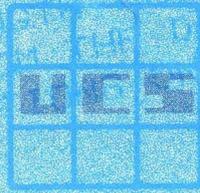# CALMA

INTERACTIVE GRAPHIC SYSTEMS
A division of United Computing Systems Inc

UCS

rdos/cli programmer's reference manual

RDOS/CLI

Real-Time Disk Operating System
and
Command Line Interpreter

Programmer's Reference Manual

FIRST EDITION

Data in this manual is subject to change during future developments. At the time of release, the data in this publication was as accurate and current as possible. However, CALMA is not responsible for any damages (including consequential) caused by reliance upon materials provided.

NOTICE

CHANGE AND REVISION NOTICE

This page provides notification of all changes and revisions incorporated in this manual.

A CHANGE is issued for corrections or additions to a publication. Reprinted pages are provided where there are extensive changes on a page and new pages are provided as required for additional data. Reprinted pages carry a change number at the bottom of the page and changes in text are flagged by vertical change bars in the page margin. Where complete new pages are added or text is moved without changing content, the pages will contain a change number but change bars are not used. When minor changes are required, instructions are given for marking the affected pages. A description of each change which includes the change number, change date, reason for the change, and applicable change instructions are given on this page.

A REVISION is a complete reissue of a publication. All previous changes are incorporated in the revision so that a complete new manual is presented. The current revision level, if other than original, is given on this page.

PREFACE

This manual provides programmers with information required to use the CALMA Real-Time Disk Operating System (RDOS) and Command Line Interpreter (CLI). In compatible CALMA hardware systems, RDOS controls the computer and input/output devices within the parameters established by an interactive system generation (**SYSGEN**) procedure. The user communicates with RDOS using the CLI.

The CALMA RDOS and CLI are enhanced versions of the RDOS and CLI developed by Data General Corporation. Data General's RDOS was developed with foreground and background programming. Enhancements added by CALMA include programming in up to 16 separate grounds.

Information contained in this manual is adequate for use by programmers that understand the Data General versions of RDOS and CLI. For an in-depth understanding of these programs, refer to the following Data General publications:

- Real Time Disk Operating System Reference Manual, 093-000075

- RDOS Command Line Interpreter Reference Manual, 093-000109 (for NOVA computers) 093-000146 (for ECLIPSE computers)

CALMA versions of RDOS and CLI are described in the three sections of this manual as follows:

- Section I contains an introduction and description.
- Section II contains information about CALMA RDOS.
- Section III contains information about CALMA CLI.

ECLIPSE® and NOVA® are registered trademarks of Data General Corporation, Southboro, Massachusetts.
Teletype® is a registered trademark of Teletype Corporation, Skokie, Illinois.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

GENERAL DESCRIPTION

1.1 INTRODUCTION

This section contains information about the organization and use of CALMA RDOS and CLI. Information in this section includes the following:

Allocation of Core
RDOS Disk Storage
Referencing Disk Files
File Attributes
File Organization
File Names
Linked File Names

1.2 ALLOCATION OF CORE

The allocation of core memory for NOVA computers is shown in Figure 1-1. Allocation of core memory for ECLIPSE computers using CALMA's enhanced RDOS follows the same general philosophy.

The ECLIPSE computer provides memory mapping which defines a logical address space for each ground. The first 17 words of each address space are required for use by RDOS. A few addresses have special hardware significance, such as the auto-increment registers or the stack pointer in an ECLIPSE, but the remainder of the address space is completely defined by the user.

≤32K

| System Buffers |
| Resident RDOS |
| Background NREL |
| Background overlay area(s) |
| Background NREL |
| Background TCB pool |
| Background overlay directory |
| **400** Background UST |
| Background page zero |
| **15** RDOS page zero |

User and RDOS Address Space, (Unmapped)

8K≤
physical addresses

| System Buffers |
| Resident RDOS |
| **0** RDOS page zero |

RDOS Address Space (Mapped)

≤32K
logical addresses

| program NREL |
| overlay area(s) |
| program NREL |
| TCB pool |
| overlay directory |
| **400** UST |
| **0** program page zero |

**Foreground and Background
User Address Space (Mapped)**

NOTE
Page zero is the first 1024 (1K) words in a designated memory space.

Figure 1-1.   Address Space Configuration for NOVA Computers

The ECLIPSE mapping unit increases the use of memory space compared to NOVA systems. With memory mapping, the logical (user) address space is not limited since the RDOS system uses a different map, and total address spaces in excess of 32K are possible.

The mapping of logical addresses into physical addresses (memory available to the CPU) is controlled by the RDOS system.

1.3   RDOS DISK STORAGE

A storage disk is divided into sectors, or storage blocks, that will contain up to 256 words of data. For an RDOS disk, the first two sectors are reserved for HIPBOOT, a small program that moves the Bootstrap program into core. The next four sectors (3 thru 6) are also reserved and setup by the Disk Initializer Program. All remaining sectors are available for system use or user file data storage.

1.3.1   Data Storage Area

Data is stored in a Primary Partition, a Secondary Partition, or a Subdirectory.

A Partition is an RDOS storage area capable of managing disk storage allocation separately. A Subdirectory is a storage area within a Partition that is controlled by the Partition. Partitions and Subdirectories are discussed in more detail in the following paragraphs.

1.3.1.1   PRIMARY PARTITION

The Primary Partition consists of the physical disk except the reserved sectors on the disk containing the controlling RDOS Operating System.

The first portion of the Primary Partition contains:

SYS.DR	Lists the names of all Secondary Partitions, Subdirectories, and files on the disk.
MAP.DR	Identifies all unused and used sectors on the disk.
Push Space	Storage area for core images to be temporarily pushed into during program swaps.

NOTE
Push space is a term used to describe a temporary storage area, but **SYS.DR** and **MAP.SR** are actual files.

PRIMARY PARTITION
PICTORIAL REPRESENTATION

Entire disk area
with first six
sectors protected.
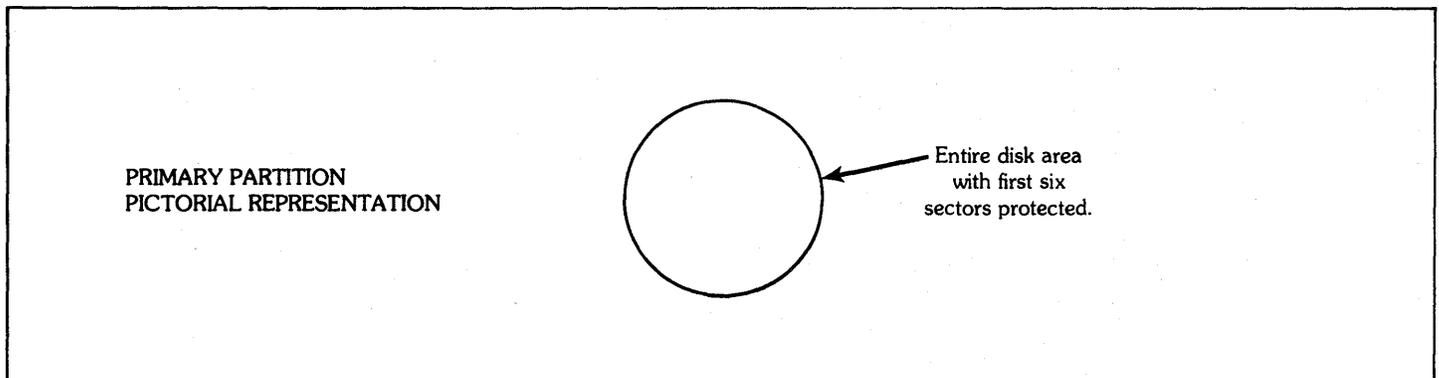
1.3.1.2 SECONDARY PARTITION

A Secondary Partition is a subdivision of the Primary Partition. A Secondary Partition has a fixed length which is set when it is created and is limited to a maximum of 64K sectors. All or part of a Primary Partition may be divided into Secondary Partitions.

NOTE

A large number of Secondary Partitions should not be created since this generally results in a waste of space due to fragmentation.

1.3.1.3 SUBDIRECTORY

A Subdirectory is a variable storage area within a Primary Partition or a Secondary Partition. The variable storage space in a Subdirectory increases or decreases within the total storage space available in the parent partition.

For example, if the current directory is **DC0**;
CDIR SUB0 cr   Creates a Subdirectory on **DC0**.
CDIR DC1:SUB1 cr   Creates a Subdirectory on **DC1**.

Only the Primary Partition exists when a disk is first initialized as an RDOS disk.

The CLI command, **CPART**, is used to create Secondary Partitions. The CLI command, **CDIR**, is used to create Subdirectory storage areas within a Partition. Each time the RDOS System is started, it must be bootstrapped. Bootstrapping does not affect any Partitions, Subdirectories, or files on the disk.

A Primary Partition cannot be deleted. Secondary Partitions and Subdirectories may be deleted. If a Secondary Partition is deleted, all Subdirectories within the Secondary Partition are also deleted; and if a Subdirectory is deleted, all files within the Subdirectory are deleted. In either case, RDOS automatically updates the system directory and map directory.

SECONDARY PARTITION
PICTORIAL REPRESENTATION
SECONDARY PARTITION
PRIMARY PARTITION
ANOTHER PRIMARY PARTITION IN MULTIDISK SYSTEMS

SUBDIRECTORY PICTORIAL REPRESENTATION
SUBDIRECTORY IN PRIMARY PARTITION
PRIMARY PARTITION
SUBDIRECTORY IN SECONDARY PARTITION
SECONDARY PARTITION
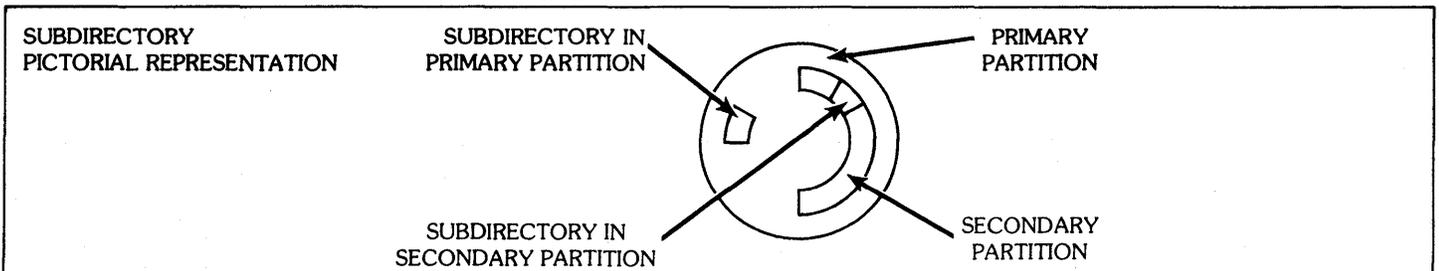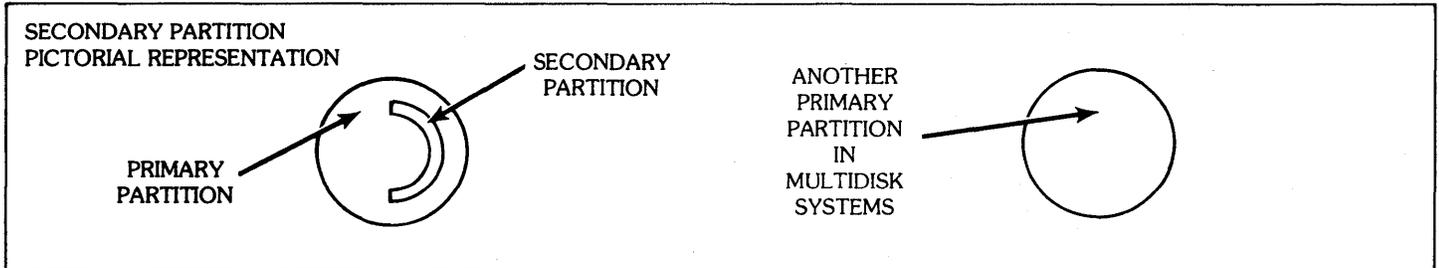
1.3.2 Directories

A directory is an index pertaining to a storage area on the disk.

Each partition contains a Map Directory of the used and available storage space in the partition. Subdirectories do not have unique Map Directories, but use the parent partition's Map Directory. RDOS automatically names every Map Directory **MAP.DR**.

Each storage area on the disk (Primary Partition, Secondary Partition, Subdirectory) has a System Directory that lists by name all subdivisions and files in its storage area.

A directory or partition name contains from one to ten alphanumeric characters. This follows the standard file naming convention with the provision that the **.DR** extension is automatically used. The user should avoid names which could be confused with device names, such as **$LPT**, and make sure that all directory and partition names are unique. For example, while it is possible to have partitions **A** and **B** each containing directory **C**, only one of the directories named **C** could be initiated at any one time. If one of the directories named **C** were renamed to **D** (possible only when it is not initiated), then operations between the two directories are possible. Table 1-1 provides a list of system directory entries.

Table 1-1. System Directory Entries

SUBDIVISION TYPE	SYSTEM DIRECTORY CAN CONTAIN
Primary Partition	Names of Secondary Partition in Primary Partition. Names of Subdirectories in Primary Partition. Names of files stored in Primary Partition. **NOTE** The RDOS System automatically assigns the file name of the device containing the Primary Partition to the directory for the Primary Partition. If, for example, the Primary Partition is on disk pack **DC0**, the directory for the Primary Partition is named **DC0**.
Secondary Partition	Names of Subdirectories in Secondary Partition. Names of files in Secondary Partition
Subdirectory	Names of files in Subdirectory.

Every System Directory, whether a Primary Partition directory, a Secondary Partition directory, or a Subdirectory directory is automatically named **SYS.DR** by RDOS.

Each **SYS.DR** contains the number of files in it followed by a system directory entry for each file. The information in the system directory entry is listed in Table 1-2.

Table 1-2. Information Contained in System Directory

TYPE OF INFORMATION	DESCRIPTION
File Name	One to ten character name assigned to file.
Extension	Extension appended to file name.
Attributes	Data attributes of file.
Link Attributes	Link attributes of file.
Block Count Byte Count in Last Blk. First Address	Programming data. Not available for general use.
Year/Day Last Accessed	Date file last used.
Year/Day Created	Date file created.
Hour/Minute Created	Hour and minute file created.
Variable Data	Programming data. Not available for general use.
User Count	Number of users currently accessing file. A non-zero number indicates one or more users are using the file.
DCT Link	Programming data. Not available for general use.

The CLI command, **LIST**, lists system directory entries for all files in the named Primary Partition, Secondary Partition, or Subdirectory. To allow user files to be listed separately, each user should create his own Subdirectory to contain his files.

Secondary Partitions should not be created for each user because a fixed amount of disk space is specified for each Secondary Partition. the unused space is wasted because it is not available to other users. A Secondary Partition is useful when more than one group of users share the same disk.

RDOS automatically removes deleted file names from the appropriate directory. This means that deleted file names cannot be undeleted, and the user does not maintain an RDOS directory.

1-4

1.4 REFERENCING DISK FILES

Before RDOS can read from or write to a disk file, the user must:

- Identify the directory that contains the file name.
- Identify the file by name.

The user may identify the directory as the current directory or with a directed file access.

1.4.1 Current Directory

Any System Directory on the RDOS disk may be identified as the current directory. For example, the current directory may be the directory for the Primary Partition, for a Secondary Partition, or for a Subdirectory.

The current directory is identified as follows:

- When RDOS is bootstrapped, the current directory is the directory for the Primary Partition on the disk pack containing the controlling RDOS operating system.
- Once RDOS is bootstrapped, the CLI command, **DIR**, may be used to identify any directory on the disk as the current directory.

1.4.2 Directed File Access

If the file name is not in the current directory, the user may prefix the directory name to the file name. If, for example, a file name **FILE** is stored in a Secondary Partition named **SECOND**, the command to type the contents of the file is:

TYPE SECOND:FILE cr

This procedure is called a directed file access.

1.4.3 Directory Name

The complete directory name consists of the Primary Partition name, the Secondary Partition name, if any, and the Subdirectory name, if any.

Since all directories initiated at the same time must have unique names, a working directory name is sufficient as a file name prefix. Assume partition **DC0** contains Subdirectory **X** and **X** contains file **A.SR**. If **X** is not the current directory, a reference to file **A.SR** from some other directory is possible only after **X** is initiated. The complete name, **DC0:X:A.SR** may be used, but the working name, **X:A.SR**, is sufficient.

Each storage area contains a directory identifying the data stored in the area. Therefore, the areas and the directories form a hierarchy. (See Figure 1-2.) The compiler name and the working name for some of the files in Figure 1-2 are listed in Table 1-3.

NOTE
It is possible for two directories to have the same name if they are contained in different partitions, but they cannot be initiated at the same time.

Table 1-3. Directory Names

FILE NAME	PARENT DIRECTORY	COMPLETE DIRECTORY NAME	WORKING NAME
FILEA	DC0	DC0:FILEA	DC0:FILEA
FILEG	SUB1	DC0:SUB1:FILEG	SUB1:FILEG
FILEJ	SP1	DC0:SUB1:FILEJ	SP1:FILEJ
FILEP	SPSUB2	DC0:SP1:SPSUB2:FILEP	SPSUB2:FILEP
FILED	SUB2	DC0:SUB2:FILED	SUB2:FILED
FILEM	SPSUB1	DC0:SP1:SPSUB1:FILEM	SPSUB1:FILEM

Primary Partition

Dir. Name **DC0**
FILEA
FILEB
FILEC

Secondary Partition

Dir. Name **SPI**
FILEJ
FILEK
FILEL

Subdirectory

Dir. **SUB1**
FILEG
FILEH
FILEI

Subdirectory

DIR. **SUB2**
FILED
FILEE
FILEF

Subdirectory

DIR. **SPSUB1**
FILEM
FILEN

Subdirectory

DIR. **SPSUB2**
FILEO
FILEP

The directory hierarchy for this illustration is: **DC0**
    SP1
      SPSUB1
      SPSUB2
    SUB1
    SUB2

Note that **SP1, SUB1,** and **SUB2** are all at the same level.
DC0 is the parent directory for **SP1, SUB1,** and **SUB2**.
SP1 is the parent directory for **SPSUB1** and **SPSUB2**.

Figure 1-2.   Directory Hierarchy

1.4.4 Directory Initialization

A directory must be initialized before a file name in it can be referenced (entered) with a CLI command.

If the **DIR** command is used to identify the directory as the current directory, initialization is automatic.

However, if a directed file access is entered (the complete directory name is prefixed to the file name) all directories in the complete directory name must be initialized before the directed file access is entered with the CLI command. The CLI command, **INIT**, is used to initialize directory names. The directory names must be in descending order. That is, the highest name in the directory hierarchy is entered first, then the second highest name, etc.

The following initialization examples refer to the directory hierarchy illustrated in Figure 1-2. Each example assumes only **DC0** is initialized.

File to be Accessed	Initialization Command Required
FILEJ	**INIT SP1**
FILEN	**INIT SP1:SPSUB1**
FILED	**INIT SUB2**

The number of directories that may be initialized at one time is specified during the interactive system generation procedure called **SYSGEN**. If this number is reached, and other directories must be initialized in order to access a file, some previously initialized directories must be released. See CLI **RELEASE** command.

Table 1-4 lists several command sequences referencing disk files where up to three directories are initialized at the same time. The structure of these commands as they are stored on the disk is shown in Figure 1-3.
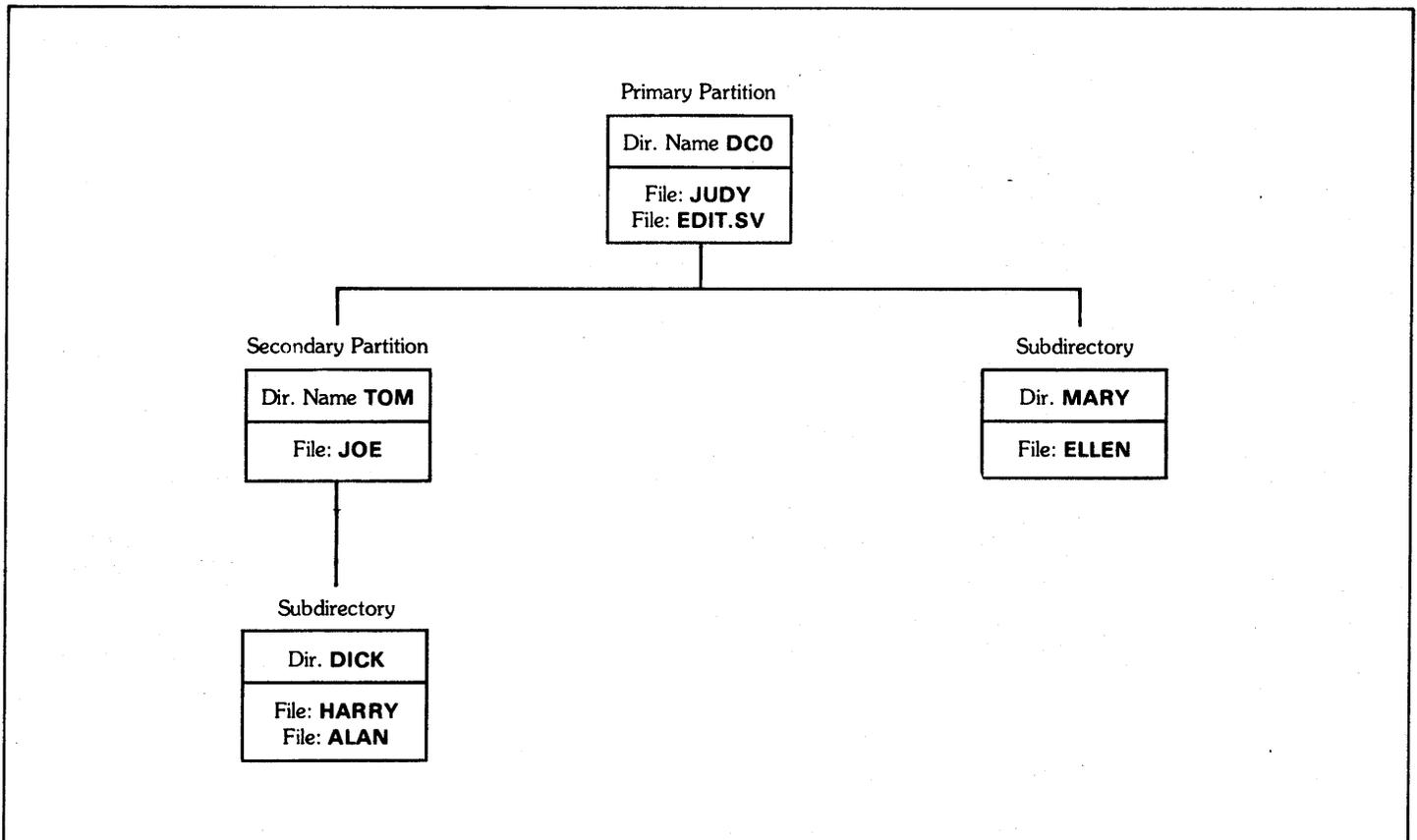
Figure 1-3. Typical Command Sequence Structure

Table 1-4. Typical Command Sequence

COMMAND SEQUENCE	EXPLANATION
DIR DC0 cr	Makes directory for Primary Partition the current directory. The **DIR** command automatically initializes the directory.
TYPE JUDY cr	Types contents of file named **JUDY** at user's console.
INIT MARY cr	Initializes directory for Subdirectory, **MARY. DC0** remains the current directory.
TYPE MARY:ELLEN cr	Types contents of file named **ELLEN** in Subdirectory **MARY**.
RELEASE MARY cr	Releases (unitializes) directory **MARY**. (Only 3 directories can be initialized at one time in this example.)
INIT TOM:DICK cr	Initializes directories **TOM** and **DICK**.
DIR TOM cr	Makes directory for Secondary Partition **TOM** current directory.
TYPE DICK:HARRY cr	Types contents of file named **HARRY** in Subdirectory **DICK**. (Subdirectory **DICK** is in Secondary Partition **TOM**.)
	NOTE
	TYPE TOM:DICK:HARRY could be entered instead of **DIR TOM** and **TYPE DICK:HARRY**; however, **DC0** would remain in the current directory.

COMMAND SEQUENCE	EXPLANATION
TYPE JOE cr	Types contents of file named **JOE** at user's console. No directory specifier is needed because **JOE** is in the current directory, **TOM**.
LINK EDIT.SV/2 cr	Creates a link named **EDIT.SV** in the current directory, **TOM**, which is linked to **EDIT.SV** in the next higher level directory i.e. the Primary Partition.
DIR DICK cr	Makes directory for Subdirectory **DICK** current directory.
DELETE HARRY ALAN cr	Deletes files **HARRY** and **ALAN** in current directory.
DIR TOM cr	Makes directory for Secondary Partition, **TOM**, current directory.
RELEASE DICK cr	Releases (unitializes) directory, **DICK**.
DELETE DICK.DR cr	Deletes Subdirectory, **DICK**. The file space formerly allocated to Subdirectory, **DICK** is reallocated to Secondary Partition, **TOM**. Secondary Partition, **TOM**, contains only file **JOE** and the linked file named **EDIT.SV**.

1.5   FILE ATTRIBUTES

File attributes are defined as characteristics assigned to a disk file that are automatically set when the file is created and may be changed after the file is created.

CLI commands **CHATR** or **CHLAT** are used to change some of the attributes, and CLI command **LIST** lists the attributes assigned to the file.

The file attributes are stored in the system directory entry of the file and can be listed for reference (refer to the example given with the CLI **LIST** command in Section III).

File attributes consist of:

Data Attributes	Attributes relating to the use of the data in the file when it is accessed directly (refer to Table 1-5).
Link Attributes	Attributes relating to the use of the data in the file when it is accessed through an alias file name as a resolution file (refer to Table 1-6 and Paragraph 1.8).
Structure Attributes	Attributes relating to the design or structure of the file on the disk (refer to Table 1-7).

Table 1-5. Data Attributes

ATTRIBUTE CODE	DESCRIPTION	EXPLANATION
N	No-Resolution Allowed	File cannot become a resolution file. Therefore, alias file names cannot be linked to the name of the file.
P	Permanent	Files cannot be deleted. **NOTE** A Permanent file can be deleted if the **P** attribute is removed.
R	Read-Protected	Data cannot be read from the file.
W	Write-Protected	Data cannot be written to the file.
NOTE The following attributes are not normally set by users.	
A	Attribute-Protected	Attributes of file cannot be changed. **CAUTION** After the **A** attribute is set, it cannot be removed.
S	Save	File contains an executable program. Normally set only if a save file is created by RLDR. However, after this attribute is set, it cannot be removed.
? &	User Defined User Defined	These two attributes are available for use by programmers.
NOTES The user should be careful not to set overly restrictive attributes to a file. Never assign attributes **P** and **A** to the same file.	

Table 1-6. Link Attributes

ATTRIBUTE CODE	DESCRIPTION	EXPLANATION
/N	No-Resolution Allowed	File cannot become a resolution file. Therefore alias file names cannot be linked to the name of the file.
/P	Permanent	File cannot be deleted.
/R	Read-Protected	Data cannot be read from the file.
/W	Write-Protected	Data cannot be written to the file.

NOTE
The following attributes are not normally set by users.

ATTRIBUTE CODE	DESCRIPTION	EXPLANATION
/A	Attribute-Protected	Attributes of file cannot be changed.

CAUTION
After the **A** attribute is set, it cannot be removed.

ATTRIBUTE CODE	DESCRIPTION	EXPLANATION
/S	Save	File contains executable program.
/?	User Defined	These two attributes should be used by Assembler
/&	User Defined	Language Programmers only.

Table 1-7. Structure Attributes

ATTRIBUTE CODE	DESCRIPTION	EXPLANATION
none	Sequential	Sequential files are discussed under File Organization, Paragraph 1.6.2.
D	Random	Random files are discussed under File Organization, Paragraph 1.6.3.
C	Contiguous	Contiguous files are discussed under Contiguous Organization, Paragraph 1.6.1.
L	Link	Linked file name.
T	Partition	File is a partition.
Y	Directory	File is a directory.

1.6 FILE ORGANIZATION

Data is stored on the RDOS disk in sectors called storage blocks. A storage block is an area in which 256 words of data can be stored. RDOS assigns an address to each block and then locates the appropriate block of data by looking for the block address.

The organization of files is contiguous, sequential, or random depending on how the blocks of data in the file are arranged. The user selects the file organization through system calls or CLI commands.

1.6.1 Contiguous File Organization

Contiguous files consist of a fixed number of disk blocks located in an unbroken series of addresses. In order to access a contiguous file, the RDOS only needs the address of the first block in the file. For example, if a file contains 5 blocks of data and the address of the first block is 22, then the other four blocks are 23 through 26 as shown below.

The advantage of a contiguous file is that RDOS only needs one address to access the complete file; therefore, file access is faster. The disadvantage of a contiguous file is that the number of blocks reserved for the file is set when the file is created. This fixed length does not change if the amount of valid data in the file (as determined by the user) decreases.

Usually, the only files stored contiguously are files with fixed lengths that should be accessed quickly, such as partitions, **MAP.DR**, and push temporary storage space.

1.6.2 Sequential File Organization

Sequential files consist of a variable number of disk blocks usually scattered throughout the disk. Therefore, in order to access a sequential file, the RDOS needs a link field in each block of the file. The link field is defined as the exclusive OR of the previous and subsequent block addresses. This procedure allows the system to scan a sequential file in either direction by maintaining a few extra words of core storage. For example, a sequential file contains six blocks, and their addresses are 7, 9, 18, 32, 10, and 51.

