX-11S unmapped system SLP and PAT correction files applied by Autopatch, and system image and task image files
64	RSX-11S mapped system image and task image files
200	SYSGEN indirect command files

B.3 EXAMPLE SYSTEM USER FILE DIRECTORIES

The system UFDs are composed of a group number and a member number. These numbers identify the specific contents of each directory. For example, if you wanted to locate all the MCR listing and map files, you could find the group number for MCR files (12) and the member number for listing and map files (34) and combine them to specify the UFD [12,34]; this UFD contains the MCR listing and map files. Table B-3 lists the MCR UFDs and the files contained in each UFD. Table B-4 lists the UFDs used by the RSX-11M/M-PLUS operating system.

Table B-3
MCR User File Directories (UFDs)

[12,10]	MCR source files
[12,20]	Unmapped MCR object modules, assembly command files, and library build command file
[12,24]	Mapped MCR object modules, assembly command files, and library build command file
[12,30]	Unmapped MCR listing and Map files
[12,34]	Unmapped MCR listing and map files
[12,40]	SLP correction files for files under [12,10]

Table B-4
System UFDS

[1,1]	System and macro object module libraries
[1,2]	System message and help files

(continued on next page)

Table B-4 (Cont.)
System UFDS

[1,3]	Lost files found by VFY
[1,4]	Postmortem and snapshot dumps
[1,6]	Error logging files and, on RSX-11M-PLUS, Accounting
[1,7]	Spooling queue file and transparent listing file
[1,11]	RSX-11M-PLUS work space for system maintenance
[1,20]	RSX-11M object module libraries and task-build overlay descriptor files for nonprivileged tasks
[1,24]	Object module libraries, task-build command files for mapped tasks, and overlay descriptor files for privileged tasks
[1,34]	Executive and task maps
[1,54]	Executive and system tasks
[2,300]	UETP files
[200,1]	Sample files for system introduction
[200,2]	Field service files

B.4 SYSTEM FILE NAMING CONVENTIONS

System file names have the following general format:

nnnsss.typ

where:

nnn is the system identification for the task. For example, DMP is the identification for the File Dump Utility Program.

sss is a suffix for noting the relationship of the file to the task specified in the nnn identification. For example, PIPBLD.CMD is the command file for building PIP.

Suffixes:

ASM Assembly command file
BLD Task Builder command or overlay descriptor file
RES FCSRES version of the task
FSL FCSFSL version of the task (RSX-11M-PLUS)

typ is the standard mnemonic for file type. RSX-11M/M-PLUS has a set of conventional and default file types that are used to reflect the contents of a file. See Table B-5.

Table B-5
System File Types

Type	File Contents
.BAS	BASIC-11 language source file
.BAT	Batch file
.BLD	Indirect command file used as input to SYSGEN
.B2S	BASIC-PLUS-2 language source file
.CBL	COBOL language source file
.CDA	Crash dump binary file
.CFS	Error logging control file string
.CLB	Indirect Command Processor command file library
.CMD	Indirect command file
.COR	SLP correction file
.CRF	Cross-Reference Processor symbol table file
.DAT	File containing data (as opposed to a program)
.DIR	Directory file
.DMP	File Dump Utility output file
.ERR	Error Logger output file
.FTN	FORTRAN IV, FORTRAN IV-PLUS or FORTRAN 77 language source file
.HLP	Help file
.LOG	Batch or console log file
.LST	Listing file
.MAC	MACRO-11 source file
.MAP	Task Builder memory allocation map
.MLB	Macro library file
.OBJ	Object module (output from either the MACRO-11 Assembler or a compiler)
.ODL	Task Builder overlay descriptor file
.OLB	Object module library file
.PAT	PAT Utility Correction file
.POB	Patched object module used by the PAT Utility

(continued on next page)

Table B-5 (Cont.)
System File Types

Type	File Contents
.SML	System macro library file
.STB	Task symbol table file
.SYS	Bootable system image or other system file
.TMP	Temporary file
.TSK	Task image file
.TXT	Text file
.ULB	Universal library file

INDEX

RSX-11M/M-PLUS
System Management Guide
AA-L679B-TC, AD-679B-T1

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

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

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

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

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03061

Do Not Tear - Fold Here

Cut Along Dotted Line