The link field for block address 32 is the exclusive OR (in binary) of address blocks 10 and 18. When written in binary:

$$10_{10} = \quad 01010$$
$$18_{10} = \quad 10010$$
$$24_{10} = \quad 11000$$

For the last blocks in the file, the link field is the first block address.

On large disks, the last two words in each block contain the link field. In all programs supplied by Data General or CALMA, sequential file access is independent of the size of the link field.

The advantage of a sequential file is that the number of blocks in the file increases or decreases as the file grows larger or smaller.

The disadvantage of a sequential file is that positioning to a random location in a file requires that all intervening blocks between the current block and the desired block must be read.

	File Sequence:	Block 1	Block 2	Block 3	Block 4	Block 5
CONTIGUOUS FILE ORGANIZATION	Disk Block Address:	Block 22	Block 23	Block 24	Block 25	Block 26

	File Sequence:	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6
SEQUENTIAL FILE ORGANIZATION	Disk Block Address:	Block 7	Block 9	Block 18	Block 32	Block 10	Block 51
	Link Field:	9	21	41	24	19	7
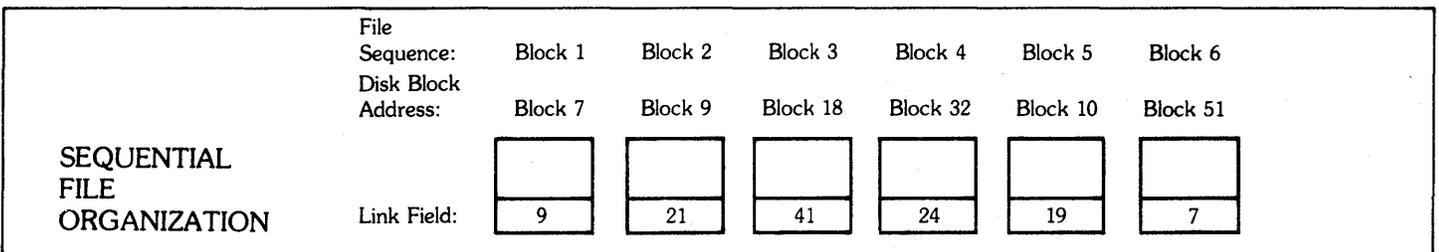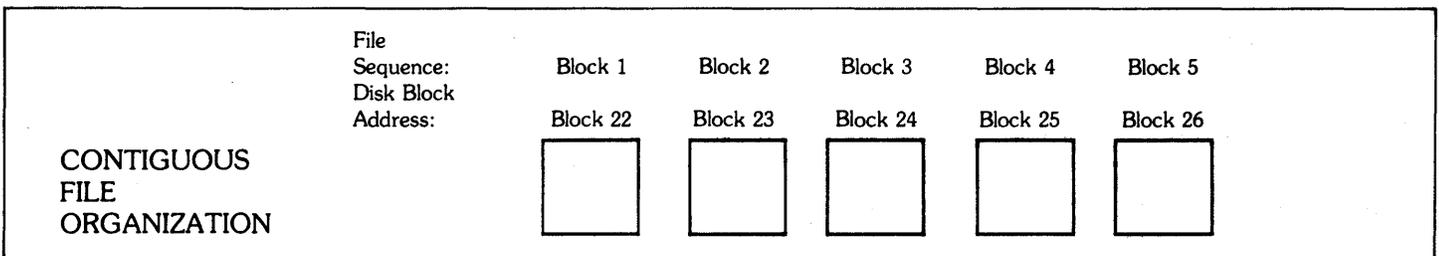
1.6.3 Random File Organization

Random files, like sequential files, consist of a variable number of disk blocks scattered throughout the disk. However, the first block on a random file contains an index of the addresses of every block in the file. The address of the first block containing data is the first address in the file index, the address of the second block containing data is the second address in the file index, etc. If, for example, the physical addresses of the blocks of data in a random file are 97, 109, 216 and 167, the random file would contain the blocks shown below:

If the file index contains more than 255 single word addresses or 127 double word addresses, another block is sequentially linked to the first file index block.

Random files combine the advantages of contiguous and sequential file orgranization. RDOS only needs one address (the address of the file index) to access the complete file, and the number of blocks in the file increases or decreases with the size of the file. In addition, each block of data can be directly accessed so a block of data can be read without reading all data stored before it in the file.

Most files are random because they can be directly accessed, and still take advantage of disk space.

1.7 FILE NAMES AND EXTENSIONS

A file is defined as any collection of data on a mass storage device. All I/O devices are referenced through file names as if they were files.

1.7.1 File Names

The following rules apply to file names:

1. A file name may contain from one to ten alphanumeric characters or the special character, $.

2. A file name cannot duplicate a name that already exists in the directory. If the name already exists, an error message is printed.

3. A one or two character extension preceded by a period may follow the file name.

4. The file name cannot duplicate an I/O device. All input and output hardware devices are assigned file names. I/O file names usually begin with a $ (dollar sign) and contain four or five characters. Table 1-8 lists the file names reserved for I/O devices.

1.7.2 Temporary File Names

Filename Mapping is performed by mapped versions of RDOS. This allows several grounds to do work in the same default directory without conflict. Then, certain filenames, called temporary filenames are no longer used in the normal RDOS sense. Temporary filenames are defined as follows:

1. Filenames that begin with a double dollar sign ($$).

2. Filenames with an extension that has a dollar sign ($) as the first character.

3. Filenames that have a .**CM** extension.

File Sequence:	Index Block	Block 1	Block 2	Block 3	Block 4
Disk Block Address:	Block 20	Block 97	Block 109	Block 216	Block 167
RANDOM FILE ORGANIZATION	97 / 109 / 216 / 167				
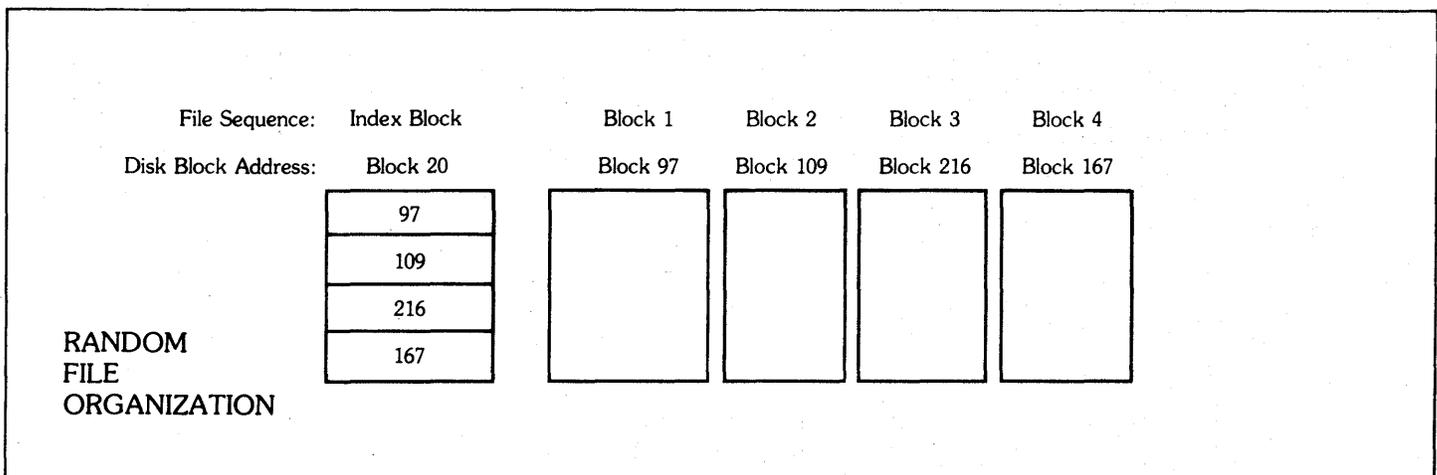
Do not use digits in the extensions to permanent (nontemporary) file names because these digits are used for filename mapping and may result in loss of data. Temporary filenames are only used by one program or series of programs that all run in the same ground. Also, temporary filenames are normally deleted after use and are not assumed to exist before the program or group of programs that uses them has run.

To use filename mapping, CLI command files that have a .CM extension, and are not already temporary files, must be renamed and given a new extension.

Table 1-8. I/O File Names

RESERVED I/O NAME	I/O DEVICE
.DC0, DC1	· Century Disk Drive 0 or 1 (CALMA Convention).
BD0...BD3	Calma Storage Module Disks 0, 1, 2, 3.
	NOTE The primary device is defined as the device on which the controlling RDOS Operating System resides.
MT0...MT7	Magnetic Tape Controller No. 1 for Tape Drives 0, 1, 2, 3, 4, 5, 6, or 7.
MT10...MT17	Magnetic Tape Controller No. 2 for Tape Drives 10, 11, 12, 13, 14, 15, 16 or 17.
$ICR	CPU to CPU Command Receiver (resembles Teletype Input).
$ICX	CPU to CPU Command Transmitter (resembles Teletype Output).
$IXR	CPU to CPU Command Transceiver (resembles Magnetic Tape).
$KEY0...$KEY7	Calma Station Keyboard 1 thru 8.
CRT:0...7	Calma Station CRT 1 thru 8.
QTY:0...QTY:7	Asynchronous Data Communcations Multiplexer 0 thru 7.

RESERVED I/O NAME	I/O DEVICE
$TTI $TTO	Teletype Keyboard 0 } For all system Teletype Printer 0 { messages and responses regarding .OPEN, setup, and ground terminations.
$TTR	Teletype Reader 0.
$TTP	Teletype Punch 0.
$TT11	Teletype Keyboard 1.
$TT01	Teletype Printer 1.
$TTR1	Teletype Reader 1.
$TTP1	Teletype Punch 1.
$LPT	Line Printer 0.
$LPT1	Line Printer 1.
$CDR	Card Reader.
$CDP	Card Punch.
$PTR	High speed paper tape reader.
$PTP	High speed paper tape punch.
$PLT, $PLT1, $PLT2	Plotter 1, 2, or 3 (CALMA Data Channel).
$STA	CALMA Station Lights and Digitizer.
$VMDn	CALMA VMD's (n = 0-7 for eight VMD's)

1.7.3 Extensions

Extensions normally indicate the type of data the named file contains. For example, the extension .SR is appended to the name of the file that contains an assembly language source program.

Extensions are automatically added to many RDOS and CALMA convention files. These extensions are listed in Tables 1-9 and 1-10. The user should become familiar with the filenames and their applicable extensions.

The user may differentiate among files of related data by giving each file the same name, but a unique extension.

The following rules apply to extensions:

1. Extensions are appended to disk file names only.

2. An extension contains one or two alphanumeric characters.

3. The extension is separated from the file name by a period. For example, **FILE.EX**.

4. Do not use a **$** (dollar sign) in an extension. The **$** character denotes a temporary file to many utility programs.

5. Do not use digits in the extension.

6. Do not use an RDOS or CALMA convention extension in other than its intended context.

Table 1-9. Typical RDOS File Extensions

EXTENSION	DEFINITION
.AL	DGL or ALGOL source program.
.BK	Backup of a source file. }
.BU	Backup of a source file. } used by editors
.CM	Temporary command file, e.g., **COM.CM**.
.DR	Directory or subpartition file.
.ER	Error file.
.FR	Fortran source program.
.LB	Library of relocatable binary file.
.LS	Listing file.
.MC	RDOS/CLI Macro.
.OL	Overlay file
.RB	Relocatable binary file.
.SC	Temporary write file used by editors.
.SR	Assembler or Basic source program.
.SV	Save file
.TU	Tuning File.

1.8 LINKED FILE NAMES

Data is stored in a file in the Primary Partition, in a Secondary Partition, or in a Subdirectory. Each Partition or Subdirectory has a directory that lists the name and identifying characteristics of the files stored in it.

Storage space on the RDOS disk is conserved when only one copy of a file containing commonly used data is stored on the disk. Since RDOS users can access files in other directories, a file containing data needed by more than one user may be stored in any Partition or Subdirectory on the disk.

1.8.1 RDOS File Links

An RDOS link allows a directory entry to point to a file in another directory without explicitly specifying that other directory.

Table 1-10. Typical CALMA Conventions for File Extensions

EXTENSION	DEFINITION
.DA	DAL source file.
.DB	Data base.
.DC	Documentation
.DF	Data file
.GO	GPL II™ object
.GS	GPL II source
.GW	GPL II work area
.JD	Background job descriptor file
.JT	Background job template
.LS	DAL listing file
.MB	DDM model backup file
.ML	DAL meta language file
.MR	DDM macro
.M2	DDM model file
.OV	GDS II or DDM overlay file
.PS	DDM permanent symbol file
.S1	DDM symbol file
.S2	DDM symbol file backup
.SA	Schema
.SS	Script source

When file names are linked, the file containing the data is called the resolution file. The directory entry in the directory of the resolution file contains the file name and all identifying information related to the data identified by the file name. The directory entry in the directory of the link file contains the link file name, link attributes (if any), and the file name to which the alias is linked. It contains no other information because no data is stored in the Partition or Subdirectory identified by the directory that contains the alias file name. For example:

```
LIST/E ALIAS.SV   RESOLUTION.SV
ALIAS.SV          DC0: RESOLUTION.SV
RESOLUTION.SV     24576 SPWD 12/13/76 01:05 12/14/76
                  [003505]
```

An alias file name can be linked to the file name of the resolution file or to another link. A chain of linked file names may be up to ten long.

Links may contain a directory specifier, but more often the directory specifier is omitted. The directory is then assumed to be in the parent partition of the subdirectory or secondary partition where the link entry resides. If the link resides in a primary partition the assumed directory is the primary partition. When the CLI **LIST** command is used to list link entries, an at sign and colon (@:) are used to indicate links without directory specifiers. For example:

```
R
LIST LNK
LNK   @:RESOLUTION.RN
R
```

1.8.2   Resolution File

A resolution file is the file that is being linked-to when a CLI **LINK** command is used. More than one link entry may point to a resolution entry, and more than one user at a time may open the resolution file for reading and writing. Before a resolution file is opened to read or write, the directory where it resides must be initialized, and all higher level directory specifiers required for a complete file reference must be initialized also.

The entry in the system directory for a resolution file contains two sets of file attributes:

> Data Attributes
> Link Attributes

The Data Attributes apply to direct users of the data in the resolution file. These attributes are specified when the resolution file is created and may be modified with the CLI command, **CHATR**.

The Link Attributes apply to linked users of the data in the resolution file. These attributes are also specified when the resolution file is created and may be modified with the CLI command, **CHLAT**.

Whenever a resolution file is accessed via a linked file name, the attributes assigned to the linked file name are a combination of the Data Attributes and the Link Attributes of the resolution file. The Data Attributes and the Link Attributes of the resolution file are automatically copied to the user file description for the linked file name. The file attributes in the user file description for the linked file name cannot be altered with the CLI command, **CHATR**.

SECTION II

CALMA RDOS

2.1 GENERAL

This section contains information about the CALMA version of RDOS. RDOS was developed by Data General Corporation and the information in this section assumes that the programmer is familiar with the original RDOS. If additional information about RDOS is required, refer to the Data General publications listed in the preface to this manual. The CALMA version of RDOS has been enhanced to make it more useful with CALMA developed hardware systems. Information in this section is arranged as follows:

    Incompatabilities with Data General RDOS
    CALMA RDOS I/O Enhancements
    Multiground Support for CALMA RDOS
    General Enhancements of CALMA RDOS
    System Generation and Bootstrap Enhancements

2.2 INCOMPATABILITIES WITH DATA GENERAL RDOS

Several incompatibilities resulted from the enhancement of RDOS. These incompatibilities are:

1. Support for the multiprocessor communications adapter (MCA) is removed.

2. Support for the Data General fixed head disks (DK0, DK1) and Zebra disks is removed.

3. Operator message support was removed except for .SYSTM .WROP which is executed by any ground without doing a system generation (SYSGEN) procedure.

4. Common read and write calls .ICMN, .RDCM, and .WRCM are redefined for multiground RDOS.

5. System messages and responses regarding device .OPEN, setup, and ground termination are routed to devices $TTI or $TTO.

6. Support is no longer provided for temporarily suspending one background program so that a new b ckground program can be executed (checkpointing). The checkpointing command (.EXBG) is redefined to start up new backgrounds (refer to Paragraph 2.4.2).

7. The get memory command (.GMEM) and set memory command (.SMEM) are redefined to allow

specifying of up to 16 memory partitions (refer to Paragraph 2.4).

8. To provide better recovery from faults in user programs, error traps in a ground cause the ground to pop to the top of its push stack, resets the default directory to the master directory, and restarts a new copy of the command line interpreter (CLI) program. Traps in a multitask environment may be intercepted using the command to reserve a program interrupt task (.INTAD) and setting up a user status table (USTBR). Refer to Paragraph 2.5.

9. Panic code 100013 is deleted so when the background attempts to pop from level 1, the default directory is reset to the master directory and CLI is restarted similar to error traps, except that the extended trap exit is not taken. Traps in user clock routines are also treated in a similar way.

10. Systems Generation (SYSGEN) and bootstrap procedures are enhanced (refer to Paragraph 2.6).

11. Support for CALMA data channel plotters replaces the support for byte device plotters (refer to Paragraph 2.3).

12. In unmapped environments, the foreground load command .EXFG is not supported.

13. Temporary filenames are mapped (refer to Paragraph 1.7.2).

2.3 CALMA RDOS I/O ENHANCEMENTS

The following paragraphs describe changes made in the Data General RDOS program to provide enhanced I/O support for CALMA RDOS.

2.3.1 Magnetic Tape

INIT and RELEASE of magnetic tape now obtains exclusive access to a given tape unit for the ground INITing it. Also, RELEASE is done only by the ground that INITed.

When a magnetic tape file is opened (.OPEN or .MTOPN) the contents of accumulator 1 (Ac1) are not used as a characteristic mask, but as a physical device mask for seven track tape. Setting 1B0 in Ac1 causes all seven track tape

I/O to operate in the special three-byte mode that CALMA hardware supports; this also disables the standard RDOS software seven track mapping. To obtain normal RDOS seven track operation, set Ac1 to zero.

Magnetic tape positioning options are added for more control and flexibility of the tape drives. This requires special file names when the tape file is opened (these options work with either .OPEN or .MTOPD):

MTn:XX   Position to first unused file on the tape, if tape is fully initiated, then position to load point.

MTn:SS   No positioning, leave tape where it was.

MTn:NN   Space tape forward one file. If used with non-direct I/O, this command writes or reads the next file on tape.

2.3.2 Disk Devices

A special file, $DISK, is defined in the primary partition, and makes the entire disk appear to be one very large contiguous file, with HIPBOOT as its first block. This file is write-protected and is read with .RDB or .ERDB. On disks larger than 64K sectors, additional files are defined to allow access to 64K pieces of data. The additional files are $DISK1, $DISK2, etc. This feature is used to quickly save the disk on tape. This RDOS program is called BACKUP.

The CALMA disk drives do not use the standard Data General disk key of 'DPn'. Instead, they use new three letter keys. Century disks are 'DCn' as follows:

DC0   Century; Unit 0; Device 20
DC1   Century; Unit 1; Device 20

The 80 megabyte (80 MB) and 300 megabyte (300 MB) storage module disks are 'BDn' as follows:

BD0   Storage Module; Unit 0; Device 26
BD1   Storage Moduel; Unit 1; Device 26
BD2   Storage Module; Unit 2; Device 26
BD3   Storage Module; Unit 3; Device 26

Extended Block I/O is enhanced to allow I/O into a non-contiguous series of memory blocks. A restriction is imposed by this new form so that an I/O transfer begins on an even 1K word boundary. The new calling sequence uses a formerly illegal combination of register values. This enhanced form of the call is limited to 64 blocks maximum per transfer.

Ac0 = address for list of physical I/O blocks.
    (bit 0 set)
Ac1 = disk address into file.
Ac2 = (left byte: number of sectors); (right byte: channel).
.SYSTM
.ERDB/.EWRD
<error return>
<normal return>

A .SYSTM call (.SDISK) is added to CALMA RDOS to allow more powerful manipulation of random disk files. The .SDISK call is used to change the size of a random file or to free a sector within a random file.

Change size of random file:

Ac0 = address of UFD with UFTBK and UFTBC set up.
Ac1 = 0
.SYSTM
.SDISK <chan>
<error return>
<normal return>

The size of the random file open on chan is adjusted to match the UFTBK and UFTBC values given. If the size of the file is being reduced, all sectors past the new end of the file are freed and any data contained in them is lost.

Free a sector in a random file:

Ac0 = relative sector number.
Ac1 = 1
.SYSTM
.SDISK <chan>
<error return>
<normal return>

The relative sector given by Ac0 is freed and any data contained in it is lost. If the sector is read before it is rewritten, it appears as all zeros.

2.3.3 CPU to CPU Devices

The CALMA CPU-to-CPU hardware is supported as an RDOS SYSGEN option. When it is selected, three new devices are available for use. The command receiver is the '$ICR' that supports the command, .RDL. The command transmitter is '$ICX' and is written to using .WRL or .WRS. A .WRR to the $ICX has the special meaning of clear the CPU-to-CPU handler and send an abort/clear to the other CPU. The last device is '$IXR' which supports the CPU-toCPU data channel device and is used to do large transfers of data. This device supports .RDS for input and .WRS for output.

2.3.4 Station Devices

CALMA station keyboards and CRT's are supported as RDOS system devices using '**$KEYn**' and '**CRT:n**' respectively. The station lights and digitizer are also accessed using the single device '**$STA**' as well as '**$KEYn**'. (n is the number of the station; 0 <= n <= 7).

$KEYn   The keyboard is read as a standard byte-oriented input device. Commands **.RDL** and **.RDS** are supported. The **.RDL** command prompts the user with a cursor on the CRT and echoes the characters typed. The **.RDS** command inputs with a cursor but does not echo.

.WRS is used to output to the keyboard lights. The following data is supplied to this command:

> word 0 - Keyboard lights word 1
> word 1 - Keyboard lights word 2
> word 2 - Keyboard lights/Speaker word 3

Data to **.WRS** must be word aligned and the byte count even. All the words do not need to be output. If the byte count is less than 6 then the words loaded are truncated with the bottom up.

The digitizer is read, and optionally a character input, using **.RDR**.

Ac0 = address of six word area.
Ac1 = 0 for no keyboard input or wait
    <> 0 for keyboard input.
.SYSTM
.RDR <chan>
<error return>
<normal return>

If Ac1 is zero, only the digitizer is read and the call does not wait unless there is another keyboard read outstanding for the same station keyboard. If Ac1 is nonzero and the device buffer is empty, one character is input and the command waits. The data is read as follows:

> word 0 - character (right justified)
> word 1 - X low order
> word 2 - Y low order
> word 3 - High order X and Y
> word 4/5 - (reserved for future use)

.WRR is used to turn off the processing of control characters and interrupts from the keyboard and is also used to perform a hardware reset of the digitizer.

Ac1(1B15) = 0 disable keyboard translation
Ac1(1B15) = 1 enable keyboard translation
Ac1(1B0) = Reset the digitizer.
.SYSTM
.WRR <chan>
<error return>
<normal return>

To disable translation of keyboard input, issue the **.WRR** command to **$KEYn** with Ac1 = 0.
The translation mode is also enabled by **.RESET** and **.RTN**.

To use the keyboard separately from the CRT for a given station, the **$KEYn** device is **OPEN**ed with a mask in Ac1 of 1B0; this also implicitly masks out the **DCKEY** bit and disables keyboard translation.

Three special characters are used in keyboard translation mode as follows:

<Control> causes the next character typed to be masked to 6 bits before further processing.

<Clear> causes any input on this line to be flushed, the screen cleared, and cursor positioned to the upper left corner.

<Exc> is the ASCII Escape, octal 33.

$STA   The '**$KEYn**' commands **.WRS** and **.RDR** are also used with '**$STA**', except the command never waits on **.RDR** regardless of the value given in Ac1. The left byte of Ac2 is used to specify the number of the station that is being addressed. This allows the digitizer to be read while a task is waiting on a character from the keyboard.

CRT:n   **.WRR** (count > 0) causes output of hardware scope commands to the appropriate CRT. Ac0 is the word address of the data and Ac1 is the word count. **.WRR** (count <= 0) causes an output of (- count) words. Following this, if (count <> 0) the buffer is finished forcing out the commands.

.**WRS** and .**WRL** generate characters using a 5x7 character matrix augmented with descending characters. Output stops either at the end of a page (32 lines), or when a form feed is processed. A form feed causes a special 'Pp' symbol to be typed on the screen in addition to waiting for keyboard input. After a key is struck on the keyboard the CRT screen is cleared and output begins again at the upper left corner. If **DCNAF** is masked out on .**OPEN**, at page end or form feed and continue, .**WRS**/.**WRL** do not wait but flash the screen immediately.

Direct I/O to the CRT is possible using .**MTOPN** and .**MTDIO** commands. This is similar to magnetic tape except the command field in Ac1 to .**MTDIO** is ignored and the status returned in Ac2 is not defined.

Ac0 = address of data buffer.
Ac1 = word count (0<=count<4096)
      count = 0 for 4096 word write.
.**SYSTM**
MTDIO <chan>
<error return>
<normal return>
Ac1 is word count transferred (after normal return).
Ac2 is undefined (after normal return).

NOTE
Commands .**RDR** and .**WRS** to **$KEYn** and .**WRR** to the CRT are implemented in non-overlay code to provide minimum overhead for interactive systems. The use of the line read and write is not recommended for this type of application. **MTOPN** and .**MTDIO** also yield efficient CRT I/O, especially when large amounts of data are output.

2.3.5 Data Channel Plotters

CALMA Data Channel Plotters are implemented in CALMA RDOS using the .**MTOPN** and .**MTDIO** commands. The plotters are represented by '**$PLT**' (device code 15), '**$PLT1$**' (device code 21) and '**$PLT2**' (device code 23). Support is similar to .**MTDIO** for the CRT. The error return is never taken for a device error but the user program detects device errors from the returned status word. The .**MTDIO** returns a partial count transferred, like the magnetic tape, if the transmission was not completed.

Ac0 = address of data buffer
Ac1 = word count (0<count<4096)
      count = 0 for 4096 word write.

.**SYSTM**
.**MTDIO** <chan>
<error return>
<normal return>
Ac1 - word count transferred (after normal return).
Ac2 - device status from DIA (after normal return).

2.3.6 Vector Memory Displays

CALMA VMD's are supported as devices '**$VMDn**', where n goes from 0 to 7 for up to eight VMD's. Support for the VMD is basically .**MTOPN** and .**MTDIO**, except .**WRR** is done to text or wait for the return message.

.**MTDIO** is standard except the command code 0 is read and command code 1 is write.

Ac0 = address of data buffer
Ac1 = (0 for read; 10000₈ for write) + word count
      0<=count<256
.**SYSTM**
.**MTDIO** <chan>
<error return>
<normal return>
Ac1 = word count transferred (after normal return).
Ac2 = device status from DIA (after normal return).

.**WRR** uses Ac1 as a parameter to tell it whether or not to wait for a message. Zero means not to wait and nonzero means to wait. Ac0 is returned with the word count that is read on the .**MTDIO**, which must be read next. If Ac1=0, no waiting was selected on the call to .**WRR**, and no read count is received so far, then error **ERDLE** is returned. When .**WRR** returns a word count it resets the word count register until another count is sent.

Ac1 = 0 if no wait for return message
      <> 0 to wait for return message and count.
.**SYSTM**
.**WRR** <chan>
<error return>
<normal return>
Ac0 = word count to use in Ac1 for next .**MTDIO** call.

The normal calling sequence for VMD I/O is:

.**MTOPN**	; open the VMD
.**MTDIO**	; issue write command
.**WRR**	; wait for VMD's response, Ac1 <> 0
.**MTDIO**	; use count from .**WRR** to read the response from the VMD

The last three calls are repeated for additional data.

2.3.7 Enhanced QTY Support

The asynchronous data communications multiplexer (QTY) support in CALMA RDOS is enhanced to allow any QTY station to serve as the master input and output console for a ground. When the .SCIN and .SCOUT calls are used to select one of the QTY units as the master console, one of the UFT's for the ground is 'stolen' and kept for the exclusive use of the QTY. This is restricted in that .SCIN and .SCOUT cannot select different QTY units as the input and output consoles for the same ground. If a QTY is the master console for a ground, commands .GCHAR and .PCHAR are directed to that QTY line. The restriction against simultaneous read or writes to a QTY line is deleted. In addition, the output hold characters ↑S and ↑Q are redefined as they are for the teletype. Type ahead is allowed up to the rather small size of the QTY input buffer, with data being flushed if ↑A is used to abort the ground doing input from the QTY. If the QTY unit is the ground's master console, then .EXEC and .RTN do not affect the QTY buffer for that line. When the line is not a master console, then .EXEC has the effect of a .RESET, and it as well as .RTN flush any buffered characters for that line.

The task command .ABORT is used to abort QTY I/O in the same manner as other commands.

Lower to upper case conversion is the default mode for QTY input with the bit DCLTU masking this off upon .OPEN. If the QTY line is master console for a ground, the last .OPEN done to the QTY line determines the setting of DCLTU. If the QTY is not the master console, then the first .OPEN determines DCLTU for use of the QTY. .RESET and .RTN reset the masking of DCLTU if the QTY line is the ground's master console.

CALMA Swappable Grounds (see Paragraph 2.4) do not support any QTY I/O.

2.3.8 Initiate and Release of Devices

The .INIT and .RLSE commands are extended in CALMA RDOS to byte devices for obtaining exclusive use by the INITing ground. If .INIT is issued to a device that is not .OPEN or one that is .OPEN by the calling ground, the device is reserved for the exclusive use of that ground until that ground terminates or until it issues a .RLSE of the device. .INIT returns an error if any attempt is made to .INIT a device currently .OPEN by another ground even if not INITed by that ground, the error return is also taken if the device has already been INITed.

2.3.9 Test for Character Command

A new command (.TCHAR) allows single task programs to test for keyboard input without having to wait until a character is typed. .TCHAR is defined for the $TTI, $TTI1, $ICR, $KEYn, $DPI, and QTY:n devices. For devices $TTI, $TTI1, and QTY:n, echoing is done at interrupt level and .TCHAR specifies whether typed characters are echoed. This is all .TCHAR does on the actual input process. It does not buffer or process the actual input data.

AC0 = 0 = don't echo. <> 0 = echo.
.SYSTM
.TCHAR <chan>
<error return>
<normal return>

The error return is taken unless at least one character is waiting in the device buffer. If .TCHAR is followed by a .RDS of length one it is assured the call will not wait.

2.3.10 Smart Line Printer Spooler

The line printer device is supported in multiground RDOS. Before, the spool was a simple FIFO with the restriction that only one ground could input at a time. The new spooler, known as the 'Smart Spooler', allows each ground to have its own $LPT output at the same time. Printing does not begin until the file is closed for the last time. At that point, a disk file created by the $LPT open routine is queued up for output to the line printer. The queue is kept in a disk file of the master directory called 'SSPOOL.SL'. The temporary disk files are of the form 'LSPLnnnn.SP' where nnnn ranges from 0000 to 9999. Since all I/O is done directly to disk, these disk I/O system routines reduce the amount of time required to queue output to the $LPT. Also, since the spooler operates directly from the file to be printed there are fewer pauses in the operation of the line printer. The Smart Spooler keeps track of even and odd pages on the line printer and adds an extra form feed at the end of every job that contains an odd number of pages to make it come out even. Also, whenever the spool empties out after printing at least one file, the spooler puts out four extra form feeds to move the last page into position for tearoff. This makes it possible to operate the line printer without operator intervention except when paper runs out or the device malfunctions.

The following two commands are new for the Smart Spooler:

.SCLOSE <chan> Like **.CLOSE** except for **$LPT**. For **$LPT** this command tells the system that the **$LPT** output has more to come in later core images and not to queue it which would force all the output to be printed continuously and without separating heading pages.

NOTE

This call is patched into MAC, DGL, FORTRAN5 etc.

.SSCTL Argument in Ac0 contains command for spooler.

Commands are:

1	STOP printing at the end of the current line.
2	PAUSE at the end of the current page.
4	BACKSPACE and restart printing at the top of the current page. (This argument assumes operator intervention occurred and does not necessarily preserve forms parity.)
10₈	RESTART current job from the beginning. (Assumes operator intervenes and puts paper on even form feed parity.)
20₈	ABORT current job at end of next line. (Preserves parity by outputting a form feed if needed.)
40₈	CONTINUE current job if STOPed or PAUSEd or inactive. This command is used after a user program adds commands to **SSPOOL.SL** to reactivate the spooler.
100₈	TERMINATE the spooler after finishing the current job and suspend operation until CONTINUEd.
200₈	UNTERMINATE the spooler if it was TERMINATED previously.
400₈	BLOCKBEGIN delimits a group of data to be spooled into a single file for smart spooler output, regardless of **.OPEN**'s, **.CLOSE**'s, and program swaps.
1000₈	BLOCKEND causes the group of output, started by BLOCKBEGIN, to terminate and be enqueued to output. BLOCKEND also enqueues any output left by **.SCLOSE** with no following **.CLOSE** (or **.RESET**).

2.3.11 Current Directory Status Information

The **.RSTAT** command is enhanced with an option to obtain additional information about the file being looked up. This option is specified by setting bit 0 of the address passed in Ac1 and results in three extra words being returned. These extra words are appended to the end of the normal UFD from **.RSTAT**. The words and their contents are as follows:

UFDXU If disk or magnetic tape, the unit number of the device.

UFDXM The I/O **.SYSTM** calls-allowed mask, defined as follows:

1B0	.OPEN
1B1	.CLOSE
1B2	.RDS
1B3	.RDL
1B4	.RDR
1B5	.WRS
1B6	.WRL
1B7	.WRR
1B8	.APPEND
1B9	.ROPEN
1B10	.EOPEN
1B11	.TOPEN
1B12	.MTOPD
1B13	.TCHAR

UFDXC The device characteristic mask (0 if disk).

2.4 MULTIGROUND SUPPORT FOR CALMA RDOS

CALMA Multiground RDOS supports up to 16 grounds rather than foreground and background only. The additional grounds are all backgrounds in their USTPC value and in their return to the **.SYSTM** **.FGND** call. Impact on existing RDOS calls and conventions are kept to a minimum, but changes were required in several areas that could produce incorrect results. Testing of all standard RDOS programs is strongly suggested.

An additional feature is incorporated in the multiground system that allows certain extra grounds to be designated during system generation as swappable and sharing the same physical memory. This allows non-interactive, background computing that requires large core space to proceed without serious impact on the response time of the grounds in core.

Regular grounds, including the additional backgrounds, support all the documented facilities of RDOS, including user .IDEF's and interrupt servicing. The .SCIN/.SCOUT commands allow these grounds to specify their own console devices which also support the ↑A and ↑C console interrupts. The ↑F interrupt of the foreground is issued only from the $TTI device when it is selected as the master console for the background (that is the main background ID = 1). Also, if a spoolable device is selected as the output console by .SCOUT, then ↑A interrupts cause an .SPKILL to be performed to that device.

Swappable grounds are restricted in the area of interactive and real time .SYSTM calls. Keyboard reads, .GCHAR, QTY I/O, user .IDEF's, virtual memory, and virtual overlays are not allowed from a swappable ground because the ground is often on disk and cannot respond in time to service the requests. Swappable grounds may use .SCOUT in the normal manner and use .SCIN to specify a console that may interrupt, although no keyboard input is allowed. A call is provided to suspend swapping so that certain I/O, such as typing a message to the master console, is completed without interleaving messages from other swappable grounds. Scheduling of swappable grounds in core is done on a program priority basis in the same manner as nonswappable grounds (refer to the .PGND command). If there is more than one swappable ground with the lowest numerical priority, each ground runs for its current swap time (refer to the .TSWAP command). The next ground with equal priority then runs and the sequence continues in a round-robin fashion.

Backgrounds other than background 1 (the original background) may issue a .RTN or .ERTN from push level 1 to terminate the ground. This is similar to the foreground termination. When a ground terminates, all the directories INITed by it are automatically .RLSEed. The .FGND call does not distinguish between the several backgrounds returned in the accumulators. It only indicates if the single foreground is running and the number of active backgrounds is irrelevant.

The following paragraphs describe the new and modified .SYSTM calls that support CALMA Multiground RDOS.

2.4.1 Set Master Consoles for a Ground (.SCIN and .SCOUT)

Ac0 = Byte Pointer to input/output device name.
Ac1 = ID of Ground whose consoles are to be set,
    (or –1 for current ground)
    (see .EXBG for Ground ID definitions).

.SYSTM
.SCIN/.SCOUT
<error return>
<normal return>

These .SYSTM calls are used to change the master console of a ground. The caller's own console may be changed at any time and in addition the consoles of any ground that is currently inactive may be set. This prevents a ground from interfering with the consoles of another ground that is currently running. The device names may be any filename of six or fewer characters. Since some programs assume that there is a trailing null included, names of five or fewer characters should be used for complete compatibility with all .GCIN and .GCOUT callers. If the teletype devices, CALMA station devices, or one of the QTY lines is selected; .GCHAR and .PCHAR will work for the ground. Otherwise these calls give 'device not in system' error returns. This does not prevent other devices or even disk files from being used as arguments to .SCIN or .SCOUT if the application does not require that .GCHAR and .PCHAR work. Also, .SCIN determines the console that can issue ↑A and ↑C for the ground, provided that the console supports .GCHAR as described above. .SCOUT causes an automatic .SPKILL to be issued to the device after ↑A if it is spoolable.

2.4.2 Initialize a New Background (.EXBG)

Ac0 = Byte Pointer to filename.
Ac1 = (left byte: Priority less one.
      right byte; Ground ID).
Ac2 = Passed to called program.
.SYSTM
.EXBG
<error return>
<normal return>

The .EXBG command is used to start up a 'Multiground' background. It follows the .EXEC command except for the use of Ac1. The left byte of Ac1 contains the priority of the new ground less one (priorities range from 1 to 377; or priorities minus one from 0 to 376). The right byte contains the number of the new ground:

0 - Foreground
1 - Background (not allowed since already running)
2 - BG2 - second Background
3 - BG3 - third Background
    .
    .
    .
17 - BG15 - last Background

2.4.3 Abort a Ground (.AGND)

Ac0 = 0 = ↑A type abort that **.ODIS** disables,
     1 = ↑F type abort that **.ODIS** disables,
     −1 = ↑F type abort that **.ODIS** does not disable.
Ac1 = Ground ID (0-17).
.SYSTM
.AGND
\<error return\>
\<normal return\>

The **.AGND** command is used to send the equivalent of a ↑A to any active ground. It is issued by any ground and refers to any ground, including itself. Ac1 contains the ID of the ground to abort (from 0-17, see **.EXBG**). The **.ODIS** **.SYSTM** disables the action of **.AGND** on the selected ground if Ac0 is 0 or 1, just as it prevents keyboard ↑A and ↑C interrupts.

An ↑F type abort causes the ground to pop all the way to the top of its push stack, with USTIT and USTBR addresses being ignored during this operation. The aborted ground is terminated if there is at least one other ground running. If no other ground is running, the aborted ground is restarted with a fresh copy of the CLI in the master directory.

If **.AGND** aborts a swappable ground that is currently swapped out, the swappable ground currently in core is swapped out as soon as possible and the aborted ground swapped in for the fastest response to the abort. **.AGND** resets the swap time quantum for the ground being aborted to 10 seconds which allows the abort to proceed if it has been intercepted. The swap time must then be reset.

2.4.4 Set Priority of a Ground (.PGND)

Ac0 = Priority less one (0-376).
Ac1 = Ground ID (0-17).
.SYSTM
.PGND
\<error return\>
\<normal return\>

The **.PGND** command is used to alter the priority of a ground. Priorities range from a high of 1 to a low of 377. The scheduler unconditionally gives control on the basis of priorities, and also optimizes the cases of multiple grounds with the same priority. Ac1 contains the ID of the ground whose priority is to be altered and Ac0 contains the new priority less one.

2.4.5 Return Information About a Ground (.GROU)

Ac0 = address of 16 word area to receive information
Ac1 = Ground ID (0-17) or −1.
.SYSTM
.GROU
\<error return\>
\<normal return\>

To obtain various run parameters about a ground from the RDOS program table, the **.GROU** call is used. Ac1 contains the ID of the desired ground, or −1 is used to select and return the ID of the current ground in Ac1. The information returned is defined as follows:

        word 0 - RDOS Status
        word 1 - Priority
        word 2 - RDOS Flags
        word 3 - Push level
        word 4 - Number of 1K memory blocks assigned
        word 5 - Number of 1K memory blocks free
        word 6 - Swap flags
        word 7 - Swap time interval
        word 10 - Number of UFT's
        word 11 - CPU Time in clock ticks (low order)
        word 12 - CPU Time (high order)
        words 13-17 - \<reserved for future use\>

2.4.6 Set Swap Time Interval (.TSWAP)

Ac0 - Swap time interval (0-300 seconds).
Ac1 - Ground ID
.SYSTM
.TSWAP
\<error return\>
\<normal return\>

To set the swap time interval the **.TSWAP** call is used. The swap time interval is the amount of time the given ground runs in core during its run of the round-robin swap scheduling. If the time is set to zero, then the ground is not swapped in to run until the time interval is again reset to a non-zero value. **.TSWAP** does not change the ground currently in core until the next time it is swapped in. **.EXBG** initially sets the swap time interval to 10 seconds.

2.4.7 Disable/Enable Swapping (.SWDIS/ .SWEBL)

.SYSTM
.SWDIS/.SWEBL
<error return>
<normal return>

These calls are used to disable and reenable swapping of the calling ground. After the **.SWDIS** is executed the ground remains in core until the **.SWEBL** call is issued or a **.RESET, .RTN, .ERTN,** or †A interrupt occurs. Since the **.SWDIS** call locks all the other swappable grounds out of core, its use for a long period degrades background response.

2.4.8 Swap Current Ground Out (.SWBK)

.SYSTM
.SWBK
<error return (not used)>
<normal return>

The **.SWBK** command is used by swappable grounds to force swapping to take place immediately if there are any other swappable grounds waiting to run. It has no other affect and does not alter the basic time slice allocated to the swappable ground.

2.4.9 Get and Set Memory Partitions (.GMEM/ .SMEM)

Ac0 = Address of 16 word area.
Ac1 = -1 (used as flag for MG RDOS version).
.SYSTM
.GMEM/.SMEM
<error return>
<normal return>

Multiground versions of these calls follow the normal mapped RDOS conventions except for the number of parameters and the method of specifying them. The core partitions are saved/returned in core rather than accumulators and Ac1 is set to -1 to prevent old programs from being destroyed by not setting up the address in Ac0. The sum of the arguments for **.SMEM** must remain the same as it was from a previous **.GMEM**. If swappable grounds are generated then only the first swappable ground is considered as far as allocating memory for **.SMEM**. The same core partition value is returned for all the swappable grounds by the **.GMEM** command, which in turn gives a sum of all core partitions, including all swappable grounds, that are greater than the size of real user core on the machine.

2.4.10 Common Commands (.ICMN/.RDCM/ .WRCM)

Common commands support of RDOS is modified to allow any ground to read or write to the common area of any other ground. In order to accomplish this, the calling sequences for **.RDCM** and **.WRCM** were changed. Ac2, which contained the length to be read/written, now specifies both the length and the ID of the ground where the request is directed. This allows communication between all grounds currently in core. The new format for Ac2 is ground ID in the left byte and length in the right byte. To specify a length of 256 words, zero is used since only 8 bits are available for the length field. Requests directed to swappable grounds are made while the ground is in core. This is determined fairly accurately using the **.GROU** call. There is a possible race condition if the ground gets swapped out after the **.GROU** is issued but before a **.WRCM/.RDCM** is used. In this case, an error return for illegal address results and the common command is tried again the next time the swappable ground is in core.

The size of the communications region specified by **.ICMN** can be as large as the user core size of the ground.

Another feature is added to **.WRCM** which supports the start up of a task in the ground being written to. Whenever **.WRCM** is executed, the target ground is checked for any **.REC** task calls that are waiting by using a message address equal to the first word in the transmission by **.WRCM**. This may be any word in the ground's communications region since **.WRCM** uses the offset parameter to start the transmission at any word. If there is a **.REC** waiting, the effect is the same as if a **.XMT** was issued. The first word is returned in the **.REC** task's Ac1 and the word in core is zeroed.

2.4.11 Transmit to Common (.XCMN)

Ac0 = Message to send to other grounds (non-zero).
Ac1 = Offset into Communications Region.
Ac2 = (left byte) Ground ID of Destination.
.SYSTM
.XCMN
<error return>
<normal return>

This command causes an **.XMT** to any word in the other program's communications area. Like **.XMT** it gives an error return if the addressed word is already non-zero, and any task waiting for **.REC** on the addressed word is activated similarly to **.WRCM**. This command is used for process synchronization between grounds in the same manner that **.XMT** and **.REC** are used among tasks within a ground.

2.5 GENERAL ENHANCEMENTS OF CALMA RDOS

The following paragraphs describe changes made in the Data General RDOS program that provide general enhancements used for CALMA RDOS.

1. **.RENAME** accepts arguments that refer to different file directories if they both use the same **MAP.DR**. This means that they are both subdirectories of the same partition (primary or secondary) or that one is the subdirectory and the other is the containing partition.

2. **.REMAP** is extended to include another calling sequence to allow a noncontiguous sequence of blocks to be remapped in a single call.

   Ac1 = address of list of logical/physical blocks.
       (bit 0 set)
   Ac2 = number of words in list.
   .REMAP
   <error return>
   <normal return>

   Entries in the list of logical/physical blocks have the logical window offset in the left byte and the physical block number in the right byte. To prevent access to a logical block and validity protect a 1K block, the special value 377 is used as the physical block number. The new form is an illegally large number of blocks to remap in the old command format.

3. The system scheduler is enhanced to consider the priority of the task ready to run in a ground as well as the overall priority of the ground when scheduling the various grounds. The net dispatching priority is computed as follows:

   Dispatching Prio = PPRI*20+(IF IPRI>100 THEN
                     TPRI ELSE 200)

   WHERE: PPRI = ground priority, IPRI = task priority

   The system selects the ground with the lowest value of dispatching priority that is ready to be run. The reason for the test of the TPRI less than 101 is to prevent single task, and old multitask programs that use the lowest numeric values of TPRI, from interfering with the global priority scheduler. Mapping of the old low values onto 200 allows new multitask programs to set themselves above or below the normal as required.

4. The panic core dump routine is modified to allow dumping to magnetic tape. To use this option, mount a tape with a write ring on MT0 and set the CPU data switches to -1. Press CONTINUE twice. The format of the tape is 1024 word records with two EOF's at the end.

5. Traps are intercepted in mapped RDOS by setting up an interrupt task with **.INTAD** and specifying an exit address in USTBR. When this is done, traps cause the trapping TCB to be suspended by setting 1B0 in TPRST, status info to be saved in low core, and the **.INTAD** task readied to process the trap condition. The trap status is stored as follows:

   loc 2 = map status
   loc 3 = carry (1B0)+PC
   loc 4 = Ac0
   loc 5 = Ac1
   loc 6 = Ac2
   loc 7 = Ac3

   Also Ac1 of the **.INTAD** task is set up with the address of the TCB that caused the trap. If the trap was in the scheduler, Ac1 is zero and no TCB is suspended; also if USTBR is used for a ↑C or **.BREAK** exit, then Ac1 is set to -1.
   When the extended trap exit is taken, both USTBR and USTIT are set to -1 to decrease the chances of a trap loop.

6. Save files used as arguments to **.EXEC, .EXFG, .EXBG** may be contiguous as well as random in CALMA RDOS. The contiguous file results in a slightly faster program swap.

2.6 SYSTEM GENERATION AND BOOTSTRAP ENHANCEMENTS

The following paragraphs describe changes made in the Data General RDOS program that provide enhancements to the system generation (SYSGEN) and bootstrap programs used for CALMA RDOS.

1. CALMA RDOS is available in the Unmapped NOVA (URDOS), Mapped S/200 Eclipse (ARDOS), and Mapped S/230 Eclipse (ZRDOS) versions. NOVA RDOS is generated using 'SYSGEN', while Mapped Eclipse RDOS is generated using 'BSYSGEN' for non-multiground and 'MSYSGEN' for Multiground RDOS. The SYSGEN dialogs are altered to reflect the new CALMA I/O devices and other changes.

2. The **.BOOT** command is also modified. The filename argument is looked up starting in the master directory, thus to boot elsewhere the name must be fully qualified with directory specifiers. The contents of Ac2 to **.BOOT** are now passed to the booted program in location SCGDS in ROOT OF BOOT (this value was not previously used by RDOS systems). In addition, the current time and date are passed to the new program in ROOT of BOOT. If RDOS is being rebooted, these values are automatically set into the new system and the usual prompt for time and date is omitted.

3. When the system is booted up, a command file is set up for the CLI to execute before it accepts any typed in user commands. This command file must exist in the same directory as the RDOS system, and has the name **'system.IN'**, where system is the name of the current RDOS system. This allows each different system to provide its own custom initialization command file. Functions done at this time often include setting up of links to programs that only run under certain systems (such as all multitask **.SV** files) automatic clearing of files with the wrong use counts, **INIT**ing of directories, and setting of memory partition sizes.

4. Logon support is provided if it is selected during the SYSGEN dialog. This is done by **.EXEC**ing 'LOGON' whenever the system is booted from the front panel switches. The program then requires a user ID and password before CLI is allowed to run. At other times, or if Logon is not selected by SYSGEN, the program **'MINILOGON'** is chained to the user program. It implements the **'system.IN'** set up when LOGON is not used.

    To disable Logon at all times, **'LOGON.SV'** is linked to **'MINILOGON.SV'**. To enable Logon at all times, both **'LOGON.SV'** and **'MINILOGON.SV'** are linked to **'DOLOGON.SV'**. The normal case has **'LOGON.SV'** linked to **'DOLOGON.SV'** so Logon only occurs if front panel booting and if it was selected by SYSGEN.

5. Certain SYSGEN's provide a basic level of support for all CALMA hardware. These systems are as follows:

32KNOVA	System for 32K NOVA that includes one station and plotter
24KNOVA	Minimal NOVA system with no station
IXRNOVA	System for NOVA ICPU link with 32K and no station
ECLIPSE	Standard ECLIPSE system with several extra backgrounds and three stations. No QTY support or second teletype.
ZECLIPSE	Like Eclipse except for 230 map.
SHARED	ECLIPSE system for shared use that includes one station, **$LPT**, QTY, and second terminal as well as two magnetic tape controllers.
ZSHARED	Like SHARED except it is for a 230 ECLIPSE with two stations, and only one terminal.

SECTION III

CALMA CLI

3.1 GENERAL

This section contains information about the CALMA version of the RDOS Command Line Interpreter (CLI). Like RDOS, CLI was originally developed by Data General Corporation and the information in this section assumes that the programmer (user) is familiar with the original CLI. If additional information is necessary, refer to the Data General publications listed in the preface to this manual. The CALMA version of CLI has been enhanced to make it more useful with the CALMA version of RDOS. The information in this section is arranged with general information and instructions on using CLI given first. Each of the CLI commands are then provided in alphabetic order with a description of the command and how it is used.

3.2 PURPOSE OF CLI

The Command Line Interpreter (CLI) is the main interface between users at a system console and RDOS. Users communicate with the system by using CLI commands. When CLI commands are manually entered from a system console, (such as a Teletype, Decwriter, or CALMA station) the commands are translated, and in some cases executed, by CLI. Usually command execution is done by RDOS. The CALMA multiground RDOS services CLI command requests from several users at the same time. To receive service, the user is assigned a ground with enough core memory to run CLI. Depending upon the available system consoles and the amount of available memory, up to 16 users can be serviced by RDOS. When started, RDOS automatically activates a copy of CLI for each user. Each user then initiates a directory that provides the index to the assigned ground.

3.3 USER AND CLI INTERFACE

When using CLI, a user is independent of all other users. However, interference can occur since they all use the same disk. To help prevent interference while compiling, users each require a different directory because the compiler always uses the same set of scratch file names. Generally, any action taken by a user does not affect any action taken by all other users. Unless otherwise stated, the descriptions in this section apply to single users.

Specific error messages are output to the system console to indicate errors as they occur.

3.3.1 Ready Message Formats

To tell the user when CLI is ready to accept commands, a ready message is output to the currently specified system console. The ready message is output when CLI is activated or regains control of the user's ground. A user can select the ready message output in either of two formats; a short format or an extended format. The short format is the letter R and a carriage return. If the extended format is selected, CLI outputs the letter R, a carriage return, the current time, and a second carriage return. To change the ready message to either the short or extended format, the user waits until CLI is ready to accept a command and then types a period followed by a carriage return.

3.3.2 Command Input

Commands are input by the user after a ready message is received from the CLI. Characters typed on the user's input device are immediately echoed to the output device starting on the next line after the ready message. Usually a string of characters is required to create a command that will cause some action to occur. The exceptions are immediate action characters.

3.4 IMMEDIATE ACTION CHARACTERS

Immediate action characters cause action to occur when they are typed or in some cases when they are followed by a carriage return. These include characters used for input editing, abort, output control, and special purposes such as command termination. The following paragraphs describe how immediate action characters are used.

Some immediate action characters require that two keys are pressed at the same time. These are indicated by **CTRL n**, where **CTRL** is the control key marked **CTRL** and n is a specific character on the keyboard such as A, C, L, etc. For example, **CTRL A** is used as a command execution interrupt.

3.4.1 Input Editing Characters

Input editing characters are used to modify or correct data as it is typed at the user's console. This includes characters used to delete a previous character, delete an entire line, or continue a command string on a second line. These input editing characters can be used until a command terminator is typed. After typing a command terminator, editing

requires a text editor program (refer to **EDIT** command). The characters used for input editing are described below:

1. **BACKSPACE** is used to delete a previous character or several previous characters. How the **BACKSPACE** is echoed depends on the type of output device. For Hazeltine alphanumeric CRT's, the previous character is erased and the cursor moves one space to the left; on Tektronix storage CRT's, the deleted character is underlined; on Decwriters, the carriage is moved one space to the left so that an overstrike occurs when the next character is typed. On Teletypes, when **RUBOUT** is pressed, a left arrow is echoed to indicate deletion of the previous character.

2. Reverse slash or a **CTRL L** is used to delete an entire line. The reverse slash is echoed followed by a carriage return to indicate the line was deleted.

3. An up arrow (↑) followed by a carriage return is used at the end of a physical line in order to continue the same command string on the next line. On Teletypes the physical line is limited to 80 characters while Decwriter and other consoles have 132 characters per line. However, the logical command string can be extended beyond one line by typing an up arrow followed by a carriage return. A carriage return, that follows any character except an up arrow, is a command terminator.

3.4.2 Abort Characters

An abort character is typed to abort input collection or command execution. Either of two abort characters can be used depending upon the desired results. The two abort characters are **CTRL A** and **CTRL C**. If **CTRL A** is used while typing in a command line or during command execution, the CLI aborts input collection or execution respectively and then outputs a carriage return, prints out **INT** to indicate an interrupt occurred, prints out a ready message, and waits for the next command. Typing a **CTRL C** causes the same operations to occur except **BREAK** is printed instead of **INT** and the current memory contents are written to disk in a file named **BREAK.SV**.

3.4.3 Output Control Characters

Output control characters are used to control certain output features on a CRT. For alphanumeric CRT's the **CTRL S** and **CTRL Q** are used. **CTRL S** stops the data output so it can be read before it scrolls off the screen and it

also suspends the echo command. **CTRL Q** continues the data output and scrolling. The **CTRL Q** should also be tried if the console does not respond (echo command suspended) without apparent reason.

Whenever a storage display (CRT:n) is filled, it automatically pauses until an erase command is received. To erase the current display and continue outputting, type any character. The stored display and the current input command line are erased by typing **CTRL G**.

3.4.4 Special Characters

Special characters used for immediate action include the command terminators and keystrokes defined by the user program. The command terminators are carriage return (**RETURN**), form feed (**FF**), and semicolon (;). A carriage return indicates that a complete command was defined. It is echoed to the console output device as a physical return to column one and a line feed in preparation for the next output. Form feed also indicates that a complete command was defined but is not echoed as a carriage return and line feed. Semicolons are used to separate commands when two or more commands are strung in the same command line. A more complete description of command terminators is given in Pargraph 3.5.

User programs can specify special immediate action characters. For example, when the input file is a system console, the end of file could be specified as **CTRL Z**, which allows a return to the main program with a single key stroke.

3.5 COMMAND DEFINITION

Commands require specific formats, names, and terminators for interpretation or execution by CLI program. These parts of the command structure (syntax rules) are defined in the following paragraphs.

3.5.1 Command Line Formats

A complete command contains a legal command name and a command terminator. Most commands also require arguments, consisting of parameters and switches, that specify what the command will do. Parameters are separated by spaces or commas and switches, which apply to command names or parameters are indicated by a slash followed by a switch letter. If there is no command or parameter before a slash or the slash is the first character in a line without at signs (@), the line is ignored. This provides a way to include comments in a macro file. When two or more commands are strung in a command line, the commands are separated by a semicolon. To continue a

command line on a second physical line without terminating the command, an up arrow is typed followed by a carriage return.

3.5.2 Command Names

Command names are similar to file names in that only the first ten characters of the name are recognized by the program. These ten characters can be alphanumeric or the dollar sign (**$**). Examples of legal file names are **GTOD, TAPE$DISK,** and **$1000**. For commands that provide execution of user save files, a file name extension or directory prefix can be added to the command name. File names and directories are described in Section I.

3.5.3 Command Terminators

Commands are terminated by a carriage return (**RETURN**), form feed (**FF**), or semicolon (**;**). A carriage return signals CLI that a complete command line was defined and directs the program to interpret or execute the command line. It also echos a carriage return and line feed to the console output terminal so any subsequent output begins in column one of the next line. Since the line feed occurs automatically after a carriage return, the line feed character is ignored by CLI. A form feed performs the same functions as a carriage return except it does not echo a carriage return and line feed. Semicolons are used to separate individual commands in a command line so the commands are interpreted successively.

3.5.4 Command Processing

The CLI is a system program that accepts command lines, manually entered from the system console keyboard. Command lines are translated into commands that are recognized by the operating system (RDOS). A command line consists of one or more CLI commands followed by a command terminator such as a carriage return. Except for a number of simple commands that CLI executes directly, the command portion of the command line is interpreted by CLI using an associated file name. Therefore, every CLI command must have an associated file name before the command can be interpreted. When a command is typed, CLI builds a file called **COM.CM** that contains the edited command line and then loads a save file for execution. For example, the programmer (user) may type:

ASM $PTR (RETURN)

This command causes CLI to load a save file named **ASM.SV** for execution. The **$PTR** is the file name of the high-speed paper tape reader from which the file is to be

assembled. (In this example, the system requests the loading of **$PTR** twice since **ASM.SV** is a two-pass assembler.) Throughout this section, (**RETURN**) indicates the keyboard key marked **RETURN** or **CAR RET** is pressed.

Any file directory can contain a number of entries having the same file name but different extension (directories, file names, and extensions are described in Section II), for example:

A.SV	Relocatable binary save file
A.RB	Relocatable binary file
A.SR	Assembler or Basic source program
A.LS	Assembly or Compile source program
A.XX	User assigned extension

File names used as arguments for commands must specify the file name and the extension. Some commands will search for the specified file name and extension, and if unable to find the file name as specified, will search for the file name without an extension. For example:

ASM A (RETURN)

This command, when typed, causes a search for file name **A.SR** and if **A.SR** is found it is used as a source file for the assembly. Otherwise a search is made for **A**. If instead, the assembler source file **A** has extension **.XX** or no extension, a search is made for the specified file name and no other.

When the following command is typed:

RLDR A (RETURN)

A search is made first for an **A.RB** file name and if not found, then for file name **A**. If either file is found, the CLI creates an output file for the relocatable loader called **A.SV**, which is the relocatable binary save file. The commands **SAVE** and **MKSAVE** also have save files as outputs and CLI adds the extension **.SV** to the output file name. If another file name extension is substituted, the substitute extension is ignored and CLI gives the file a **.SV** extension.

The command **MKABS** has an input save file so when the user types:

MKABS A $PTP (RETURN)

a search is made first for an **A.SV** file name and if not found, then for file name **A**. If **A** is found, CLI creates a file named **A.SV**.

To execute a file named **A.SV**, the user types:

A (RETURN)

and CLI then calls the operating system to load the file named **A.SV** into memory and transfers control to its starting address. This is a special case of loading a save file and the only search is for the file name with the **.SV** extension.

Most other commands require appropriate file name extensions to be specified explicitly. For example, if the user types:

DELETE A (RETURN)

only the file **A** is deleted and file names with extensions would not be deleted.

If the user types the command:

RLDR A FOO/S (RETURN)

the /**S** option indicates the user wants the save file output of the loader to be named **FOO.SV**.

If the user specifies a file name and assigns his own extension, the file is always referenced by the full file name and extension.

3.6 GLOBAL AND LOCAL SWITCHES

Global and local switches are used to modify command names and their parameters. A switch is indicated by a slash followed by a single letter. Each letter has a distinct meaning depending upon the command or parameter that it follows. When a switch follows a command name, it is a global switch that applies to all the command parameters. A switch that follows a parameter is a local switch that applies only to the parameter. Any number of spaces, or no space, may separate a switch from its command name or parameter. Several global and local switches can be used in the same command. The order that multiple switches are specified is irrelevant and specifying duplicate switches does not cause errors. Examples of global and local switches are given in the following paragraphs.

3.6.1 Use of Global Switches

Global switches always use the same syntax, but their specific meaning depends upon the command (or program) being executed. A global switch is appended to a command name and is used to limit, modify, or extend operation of the command. Some command names do not have switches

and other commands have many switches. An example of a command with many switches is the **LIST** command. The **LIST** command normally lists all file names in the current directory, the resolution file name if the file name is a link, and the organization code if data is contained in the directory. The list is in random order. When the global switch /**B** is added to the command, a brief form that contains only filenames is listed. Global switch /**S** sorts the file names, and /**E** lists everything in the directory for each file name. Some variations for the **LIST** command are as follows:

LIST	The basic command.
LIST/B/S **LIST/S/B**	These variations are equivalent since the order of the switches does not matter.
LIST/S/E **LIST/B/S/E**	These variations are also equivalent since the /**B** and /**E** switches are contradictory and /**E** has higher priority than /**B** in this command.

3.6.2 Use of Local Switches

Local switches are used to limit, modify, or extend parameters and only affect the parameters where they are used. For most commands, the order of parameter values is irrelevant, but in some cases the parameter value does not indicate its intended use. To identify the intended use, a local switch is added to the parameter.

For example, the command:

ASM A B $LPT/L (RETURN)

causes files **A** and **B** to be assembled and a listing of the assembly is output to the line printer.

3.7 ARGUMENT FORMAT

The argument format determines how argument values are passed to a command. Providing the values are defined according to the rules for file names given in Section I, the file name format is general enough to pass small numbers as a string of characters. For example, **$1000** is a value and a legal file name.

Enclosing arguments in quotes does not change their meaning so **"ABC"** is equivalent to **ABC**. Arguments with illegal characters (other than alphanumeric or dollar sign) are deliberately enclosed in quotes to prevent erroneous

interpretation. Quotes allow **"AB@,10"** and **"67.375"** to be legal arguments. This feature allows encoding of special characters and passing messages to the user in a command macro.

3.8 ASTERISK AND DASH CONVENTIONS

The asterisk and dash conventions are used to select a group of files with similar names or extensions. These conventions are not part of the general CLI syntax and do not apply to user programs unless the programs are specifically written to include them.

3.8.1 Use of the Asterisk Convention

An asterisk is used to represent one character in a file name or extension. For example:

DELETE A**M	Deletes all non-permanent files in current directory that have a four-character name beginning with **A** and ending in **M** and no extension. This command deletes files named **ATOM ADAM, A22M,** and **AARM** but does not delete files named **ATOMIC, ITEM, ALAS, ADA,** or **ATOM.SR.** If the deleted names are links, the resolution file is deleted.
LIST/B/S B*	Lists, in alphabetic order, all two character file names or links that begin with **B** in the current directory.
LIST/B/S B*.SR	Similar to the previous example except only files with **.SR** extensions are listed.
LIST *	Lists all single character file names without extensions.

3.8.2 Use of the Dash Convention

A dash is used to represent any number of characters in a file name or extension. For example:

DELETE A-M	Deletes all non-permanent files in current directory that begin with **A** and end with **M** and have no extension.
LIST -.-	Equivalent to **LIST**.
LIST ****.-	Lists files with a four character name and any extension.
LIST ****-.-	Lists all files with four or more characters in the name and any extension.

CAUTION

Do not use the command **DELETE-.-** because it will delete all files on the disk. There is no way to recover the deleted files.

3.9 ADDITIONAL FEATURES

The previous paragraphs in this section provide enough information to correctly use any of the CLI commands provided later in this section. Additional features useful to the experienced user are described in the following paragraphs. These additional features include numeric switches; commas, angle brackets, and parentheses; indirect files; macro files; and registers.

3.9.1 Numeric Switches

Numeric switches are used as a shorthand notation to indicate how many times an argument value is used and where the first argument is the command name. A numeric switch is indicated by a slash followed by a number. Any number of spaces, or no space, may separate a switch from its argument. Switches **/0** and **/1** are the same as no switch, and switch **/6** is the same as switches **/2/0/4** when applied to the same argument. An application of a numeric switch is given in the following example.

> **LINK MAC.PS/2 (RETURN)**   is equivalent to,
> **LINK MAC.PS MAC.PS (RETURN)**

3.9.2 Comma, Parenthesis, and Angle Bracket Conventions

CLI commands can be repeated with each of several arguments or argument strings by first separating the arguments or strings with commas and then enclosing all the arguments with parenthesis. An application of enclosing arguments in parenthesis is shown in the following example:

DUMP MT0:0 FILE(A,B,C).SR
The commands as entered.

DUMP MT0:0 FILEA.SR
DUMP MT0:0 FILEB.SR
DUMP MT0:0 FILEC.SR
The above commands as interpreted by CLI.

Also, a series of command names can use a common argument by separating the command names with commas and enclosing them in parenthesis. An example of enclosing

command names in parenthesis is shown in the following example:

(PRINT, DELETE) ALONGFILENAME.LS
The commands as entered.

PRINT ALONGFILENAME.LS
DELETE ALONGFILENAME.LS
The above commands as interpreted by CLI.

Parentheses cannot be nested, but multiple sets of parenthesis are permitted in the same command. An application of multiple parentheses is shown in the following example:

MAC/N PART:(TEST,B,C,D).SR OVLY.SR MT0:(0,1,2,3)/L
The commands as entered.

MAC/N PART:TEST.SR OVLY.SR MT0:0/L
MAC/N PART:B.SR OVLY.SR MT0:1/L
MAC/N PART:C.SR OVLY.SR MT0:2/L
MAC/N PART:D.SR OVLY.SR MT0:3/L
The above commands as interpreted by CLI.

Angle brackets are used for in-line expansion of arguments within a single command. Arguments enclosed in angle brackets can be separated by commas or spaces. An application of angle brackets is shown in the following example:

DUMP MT0:0 FILE<A,B,C>.SR
The command as entered.

DUMP MT0:0 FILEA.SR FILEB.SR FILEC.SR
The above command as interpreted by CLI.

Angle brackets can be nested to any depth, but a single angle bracket cannot be enclosed with parenthesis. For example, (<>) and <( )> are legal but <(>) is not legal. Angle brackets are evaluated before parenthesis and both must be used in sets.

3.9.3   Indirect Files

Paired at signs (@) around a file name represent the contents of the file rather than the file itself. These are indirect files that allow frequently used data to be saved in the file. Indirect files are used in a command string and are expanded when the command is executed. The contents of the file must legally fit in the command string where it is used and the maximum nest depth for indirect files is three levels. A frequent use for indirect files is to specify a list of libraries for loading with relocated binary programs. An application of indirect files is provided in the following example:

RLDR DGLBINARY @ALIB@
Where **ALIB** contains a space and the following text.

A5<TASK,SYS,RDOS,IO,FMT,SUBS,ARITH,INIT>.LB

Indirect files can be created using the **BUILD** command.

3.9.4   Macro Files

Macro files allow frequently used sets of commands to be defined once and then referenced as desired. In its simplest form, a macro file is equivalent to an indirect file that contains one or more CLI commands. The CALMA version of CLI has enhanced macro file capability that includes the loading and testing of five registers as part of a macro call (refer to Paragraph 3.9.5). Whenever a macro file is executed, a **COM.CM** file is created as if a save file was being executed. The macro files work as follows:

1. The programmer (user) creates a macro file using any of the editors described in Appendix A. A macro file can contain any legal CLI command string.

2. The macro filename is given a **.MC** (macro) extension.

3. For execution, the macro file name is used like a command. On input, if no extension is given, the system searches for the macro filename as a CLI command, then as a macro file, and finally as a save file. If none of these are found an error occurs. To avoid errors during execution, do not use macro file-names that duplicate CLI command or save file names. If a CLI command or macro file was used, always use a **.SV** extension to specify the save file.

4. For example, the following command macro string could be used to prevent multiple lists from being generated during editing.

    BIGFILE.MC   macro filename
    ERROROFF; DELETE #0.LS; ERRORON   command string
    ASM/A/L/U/X #0   macro execution

    When **BIGFILE** is executed, the macro command string deletes the listing for the contents of register #0 while error messages are suppressed. A new listing is then created to replace the one that was deleted.

5. The CLI command macro does not use the macro assembler.

6. See the **MESSAGE** command for outputting text in a macro.

7. The slash sign "/" at the beginning of a macro file line denotes a comment statement.

8. The following example of a macro file is named **F5.MC**.

NOTE
Lines are numbered for description only; the numbers are not used in macros.

```
(1)    / THIS IS DESIGNED TO COMPILE A LIST OF FORTRAN FILES
(2)    /
(3)    / MESSAGE/N "FILES?";–0
(4)    / DELETE ANY GARBAGE
(5)    ERROROFF
(6)    DELETE <#0>.<LS, SV, RB>
(7)    ERRORON
(8)    / NOW COMPILE THEM
(9)    / MESSAGE "COMPILING" %TIME%
(10)   FORTRAN/L/B (#0)
(11)   MESSAGE "DONE AT" %TIME%
(12)   LIST/C <#0>.RB
(13)   PRINT <#0>.LS
(14)   / END
```

Lines 1, 2, 4, 8, and 14 are comment statements that describe the operation but are not printed by the macro and lines 3, 9, and 11 are messages to be printing by the macro. On line 3 the message is not followed by a carriage return (/**N**) and the left arrow (←) indicates the next character is a register number. The left arrow is a Teletype character which is replaced by an underscore character (—) when a keyboard or Decwriter are used. For more information on registers, refer to Paragraph 3.9.5.

The CLI commands used in the macro are shown in bold characters and are described in separate listings following Paragraph 3.11. Angle brackets (< >) are described in Paragraph 3.10.2.

On lines 9 and 11 the message is followed by the term %TIME%. This is a system variable that allows the actual time to be printed instead of the term. Other system variables include %DATE% to print the actual date and %MDIR%, %GDIR%, and %LDIR%. These terms print out the actual names of the master directory, current directory, or the last directory used respectively.

3.9.5 Registers

The CALMA version of CLI makes five registers available to the user. Each register holds up to 300 characters and is identified by a number from #0 through #4. Registers can be used interactively but are best used in macros. A register is set as part of a macro call, set by a constant in the macro, or set by the operator at run time. Contents of a register can be inserted in a command line at any point and can be tested to provide conditional command sequences for execution under macro control.

To load a register as part of a macro call, the register parameters are specified after the macro file name. The first parameter is loaded into register #0, the second into register #1, the third into register #2, etc. If less than five parameters are specified, the unused registers are initialized to null strings (all zeros). A parameter value is specified as a parameter (**ABC**) or a character string (**"A B C"**). Also, a value can be set from a register value. For example, macro #1 moves the current contents of register 1 into register 0.

Register loading is conditioned on the normal CLI command lookup. If the macro is executed as an indirect file (**@MACRO.MC@**), the current register contents are not changed. If any registers are loaded as part of a macro call, global and local switches are ignored.

A register is loaded from a constant string (a quantity in single quotes) using the form:

←n'constant string', where ←indicates register loading and n is the register number.

A user can load a register at run time by using the form:

←n, where ←indicates register loading and n is the register number.

When this form is used in a macro, CLI requests one line of input from the console input device. The user's response is terminated by a carriage return which is removed when the register is loaded.

NOTE
When using a keyboard or Decwriter in place of a Teletype, the left arrow (←) is replaced by an underscore (—).

Contents of a register are inserted into a command line using the form:

#n where n is the register number.

To test the contents of a register, braces are used in the syntax. The syntax is a left brace, number of register to be tested, relational operator, reference value enclosed in

single quotes, text to be inserted into the command line (if relationship is true), and a right brace. A relational operator is either equal or not equal as denoted by an equal sign or left and right braces respectively. Text can be inserted as a result of a conditional expansion which may include another conditional expression to test a different register.

For example, to allow an optional listing file to a **MAC** assembly, the first macro parameter could be macro **MA.MC**:

<p align="center">MAC/U {O{}' '#0/L} MACFILE</p>

MA   generates   **MAC/U MACFILE**
MA LIST   generates   **MAC/U LIST/L MACFILE**

To assemble a macro file, either on a command line, or prompted for (if no argument appears at **MASM.MC**), the following form could be used:

<p align="center">{O=' 'MESSAGE/N "ENTER FILENAME:";−O};ASM #0</p>

MASM MYU results in **ASM MYU**
MASM prompts for a file with:
    ENTER FILENAME: xxxxxx
The user types **xxxxxx** which results in **ASM xxxxxx**.

3.10 CLI ERROR MESSAGES

During operation and execution of CLI, error messages may print out to indicate an incorrect entry or sequence was attempted. Following an error message print out, the system usually returns to the CLI prompt (prints a letter R) to wait for further instructions from the user. A list of CLI error messages and their meaning is provided in Table 3-1.

Table 3-1. CLI Error Messages

ERROR	MEANING
ADDRESS ERROR	Attempt to reference an address outside user address space (mapped systems only).
ATTEMPT TO READ INTO SYSTEM SPACE	Attempted access of unmapped system area.
ATTEMPT TO WRITE AN EXISTING FILE	Attempt to write an existing file.
ATTEMPT TO RELEASE AN OPEN DEVICE	Attempt made to release an open device.
BAD DISK ADDRESS	The RDOS file structure contains bad data (perhaps due to a **MAP.DR** error) and this file cannot be read. Reinitialize the disk as soon as possible after this error occurs.
BRACKET ERROR	Unmatched brackets.
CHECKSUM ERROR	Checksum error detected during input.
CHANNEL ALREADY IN USE	Attempt to open a channel which is already in use.
COMMON SIZE ERROR	The communications area specified for interprogram communications is too small.
COMMON USAGE ERROR	No communications area is defined in the addressed ground.
CHANNEL CLOSED BY ANOTHER TASK	Two tasks share a common I/O channel. One task closes the channel before the other task was able to complete its I/O.
COMMAND LINE TOO LONG	Command line exceeds limit.
CONSOLE INTERRUPT RECEIVED	Control A or Control C was given to a program that intercepted the interrupt, and then terminated.
DIRECT I/O ACCESS ONLY	Attempt to perform sequential I/O on a file with the I attribute.
DEVICE NOT IN SYSTEM	Attempt to reference an uninitialized directory or device.
DEVICE ALREADY IN SYSTEM	Illegal (and unnecessary) attempt to re-initialize a directory device. Alternatively, an attempt to identify a system device as a user device.
DIRECTORY SIZE INSUFFICIENT	An attempted CPART specified too few disk blocks.

Table 3-1. CLI Error Messages (Continued)

ERROR	MEANING
DIRECTORY DEPTH EXCEEDED	Attempt to create a tertiary partition or a secondary subdirectory.
DIRECTORY IN USE	Attempt to assign a logical name to a directory or device which has already been initialized. Alternatively, an attempt ro release a directory which has files in it that have not been closed.
DIRECTORY NOT INITIALIZED	Attempt to reference an uninitialized directory or device.
DIRECTORY SHARED	A released directory is in use by another ground (this is merely a warning).
DEVICE TIMEOUT	10 second disk timeout occurred in other than the master device.
DEVICE PREVIOUSLY OPENED	Attempt to open a previously opened device.
DISK NOT RDOS FORMAT	Attempt to .**INIT** a non-RDOS disk.
END OF FILE	End of file detected.
ERROR IN USER TASK QUEUE TABLE	Bad table input to .**QTSK**.
FILE READ PROTECTED	Attempt to read a read-protected file.
FILE WRITE PROTECTED	Attempt to modify a write-protected file.
FILE ALREADY EXISTS	Attempt to create a file with a name which is already in use.
FILE DOES NOT EXIST	Attempt to reference a non-existent file.
FILE ATTRIBUTE PROTECTED	Attempt to change the attributes of an attribute-protected file.
FILE NOT OPEN	Attempt to access an unopened file.
FILE SPACE EXHAUSTED	Out of user disk space of mag tape EOT reached before end of write.
FATAL SYSTEM UTILITY ERROR	An unrecoverable error was detected within a utility.
FILE DATA ERROR	File read error.
FILES MUST SHARE SAME **MAP.DR**	Improper use of **RENAME** command.

3-9

Table 3-1. CLI Error Messages (Continued)

ERROR	MEANING
FILE IN USE	Attempt to modify a file which in in use.
FILE POSITION ERROR	Attempt to illegally position a file.
GROUND IS SWAPPABLE	Certain operations are not allowed in swappable grounds.
ILLEGAL ARGUMENT	Illegal character in an argument.
ILLEGAL NUMERIC ARGUMENT	Non-numeric character in a numeric argument.
ILLEGAL ATTRIBUTE	Undefined attribute specified.
ILLEGAL BLOCK TYPE	Attempted **LOAD** of a file which is not in **DUMP** format.
ILLEGAL CHANNEL NUMBER	Channel number exceeding 77 octal was input to a system command.
ILLEGAL FILE NAME	Illegal character in file name string.
ILLEGAL SYSTEM COMMAND	Attempt to reference an uninitialized directory or device or to release an uninitialized device.
ILLEGAL COMMAND FOR DEVICE	Attempt to perform illegal I/O (e.g., direct I/O to paper tape).
INSUFFICIENT MEMORY TO EXECUTE PROGRAM	Attempt to allocate more memory than is available.
ILLEGAL OVERLAY	The specified overlay does not exist.
INVALID TIME CHANGE	Attempt to set illegal time or date.
INSUFFICIENT CONTIGUOUS BLOCKS	Insufficient number of free contiguous disk blocks to create the desired file.
ILLEGAL DIRECTORY NAME	Illegal character in link resolution name string was given in a **LINK** command.
ILLEGAL OR NON-EXISTANT GROUND	An argument to **.EXBG** is not **SYSGEN**'ed into the system or is already running; also if **.AGND** refers to a ground that is not active.

Table 3-1. CLI Error Messages (Continued)

ERROR	MEANING
INSUFFICIENT ROOM IN CHANNEL MAP	Improper data channel size specified .**IDEF** system command.
ILLEGAL PARTITION VALUE	One of the following conditions was detected in an **SMEM** command. The background and foreground allocation arguments do not equal the total user address space which is available; the new memory settings would not leave enough room for the background CLI (five 1024 word blocks are needed); the **SMEM** command was not issued from the background CLI while no foreground program was running.
LINK NOT ALLOWED	Attempt to link a file which has the "no link resolution" attribute set.
LINK DEPTH EXCEEDED	Attempt to reference a resolution file via more than 10 levels of links.
LINE TOO LONG	Line limit exceeded on read or write line I/O.
MAP.DR ERROR	File system **MAP.DR** inconsistency detected in a directory device.
NO MORE DCBS	Attempt to initialize too many devices and/or directories. **SYSGEN** a new system and specify a greater number of subdirectories/subpartition to be accessible at one time. Alternatively, release one or more currently initialized devices or directories.
NO DIRECT I/O	Direct I/O allowed only on magnetic or cassette tape files.
NO DEBUG ADDRESS	The debugger was not loaded with this save file and the **DEB** command was issued.
NO STARTING ADDRESS	Attempt to execute a non-save file.
NO ROOM FOR UFTS	Insufficient number of channels specified at **SYSGEN** time.
NOT A LINK ENTRY	Attempt to delete an entry which lacks the link characteristic. Attempting to **DELETE** a link causes the resolution file to be deleted.
NOT A SAVED FILE	File requires the save attributes and the random or contiguous characteristic.

Table 3-1. CLI Error Messages (Continued)

ERROR	MEANING
NO SUCH DIRECTORY	Directory specifier unknown.
NO FILES MATCH SPECIFIER	No match found for any argument containing the * or — conventions.
NO SOURCE FILE SPECIFIED	CLI command requires a source file.
NOT A COMMAND	Fatal CLI error due to modification of **CLI.SV** or **CLI.OL**.
NOT ENOUGH ARGUMENTS	More arguments are required by this command.
OTHER GROUNDS ACTIVE	**.BOOT** or **.RLSE** of the master device was attempted while more than one ground was active.
OUT OF TCB'S	Task Control Blocks all used.
PHASE ERROR	Attempt to reset location counter back over current value (only during **MKSAVE**: setting the counter ahead is permitted).
PERMANENT FILE	Attempt to delete or rename a permanent file; attempt to **LOAD** or **DUMP** a permanent file without using the /**A** global option.
PARITY ERROR	Parity error on read line or read sequential of mag tape.
PUSH DEPTH EXCEEDED	Attempt to push too many levels (limit of 4).
SPOOL FILES ACTIVE	**.BOOT** or **.RLSE** of master device attempted while spooling was in progress.
SYSTEM DEADLOCK	System run out of resources i.e., buffers.
SIGNAL TO BUSY ADDRESS	Multiple transmissions to same address with no receive.

Table 3-1. CLI Error Messages (Continued)

ERROR	MEANING
STACK OVERFLOW	Fatal CLI runtime error, generally caused by running in a ground that does not have enough memory allocated.
SYSTEM STACK OVERFLOW	System stack capacity exceeded.
SQUASH FILE ERROR	Attempt to **SQUASH** a file that has already been squashed.
SYS.DR ERROR	File system **SYS.DR** inconsistency detected.
TASK ID ERROR	Task ID violates syntax requirement.
TASK NOT FOUND FOR ABORT	Attempt to abort a task that does not exist.
TOO MANY ACTIVE DEVICES	Too many devices active at the same time.
TOO MANY ARGUMENTS	Too many arguments input to a command where the number or position of arguments is significant.
UNIT IMPROPERLY SELECTED	Magentic tape controller not turned on, tape unit ON LINE. Disk unit not ready.
UNMATCHED@	Command line has an odd number of @ signs.
YOU CAN'T DO THAT	Attempt to end console logging without using password. Attempt to run an ECLIPSE program on a NOVA. Alternatively, this message is given to the CLI (e.g., non-upper case ASCII characters). Also, any attempt to full **.INIT** a disk primary partition (see **EQUIV**).

3.11 CLI COMMANDS

The CLI commands provided in the following paragraphs are in alphabetic order by command names. Command names are in all capital letters and are followed by the arguments required to complete the command format. Where special characters (such as right slash, at sign, space, comma, period, asterisk, dash, angle bracket, or parenthesis) are shown, they are part of the normal command format and should not be ignored. Square brackets are not part of the command format and are used to show alternate or additional arguments. If the square brackets enclose alternate arguments, the arguments are stacked and the user selects one of the stacked arguments. Additional arguments are on the same line and enclosed in brackets to show the command format is complete without the additional arguments. Two or more of the same additional arguments are indicated by etc., which follows the repeatable argument in square brackets. Arguments that are not enclosed in brackets must be used to complete the command line. Words used in the argument format describe the type of data to be included in the command line. These words are all lower case and are usually two or three short words run together to save space.

An application of the command format is shown in the following example:

APPEND newfilename oldfilename [oldfilename etc.]

In this example, **APPEND** is the command, **newfilename** is the name of the new file created by appending an existing (old) file, and **oldfilename** is the name of the old file to be appended. Square brackets enclose a second and subsequent **oldfilename** indicating that two or more existing files can be appended to create a new file with this command. The square brackets are not used in the command when second and subsequent **oldfilenames** are used.

Following the command name and format, separate paragraphs are used as applicable to describe the purpose of the command, arguments not apparent in the command format, attributes, switches, examples, and restrictions. These paragraphs contain the information necessary to understand and properly use each command.

NOTE
Command terminators are not shown in the separate command descriptions. When two or more commands are entered on the same line they are separated by semicolons. The command line, whether one or more complete commands, is usually terminated by a carriage return (RETURN). For more information on command terminators, refer to Paragraph 3.5.3. In the examples of command use shown in the following command descriptions, cr indicates that the keyboard RETURN key is depressed.

ABGROUND [/switches] ground id

Purpose
Aborts a background job stream.
Under certain abnormal situations, a user may lose control of a background job stream and require an abort to regain control. For example, if a ground is directed to execute commands from a disk file which is improperly terminated or if a user program brings in a background for which no console input is defined, an abort from another ground could be necessary.

The **ground id** is the identifier of an RDOS ground:
FG, BG, BG2, BG3,..., BG15. Most RDOS systems are not generated with sixteen backgrounds and only the generated grounds are used as parameters to **ABGROUND**.

Global Switches
/A Perform Control A Abort. (Normally aborts the current program and returns to CLI if it is running in the identified ground.)
/F Perform Control F Abort only if enabled. (Complete ground abort except, if the program is currently running with console interrupts disabled by a .ODIS system command, no abort occurs.
With no global switch given, a complete ground abort occurs and the .ODIS system command is ignored so the interrupt cannot be prevented. This is the absolute method of terminating a ground.

Example
ABGROUND BG4 cr
This command aborts the job running on background number 4, providing background four was generated and is not the caller's ground.

Restrictions
If **ABGROUND** is accidentally used on the caller's ground it has the effect of killing the ground just like any other ground; also, if it is the last running RDOS ground, then CLI is restarted instead of the ground being aborted.

ALGOL inputfilename [outputfilename etc.]

Purpose
Compiles an ALGOL source file (**inputfilename**) with the ALGOL 4 compiler. By default, execution of the command produces an intermediate source file (**inputfilename.SR**) for the compiler output and a relocatable binary file (**inputfilename.RB**) for the assembler output. When assembly is complete, the intermediate source file is deleted. A listing file is also produced if a global option /**L** is selected.

On input, the system searches for **inputfilename.AL**, and if not found, it searches for **inputfilename**. Output is **inputfilename.RB** by default and other files with .**LS** or .**SR** extensions as determined by options.

Global Switches
/**A**	Suppress assembly.
/**B**	Brief listing; compiler source program input only.
/**E**	Suppress compiler error messages at console. Assembler error messages are not suppressed.
/**L**	Produce listing to **inputfilename.LS**.
/**N**	Do not produce relocatable binary file.
/**S**	Save intermediate source file.
/**U**	Append user symbols to binary output.

Local Switches
/**B**	Relocatable binary output file (overrides global /**N**).
/**E**	Error listing to given file name.
/**L**	Listing output to given file name (overrides global /**L**).
/S	Intermediate source output to given file name.

Examples
ALGOL MAIN $LPT/L cr
Compile **MAIN.RB** and produce line printer listing.
ALGOL/A SUBR MT0:2/S cr
Suppress assembly and output intermediate source file to magnetic tape.

Restrictions
For ECLIPSE computers, the DGL compiler (see **DGL** command) is recommended because it generates better object code.
Command name **ALGOL** cannot be changed.
The asterisk and dash convention is not permitted.

APPEND newfilename oldfilename [oldfilename etc.]

Purpose
Creates a new file (**newfilename**) by concatenating one or more old files in the order in which their names appear as arguments. The old files are unchanged.

Global Switches
/**C**	Create a contiguous new file if the old file is a disk file.
/**R**	Create a random new file if the old file is a disk file.

Examples
APPEND DC1:A2.SR A.SR B.SR DC0:C.SR cr
Produce **A2.SR** on disk drive unit 1 with the contents of files **A.SR** and **B.SR** from the default directory and **C.SR** from disk drive unit 0.

Restrictions
The asterisk and dash convention is not permitted.

ASM filename [filename etc.]

Purpose

Assembles one or more source files using the extended relocatable assembler. By default execution of the command produces a relocatable binary file (**filename.RB**). A listing file is also produced if a global switch /L is selected.

On input the system searches for **filename.SR**, and if not found, it searches for input filename unless the filename was given with an extension. Output is **filename.RB** by default and files with **.LS** extension as determined by option, where filename is the first source file specified without local switches /**S**, /**L**, or /**B**.

Global Switches

/**A**	Include **.DUSR's** in **XREF** (GDSASM only).
/**E**	Suppress error messages.
/**L**	Produce listing to **filename.LS**.
/**N**	Do not produce relocatable binary file.
/**P**	Perform **.BREAK** and save a core image with symbols that can be executed to create new versions of **ASM**.
/**S**	Skip pass 2. A **BREAK** is signalled after pass 1 to allow saving a version of the assembler that contains the semipermanent symbols.
/**T**	Do not produce symbol table in listing.
/**U**	User symbols appended to relocatable binary.
/**X**	Produce symbol table cross reference.

Local Switches

/**B**	Relocatable binary. (Overrides global /**N**).
/**E**	Error message file.
/**L**	Listing file (overrides global /**L**).
/**N**	Do not list this file.
/**S**	Skip this file on pass 2.

Examples

ASM Z $LPT/L cr
  Produce **Z.RB** and a listing on the line printer.
ASM/U/L/X/A QQXV $LPT/L cr
  Assemble program **QQXV** including user symbols for subsequent debugging and print a listing with a cross reference which includes permanent user symbols.

Restrictions

Command name **ASM** cannot be changed.
The asterisk and dash convention is not permitted.

NOTE
To run **GDSASM**:
LINK ASM.SV GDSASM.SV
LINK XREF.SV GDSXREF.SV

ASMREF [filename etc.]

Purpose

Produces a cross reference listing for multiple assembly language programs.

Global Switches

/**D**	Minimize disk space during the sort phase.
/**N**	Assume standard NOVA instruction set.
/**P**	Include pseudo-ops in the cross reference.

Local Switches

/**L**	Specifies list file (**$LPT** is the default).
/**X**	Specifies a scratch directory for the sort phase.

Example

BUILD X -.SR; ASMREF @X@ cr
  Produces a cross reference history for all assembly language programs in the current directory with the output directed to the line printer. Since the global /**N** option was omitted, ECLIPSE instructions, such as **ELDA**, are not included in the output.

Restrictions

1. The utility program **ASORT** must be available (perhaps as a link) in the current directory.
2. The following names are used for scratch files by this program: **SORT.CM, FSORT.CM**. Some scratch files having the **$$** extension are also used.
3. **ASMREF** runs on ECLIPSE computers only.

BACKUP [filename etc.]

Purpose
Creates a complete image of the disk. The disk is copied to magnetic tape sector by sector without any interpretation or validation of the contents of each sector. This means that partial reload or reformatting options do not apply when reloading a **BACKUP** tape. **BACKUP** supports multiple reels of tape if the amount of data exceeds what a single reel holds.

For complete procedures to create and reload a Backup Tape, refer to Section 4, Paragraph 4.4.

Global Switches
/N   Do not rewind the tape (**RELEASE MT0**) after writing the tape or after booting the tape on the subsequent reload.

Local Switches
/B   Dump a Background CGI Overlay File. (Used with option **/C**).

/C   Dump a CGI Overlay File. (See **/B** and **/S**.)

/G   Dump a GDS I Area File.

/N   Do not dump a contiguous file.

/P   Dump a contiguous file in part, with number of sectors supplied as the next argument in the command line. The number of sectors is given in octal.

/S   Dump a Small CGI Overlay File. (Used with option **/C**.)

Examples
BACKUP cr
    Creates a disk image on magnetic tape mounted on unit 0
BACKUP GDS/G cr
    Backup the disk on magnetic tape, including a GDS Area File in the current directory.

Restrictions
Two versions of **BACKUP** exist: one to produce tapes from Century disks and another to produce tapes from Calma Storage Modules (Big Disks). These tapes are not interchangeable. Also, Calma Storage Modules are available with 80MB and 300MB disks and Backup Tapes made from them are not interchangeable either. Separate versions of **BACKUP** also exist for NOVA and ECLIPSE computers. The existing versions of **BACKUP** are:

NBACKUP.SV	For NOVA with Century disks
ABACKUP.SV	For ECLIPSE with Century disks
NBBACKUP.SV	For NOVA with Storage Modules
ABBACKUP.SV	For ECLIPSE with Storage Modules

Normally, the **BACKUP.SV** file is linked automatically to the correct version in the master directory. However, if a system has both types of disks the appropriate version must be selected.

BASIC

Purpose
Invokes the **BASIC** programming language interpreter to execute a **BASIC** program. Configuration and load procedures for **BASIC** language under RDOS are described in the Data General **BASIC** user's manual.

Example
BASIC cr
    Once the **BASIC** interpreter is invoked, it initiates a
    short dialogue with the user as to program
    requirements before starting up the **BASIC** system. The user
    can exit **BASIC** and return to the CLI by using the
    KILL command.

Restrictions
A **BASIC** save file, **BASIC.SV**, must have been configured via **BSG** when the **BASIC** command is executed.
BASIC requires that directory **BASIC.DR** be **INIT**ed or a subdirectory/subpartition of the directory in which **BASIC** is executed.
There are no switches, and the asterisk and dash convention is not permitted.

BATCH [jobfile etc.] [outputfile/O] [logfile/G]

Purpose
Initiates execution of one or more job files to create a **BATCH** job stream. Each jobfile is an input file with one or more user jobs and is complete with job control commands and optional data sets. The card reader is the default input file, the line printer is the default output file, and the Teletype printer is the default log file. Also by default, **BATCH** searches for job files with the **.JB** extension. For further information, refer to the Data General **BATCH** user's manual.

Local Switches
/G   Produce log file.
/O   Produce output file.

Examples
BATCH cr
> Jobs input from card reader are executed and output is printed on the line printer. The log file is the **$TTO**.

BATCH A.JB B.JB MT0:1 LOG/G cr
> Jobs **A.JB** and **B.JB** and the job on file 1 of **MT0** are executed and the output is printed on the line printer. The log file is a disk file named **LOG**.

Restrictions
The asterisk and dash convention is not permitted.

BOOT newsystem

Purpose
Releases the current system and performs a disk bootstrap. The arguments in **newsystem** may be the name of a system save file in the current partition, the name of the primary partition from where the system will be bootstrapped, or one of the following formats as applicable:
1. primary partition
2. [primary partition:] secondary partition
3. [primary partition:] [secondary partition:] save filename

There are no spaces between colons and arguments and all partitions must be initialized before they are valid arguments. If directory specifiers are omitted, **newsystem** is looked up in the master directory and not in the current (default) directory.

Examples
BOOT DC0 cr
> Since, in this case, HIPBOOT resides in **DC0**, when **BOOT** is executed the program queries for **FILENAME?**. Respond by typing the system save filename or press the RETURN key to default to **SYS.SV/SYS.OL**. When only the primary partition is given as the command argument, the response to **FILENAME** must be a disk file name.

BOOT SYS cr
> Reboots the default system on the current master device.

Restrictions
Execute the **BOOT** command only when other RDOS grounds are not active since it brings down the entire RDOS system.

Where RDOS is dying and a reboot with minimum data loss is desired, a special technique is required to override the interlocks that prevent a **BOOT** when other activity is going on. This technique must be used with extreme care after ensuring that other system activities will not be disturbed. To perform this emergency boot, proceed as follows:

1. Push the CLI to an extra level using DIR %MDIR%; CLI. The system responds with the CLI prompt (R).

2. Give the **BOOT** command again. The bootstrap is then executed and other running grounds are aborted. However, some files may be left open and the **CLEAR/A/D** command might be required in some directories afterwards.

BPUNCH filename [filename etc.]

Purpose
Punches a given file or files in binary on the high speed punch. Compare with **PUNCH** command.

Example
BPUNCH A B DC1:MFILE cr
  Punch files **A** and **B** from the current directory and **MFILE** from **DC1** on the **$PTP**.

BSYSGEN

Refer to **SYSGEN** command for purpose and use information.

BUILD outputfilename [filename etc.]

Purpose
Builds file **outputfilename**, containing names of all current directory files specified. The asterisk and dash convention may be used. If an extension is not given, the filename or names included are those without extensions.

Global Switches
/**A**   Include all permanent and nonpermanent filenames.
/**K**   Do not include links.
/**N**   Do not include extensions to filenames.

Local Switches
/**A**   Include only files created this date or after, where the parameters have the form mm-dd-yy and mm and dd are one or two digits.
/**B**   Include only files created before this date. The argument has the form given for local /**A**.
/**N**   Do not include files matching this name.

Examples
BUILD OUT-.- cr
  Produce file **OUT** containing names of all nonpermanent files in current directory.
BUILD ABFILE A-.- B-.- cr
  Produce file **ABFILE** containing names of all files beginning with the letter **A** or the letter **B** and having an extension.
BUILD LISTNM -.LS cr
  Produce file **LISTNM** containing names of all files in the current directory with the extension .**LS**.

CCONT filename blkct [filename blkct etc.]

Purpose
Creates one or more contiguously organized files in a directory. The argument specifies filename(s) with no attributes and a length given in decimal disk blocks (**blkct**). Disk storage is allocated in decimal disk blocks of 256 (decimal) words each.

Global Switch
/N   Do not zero file. Provides faster execution.

Examples
CCONT Z 25 cr
  Create a contiguous file **Z** in the default directory with a length of 25 blocks.
CCONT TEXT 28 DC0:KG 57 cr
  Create contiguous file **TEXT** in the default directory with a length of 28 blocks and file **KG** in primary partition **DC0** with a length of 57 blocks.
CCONT/N X 20000 cr
  Create contiguous file **X** in the default directory without zeroing it out.

Restrictions
The asterisk and dash convention is not allowed.

CDIR subdirectoryname

Purpose
Creates a subdirectory with a **.DR** extension.

Examples
CDIR C cr
  Create subdirectory **C.DR** in the current partition.
CDIR DC0:A:B cr
  Create subdirectory **B.DR** within secondary partition **A** of primary partition **DC0**.

CHAIN savefilename

Purpose
Overwrites CLI by performing a program chain to **savefilename**. The command is used with considerable caution. After chaining the ground normally terminates (unless it is the only ground running, which causes CLI to restart). Also, the program can chain back to CLI via the **.EXEC** system command.

Example
CHAIN OSIP cr
  The CLI is overwritten by **OSIP.SV** when the command is executed.

CHAIN SWAP inputconsole outputconsole

Purpose
Allows any ground to reset its console devices. The command does this by issuing **.SYSTEM** calls **.SCIN** and **.SCOUT** and the chaining to the CLI to reinitialize it.

Global Switch
/D   Start at debug address.

Examples
CHAIN SWAP $TTI $TTO cr
  Reset current ground to first **TTY**.
CHAIN SWAP $TTI1 $TTO1 cr
  Reset to second **TTY**.
CHAIN SWAP $ICR $ICX cr
  Reset to **$ICR/ICX** (like old GDSCLI).
CHAIN SWAP $KEY0 CRT:0 cr
  Reset to station 1 CRT/Keyboard (like old CRTCLI).
CHAIN SWAP $KEY1 CRT:1 cr
  Reset to second station.
CHAIN SWAP QTY:2/2 cr
  Reset to unit 2 **QTY**.

Restrictions
Use extreme caution if specifying disk files because, in that case, the CLI cannot be aborted except by ABGROUND.

CHATR filename attribute [filename attribute etc.]

Purpose

Changes, adds, deletes, or otherwise modifies the resolution file attributes of a given file or files. All current attributes are replaced by the attributes given in the parameters. The attributes permitted are one or more of the following:

N   No linking is permitted to this file.
P   Permanent file. Cannot be deleted or renamed.
R   Read-protected file. Cannot be accessed for reading.
S   Save file. Once set, this attribute cannot be removed.
W   Write-protected file. Cannot be altered.
O   No attributes are set.
*****   All current attributes are retained.
?   User-defined attribute (bit 9).
&   User-defined attribute (bit 10).

When two or more attributes are given for a file, they are concatenated as a single parameter.

Example
CHATR ALPHA R* BETA O cr
   Add read protected attributes to **ALPHA**; **BETA** is stripped of all attributes, except save.

CHLAT filename attribute [filename attribute etc.]

Purpose

Changes, adds, deletes, or otherwise modifies link access of a file or files. All current link access attributes are replaced by the attribute given in the parameters. The attributes permitted in the attribute parameters are one or more of the following:

N   No linking is permitted to this file.
P   Permanent file. Cannot be deleted or renamed.
R   Read-protected file. Cannot be accessed for reading.
S   Save file. Once set, this attribute cannot be removed.
W   Write-protected file. Cannot be altered.
O   No attributes are set.
*****   All current attributes are retained.
?   User-defined attribute (bit 9).
&   User-defined attribute (bit 10).

When two or more attributes are given for a file, they are concatenated as a single parameter.

Example
CHLAT ALPHA W cr
   Set write protect attribute for file **ALPHA**.

CLEAR [filename etc.]

Purpose
Clears the file use count of one or more **SYS.DR** entries given by filenames. The command is used when files left open on a system failure or if the system is shut down without releasing the master device. This command is issued from any level in the foreground or from level zero background.

Global Switches
/A Clear use count in all files in current directory except **CLT.OL, CLI.ER, SYS.TU, system.OL,** and **LOG.CM**. (Refer to Table 1-9 for a description of file name extensions.)

/D Set device use counts to zero in current directory.

/V Verify files cleared with list on console. (Used with **/A**.)

Examples
CLEAR/A/D cr
This could be used after system failure when one or more files in the current directory were left open. When use counts are zeroed, the files can be deleted or renamed.

CLEAR DC0:A:B cr
Zero count of file **B** in subdirectory **A** of **DC0**.

CLEAR DC1:A:B cr
Zero count of file **B** in subdirectory **A** of **DC1**.

CLG filename [filename etc.]

Purpose
Performs FORTRAN IV compilation, loading, and execution of one or more FORTRAN IV source files. Output includes one or more intermediate source files, one or more relocatable binary files, and an executable save file. The save file is created by the relocatable loader, using the relocatable binary files and four of the FORTRAN IV libraries which were merged into a single library named **FORT.LB** (see **LFE M** command).

If a listing device is not specified with a global switch, but is specified by a local switch, listings of each FORTRAN IV compilation, assembly, and load map are output to the listing file specified by the local switch.

On input, the system searches for **filename.FR**, and if not found, it searches for filename. If local switch **/A** is specified, the system searches for **filename.SR**, and if not found, it searches for filename. When local switch **/N** is specified, the system searches for **filename.RB**, and if not found, it searches for filename. On output, the system produces temporary assembler source files, **filename n.SR**; produces relocatable loader input files, **filename n.RB**, (Where **n** is the successive number of specified filenames); and produces save file, **filename.SV**, for the first specified filename.

Global Switches
/B List only compiler source program.

/E Suppress compiler error messages.

/M Suppress loader map.

/T Multitasking mode (multitask FORTRAN library, **FMT.LB**, must be on disk).

Local Switches
/A Assemble and load only; do not compile.

/L Listing output directed to given file name.

/O Load (no compilation or assembly).

In addition, each **filename.SR** may have local switches appropriate for assembly as listed for the **ASM** command, and each **filename.RB** may have local switches appropriate for loading as listed for the **RLDR** command. Overlays can be created with the **CLG** command in a similar manner to the **RLDR** command.

Example
CLG/T MAIN AB/O AC/A FORT.LB $LPT/L cr
Program **MAIN** is file in FORTRAN IV language, **AB** is in relocatable binary and **AC** is in assembly language. Loading of the multasking library **FMT.LB** is from disk file (**/T** option). When compiling, assembling, and loading are completed, **MAIN.SV** is executed with a listing to the line printer.

Restrictions
The asterisk and dash convention is not allowed.

The **CLG** command differs from the **FORT** command because **CLG** treats source input files individually where some files require loading, other files require assembly and loading, and still others also require compilation. The **FORT** command produces a relocatable binary file, but cannot produce a save file and execute it as **CLG** does.

CPART partitionname blkct

Purpose
Creates a Secondary Partition, with a **.DR** extension, as a contiguous file whose length in blocks is given by **blkct**, where the number of blocks is an integer multiple of 16 decimal and greater than or equal to 96 decimal. If **blkct** is not a multiple of 16 decimal, it is truncated to the next lower multiple. The maximum partition size is 64K sectors.

Example
CPART DC0:APART 100 cr
    Secondary partition, **APART**, is created in primary partition **DC0**. **APART** is 96 blocks in length (100 truncated to 96).

CRAND filename [filename etc.]

Purpose
Creates one or more randomly organized files in a given directory, each having a length of zero and the attribute D.

Example
CRAND RFILE SFILE DC2:CASE1 cr
    Create random files **RFILE** and **SFILE** in the current directory and random file **CASE1** in **DC2**.

CREATE filename [filename etc.]

Purpose
Creates one or more sequentially organized files in a given directory, each having a length of zero and no attributes.

Example
CREATE TEST TEST1 DC1:TEST2 cr
    Create files **TEST** and **TEST1** in the current directory and file **TEST2** in **DC1**.

DEB filename

Purpose
Used to debug a save file .A symbolic debugger must have been loaded as part of the program save file. (See **RLDR** command.) When executed, the **DEB** command transfers control to the debugger in this save file and awaits debugging commands.

During debugging, memory can be examined, break points set, the program run, etc. After making any desired changes in the program, issuing the **$V** debug command, returns control to CLI, and then save the core image under an appropriate file name, as described under the **SAVE** command.

Examples

DEB A cr	Debug **A.SV**
. cr	
. cr	Debugger commands
. cr	
$V cr	Return to CLI
BREAK	CLI issues break
SAVE A cr	Changed version of **A** saved
A cr	Execute **A**

DELETE filename [filename etc.]

Purpose
Deletes the files in a directory having names given in the argument list. A filename is preceded by a directory specifier if the directory is initialized. Resolution files may be deleted, but an attempt to delete a link causes the link's resolution file, if any, to be deleted. To remove a link entry, use the **UNLINK** command.

Global Switches
/C   Confirm each deletion. Repeat each filename and wait confirmation that the file is to be deleted. Pressing carriage return deletes the file; pressing any other key prevents deletion.
/L   List deleted files on $LPT (overrides /V).
/V   Verify deleted files with a list on the console.

Local Switches
/A   Delete only files created this date or after, where the preceding parameter has the form mm-dd-vv and mm and dd are one or two digits.
/B   Delete only files created before this date, where the preceding parameter is the same for as local option **/A**.
/N   Do not delete files matching this name.

Examples
DELETE/V LIMIT.- MAP cr

DELETED LIMIT.SR
DELETED LIMIT.RB
DELETED LIMIT.SV
DELETED MAP.
Deleted files are verified

Deletes all files having the name **LIMIT** and any extensions (including null) and verifies deleted files on system console.

DELETE/C A-E cr

APPLE cr*
ADLE cr*
ALICE #
ACME #
ABLE cr*

Confirms each file to be deleted by printing out the file name. A carriage return (cr) response deletes the file and the system echos an asterisk (*). Any response (# in this case) other than cr saves the file from deletion.

Restrictions
Deleted files cannot be restored. For example, -.- as a parameter wipes out all non-permanent files on the disk. Be extremely careful using the dash convention with the delete command.

The asterisk and dash convention is used only when the filename parameter is in the current directory.

DGL [/switches] filename [/switches]

Purpose
Compiles DG/L source files.

Global Switches
/B   Output brief listing of source code and storage map only.
/C   Check syntax without ckecking semantics or generating any code.
/E   Check for errors and produce listing and cross-reference of any errors found.
/F   Generate code for a program which does not require a floating point unit.
/H   (NOVA only) Use hardware multiply/divide unit.
/I   Don't list include files.
/L   Produce a listing to **.LS** file.
/M   Generate code with short LEF's.
/N   Do not generate object code.
/O   Restrict Optimization.
/P   Generate code for external procedures assuming the correct number of parameters.
/R   Use floating point for integer divisions in sub-expressions.
/S   Generate code for subscript checking.
/T   Generate code for string overflow checks.
/W   Include warning message (may cause voluminous output).
/X   Generate full cross reference table, including literals.
/Z   Generate code assuming all external integers are resolved to page 0 locations.

Local Switches
/B   Specifies binary object code file (**source.RB** by default).
/C   Specifies NOVA or ECLIPSE object code, (use **N/C; E/C** is the default).
/E   Specifies error message output file.
/L   Specifies list output file.
/O   Identifies system code string for conditional compilation.

Examples
DGL/A/M TEST TEST.LS %GCOUT%/E cr
RLDR/D TEST @ALIB@ cr
Compiles the program **TEST.AL**, omitting an assembly-level listing, with code generated for a mapped ECLIPSE computer. The compiled listing is written to the file **TEST.LS**. Error messages are output to the current console. The relocatable loader command demonstrates that a **DGL** program typically references one or more programs found in libraries specified by the file **ALIB**.

Restrictions
The **DGL** compiler can generate code for NOVA computers, but the compiler can be run only on ECLIPSE computers.

DGLREF [/switches] [filename/switches etc.]

Purpose
Produces a cross-reference listing for multiple **DGL** or **ALGOL** programs.

Global Switches
/D   Disk space is minimized in the sort phase.
/I   Include references located in an include file.
/K   Include keywords.

Local Switches
/L   List file (**$LPT** is the default).
/X   Scratch directory for the sort phase.

Example
BUILD X -.AL; DGLREF @X@ cr
  Produces a cross reference listing for **DGL** or **ALGOL** programs in the current directory with output directed to the line printer.

Restrictions
  1. The utility program **ASORT.SV** (or a link) must be available in the current directory.
  2. **DGLREF** is used only on ECLIPSE computers.

DIFF [/switches] newfile oldfile [differencefile]

Purpose
Determines changes in a source file between **newfile** and **oldfile**. The output goes to **differencefile**, which defaults to **DIFF.DI** if omitted.

The **differencefile** contains several lines of text wherever the new and old files differ. The output identifies the page and first line number of the difference in the new file along with the text which does not match the old file. The corresponding data from the old file is also output. While the last lines from output file always match (as a further aid in reconciling the location of the difference with the listing), the difference program requires that four consecutive lines match to terminate the difference entry.

If one hundred lines of each file do not match in any four consecutive lines and if the automatic option (/A) is not specified, the program asks for help from the user to get the files synchronized. The program prints the location in each file and requests the user to respond with A, C, M, N, or O. The meaning of these responses are:

A - Abort   Do not list any more differences.
C - Continue   Try an additional 100 lines in file for a match.
M - Match   Output the unmatching blocks and try again.
N - New   Output the unmatched block from the new file and try again.
O - Old   Output the unmatched block from the old file and try again.

Global Options
/A   Automatic mode. Do not ask user if files differ by more than 200 lines.

Example
DIFF/A TEST.SR TEST.BK TEST.DI cr
  This command creates a difference listing of the files **TEST.SR** and **TEST.BK** with the output placed in file **TEST.DI**. It is assumed that the user wants to check the results of an edit.

DIR directoryspecifier

Purpose
Changes the current directory or current default directory device and initializes the new directory or directory device if necessary. **Directoryspecifier** is given as:

primaryparition:secondarypartition:subdirectory

Any area or combination of areas can be selected. If **primarypartition** is not specified, the command references the current primary partition.

Examples
DIR DC1 cr
Direct all default filename references to files primary partition **DC1**.
DIR DC2:DEF:GRR cr
Direct all filename references to files to subdirectory **GRR** in secondary partition **DEF** of primary partition **DC2**.

Restrictions
The **DIR** command implies an **INIT**, and all initialized directories must be unique.

DIRECTORY

Purpose
Lists all directories currently initiated (**INIT**) by all active grounds when called from the master directory.

Example
DIRECTORY cr

DIRECTORY   GROUNDS INIT'ED
DC0   **BG**
SYSGEN   **BG**

9 DCB's FREE

The message indicates directories **DC0** and **SYSGEN** are initiated and they are both in the main background (**BG**). Also, there are 9 disk directory blocks available for new directories in this system.

DISK

Purpose
Obtains a count in decimal of the number of blocks used and number still available in the current partition. If the current directory is a subdirectory, the size of the parent partition is indicated.

Example
DISK cr
LEFT: 692, USED: 332
The message indicates that 692 of 1024 blocks in the current partition are still available for use.

DUMP outputfile [filename etc.]

Purpose

Dumps a file or files to a given file or device.

In the list of filenames, filenames can be paired as follows: **oldfilename/S newfilename**, where **newfilename** is the name of the file in the dump and **oldfilename** is the name retained in the current directory.

Directory information for each file, including name, length, attributes, creation and last access time; are written as a header to each dumped file. If file names are given, names are not preceded by a device specifier. Filename is either a partition or subdirectory. Dumping a directory dumps the contents of the directory also.

Global Switches

/A	Dump all files, permanent and nonpermanent. (If not given and there is no list of files to be dumped, all nonpermanent files are dumped.)
/K	Do not dump links.
/L	List dumped files on **$LPT**. (Overrides **/V**.)
/S	Dump file as segments of paper tape of up to 20K bytes each. (Tape segments are suitably numbered for later reloading in proper sequence.)
/V	Verify dumped files with console list.

Local Switches

/A	Dump only files created this day or later. (Preceding parameter has the form mm-dd-vv, where mm and dd are one or two digits.)
/B	Dump only files created before this day. (Preceding parameter has the form mm-dd-yy, where mm and dd are one or two digits.)
/N	Do not dump files matching this name.
/S	Retain preceding **oldfilename** in directory but rename in dump files. (See Purpose.)

Examples

DUMP/A MT0:0 cr
Dump all files to **MT0:0** (magnetic tape file 0).
DUMP/V DUMPFI -.SV cr
EDIT.SV
ASM.SV
RLDR.SV
A.SV
B.SV
Files dumped to **DUMPFI** are verified by listing on the console.
DUMP SOURCE -.SR 7-22-74/A cr
Dump all files with the **.SR** extension created on or after July 22, 1974 to the file **SOURCE**.

Restrictions

If while dumping all files (**DUMP/A**) in an environment containing several directories, dumping is aborted by typing **CTRL A**, CLI must be explicitly directed to a specific directory before making any further file references. This is done since it is impossible to determine what directory was being dumped and was therefore the current directory at the time dumping was aborted.

The asterisk and dash convention is permitted only when the filename is in the current directory.

EDIT [filename]

Purpose

Invokes the text editor to build a new source file or edit an existing source file. When the command is executed, the editor responds with an asterisk (*). Use of this editor is described in Appendix A of this manual.
The user terminates editing and returns to the CLI by typing the following characters: CTRL D, ESCAPE, ESCAPE. These characters are echoed as ↑D$$.

Global Switches

/A   Use all of memory for Edit Buffer, not just 16K.
/L   Allow lower case input from console (not required if Hazeltine).

Examples

EDIT cr
*

  The editor is invoked and the user issues commands indicating the input file and the output file to be used.
EDIT MYSYSTF cr
*

  Adding a filename causes the editor to be invoked and to issue the editing command **UYMYSYSTF$$**.

Restrictions

Two copies of the **EDIT** program exist; **AEDIT** for ECLIPSE computers and **NEDIT** for NOVA computers. The **EDIT** command is linked to the correct version when a standard system is executed.

ENDLOG [password]

Purpose

Closes the log file opened by a previous **LOG** command and terminates recording of CLI dialog. Password must match the password, if any, given in the **LOG** command.

Example

ENDLOG JSMITH cr

  This command closes the current log file which was opened with the password **JSMITH**. The file must be closed before any **TYPE, PRINT,** etc. commands are executed.

EQUIV newname oldname

Purpose
Assigns a new name to a multiple file device before the device is initialized. The equivalence command also allows an alias to be established for a secondary disk pack.

Examples
Assume that a program is coded that references the file **TAPE**. If prior to running this program, the **EQUIV TAPE MT0** command is issued, then all references to **TAPE** are interpreted as references to the corresponding **MT0** file.

Assume that it is desired to perform a full initialization on a disk pack in **DC1** when using a dual disk system. Since the straight forward method is disallowed as a safety feature, the following command sequence works:

EQUIV X DC1 cr
INIT/F X cr
RELEASE X cr

Restrictions
The **EQUIV** command cannot be issued for a master disk device nor can it be issued to a device that is initialized. After release, a device's default global specifier is reassigned to it. Subdirectories and Secondary Partitions cannot be equivalenced.

ERROROFF

Purpose
Suppresses the output of CLI error messages during the execution of a command macro or indirect file.
Error message output is reenabled if an **ERRORON** command is executed or the command execution is completed and the CLI prompts the user for another command.

Example
Assume the macro **LOAD.MC** contains the following commands:
ERROROFF; INIT SGDS:ALGOL5; ERRORON
RLDR/D TEST @ALIB@
    Execution of this macro command;
LOAD cr
    Initializes the directory **ALGOL5** in partition **SGDS** and
    performs the relocatable load. If the same macro is
    repeated, error messages to the effect that the directory
    is initialized are suppressed while any error messages
    from the load are output.

ERRORON

Purpose
Enables the output of CLI error messages during the execution of a command macro or indirect file containing the command **ERROROFF**.

Example
ERRORON cr
  Turns on the error message output.

EXBG groundid program priority [inputconsole ↑ [outputconsole [datum]]]

Purpose
Starts execution of another background if the ground is inactive and has a memory partition assigned.

Arguments
The **groundid** consists of the RDOS grounds: **FG, BG, BG2, BG3,..., BG15**. Most RDOS systems are generated with fewer than sixteen backgrounds and only those grounds actually generated are used as parameters to **EXBG**. (The **GMEM** command lists all generated grounds and the core sizes assigned to each.)

The **program** is the name of a save file that is to receive control and the **.SV** extension must be used. **Priority** specifies the ground priority in a range from 1 to 256. The recommended value is 2 which is used to establish the same priority as the main background.

The **inputconsole** and **outputconsole** parameters specify the device used by the new ground for any interaction such as CLI commands, error messages, etc. If these parameters are omitted, the current console is not changed. If the ground was designated as swappable, no console input is allowed if the console input is a keyboard input device. The input and output console names are limited to five characters.

The **datum** is the quantity passed in Ac2 to the program (16 bits).

If the **EXBG** command is used without arguments, the program pormpts for the arguments interactively. The system will print **BG#**, which requires the entry of an integer (number of the ground). It then prompts for a **DATUM** word to pass to the new program. Enter any decimal integer value.

Global Switches
/D   Start program at debugger address. **DC0** must contain **DEB.SV** to use this switch.

Examples
EXBG BG2 CLI.SV 2 QTY:0/2 cr
  Starts **BG2** running the command line interpreter at priority **2** with both the console input and console output devices set to **QTY** unit 0.
EXBG BG6 JOBIN.SV 2 CIN COUT cr
  Starts **BG6** running the program **JOBIN** at priority **2** with disk files **CIN** and **COUT** specified for the console input and console output units.

Restrictions
The command **EXBG** is issued only in a mapped environment.
If the console input unit device is a disk file and the CLI will run, the input file should conclude with a **POP** to deactivate the ground or a **CHAIN SWAP** to some console.

EXFG executable stream

Purpose
Loads an **executable stream** from a background into core for foreground execution. The **executable stream** is either a save file or a CLI command stream. A foreground save file is not loaded for execution if it would cause overwriting of the CLI. Loading a utility command stream causes creation of a foreground file, **FCOM.CM**, similar to **COM.CM**.

Global Switches
/D   Pass control to the debugger, not save file.
/E   Foreground and background have equal priority.

Examples
EXFG DC1:MON cr
   Load and execute **MON.SV** in **DC1**.
EXFG CLG MAIN cr
   Execute the command stream **CLG MAIN**, causing compilation, assembly, loading, and execution of FORTRAN IV program **MAIN**.

Restrictions
This command is less powerfule then **EXBG**. The **EXBG** command will also start a foreground, but cannot build an **FCOM.CM** file as **EXFG** does.

FDUMP [/switches] [list file/switch]

Purpose
Creates a backup copy of the current directory on magnetic tape as a faster alternative to **DUMP**. **FDUMP** does multitasking and also supports multiple reel dumps. Compare with **BACKUP** and **DUMP** commands.

Global Switches
/L   List file names on the line printer.
/V   List file names on the console. (Verify)

Local Switch
/L   List file names in the specified file.

Example
To dump the directory **DIR1** on magnetic tape drive unit 0, file 0, mount a scratch tape on magnetic tape drive unit 0.
DIR DIR1 cr
   To make **DIR1** the current directory.
FDUMP/V cr
   Magnetic tape drive unit 0, file 0 (**MT0:0**) is assumed and file names are listed on the system console by option **/V**.

Restrictions
FDUMP is not used to dump more than one directory per magnetic tape, and files cannot be stacked. This command is not selective so that all files in the current directory are copied. Use **FLOAD** to reload the directory dumped by **FDUMP**.

Error messages that can be returned by **FDUMP** include:

"FILE-NAME"
   Indicates a file name error in the dumped file.
"OPEN ERROR-FILE NOT DUMPED"
   Indicates the directory to be dumped was open (current directory).
"SYS ERR RTN-OFFSET nnnnn IN xxxxxx"
   Indicates an RDOS error in **FDUMP**.

FILCOM filename filename [filename/L]

Purpose

Compares two files word by word and prints any dissimilar word pairs. The displacement (difference) is printed, followed by a slash, followed by the dissimilar word pair, for example:

```
 15145/      000000      020044
   ↑           ↑           ↑
displace-    file 1      file 2
ment         word        word
```

Local Switches

/L   Listing file. Default listing is to the console.

Example

FILECOM TEST1 TEST2 $LPT/L cr

    Compare files **TEST1** and **TEST2** word by word and print dissimilar word pairs on the line printer.

FLOAD [/switches] [file/switch]

Purpose

Reloads a tape created by **FDUMP** into the current directory. Compare with **LOAD**.

Global Switches

/L   List loaded file names on the line printer.

/V   (Verify) list loaded file names on the current output console.

/N   (No Load) List file names according to global /L, /V or local /L.

Local Switch

/L   List loaded file names in the specified file.

Example

To load a directory that was previously dumped by **FDUMP** on magnetic tape drive unit 0, file 0 (**MT0:0**): Mount the tape containing the directory dumped by **FDUMP** on magnetic tape drive 0.

DIR DIR1 cr

    To make **DIR1** the current directory.

FLOAD/V cr

    Magnetic tape drive unit 0, file 0 (**MT0:0**) is assumed and file names are listed on the system console by option /**N**.

Restrictions

Error messages that can be returned by **FLOAD** include the same messages listed for **FDUMP**.

FORT inputfilename [filename]

Purpose
Compiles a FORTRAN IV source file **inputfilename** with output directed by local switches to **filename**. The output is a relocatable binary file, an intermediate source file, a listing file, or combinations of all three. Execution of the command produces an intermediate source file, **inputfilename.SR** (output of compilation), and a relocatable binary file, **inputfilename.RB** (output of assembly).

When assembly is complete, the intermediate source file is deleted. No listing is produced unless the /L switch is given.

Global Switches
/A   Suppress assembly.
/B   List compiler source input only.
/E   Suppress compiler error messages.
/F   Equivalence FORTRAN labels and variable names to assembler acceptable symbols.
/L   Produce listing to **inputfilename.LS**.
/N   Do not produce relocatable binary file.
/P   Process only 72 columns/record or characters/line
/S   Save the intermediate source file.
/U   Output user symbols during assembly. (Must be used with /F.)
/X   Compile statements with X in column 1.

Local Switches
/B   Relocatable binary output file (overrides /N).
/E   Error messages file.
/L   Listing file (overrides global /L).
/S   Intermediate source file.

Examples
FORT/F/U DC1:TABLE $LPT/L cr
    Compile FORTRAN IV file **TABLE** in **DC1** with relocatable binary output ot **TABLE.RB** and listing on the line printer. User symbols are output during assembly.
FORT MAIN $LPT.L cr
    Compile FORTRAN IV file **MAIN** and produce relocatable binary file **MAIN.RB** with a listing to the line printer.

Restrictions
The **FORT** command must be used in compiling FORTRAN source programs and the name **FORT** cannot be changed. Compare with the **CLG** command.

FORTRAN inputfilename [filename]

Purpose
Compiles a **FORTRAN 5** source file **inputfilename** with output directed by local switches to **filename**. The output is a relocatable binary file, a listing file, or both. Execution of the command produces a relocatable binary file, **inputfilename.RB**. No listing is produced unless a /L switch is given. On input, the system searches for **filename.FR**, and if not found, it searches for **filename**. Output is **inputfilename.RB** by default and **inputfilename.LS** if a global switch /L is given.

Global Switches
/B   Brief listing. (Compiler source input.)
/C   Check syntax only.
/D   Give line number and program name on all run time error messages.
/L   Produce listing, **inputfilename.LS**.
/P   Process only 72 columns/record or characters/line
/S   Generate code for subscript checking.
/X   Compile statements with X in column 1.

Local Switches
/B   Relocatable binary output file.
/E   Error message file.
/L   Listing file.

Examples
FORTRAN/B MARK $LPT cr
    Compile file **MARK** with output of relocatable binary **MARK.RB** and brief listing on **$LPT**.
FORTRAN/D/S/L TEST cr
    Compile file **TEST** with output of relocatable binary **TEST.LB** and listing file **TEST.LS**. Perform subscript checking and list line number and program name on runtime errors.

Restrictions
The **FORTRAN** command is used in all FORTRAN 5 compilations, and the name **FORTRAN** cannot be changed.

For possible error messages, refer to the Data General FORTRAN 5 User's manual.

**FPRINT filename [number/F] [number/T] ꜛ
[filename/L]**

Purpose
Prints the contents of any readable disk file in octal, decimal, hexadecimal, or byte format with any printable ASCII characters printed on the right side. Any non printing characters are output as periods (.). The location counter is always printed in octal. The default listing device is the console.

Global Switches
/B   List in byte format.
/D   List in decimal format.
/H   List in hexadecimal format.
/L   List on line printer.
/O   List in octal.
/Z   File starts at location zero.

Local Switches
/F   Print contents starting at octal location preceding switch.
/L   List to given file. (Overrides global /L.)
/T   Last location to be printed.

Example
FPRINT/B/L MYFILE 2000/F 3500/T cr
    Print **MYFILE** in byte format on the line printer from location **2000** to location **3500**.

Restrictions
FPRINT is used to dump only the first 64K words of a file.

FSR

Purpose
Allows files to be saved and restored using magnetic tape and disk. Any existing file on disk can be saved on tape and later restored to the same disk or any other disk. This special purpose utility is used to save special purpose files, to replace existing files with updated files, and to add newly developed files to an existing directory.

NOTE
    FSR is not intended to serve as a general purpose RDOS file backup as it does not perserve all information about a file.

FSR is the only utility used to move files between the RDOS system and disk management system (**DMS**). To move source files with **FSR**, the **.SR** extension is used explicitly. On tape, the files have **.SO** extensions for compatibility with **DMS**. Also, moved file names must have six or less characters to maintain compatibility with **DMS**. Differences between RDOS and DMS include the following:

1. The **READ** command in RDOS has the same function as the **PREVIOUS** command in DMS.
2. RDOS does not have equivalent functions for DMS **READ** and **NEXT** commands.
3. To specify the names of files to be dumped, the **TDUMP** command in RDOS requires a command file format produced by the CLI **BUILD** command.

The **FSR** command uses an interactive routine to transfer files from disk to tape (write) or tape to disk (read). Before using the command, the appropriate tape and disk are installed on their respective drive units. The appropriate tape and disk depends upon the direction data is to be transferred. Changes to existing files or new files can be transferred from an **FSR** tape to disk, or an **FSR** tape is created from selected files on a disk. The **FSR** interactive commands are as follows:

A   All command to read all files from tape to disk. This command requires a second prompt and a yes (Y) verification because a data loss could occur. When verified, the **A** response reads from tape to disk and unconditionally overwrites any disk file that has a duplicate name on tape.

F   Finished command used to exit the interactive routine.

L   List command will list all files from the beginning of the tape on the system console.

O   Echo command lists each command as it is written or read.

R   Read command to transfer a file from tape to disk. The system queries to find out which command to read. This query asks for a file name and a new name. If the new name exists on disk, the existing file can be overwritten or a new name can be selected.

S   Start command positions a tape to the beginning.

T   **TDUMP** command writes all files specified by the contents of a given file.

W   Write command to transfer a file from disk to tape. A message prints out if the file does not exist. An existing file is transferred to the next available space on the tape.

Z Zap command is used to initialize a new tape to receive files. It destroys all files existing on the tape. This command requires a second prompt and a yes (Y) verification because a data loss could occur. When verified, the **Z** command erases existing files and initializes the tape.

Examples
FSR cr
RDOS ALGOL FSR 24JAN77

Z, R, W, F, S, A, L, T, O
READY MTA:
  This is an error response. **READY MTA:** indicates the magnetic tape drive unit is not on line and ready.

FSR cr
RDOS ALGOL FSR 24JAN77

A, R, W, F, S, A, L, T, O
*ZAP
OK ?YES
*FINISHED
R
  This example shows the **Z** and **F** commands. Following the asterisk (*) prompt; when **Z** is typed; the system echoes **AP**, carriage return, line feed, and **OK?**. When **Y** is then typed, the system echoes **ES**, carriage return, line feed, and the prompt (*). Typing **F** then causes the system to echo **INISHED**, carriage return, line feed, and the CLI prompt (**R**).

FSR cr
RDOS ALGOL FSR 24JAN77

Z,R,W,F,S,A,L,T,O
*WRITE
FILE :**ASM.SV** cr
FILE :**AFMT.LB** cr
FILE :**LONGTRACE.RB** cr
FILE :**RLDR.OL** cr
FILE :**UMAC.PS** cr
FILE :**DGL.TX** cr
FILE : cr
*LIST

ASM.BI
AFMT.LB
LONGTR.RB
RLDR.OL
UMAC.PS
DGL.TX

*READ
FILE :**ASM.BI** cr
NAME: cr
FILE :**AFMT.LB** cr

NAME : cr
AFMT.LB ALREADY EXISTS. TYPE 'Y' TO DELETE :Y
FILE :**LONGTR.RB** cr
NAME : cr
FILE :**RLDR.OL** cr
NAME : cr
RLDR.OL ALREADY EXISTS. TYPE 'Y' TO DELETE :Y
FILE :**UMAC.PS** cr
NAME : cr
UMAC.PS ALREADY EXISTS. TYPE 'Y' TO DELETE: Y
FILE :**DGL.TX** cr
NAME : cr
DGL.TX ALREADY EXISTS. TYPE 'Y' TO DELETE :Y
FILE : cr
*START OF MT
*FINISHED
R

In this example, the **W** command was used to write some existing files from disk to tape. Typing **W** causes the system to print **RITE**, carriage return, line feed, and FILE:. The name of the file to be written is then typed and followed by pressing the RETURN key (cr). This file name prompting continues until the RETURN key is pressed before a file name is typed. The system then echoes a carriage return, line feed, and the **FSR** prompt (*).

Typing **L** causes the system to print a list of the files from tape. The name of the **ASM.SV** file was changed to **ASM.BI** on tape, and the name of **LONGTRACE.RB** was shortened to **LONGTR.RB** because it had too many characters (six maximum). All the other file names were unchanged.

The **R** command was then typed to read the files from tape to disk. In response to FILE:, the file names are typed as they appear on the list and when RETURN is pressed, the file is read to disk. Files **ASM.BI** and **LONGTR.RB** are read to disk because they are not duplicate names. The other files could have been renamed, but instead they were read to disk (**Y** response), which wrote over the existing files with the same names.

Restrictions
Exact file names and extensions must be used to transfer data. Files with .**RB**, .**SV**, .**CI**, and .**BI** extensions should not be copied between RDOS and DMS, but for other **FSR** applications, any existing file name can be saved and restored.

GDIR

Purpose
Obtains the name of the current directory on the console.

Example
GDIR cr
MYDIR2
   The current directory is **MYDIR2** and all file references are directed to the current directory by default.

GROUND

Purpose
Retrieves status information concerning the current ground.

Example

COMMAND and RESPONSE	COMMENTS
GROUND cr	
BG4	Ground identified
INPUT CONSOLE =	
QTY:2	Input console name
OUTPUT CONSOLE +	
QTY:2	Output console name
STATUS = 1000	
PRIORITY = 2	Priority established with **EXBG**.
FLAGS = 000000	
PUSH LEVEL = 2	CLI runs at push level 1
MEMORY IN USE =	
24K	Memory partition for this ground
ground	
MEMORY UNUSED	
+ OK	
NUMBER OF UFT'S	
+ 24	Maximum # of channels (User File
CPU SECONDS =	Tables) for this ground
81.66	# of CPU seconds used so far.

GMEM

Purpose
Obtains the current size of core allocated for all grounds generated by RDOS. The size is given in 1024_{10} word blocks.

Example
GMEM cr
FG:0 BG:16 BG2:32 BG3:0
   Foreground has 0K,
   Background has 16K
   Background 2 has 32K, and
   Background 3 has 0K.

Restrictions
This command is used on mapped systems only.

GSYS

Purpose
Obtains the name of the current system.

Example
GSYS cr
SYS
   The **GSYS** command causes the name of the operating system (**SYS** in this case) to be printed on the current console.

GTOD

Purpose
Obtains the current date and time of day.

Example
GTOD cr
2/15/75   23:15:20
   The current day is February 15, 1975 and the time is 20
   seconds after 11:15 PM.

IDLE

Purpose
Releases all directories and devices that were initialized in
the current ground. If the current console is a QTY unit, a
full screen message identifying the idle ground is output.

INIT logical unit specifier

Purpose
Initializes a **logical unit specifier** if it is a device specifier or
one of the following forms:

   [primary partition:][secondary partition:]subdirectory

Any area or combination of areas can be selected.

If primary partition is not specified, the command
references the current primary partition.
Partial initialization is performed by default. A file in a
directory cannot be accessed until its directory and all
superior directories are initialized.

Global Switches
/F   Full initialization. Clears all information from the
      directory. If the device is not the master device,
      HIPBOOT is written to the device. If the device is a
      tape, it is rewound and two EOFs written. (The
      default partial initialization allows access to existing
      files on the device. If the device is a disk, the
      primary partition directory is found and read into
      the system to allow access to all its files.) Full
      initialization is required for new magnetic tapes and
      will destroy any files currently written on a
      magnetic tape.

Examples
INIT DC1:A:X2 cr
   All files in subdirectory **X2** of secondary partition **A** of
   DC1 are now accessible.
INIT/F MT1 cr
   MT1 is rewound and two EOF's are written.

Restrictions
INIT/F cannot be done to disk units (safety feature),
without first **EQUIV**'ing to another name.

LDIR

Purpose
Lists the name of the last directory. When more than one directory is initiated and the name of the previously used directory is desired, issuing this command results in a printout of the name of the last directory.

Example
LDIR cr
DC0
R

    The directory previously in use was **DC0**, the directory for the Primary Partition.

LFE key filename [filename etc.] [arg etc.]

Purpose
Modifies, updates, and analyzes relocatable library files (filenames) by action involving the libraries, or relocatable binary files (**args**), or logical records within the libraries (**args**). Action is taken in accordance with the function given by key as follows:

A	Analyze global declarations of filename or of logical records (**args**) or filename.
D	Delete logical records (**args**) of filename producing output file **D.L1**.
I	Insert relocatable binary files (**args**) among logical records of filename according to local options of the logical records (**args**), producing output file **I.L1**.
M	Merge library files (filenames), concatenating the library files to produce **M.L1**.
N	Produce new library file **N.L1** from relocatable binary files (filenames).
R	Replace records (**arg,arg,...**) in filename with relocatable binary files (**arg,arg,...**), producing output file **R.L1**.
T	List logical record titles from filename.
X	Extract logical records (**args**) from filename as relocatable binary files (**args.RB**).

Global Switches
/**M**	Analyze the following two or more library filenames as one library. Use following the **A** key only.

Local Switches
/**A**	Insert the relocatable binary file that follows the switch immediately after the logical record that precedes the switch. (Used only in **LFE I** line.)
/**B**	Insert the relocatable binary file that follows the switch immediately before the logical record that precedes the switch. (Used only in **LFE I** line.)
/**E**	Error message file.
/**F**	Print analysis of each logical record on a separate page. (Used only with /**L**.)
/**L**	List to given file instead of $**LPT**. (Used only in **LFE A** or **LFE T** lines.)
/**O**	Output library file. Overrides default output file in **D, I, M, N,** or **R** lines.

Examples
LFE A/M Z1.LB MT0:2 Z2.LB $LPT/L cr
    Analyze as one library **Z1.LB**, the library in **MT0:2**, and **Z2.LB**. List analysis on the line printer.
LFE I M.LB M2.LB/O R/A HB.RB cr
    Insert relocatable binary **HB.RB** after relocatable binary **R** in library file **M2.LB**.

LINK linkfilename filename

Purpose
Creates a link entry to filename, which may be a resolution file or another link entry. The current (or default) directory is assumed unless an alternate directory is specified. If the **linkfilename** is different from the resoltution filename, the **linkfilename** is an alias.

Local Switch
/2 The link entry name is the same as the entry being linked to.

Examples
LINK ASM.SV/2 cr

A link entry, **ASM.SV** is made in the current directory to resolution file **ASM.SV** which is in the primary partition that contains the current directory.

LINK ALINK DC1:APROG cr

A link entry, **ALINK** is made in the current directory to file **APROG** in **DC1**.

Restrictions
The creation of a link does not imply that a resolution file exists. Links can be made to non-existent files. Detection of an unresolved link is made only when an attempt is made to reference the resolution file. If this occurs, the following message prints out: FILE DOES NOT EXIST.

LIST [filename etc.]

Purpose
Lists information from the current (default) directory or from any other directory about one or more files or link entries. If no **filename** parameter is given, all non-permanent and link entries in the current directory are listed.

The command format can be extended to use the primary partition name, secondary partition name, subdirectory name, and the file name to access a particular file. Also, any combination of the names can be used.

Listed information includes the file name and one or more of the following: file size in bytes, file attribute codes, file characteristic codes, link attribute codes, file creation date and time, date file was last opened, file starting address (UFTAD of UFD), and decimal file use count. The file attributes, file characteristics, and link attributes are given as letter codes and, since the file attributes and link attributes use the same set of codes, when file attributes and link attributes are listed together, they are separated by a right slash and the file attribute is listed first. The following is a list of letter codes used in the information listing.

LETTER CODE (SEE NOTE)	MEANING
A	Attribute-protected file. Attributes cannot be changed and the **A** attribute cannot be removed.
C*	Contiguous file organization.
D*	Random file organization.
L*	Link entry. This characteristic is given to the directory entry rather than to the file.
N	No resolution allowed. This attribute prevents a link to the file.
P	Permanent file which cannot be deleted or renamed.
R	Read-protected file. File cannot be read.
S	Save file (core image).
T*	A partition is defined.
W	Write-protected file. File cannot be written onto.
Y*	A directory is defined.
?	First user-definable attribute (bit 9).
&	Second user-definable attriubte (bit 10).

NOTE
Letter codes for file characteristics are indicated with an asterisk (*) that does not appear in actual listings; all other codes are for file/link attributes.

For link entries, the following information is listed: link entry name, resolution entry name with directory specifier that was given when the link was created, and an at-sign (@) when the resolution file exists in the primary partition.

Global Switches

/A List information for all files in the current directory (both permanent and non-permanent files). The information includes file name, byte count, file attributes, and file characteristics.

/B Brief listing that gives only the file name.

/C List date and time files was created month/day/ year/ hour:minute).

/E List every category of file information (overrides global options /B, /C, /F, /O, and /U).

/F List the logical file starting address or 0 if unassigned and enclose in brackets.

/K Do not list links.

/L Output list ot line printer.

/N List links only.

/O List date file was last opened (month/day/year).

/P Do not list permanent files.

/R Print total (with /T) only.

/S Sort output list alphabetically.

/T Print total size in sectors of all files listed.

/U List file use count (decimal number with a period).

Local Switches

/A List files created on this date or after. The parameter is given in the form mm-dd-yy, where mm and dd are one or two digits.

/B List files created before this date. The parameter is given in the same form as local /A.

/N Do not list files matching this name.

Examples

LIST/E/A cr

Lists information in every category for all files in the current directory using the system console (default) to output. A typical line of the listed information would be:

FLI.SV 8160 SD 03/01/74 13:56 03/01/74 [004164] 0.

file name — random save file codes — file size in bytes — date and time file was first opened — date file was last opened — logical first address of file — file use count

The logical starting address of the first block is always listed in brackets and the file use count is followed by a period to indicate a decimal number.

Typical lines that list link entries would look like the following:

ABC.SV   DC0:DEF.SV
XXX.XV   @:XXX.SV

The link entry named **ABC.SV** has a resolution file defined with **DEF.SV** in primary partition **DC0**. The link entry named **XXX.SV** has a resolution file with the same name that resides in the primary partition. Link entries are always listed as having zero byte length and only the link characteristic.

Restrictions

The asterisk and dash convention is allowed only when the filename parameter is in the current directory.

LOAD inputfile [filename etc.]

Purpose

Loads or lists a previously dumped file or files from a given device or directory. If a filename is not given, files specified by switches to the **inputfile** are loaded. The **LOAD** command loads or lists only the files that were previously dumped using the **DUMP** command.

If the load files have partition, directory, or link characteristics, the characteristics remain unchanged and a partition is recreated with the required disk space. Files can be loaded selectively from any directory that was previously dumped. The **LOAD** and **DUMP** commands do not change the file access/file creation information specified for the files.

If files were dumped in segments (**DUMP/S** command was used), the files must be loaded in the same sequence that the dump occurred. Not loading the segments in the same sequence causes a CLI runtime error message.

Global Switches

/A   Load all files including permanent ones.
/B   Brief listing.
/I   Ignore checksum errors.
/K   Do not load links.
/L   List loaded files on the line printer. (Overrides /V and listing by /N.)
/N   Do not load files but list the filenames on the system console. If global /R is used with /N, only the most recent version of the file is listed.
/O   Delete current file if it exists and replace with file being loaded having the specified name.
/R   Load most recent version of file by examining the creation date.
/V   Verify load with listing of filenames loaded on the system console. Secondary partition and subdirectory names are indicated by preceding and succeeding line feeds.
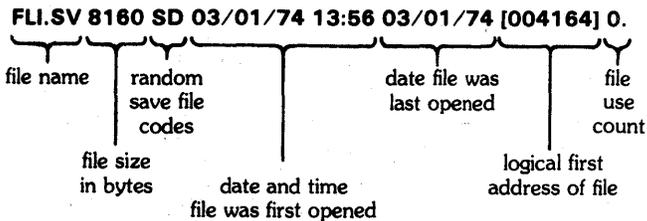/Z   Zap permanent file (used with /O or /R).

Local Switches

/A   Load files created on this day or after. The argument is given in the form mm-dd-yy, where mm and dd are one or two digits.
/N   Do not load files matching this name.

Examples

LOAD/L $PTR -.SV cr
   Load from the **$PTR** all files with the **.SV** extension and list the loaded files on the line printer.

LOAD/L $PTR -.SV 3-15-75/A TEST-.SV/N cr
   Load from the **$PTR** all files with the **.SV** extension, except files whose names begin with characters **TEST** and files created before March 15, 1975, and list on the line printer.

Restrictions

Unless the /R or /N global switches are given, the files being loaded cannot have duplicate names in the current directory.

The asterisk and dash convention is allowed only when the filename parameter is in the current directory.

LOG [password] [directory/O]

Purpose
Records CLI dialog appearing on the console in a file named **LOG.CM**. Use **directory/O** to open **LOG.CM** in a directory other than the current directory. **Password** has up to 10 alphanumeric characters and prevents **LOG.CM** from being inadvertently closed. **LOG.CM** is closed only by issuing an **ENDLOG** command (which must have the **password** if given in the **LOG** command). **LOG.CM** is examined only after it is closed.

Global Switches
/H Place a heading at the beginning of the **LOG** file as follows: Log File, current directory, master directory, current system, current date and time.

Local Switches
/O Output the file to the given directory. By default the file is output to the current directory.

Example
LOG JONESFILE cr
 All CLI dialog is directed to **LOG.CM** in the current directory.

LPT command

Purpose
Issues control commands to the Smart Line Printer Spooler when the system command **.SSCTL** was given. To enable the functions, Smart Spooler commands are:

STOP	Terminate output at the end of the next line.
PAUSE	Terminate output at the end of the next page.
RESTART	Restart output of the current job from the beginning.
CONTINUE	Restart output of the current job at current position. **CONTINUE** is used after **STOP** or **PAUSE**, and also if the hardware lost an interrupt due to pressing 'RESET' on the printer while it was printing.
BACKSPACE	Restart output of the current job at the top of the current page.
ABORT	Abort the current job in output and continue to the next job.
TERMINATE	Suspend the spooler at the end of the current job unitl the **UNTERMINATE** command is issued.
UNTERMINATE	Resume spooler operation if **TERMINATE** was used.
BLOCKBEGIN	Begin a block of output that is not separated by page headers or extra form feeds.
BLOCKEND	End a block of output begun by **BLOCKBEGIN**.

Other CLI commands for the Smart Spooler are **SPOOL** and **SPOOLQ**.

Examples
LPT ABORT cr   Abort the current **LPT** job.

Recover if paper jams:
1. **LPT STOP**
2. Fix paper in line printer and put back on line; allow it to print until buffer empties (up to 200 characters). Align paper to top of odd form (this is how it is when **LPT** is idle).
3. **LPT RESTART**

Print a bunch of files without headings:
   LPT BLOCKBEGIN
   PRINT DOGGIE.LS
   PRINT SEXY.SA DYIST.HE GREATEST.OF THEM.AL
   .
   .
   .
   LPT BLOCKEND

MAC filename [filename etc.]

Purpose

Assembles one or more source files using the macro assembler. Output may be a relocatable binary file, a listing file, or both. For more information on macro files, refer to the Data General Macro Assembler manual, 093-000081-04.

Global Switches

/A   Add semipermanent symbols to cross reference listing.

/E   Suppress error printouts if there is a listing.

/F   Produce an even number of listing pages.

/K   Keep **MAC.ST** at end of assembly.

/L   Produce listing and cross reference.

/N   Do not produce relocatable binary file.

/O   Override all listing suppression.

/S   Skip pass 2.

/U   Append user symbols to **.RB** file.

Local Switches

/B   Relocatable binary file. (Overrides global /N.)

/E   Error message file.

/L   Listing file. (Overrides global /L.)

/S   Skip this file on pass 2.

/T   Use this file instead of **MAC.PS** as the input symbol table.

Examples

MAC LIB/S (A,B,C) $LPT/L cr

Produces relocatable binary files **A**, **B**, and **C**. **LIB** contains macro definitions and is skipped on pass 2. A listing and cross-referenced symbol table are output to the line printer.

MAC/L Z cr

Assemble **Z**; produce listing to **Z.LS** and a relocatable binary **Z.RB**.

Restrictions

Do not use asterisk and dash convention with **MAC** command.

On input, **MAC** searches for assembler or basic source program in **filename.SR**. If not found and there is no extension given to **filename, MAC** searches for **filename**.

On output, **MAC** produces a relocatable binary file (**filename.RB**) by default and will also produce a listing file (**filename.LS**) if a global /L switch is given. **Filename.LS** is the first source file specified without local switches /S, /L or /B.

MDIR

Purpose

Obtains the name of the current master directory on the current console. This directory contains the system save and overlay files, the spool file, and push space for program swaps.

Example

MDIR cr

DC0

The current master directory is **DC0**.

MESSAGE/switch "message"

Purpose
Allows a macro file to write a message to the output console when the message is enclosed in double quotes.

Global Switches

/N   Output message without a carriage return.

/P   Pause until user inputs.

Example
MESSAGE/N "TAPE MADE"; GTOD cr
 When these commands are executed, the message is output without a carriage return so it reads:
  TAPE MADE 2/15/75 23:16:20
MESSAGE/P "LOAD NEXT TAPE" cr
 When this command is executed, a pasue is envoked after the program prints:
  LOAD NEXT TAPE
  STRIKE ANY KEY TO CONTINUE
 The user must then strike any key on the keyboard to continue.
MESSAGE/P "" cr
 This command causes a pause and the following message prints out:
  STRIKE ANY KEY
MESSAGE ""
 This command causes a line feed to be output.

MKABS savefilename absbinfilename [octal/S]

Purpose
Makes an absolute binary file from a save file (core image).

Global Switches

/S   Use starting address of save file specification in **USTSA** as address of absolute binary start block instead of default null start block.

/Z   Begin save file with core location zero; default is 16 octal. (See **RLDR** global switch /Z.)

Local Switches

/F   First address, relative to save file location zero, from which absolute binary file is created.

/S   Absolute binary start block has the address specified by the octal number preceding this switch.

/T   Last address, relative to save file location, to become a part of absolute binary file.

Example
MKABS FSAVE $PTP 1000/S cr
 From save file, **FSAVE**, punch an absolute binary file with a starting address of **1000**.

MKSAVE absbinfilename savefilename

Purpose

To create a save file from an absolute binary file.

Global Switch

/Z   Start save file at core location 0 rather than 16 octal.

Example

MKSAVE/Z $PTR DC0:A cr

From the absolute binary file in the paper tape reader, produce save file **A.SV** on disk 0. Begin **A.SV** at location 0.

MOVE directory [filename etc.]

Purpose

Moves entries for a given file or files in the current directory to a given directory or device directory. In the list of filenames, filenames are paired as follows:

oldfilename/S newfilename

The **newfilename** is the name of the entry in the directory and **oldfilename** is the name retained for the entry in the current directory. If no file names are given, all nonpermanent files are moved. File names cannot be preceded by a directory/device specifier. Only entries for resolution files and links are moved; partitions and subdirectories cannot be moved.

Global Switches

/A   Move all files including permanent files.
/D   Delete original files once transfer is complete. (Does not delete permanent files.)
/K   Do not move links.
/L   List moved filenames on line printer (Overrides /V and console listing by /N.)

/R   Move most recent version of file. (If there is an entry to be moved in the directory more recent than the entry in the current directory, the file is not moved.)
/V   Verify moved files with console list.
/Z   Zap (delete) permanent destination files (used with /R).

Local Switches

/A   Move only files created this date or after, where the preceding argument has the form mm-dd-yy, and mm and dd are one or two digits.
/B   Move only files created before this date, where the argument has the form given for local switch /A.
/N   Do not move files matching this name.
/S   Assign new name to moved file but retain the old name in the current directory. (See Purpose.)

Example

MOVE NEWDIR JOE.SV/S JOEJR.SR A*.SR cr

Move a file **JOE.SR** to directory **NEWDIR** and rename it **JOEJR.SR**. Also move all files whose names have two characters starting with letter **A** and having the **.SR** extension.

MSYSGEN

Refer to **SYSGEN** command for purpose and use information.

NSPEED

Refer to **SPEED** command for purpose and use information.

OEDIT filename

Purpose

Invokes the octal editor to examine and modify locations of **filename**, which may be any user file. The octal editor uses a complex syntax similar to the savefile editor (**SEDIT**) and based on the RDOS Debugger program. An advantage of **OEDIT** over **FPRINT** and **SEDIT** is that **OEDIT** can display and edit a file of any size. Refer to the Data General Octal Editor Manual for more details on using the octal editor.

To exit the octal editor and return to CLI, press the escape (<ESC>) key and then the Z key at the same time. The program echos $Z, carriage return and line feed, and then the CLI prompt (R).

Example

OEDIT FOO.SV cr

The octal editor responds with a carriage return, indicating that file **FOO.SV** is open for editing and the user can issue commands such as a request to open and display the contents of a given location.

POP

Purpose

Returns to the next higher level program in the program environment. Before the **POP** command is used, operation of the user program should have been temporarily suspended by pushing CLI into operation at the next lower program level via an **.EXEC** system command. The **POP** command then provides a means to return from the lower level so that CLI resumes operation in the user's program.

Example

A user program at level 1 is called **RESTORE.SV**. To suspend operation of **RESTORE.SV** so the CLI is usable for routine maintenance, **RESTORE** issues system command **.EXEC**, causing itself to be swapped to disk and CLI to be in operation at level 2. When maintenance operations are complete, **RESTORE** should be returned to operation. Using **CTRL A** and other means is inadequate since that does not return **RESTORE** at the point where it was suspended. The **RESTORE.SV** program is returned to core memory and its execution resumed at the point where it was interrupted with the following command:

POP cr

Restrictions

An attempt to issue a **POP** command from the top level of the last running RDOS ground is rejected with a CLI runtim error.

PRINT filename [filename etc.]

Purpose

Makes the line printer the output device and prints the contents of a given file or files.

This command is equivalent to a series of **XFER** commands.

The source files may come from any device. If a parity error is detected, a left slash (\) is printed in place of the bad character, the message **"PARITY ERROR"** is output to the console, and then printing continues.

Global Switches

/L   Allows files that were input with lower case text to be printed as lower case text.

Example

PRINT FOO.SR DC1:COM.SR $PTR cr

     Source file **FOO.SR** in the current directory, source file **COM.SR** in **DC1**, and the file on the paper tape reader are printed on the line printer.

Restrictions

For systems equipped with a smart spooler device, the **SPOOL** command is more efficient without using additional disk space.

PUNCH filename [filename etc.]

Purpose

Makes the high speed punch the output device and punches the contents of a given ASCII file or files. This command is equivalent to a series of **XFER** commands.

The source files may come from any device. If a parity error is detected, a left slash (\) is punched in place of the bad character, the message **"PARITY ERROR"** is output to the console and then punching continues.

Example

PUNCH DC0:A BETA $TTR cr

     File **A** in **DK0**, file **BETA** in the current directory, and a file on the teletypewriter reader are punched on the high speed punch.

RDEDIT [station/switch] [filename]

Purpose

Edits an ASCII file using a CALMA **station** as the system console. **RDEDIT** is the CALMA version of the NOVA text editor with many enhancements. It accepts commands from a station keyboard and outputs to the station CRT. Use of **RDEDIT** is described in Appendix A of this manual.

Local Switches

/L   Allow lower case input from keyboard (this does not effect **RDEDIT**s software lower case support).

/S   Accept input and display output at the station number where this switch is. The actual hardware codes in the CRT controller are used; so the station number entered must be one less than the CALMA station number. The default value for the station number is 0 when not running the CLI form a station, or for the current station when running the CLI from a station.

Examples

RDEDIT TEST.AL cr

     Edit the DGL source file, **TEST.AL**, at station 1.

RDEDIT 1/S cr

     Enter the editor at station 2 and wait further commands.

Restrictions

Two copies of the **RDEDIT** program exist; **ARDEDIT** for ECLIPSES and **NRDEDIT** for NOVAS. The name **RDEDIT** is linked to the correct version when a standard system is executed.

RELEASE logical unit specifier

Purpose
Prevents further I/O access to a previously initialized **logical unit specifier**. (See **INIT** command.) The command causes rewinding of tapes and is given for a disk before any disk pack is physically removed. All files in a directory must be closed before the directory is released.

If a directory other than a disk global specifier is the master directory, the directory is released before system shutdown. Releasing a master directory causes all initialized directories and multiple file devices to be released, and the system prints the message "MASTER DEVICE RELEASED". In a multiground environment, the master directory cannot be released from the foreground.

Example
RELEASE DC1 cr
    DC1 is released and the disk pack may be removed from disk drive unit 1. If **DC1** was the master directory the system prints "MASTER DEVICE RELEASED".

REPLACE savefilename

Purpose
Replaces one or more overlays in an existing overlay file **(savefilename.OL)** with one or more overlays in a replacement file **savefilename.OR** which is created by the overlay loader.

Example
REPLACE A cr
    Replace overlays in **A.OL** as specified in the previous **.OVLOD** system command that created **A.OR**.

REV filename [.SV]

Purpose
Displays the revision level of a save file. Save file **filename** must have been given the **S** attribute. The **.SV** extension appears optionally in the command format.

Revision levels are assigned in the **.REV** assembler pseudo-op and are given as a two-digit major level, followed by a period, followed by a two-digit minor level. For example, the revision level range is 00.00 to 99.99. If there is no **.REV** pseudo-op in the file, the level returned is 00.00.

Example
REV ATLC.SV cr
03.07
    User save file **ATLC** is at revision level 03.07.

Restrictions
An RDOS system save file cannot be given as an argument to the **REV** command.

RENAME oldname newname [oldname newname]

Purpose
Changes the name of a file or files. The old and new filenames can be different directories in the same partition.

Example
RENAME DC1:A DC1:ALPHA B BETA cr
    Rename file **A** in **DC1** to **ALPHA** and **RENAME B** in the current directory to **BETA**.

RLDR rbs [orb, etc. oct/s orb etc. rbs]

Purpose

Loads relocatable binary files (**rbs**) creating a save file with the default name **rb.SV**. A corresponding overlay file (**rb.OL**) is created if one or more relocatable binary files are specified as belonging to overlays (**orbs**). A left bracket ([) indicates the start of a new overlay area and **orbs** within the brackets are part of that overlay area. A comma delimits overlays within the overlay area. The argument list optionally includes a number of octal values whose meaning is specified by local switches. Argument **oct/s** is an octal number followed by a local switch.

Global Switches

/A	Produce alphabetic symbol table. (Must have local **/L** also.)
/C	Set **INMAX** to 4400 (RTOS/SOS compatible).
/D	Load a symbolic debugger.
/E	Output error messages to the console when a listing file (local **/L**) is specified.
/G	Perform special mode load.
/H	Output numerics in hexadecimal (radix 16).
/I	Start load at location zero.
/M	Suppress load map and all console output including error messages.
/N	Do not search **SYS.LB**.
/P	Print **NMAX** as each module is loaded.
/S	Leave symbol table in high memory.
/X	Allow up to 128 (decimal) system overlays. (See the Data General Relocatable Loaders Manual.)
/Y	Allow up to 256 (decimal) system overlays. (See Data General Relocatable Loaders Manual.)
/Z	Start save file at location zero (see Restrictions).

Local Switches

/C	Preceding octal value specifies the number of channels.
/E	Error message file.
/F	Preceding octal is foreground **NREL** partition address. (Loads when **/F** and **/Z** are not present.)
/K	Preceding octal is number of tasks.
/L	Listing file.
/N	Preceding octal is value **NMAX**.
/S	Save file and overlay file are to have the name specified preceding the switch.
/U	Load user symbols from the relocatable binary file preceding the switch.
/V	Create virtual overlay file for use in extended address space.
/Z	Preceding octal values specifies the forground **ZREL** partition address. (Loads are to background when **/F** and **/Z** are not given.)

Examples

RLDR RO [A,B,C] R1 R2 [E,F,G,H,D] cr

Create save file **RO.SV** containing **RO, R1** and **R2** in core image form and two overlay areas. One overlay area contains either **A, B,** or **C** and the other contains either **E, F G H,** or **D**.

RLDR 200016/F 300/Z FILE1 FILE2 cr

Create save file **FILE1.SV** containing **FILE1** and **FILE2** in core image form. Set foreground **ZREL** partition to 300 and **NREL** partition to 20016.

Restrictions

A save file produce by using a **/Z** switch will not execute under RDOS. It is used to enable loading of routines that have page zero locations 0 thru 15. The save file is then output using the **MKABS/Z** command to produce an absolute binary file to be read in the stand-alone made using the binary loader or HIPBOOT. Refer to Section IV of this manual.

By default, the first input file name is used with an **.SV** extension to form the output file name, and is also used with an **.OL** extension to form an overlay file name. User symbols are not loaded by default.

On input, the system searches for the name **rbs.RB**, and if not found, searches for **rbs**. On output, the output file name is **rbs.SV** by default, or the file name preceding the local switch **/S** with the **.SV** extension.

For multitask programming, the **/C** and **/K** local switches or the **.COMM TASK** system command must be used to avoid a fatal load error with the following message: **LOAD OVERWRITE 17**.

SAVE filename

Purpose

Creates a save file named **filename.SV** from the contents of the file **BREAK.SV** (or **FBREAK.SV** in the foreground). **SAVE** is commonly used to save the core image of a file interrupted by a **CTRL C** break or by the debugger **$V** command.

The **SAVE** command causes the most recent core image saved under the filename **BREAK.SV** or **FBREAK.SV** to be given the new name **filename.SV**. A **.SV** extension is always given to the new filename even if it had no extension or a different extension when given.

Example
DEB ALPHA cr
PP/LDA 2 @ 0 LDA 2@03
$V cr
BREAK
R

SAVE ALPHA cr
R

In this example the debugger was called to correct location **PP** in file name **ALPHA**. Debugger command **$V** exits the debugger routine and creates a **BREAK.SV** file. The **SAVE ALPHA** command then renames the **BREAK.SV** file to **ALPHA.SV**,

SDAY month day year

Purpose

Sets the system calendar. The year is given as either two or four digits. (For example, 75 or 1975.)

Example
SDAY 4 17 75 cr

Set the system calendar to April 17, 1975.

Restrictions

The usual command delimiters (space and comma) are used between parameters, not colons as in bootstrapping.

SEDIT filename

Purpose

Invokes the save file editor to examine and modify locations of **filename**. The save file editor has a complex syntax that follows the RDOS Debugger program. Refer to the Data General Save File Editor Manual for more details.

To exit **SEDIT**, press the escape (<ESC>) key and then the Z key at the same time. The program echos $Z.

SMEM fg bg bg2 bg3 etc.

Purpose

Allocates the number of 1024_{10} word blocks of core memory available to each of the grounds in a mapped system. When the system is initially booted, all memory blocks are allocated in the main background (**BG**). To divide available memory between all of the selected grounds (see the **SYSGEN** command), the **SMEM** command parameters are given in the number of 1024_{10} word blocks assigned to each ground. The parameters are given sequentially so that the foreground is first and followed by a space and the main background. Backgrounds 2, 3, 4 etc. then follow sequentially and separated by spaces.

Example
SMEM 8 12 9 4 cr

This command allocates eight (1024_{10} word) blocks to the foreground, 12 blocks to the main background, 9 blocks to background 2, and 4 blocks to background 3.

Restrictions

This command is issued from the CLI in mapped system only.

SPDIS device name [device name etc.]

Purpose
Disables standard spooling on one or more spoolable devices. The device name is any user device defined as spoolable. Any of the following devices are spoolable: **$DC0, $LPT, $LPT1, $PLT, $PLT1, $PTP, $PTP1, $TTO, $TTO1, $TTP, $TTP1** (for device names, refer to Table 1-8). Other standard spooler commands are **SPEBL** and **SPKILL**.

Example
SPDIS $PLT $PTP cr
   Disable spooling to the plotter and the paper tape punch. If output is currently being spooled to either device, the command takes effect after the current spool is completed. To reenable spooling, the command **SPEBL** is used.

Restrictions
SPDIS to the line printer (**$LPT**) is not applicable if the Smart Spooler is in the system. See **LPT** command for Smart Spooler control.

SPEED [filename]
NSPEED [filename]

Purpose
Edits ASCII text using the **SUPEREDIT** program which has multi-buffer editing, multiple I/O files, macro programming and numeric variables.

The command **SPEED** is used for the **SUPEREDIT** program on ECLIPSE computers and the command **NSPEED** is used for the **SUPEREDIT** program on NOVA computers. Refer to the Data General Corporation **SUPEREDIT** Manual, Ordering Number 093-000111-00.

Example
SPEED cr
!   **SUPEREDIT** prompt is !.
.   User may now issue a series of
.   **SUPEREDIT** commands.
.
!H$$
   User terminates **SUPEREDIT** and returns to CLI. The **$$** is echoed when the escape key (**ESC**) is pressed twice.

SPEBL device name [device name etc.]

Purpose
Enables standard spooling on any spoolable device. Refer to the **SPDIS** command for a list of spoolable devices. The other standard spooler command is **SPKILL**.

Example
SPEBL $LPT cr
  data output ot the line printer is spooled.

Restrictions
SPEBL to the line printer (**$LPT**) is not applicable if the Smart Spooler is in the system. See **LPT** command for Smart Spooler control.

SPKILL device name [device name etc.]

Purpose
Stops a standard spooling operation on one or more spoolable devices. Refer to the **SPDIS** command for a list of spoolable devices. The other standard spooler command is **SPEBL**.

Example
SPKILL $LPT cr
  Spooling of data to the line printer is stopped, causing it to output in unbuffered form and making the output terminatable by a **CTRL A** if desired.

Restrictions
SPKILL to the line printer (**$LPT**) is applicable if the Smart Spooler is not in the system. See **LPT** command for Smart Spooler control.

SPOOL [/switches] [num/C] filename

Purpose
Queues a disk file for direct output by the RDOS Smart Line Printer Spooler. Smart Spooler commands are given under the LPT command. A description of the Smart Spooler is given in Paragraph 2.3.10. The other Smart Spooler commands under CLI are SPOOLQ and UNSPOOL.

Global Switches
/D Delete the file after output is complete.
/H Do not print the heading at the beginning of the output.
/L Print upper and lower case.
/N Do not put line feeds after carriage returns.

Local Switches
num/C Specifies the number of copies to be spooled (default is one).

Example
SPOOL JOHN cr
SPOOLED : DC0:JOHN
 the file JOHN, residing in DC0, is enqueued for LPT output by SPOOL which is also running in DC0.

Restrictions
The directory containing the file to be spooled must remain initiated until the output is complete. Also, the file is not modified or .EOPEN'ed until the spooling is complete.

SPOOLQ

Purpose
Displays the current RDOS Smart Spooler output queue. Other Smart Spooler commands are LPT, SPOOL, and UNSPOOL. A description of the Smart Spooler is given in Paragraph 2.3.10.

Example
SPOOLQ cr
 The current smart spooler LPT queue is displayed on the console. The output consists of the following information:

Q	Position in queue (1 is first or being printed.)
N	Number of copies.
A	Active (asterisk (*) if job is currently in output.
D	Delete flag.
TIME	gives time the job was enqueued.
FILENAME	Gives name of file to be printed, including directory specifiers.
DIR	The current (default) directory that queued the entry.
PROG	The program name that queued the entry.
GRND	The ground name that queued the entry.

STOD hour minute second

Purpose
To set the 24-hour system clock.

Example
STOD 21 47 20 cr
The system clock is set to 9:47:20 P.M.

Restrictions
The usual command argument delimiters (space and comma) are used, not colon as in bootstrapping.

SYSDPY

Purpose
Generates a continuous system status display on a Hazeltine QTY unit. The display is similar to **SYSTAT**. There are no arguments.

SYSGEN [filename/option etc.]
BSYSGEN [filename/option etc.]
MSYSGEN [filename/option etc.]

Purpose
Generates a new RDOS system. **SYSGEN** is used for NOVA, **BSYSGEN** for ECLIPSE without multiground, and **MSYSGEN** with multiground. Switches allow a number of variations in handling system generation.

The default name of the generated system file is **SYS000.SV**. The generated system is automatically loaded by default. The **SYSGEN** dialog and any load errors are output by default on the console. Indirect command files (**SRLDR.CM**, etc.) are generated and then deleted by **SYSGEN** automatically. Typical system generation dialogs are given in Section IV of this manual.

Global Switch
/D   Load with system debugger.
/N   Do not perform automatic system load.

Local Switches
/A   Take dialog from the given dialog file instead of prompting at the console. The preceding filename appeared in a **filename/V** argument to a previous **SYSGEN** command.
/L   Output load map to specified filename instead of console.
/S   Generate system with specified filename.
/T   Search for tuning file (**filename** or **filename.TU**) and use its contents to override previously recorded **SYSGEN** responses.
/V   Save the **SYSGEN** dialog for the system being generated in filename.

Examples
SYSGEN cr
    The system generation program is activated and asks a series of questions at the console to which the user responds. When dialog is completed, **SYSGEN** invokes **RLDR** to produce an executable version of RDOS, named **SYS000**.
SYSGEN OSYS1/V $LPT/L cr
    System generation proceeds as in the previous example except that the dialog is saved in **OSYS1** and the load map is output to the line printer.
SYSGEN OSYS1/T cr
    Previously created **OSYS1** is used to generate new system. The tuning file is used to update responses in **OSYS1** wherever appropriate.
SYSGEN OSYS1/S/A cr
    Previously created **OSYS1** is used to generate new system, named **OSYS1**.
MSYSGEN ECLIPSE.DI/A DC0:ECLIPSE/S ↑
ECLIPSE.MP/L cr
    Generate Multiground ECLIPSE system into **DC0:ECLIPSE** using dialog file **ECLIPSE.DI** and put map into **ECLIPSE.MP**.

SYSTAT

Purpose
Determines the status of all grounds in the system.

Example
SYSTAT

RESPONSE

Q	GRD	LVL	PRI	PROGRAM	TOT MEM	USE MEM	NO UFT	IN CONS	OUT CONS	DIR DCB	SWAP TIME
	FG	5	1	*INACTIVE*	0	0	25	$TTI1	$TTO1		
W	BG	1	2	CLI	12	8	25	$TTI1	$TTO1	BILL	
W	BG2	1	2	CLI	31	8	24	QTY:0	QTY:0	DC0	
W	BG3	2	2	FSR	31	31	24	QTY:3	QTY:3	DC0	
*A	BG4	2	2	SYSTAT	24	24	24	QTY:2	QTY:2	SGDS	
	BG5	1	2	*INACTIVE*	0	0	25				
	BG6	1	2	*INACTIVE*	0	0	25				
	BG7	1	2	*INACTIVE*	0	0	25				0
	BG8	1	2	*INACTIVE*	0	0	25				0
	BG9	1	2	*INACTIVE*	0	0	25				0

The Q column indicates whether the ground is currently active or on the wait queue.

The * indicates the current ground was **BG4** running the program **SYSTAT**.

TPRINT filename

Purpose

Prints tuning file specified by **filename**. Tuning files are not deleted by the **TUOFF** command, so there may be more than one tuning file. One for the current system and one for each previously booted system if tuning file recording was initiated in a **TUON** command. Filename is the name of the tuning file that is to be reported. It has the same name with the **.TV** extension as the system that was geing monitored by the tuning feature. Tuning file information is printed at the console by default and all information except that on overlays is printed. The tuning file contains the following system facilities:

* buffer number,   number of requests and number of failures.
* stack number   number of requests and number of failures.
* number of system   number of requests and number
  overlay requests   of failures.
* overlay frequency
  report

Global Switches

/L   Print the tuning file on the line printer.
/O   Print the overlay frequency report.

Example
TPRINT/L BSYS cr
   Print tuning file **BSYS** on the line printer.

TUOFF

Purpose

Stops recording of system facilities in the tuning file. The tuning file is not deleted.

Example
TUOFF cr
   Recording in the tuning file is inhibited.

TUON

Purpose

Initiates recording in the tuning file. The tuning file records the number of system requests and system failures for:

* Buffer number
* Stack number
* System overlay request number

The file has the name of the currently executing operating system with the **.TU** extension.

Example
TUON cr
   If the current system is **BSYS1**, then the tuning file is opened for recording as **BSYS1.TU**.

3-53

TYPE filename [filename etc.]

Purpose

Makes the system console the output device and types the contents of a given ASCII file or files.

The source files may come from any device. When a character with bad parity is detected, a left slash (\) is typed, the message "PARITY ERROR" is output, and the output of the file continues.

Example

TYPE A.SR B.SR DC1:XX.SR cr

Files **A.SR** and **B.SR** from the current directory and file **XX.SR** from **DC1** are typed or displayed on the console.

UNLINK linkfilename [linkfilename etc.]

Purpose

Deletes link entires form a directory. The resolution file is not affected. The dash and asterisk convention is used only when **linkfilename** is in the current directory.

Global Switches

/C   Repeat each filename and wait for confirmation that the file is ready to be unlinked. A carriage return deletes the link entry; any other key retains the link entry.
/L   List deleted file on **$LPT** (overrides /V).
/V   Verify deletions with a list of deleted link entries output to the console.

Local Switches

/N   Do not delete links matching this name.

Examples

UNLINK/V MYLK JOELK TESTLK.- cr

The deleted link entries are listed on the console (/V switch).

UNLINK/C -.SV -.LB cr

Repeat each filename in the current directory having an **.SV** or **.LB** extension and wait for a carriage return before unlinking the entry. If a cr is typed after a filename, the system echoes an asterisk (*), unlinks the file, and prints the next filename if any. Typing any other response after a file name leaves the file linked and does not echo.

UNSPOOL [/switch] filename †
[*ground id/G] [bb:mm/B] [bb:mm/A] [directory/D]

Purpose

Selectively removes files that are queued to the RDOS Smart Spooler while spooling is in process. This is done by exactly match a filename, or selecting the most recent file that meets qualifiers of ground, time, or default directory.

Global Switches

/A   Unspool all files that match specifiers, not just the most recent one.
/N   Do not delete the file that is unspooled (only applies if the delete switch was specified when the file was spooled). The file currently in output cannot be inspooled without first deleting.

Local Switches

/A   Indicates that only the files spooled after the time hh:mm are unspooled.
/B   Indicates that only the files spooled before the time hh:mm are unspooled.
/D   Indicates that only the files in the given directory are unspooled.
/G   Indicates that only the files in the given ground are unspooled.

Arguments

UNSPOOL has two basic forms: first an explicit, completely qualified filename may be given in which case it is searched for and unspooled if found in the Smart Spooler queue; secondly, one or more qualifiers may be given and the most recent file found that satifies all the qualifiers is unspooled (all files that satisfy are unspooled if global /A). The qualifiers are the local switches. The asterisk (*) is a special form of the Ground qualifier that requires no /G local switch since it indicates that spool entries from the calling ground only are considered and this form is the one most often used.

Other Smart Spooler commands are **LPT, SPOOL,** and **SPOOLQ.** A description of the Smart Spooler is given in Paragraph 2.3.10.

Examples

UNSPOOL TOM/D cr

Unspool the file most recently queued from directory **TOM.**

UNSPOOL * cr

Unspool the file most recently queued from the current Ground.

UNSPOOL/A * cr

Unspool all files queued from the current Ground.

UNSPOOL/N SGDS:TEMP.SR cr

Unspool the file **TEMP.SR** from Directory **SGDS** and do not delete it.

VERIFY filename [filename etc.]

Purpose
Verifies command syntax and whether or not files exist. It is usually inserted at the beginning of CLI user written macros to ensure that all files exist. If any file is not found, the **VERIFY** command aborts the CLI macro, resets CLI, and outputs error message; FILE DOES NOT EXIST, filename. Normally, **VERIFY** is not used outside of CLI user written macros, but may be used as shown in the following example.

Example
BUILD ABFILE A-.- B-.- cr
VERIFY @ABFILE@ cr
   The first command builds **ABFILE** containing the names of all files beginning with letters **A** and **B** and having extensions. The **VERIFY** command checks whether or not the contents of file **ABFILE** exist in the proper syntax for CLI.

XFER sourcefilename destinationfilename

Purpose
Transfer contents of **sourcefilename** to **destination-filename**, organizing the destination file differently if desired. By default, the destination file is sequentially organized and the contents of the source file are copied to the destination file in precisely the same format as in the source file.

Global Switches
/A   Perform an ASCII transfer
/B   Append the source file to the destination file.

Local Switches
/C   Organize the destination file contiguously. (The source file must be on a disk if a contiguous destination file is to be produced.)
/R   Organize the destination file randomly.

Examples
XFER/A DC0:FYFILE DC1:MYFILE/C cr
   Transfer **MYFILE** in **DC0** to **MYFILE** in **DC1** and organize contiguously.
XFER/A ALPHA.SR $LPT cr
   Print **ALPHA.SR** in the current directory on the line printer.

SECTION IV

RDOS INITIALIZATION
AND BOOTSTRAPPING

4.1 GENERAL

This section contains procedures and interactive dialogs used to initialize a disk pack and boot the RDOS system in preparation for initial use or to create a new disk, to save the disk contents on magnetic tape, and to generate a system tailored to specific hardware. Information in this section includes the following:

Input Corrections
Disk Initializing and Bootstrapping
Creating and Loading BACKUP Tapes
RDOS System Generation

The procedures and interactive dialogs in this section have typical responses that may be used to operate and understand the system. Alternate responses are also given so that the programmer may alter the responses to fit any system. Before these procedures are used the programmer should know the hardware configuration including the available core memory and the type and number of device controllers.

4.2 INPUT CORRECTIONS

The procedures in this section contain interactive dialogs where the system prints a message and pauses until the programmer (user) types a response and, in most cases, presses the RETURN key. If errors occur while typing responses, the errors should be corrected before the RETURN key is pressed. The method of correcting these errors depends upon the type of input terminal being used as follows:

1. For Teletypes, each time the RUB OUT key is pressed, a left arrow (←) is echoed to indicate a previous character was deleted. Three left arrows indicate that the preceding three characters are deleted. The desired characters are then typed before the RETURN key is pressed.

2. For Decwriters, when the BACK SPACE key is pressed, the carriage moves one space to the left. Typing the desired characters then overstrikes previously typed characters and replaces them with the characters just typed.

3. For Hazeltine alphanumeric CRT terminals, pressing the BACK SPACE key erases the previous character and moves the cursor one space to the left.

4. For Tektronix storage CRT terminals, pressing the BACK SPACE key causes the previous character to be echoed and underlined to indicate that the character is deleted. New characters are then typed on the same line.

5. For all terminals, typing the reverse slash or Control L deletes all characters typed by the user on the current line. The reverse slash is echoed by a carriage return.

For most cases, illegal responses to the system query cause the query to be repeated until a legal response is entered. When applicable, a range of acceptable responses is given with the query. The user then types the desired response and presses the RETURN key. Default parameters are selected by just pressing the RETURN key.

Pressing the CTRL and A keys at the same time (CTRL A) normally aborts input collection or command execution and causes the system to do a carriage return and print **INT** on the system console to indicate an interrupt has occurred. The procedure being performed is then restarted from the beginning. During disk initialization only, pressing CTRL A causes the previous statement to be reprinted.

Pressing the CTRL and C keys at the same time (CTRL C) does the same as CTRL A except that a **BREAK.SV** file is created to store the current memory contents on disk.

All Teletype terminals have a time-out feature that turns off the motor after several minutes without activity. If a timeout occurs, press the CRTL and L keys at the same time to turn the motor on before typing.

4.3 DISK BOOTSTRAPPING AND INITIALIZING

The disk is booted and initialized from two magnetic tape programs that are supplied with the system. These programs may be supplied on one or more physical tapes. Bootstrapping and initializing are usually done with an RDOS BOOTTAPE. The interactive dialogs used for both bootstrapping and initializing a disk pack are described in the following procedures. These procedures are performed in sequence.

4.3.1 Disk Initializer Dialog

The following procedure describes how magnetic tape files **TBOOT.SV** and **.DKINIT.SV** are used to start and to initialize the disk. New or non-RDOS disks should be fully initialized before loading an RDOS program. Partial initialization is also available to check for bad disk blocks (one block contains 256_{10} words) and is shown at the end of this procedure. This procedure is applicable to a CENTURY or CDC disk drive. However, the CENTURY disk drive was used in the examples. See Figure 4-1 for a copy of a typical disk initializer dialog.

1. Turn on system power and check that all units of the system are receiving power.

2. Check that the system console (Teletype, Decwriter, etc.) and line printer (optional) are on line.

3. Lift and release the computer RESET/STOP switch to halt the computer.

4. Install a disk pack on disk drive unit 0, turn on the disk drive, and wait for the disk to get up to speed.

5. Mount and thread the magnetic tape labeled RDOS BOOTTAPE on tape drive 0. Be sure the write-enable ring is removed.

6. Set the tape drive for power on, load, and on line. If the system has a TAPE DENSITY switch, the switch setting must match the tape being used.

NOTE

Do not perform steps 7 and 8 for NOVA computers. Boot instructions for NOVA computers are provided in Paragraph 4.3.2.

7. Set the computer switch register to 100022_8 (switches 0, 11, and 14 up and all others down) to boot from magnetic tape.

8. Lift and release the computer RESET/STOP switch and then the PR LOAD/EXEC switch. The system reads **TBOOT.SV** file from magnetic tape, executes the **TBOOT.SV** program, and then prints **MTO:**.

NOTE

All responses to **MTO** refer to the list in step 9. For example, a 3 response calls the tape file with the **DKINIT.SV** Program. Different tapes have different tape file numbers for these program so the user must refer to the list supplied with the applicable BOOTTAPE program tape.

9. Type the appropriate response for the **DKINIT.SV** program (on the same line as the **MTO:** print out) and then depress the RETURN key. A typical list of responses is given below, but for accuracy, refer to the applicable BOOTTAPE program list.

MAG TAPE UNIT:FILE	PROGRAM NAME	DESCRIPTION
MTO:0	**TBOOT.SV**	Tape Bootstrap Program
MTO:1	**CLI**	CLI, BOOT, and LOGON in a dump file
MTO:2	**BOOT.SV**	RDOS Disk Bootstrap
MTO:3	**DKINIT.SV**	RDOS Disk Initializer
MTO:4	**24KNOVA.SV**	24K Century Disk System
MTO:6	**ECLIPSE.SV**	Mapped Eclipse Century System

The system then reads the selected tape file and prints:

```
CALMA DISK INITIALIZER - REV xx.xx
DISK DRIVE MODEL NUMBER?
```

The xx.xx indicates the revision level.

10. Type **CENTURY** or **CDC** (depending on type of disk drive) and press the RETURN key. The system then prints:

```
CENTURY 114 DISKPACK DRIVE TYPE
(for Century disks)
DISK UNIT?
```

NOTE

Typing an illegal response to any **DKINIT.SV** query causes an error message or a repeat query to print out.

11. Type **DC0** for a Century disk unit or **BD0** for a CDC disk unit and press the RETURN key. The system then prints:

    ENTER INTERLEAVE FACTOR (<CR> FOR 1)?

12. Press the RETURN key to enter the default interleave factor (1). The system then prints:

    COMMAND?

13. Type **FULL** and press the RETURN key. The system then prints:

    COMMAND DESTROYS ANY PREVIOUS
    RDOS DISK STRUCTURE
    RDOS INIT/F MUST BE DONE ON DISK
    AFTER COMMAND
    TYPE CONTROL-A NOW TO ABORT
    WITHOUT LOSS

    NUMBER OF PATTERNS TO RUN (1-5)?

NOTE
This command checks the disk quality by writing and then reading number patterns over the entire disk which destroys any data that previously existed on the disk. To save the data, depress the CONTROL and A keys at the same time, then change the disk before continuing.

14. Type a number from 1 thru 5 and press the RETURN key to select the number of patterns to be run. A 1 response runs only the first pattern, a 2 response runs the first two patterns, etc. Each pattern requires approximately 6 minutes to run on a century disk, 10 minutes to run on a 80MB CDC disk, and 40 minutes to run on a 300MB CDC disk. The time starts when the system prints:

    PATTERN #x (yyyyyy)

The x indicates the pattern number being run and yyyyyy is the pattern as follows:

PATTERN#	PATTERN
1	125252
2	052525
3	155555
4	177777
5	000000

After the patterns are run, the system prints one of the following messages:

MESSAGES	ACTION REQUIRED
ILLEGAL NUMBER OF PATTERNS	Type a response between 1 and 5.
CRITICAL DISK BLOCKS ARE BAD RDOS CANNOT BE BUILT, ABORTING	Reformat the disk using Data General disk formatter program Part No. 095-000071 or 095-000241 as applicable.
TOO MANY DISK ERRORS TO COMPLETE	
ALL PATTERNS RUN	Normal return after all selected patterns are run.

DO YOU WISH TO DECLARE ANY
BLOCKS BAD THAT ARE NOT
ALREADY IN THE BAD BLOCK
TABLE?

15. Type NO and press the RETURN key to advance the program. If the RETURN key is pressed before a response is typed, the system prints:

    ANSWER YES OR NO and repeats the previous message.

    When YES is typed, the system prints:

    BAD BLOCK NUMBER

    Type the number of the bad block and press the RETURN key. The system then prints:

    BAD BLOCK ENTERED
    BAD BLOCK NUMBER

This exchange continues for each block number entered. To exit from this routine, press the RETURN key before entering a new number. The system then prints:

> DO YOU WISH TO DECLARE ANY
> MORE BLOCKS BAD?

If YES is typed, the routine continues as before. When NO is typed and the RETURN key is pressed, the program advances, and the system prints:

> DEFAULT REMAP AREA SIZE IS xx
> BLOCK(S) LONG
> IT NEEDS TO BE AT LEAST yy
> BLOCK(S) LONG
>
> REMAP AREA SIZE (TYPE RETURN FOR
> DEFAULT)?

The xx and yy vary depending on the type of system disk in use.

16. Press the RETURN key to select the default remap parameters and advance the program. To change the remap area size, type the new length and press the RETURN key. The system then prints:

> REMAP AREA START BLOCK NUMBER
> (TYPE RETURN FOR DEFAULT)?

17. Type one of the following three responses:
    a. Press the RETURN key to select the default location.
    b. Type the physical block number (octal) and then press the RETURN key.
    c. Type the head, sector, and cylinder locations (separated by commas) and then press the RETURN key.

After the RETURN key is pressed, the system prints one of the following messages. Determine which message is printed and perform the action required.

MESSAGES	ACTION REQUIRED
THERE IS NO CURRENT REMAP AREA, RUN FULL INIT TO ESTABLISH ONE	Type **FULL** in response to **COMMAND**?
ILLEGAL DISK BLOCK NUMBER	Retype entry for remap area start block number.
RDOS WILL NOT RUN WITH THIS BLOCK BAD, BLOCK NOT ACCEPTED	
NO MORE ROOM FOR BAD BLOCKS IN REMAP AREA DUE TO TOO MANY BAD BLOCKS- ABORTING	Reformat disk using Data General disk formatter program Part No. 095-000017 or 095-000124 as applicable.
BAD BLOCK LIST IS FULL, UNABLE TO ENTER ANY MORE BAD BLOCKS	Remap again with larger remap area.
ILLEGAL REMAP AREA START BLOCK NUMBER	Select another area for remap.
BAD BLOCK CONTAINED IN REMAP AREA SPECIFIED, PLEASE SPECIFY ANOTHER AREA	Select an area with contiguous good blocks.
DEFAULT FRAME SIZE IS 83 MIN IS 1, MAX IS 4060	Normal program advance. Printout indicates the default frame size and limits. Default and maximum size vary with disk type.
DISK FRAME SIZE (TYPE RETURN FOR DEFAULT)?	Query to select new frame size, proceed to step 17.

NOTE

To repeat a previous query, press the **CTRL** and **A** keys at the same time. This allows another remap area to be selected. To reformat the disk, if required, use the Data General disk formatter program, Part No. 095-000071 or 095-000241 as applicable.

18. Press the RETURN key to accept the default frame size and limits. Frame size determines the number of disk blocks assigned to the RDOS system directory (**SYS.DR**). RDOS runs more efficiently when all file names for a directory fit within one frame.

To select a different frame size than the default parameters, type any number from 1 thru 4060 (disk blocks) and then press the RETURN key. If the default parameters were not selected and the number typed was not from 1 thru 4060, the system prints:

ILLEGAL FRAME SIZE

and a different selection must be typed.

When a legal selection is made, the system indicates the disk is fully initialized by printing:

FULL DISK INIT COMPLETE
COMMAND?

19. Type STOP and press the RETURN key to terminate the **DKINIT** routine. The system then does a carriage return and halts the computer. In addition to STOP, several alternate **DKINIT** commands can be typed in response to COMMAND?. These commands are described in the following list and typical responses are shown in Figure 4-2.

ALTERNATE COMMANDS	DESCRIPTION
DISK	Restarts the **DKINIT** program at the beginning (step 10).
ENTER	Enters bad blocks in the bad block accounting table.
FRAME	Allows selection of a different frame size. The message "COMMAND DESTROYS etc." prints out but is automatically bypassed without user response.
LIST	Lists the disk type and block accounting information.
PARTIAL	Provides partial disk intialization by checking for bad blocks without over writing existing data.
REMAP	Allows selection of a different remap area. The message "COMMAND DESTROYS etc." prints out but is automatically bypassed without user response.

20. Bootstrap the disk and start the operating system by loading the tape labeled BACKUP TAPE SYSGEN.DR following the procedures given in Paragraph 4.4.2.

4.3.2 NOVA Computer Bootstrap Procedure

This procedure is provided for systems with NOVA computers that are not equipped with power fail/automatic restart feature and provides alternate bootstrap procedures. Do not perform these procedures unless the system is equipped with a NOVA computer and the procedure is referenced from the preceding text.

NOTE

Switches used in this procedure are located on the front of the NOVA computer.

1. Set all of the switch register switches down and then press and release the RESET switch.

2. Set the switch register to 000376_8 (switches 8 thru 14 up and all others down).

3. Press and release the EXAMINE switch.

4. Set the switch register to 060122_8 (switches 1, 2, 9, 11, and 14 up and all other switches down) to boot from magnetic tape.

5. Press and release the DEPOSIT switch.

6. Set the switch register to 000377_8 (switches 8 thru 15 up and all other switches down).

7. Press and release the DEPOSIT NEXT switch.

8. Set the switch register to 000376_8 (switches 8 thru 14 up and all other switches down).

9. Press and release the RESET switch and then the START switch.

10. Return to the referencing procedure.

FROM MT0:**3**

CALMA DISK INITIALIZER - REV 03.00

DISK DRIVE MODEL NUMBER? CENTURY

CENTURY 114 DISKPACK DRIVE TYPE

DISK UNIT? **DC0**

ENTER INTERLEAVE FACTOR (<CR> FOR 1)? **cr**

COMMAND? **FULL**

COMMAND DESTROYS ANY PREVIOUS RDOS DISK STRUCTURE
RDOS INIT/F MUST BE DONE ON DISK AFTER COMMAND
TYPE CONTROL-A NOW TO ABORT WITHOUT LOSS

NUMBER OF PATTERNS TO RUN (1-5) ? **2**

*** PATTERN # 1 (125252) ***

*** PATTERN # 2 (052525) ***

*** ALL PATTERNS RUN ***

ANSWER YES OR NO

DO YOU WISH TO DECLARE ANY BLOCKS BAD
THAT ARE NOT ALREADY IN THE BAD BLOCK TABLE? **NO**

DEFAULT REMAP AREA SIZE IS 12 BLOCK(S) LONG
IT NEEDS TO BE AT LEAST 0 BLOCK(S) LONG

REMAP AREA SIZE (TYPE RETURN FOR DEFAULT) ? **cr**

REMAP AREA START BLOCK NUMBER (TYPE RETURN FOR DEFAULT) ? **cr**

DEFAULT FRAME SIZE IS 83,
MIN IS 1, AND MAX IS 4060

DISK FRAME SIZE (TYPE RETURN FOR DEFAULT) ? **cr**

FULL DISK INIT COMPLETE

COMMAND? **STOP**

Figure 4-1. Typical Disk Initializer Dialog

COMMAND? **ENTER**

BAD BLOCK NUMBER (TYPE RETURN TO STOP) ? **133420**
BAD BLOCK ENTERED

BAD BLOCK LIST IS FULL
UNABLE TO ENTER ANY MORE BAD BLOCKS

COMMAND? **LIST**

CENTURY 114 DISKPACK DISK DRIVE ON UNIT DC0

DISK CONTAINS 137120 BLOCKS
FRAME SIZE = 83   REMAP AREA SIZE = 12
REMAP AREA START BLOCK NUMBER = 137104
NUMBER OF BAD BLOCKS = 1

  HEAD   SECTOR   CYLINDER / BAD BLOCK NUMBER
  000005   000000   000303   / 133414

COMMAND? **PARTIAL**

*** CHECKING FOR BAD BLOCKS ***

NO NEW ERRORS DETECTED ON DISK

PARTIAL INIT RUN COMPLETE

COMMAND? **DISK**

CALMA DISK INITIALIZER - REV 03.00

DISK DRIVE MODEL NUMBER? **CENTURY**

CENTURY 114 DISKPACK DRIVE TYPE

DISK UNIT? **DC0**

**Figure 4-2.   Typical Responses to Alternate
DKINIT   Commands (Sheet 1 of 2)**

COMMAND? **FRAME**

COMMAND DESTROYS ANY PREVIOUS RDOS DISK STRUCTURE
RDOS INIT/F MUST BE DONE ON DISK AFTER COMMAND
TYPE CONTROL-A NOW TO ABORT WITHOUT LOSS

DEFAULT FRAME SIZE IS 83,
MIN IS 1, AND MAX IS 4060

NOTES

See sheet 1 of 2.

DISK FRAME SIZE (TYPE RETURN FOR DEFAULT) ? **cr**

COMMAND? **REMAP**

COMMAND DESTROYS ANY PREVIOUS RDOS DISK STRUCTURE
RDOS INIT/F MUST BE DONE ON DISK AFTER COMMAND
TYPE CONTROL-A NOW TO ABORT WITHOUT LOSS

DO YOU WISH TO DECLARE ANY BLOCKS BAD
THAT ARE NOT ALREADY IN THE BAD BLOCK TABLE? **NO**

DEFAULT REMAP AREA SIZE IS 12 BLOCK(S) LONG
IT NEEDS TO BE AT LEAST 0 BLOCK(S) LONG

REMAP AREA SIZE (TYPE RETURN FOR DEFAULT) ? **cr**

REMAP AREA START BLOCK NUMBER (TYPE RETURN FOR DEFAULT) ? **cr**

COMMAND? **STOP**

**Figure 4-2. Typical Responses to Alternate
DKINIT Commands (Sheet 2 of 2)**

4.4 CREATING AND LOADING BACKUP TAPES

A BACKUP tape is a magnetic tape containing an RDOS disk image. Whenever the disk contains information that is difficult or time consuming to recreate, a BACKUP tape should be made. This BACKUP tape is then loaded to reestablish system operation in the event of a head crash or other disk failure. Separate procedures are provided for creating and loading BACKUP tapes.

4.4.1 Creating a BACKUP Tape

A BACKUP magnetic tape is created any time the disk holds data that is critical to operation of the system. It is suggested that at the end of each working day, that a BACKUP tape be created to preserve the data in the event of a disk failure. This procedure creates a complete copy of the disk sector-by-sector without interpretation or validation of the contents. Any working disk can be dumped onto a magnetic tape to create a BACKUP tape. A typical dialog is shown in Figure 4-3.

NOTE

If the system is booted and running with the CLI prompt (R) as the last printout, proceed to step 9.

1. Install the disk pack to be copied on disk drive unit 0, turn on the disk drive, and wait for the disk to get up to speed.

2. Mount and thread a new or reuseable magnetic tape, with the write-enable ring installed, on tape drive unit 0.

3. Set the tape drive for power on, load, and on line.

NOTE

Steps 4 thru 6 boot an ECLIPSE computer. To boot a NOVA computer, skip to step 13.

4. Set the computer switch register to 100020₈ (switches 0 and 11 up and all others down) to boot from the CENTURY disk. To boot from the CDC disk, set the switch register to 100026₈ (switches 0, 11, 13, and 18 up and all others down).

5. Lift and release the computer RESET/STOP switch and then the PR LOAD/EXEC switch. The system prints:

    FILENAME?

6. Type the applicable file name such as ECLIPSE, ZECLIPSE, 32KNOVA, GDS II, etc. and then press the RETURN key. The system then prints:

    MAPPED ECLIPSE MULTIGROUND RDOS REV x.xx.xx DATE(M/D/Y)?

    The file name ECLIPSE was used and x.xx.xx indicates the current RDOS revision.

7. Type the actual date. For example, 05 11 78 is May 11, 1978. Press the RETURN key. The system then prints:

    TIME(H:M:S)?

8. Type the actual time based on a 24-hour clock. For example, 7 51 00 is 7:51 AM while 19 51 00 is 7:51 PM. Press the RETURN key. The system then prints:

    CALMA REVx CLI   (where x is the revision level)
    R   (CLI command prompt)

9. Type **GDIR** and press the RETURN key to get the name of the current directory. The system prints:

    DC0   (or **BD0** for CDC disk)
    R

    If **DC0** is not the current directory, type **DIR DC0** (or **DIR BD0** for CDC disk) and press the RETURN key. Continue when the system prints an R.

10. Type BACKUP and press the RETURN key to copy the disk image onto magnetic tape. The system prints the following message and then performs the copy operation. The amount of time required to copy the disk depends upon the amount of information on the disk.

    CALMA RDOS CENTURY BACKUP 11FEB78
    BACKUP OF CENTURY MADE 08:12:00 05/03/78
    R

4-9

The name of the disk (CENTURY) and dates and time will vary. When the copy operation is complete, the system prints an R and automatically rewinds the tape.

11. Continue with programming operations or type RELEASE **DC0** (or **BD0** for CDC disks) and press the RETURN key to release the master device. When the master device is released the system halts the computer after printing:

MASTER DEVICE RELEASED

12. Remove the BACKUP tape from the tape drive after pressing the RESET and REWIND switches, label the the tape with the information in step 10, remove the write-enable ring, and store the tape in a safe place.

NOTE

Do not perform the following steps except to boot systems with a NOVA computer.

13. Set all of the computer switch register switches down and then press and release the RESET switch.

14. Set the computer switch register to 000376₈ (switches 8 thru 14 up and all other switches down.)

15. Press and release the computer EXAMINE switch.

16. Set the computer switch register to 060120₈ (switches 1, 2, 9, and 11 up and all other switches down) to boot from the CENTURY disk. To boot from the CDC disk, set the switch register to 060126₈ (switches 1, 2, 9, 11, 13, and 14 up and all others down).

17. Press and release the computer DEPOSIT switch.

18. Set the computer switch register to 000377₈ (switches 8 thru 15 up and all other switches down).

19. Press and release the computer DEPOSIT NEXT switch.

20. Set the computer switch register to 000376₈ (switches 8 thru 14 up and all other switches down).

21. Press and release the computer RESET switch and then the START switch. The system prints:

FILENAME?

22. Perform steps 6 thru 12 to complete the procedure to create a BACKUP tape.

4.4.2 Loading a BACKUP Tape

A BACKUP tape created with the procedures in Paragraph 4.4.1 can be loaded onto disk anytime the system is up and running and an initialized disk is available. However, loading the BACKUP tape destroys any data that was previously on the disk. If a disk crash has occurred or other failure that may have damaged the disk, the disk bootstrapping and initializing procedures should be performed (refer to Paragraph 4.3).

All BACKUP tapes contain the disk bootstrap and an image of the disk at the time the tape was made. Depending on the amount of data that was stored, a BACKUP tape may consist of more than one physical tape. When more than one tape is loaded, the number 1 tape containing the disk bootstrap is loaded first and the remaining tapes are loaded sequentially using the same procedure. A typical dialog for loading a BACKUP tape is shown in Figure 4-4.

To load the BACKUP tape on an initialized disk, proceed as follows:

1. Install the disk pack, that will receive data from the BACKUP tape, on disk drive unit 0, turn on the disk drive and wait for the disk to get up to speed.

2. Mount and thread the BACKUP tape on tape drive unit 0 and set the tape drive for power on, load, and on line.

NOTE

Steps 3 thru 5 boot the ECLIPSE computer. To boot NOVA computers, skip to step 10.

3. Set the computer switch register to 100022₈ (switches 0, 11, and 14 up and all others down) to boot from magnetic tape.

4. Lift and release the computer RESET/STOP switch, and then the PR LOAD/EXEC switch. The system then reads the BACKUP tape and prints:

BACKUP OF CENTURY MADE 07:52:02 05/12/78

FILENAME? **ECLIPSE**

MAPPED ECLIPSE MULTIGROUND RDOS REV 6.01.03
DATE (M/D/Y) ? **5 11 78**
TIME (H:M:S) ? **7 51 00**

CALMA REV6 CLI
R
INIT MT0
R
BACKUP
CALMA RDOS CENTURY BACKUP 11FEB78
BACKUP OF CENTURY MADE 07:51:53 05/11/78
R
RELEASE DC0

MASTER DEVICE RELEASED

NOTE

Operator responses are shown in boldface
type and must be followed by pressing the
RETURN key.

**Figure 4-3. Typical Dialog to Create a
BACKUP Tape**

The disk name, date, and time will vary depending upon the disk drive being used and when the BACKUP tape was made. When the tape is completely read, the system automatically rewinds the tape after printing:

FILENAME?

5. Type the applicable file name such as ECLIPSE, 32KNOVA, GDS II, etc. and then press the RETURN key. If files were left open on a system failure, the system was shut down without releasing the master device, or RDOS is booted from a disk created from a backup tape, the file use counts were not cleared and the system prints:
PARTITION IN USE - TYPE C TO CONTINUE

Type C to advance the program, then the system prints:
CONTINUE
MAPPED ECLIPSE MULTIGROUND RDOS
REV x.xx.xx
DATE (M/D/Y)?

Where file name ECLIPSE was used and x.xx.xx is the current RDOS revision.

6. Type the actual date. For example, 5 12 78 is May 12, 1978. Press the RETURN key. The system then prints:

TIME (H:M:S)?

7. Type the actual time based on a 24-hour clock. For example, 8 02 10 is 8:02 AM and 10 seconds while 20 02 10 is 8:02 PM and 20 seconds. Press the RETURN key. The system then prints:

CALMA REVx CLI (where x is the current CLI revision)

If files were left open on a system failure or the system was shut down without releasing the master device, the system prints:

USE COUNTS INCORRECT - ALL DIRECTORIES BEING CLEARED
SYSGEN.DR CLEARED
DC0 CLEARED
R

The use counts are cleared in all active directories before the CLI prompt (R) is printed.

8. Continue with programming operations or type RELEASE DC0 (or BD0 for CDC disks) and press the RETURN key to release the master device. When the master device is released the system halts the computer after printing:

MASTER DEVICE RELEASED

9. Remove the BACKUP tape from the tape drive after first pressing the RESET and REWIND switches. Store the BACKUP tape, without a write-enable ring, in a safe place.

NOTE

Do not perform the following steps except to boot systems with NOVA computers.

10. Set all of the computer switch register switches down and then press and release the RESET switch.

11. Set the computer switch register to 000376$_8$ (switches 8 thru 14 up and all other switches down).

12. Press and release the computer EXAMINE switch.

13. Set the computer switch register to 060122$_8$ (switches 1, 2, 9, 11, and 14 up and all other switches down) to boot from magnetic tape.

14. Press and release the computer DEPOSIT switch.

15. Set the computer switch register to 000377$_8$ (switches 8 thru 15 up and all other switches down).

16. Press and release the computer DEPOSIT NEXT switch.

17. Set the computer switch register to 000376$_8$ (switches 8 thru 14 up and all other switches down).

18. Press and release the computer RESET switch and then the START switch. The system then reads the BACKUP tape and prints:

BACKUP OF CENTURY MADE 07:52:02 05/12/78

BACKUP OF CENTURY MADE 07:50:02 05/12/78

FILENAME? **ECLIPSE**

PARTITION IN USE - TYPE C TO CONTINUE   **C**
CONTINUE

MAPPED ECLIPSE MULTIGROUND RDOS REV 6.01.03
DATE (M/D/Y) ? **5 12 78**
TIME (H:M:S) ? **8 02 10**

CALMA REV6 CLI

USE COUNTS INCORRECT - ALL DIRECTORIES BEING CLEARED
SYSGEN.DR CLEARED

DC0 CLEARED

R

RELEASE DC0

MASTER DEVICE RELEASED

> **NOTE**
>
> Operator responses are shown in boldface type and all responses except **C** are followed by pressing the RETURN key. Dialog shows automatic clearing of use counts.

Figure 4-4. Typical Dialog to Load BACKUP Tape

The disk name, date, and time will vary depending upon the disk drive being used and when the BACKUP tape was made. When the tape is completely read, the system automatically rewinds the tape after printing:

FILENAME?

19. Perform steps 5 thru 9 to complete the BACKUP tape loading procedure.

4.5 RDOS SYSTEM GENERATION

RDOS system generation provides an interactive routine that allows the user to tailor the RDOS system to the hardware configuration and desired system features. System generation is used at anytime to reconfigure an existing system or to configure a fully initialized and booted disk. Before the system generation procedures are performed, the system must be operating with the CLI prompt (R) as the last character printed. If the system is not operating, the RDOS start up procedure in Paragraph 4.5.1 is performed. After a CLI prompt (R) is printed, system generation is performed following the procedures given in Paragraph 4.5.2.

4.5.1 RDOS Start Up Procedure

The following steps describe how to start up the RDOS system using an initialized and booted RDOS disk. A typical dialog is shown in the first six lines of Figure 4-5.

1. Turn on system power and check that all hardware units are receiving power.

2. Check that the system console (Teletype, Decwriter, etc.) and line printer (optional) are on line.

3. Install a disk pack (containing the RDOS program) on disk drive unit 0, turn on the disk drive, and wait for the disk to get up to speed.

NOTE

For systems with NOVA computers, skip to step 9.

4. Set the computer switch register to 100020₈ (switches 0 and 11 up and all other switches down) to boot from the CENTURY disk. To boot from the CDC disk, set the switch register to 100026₈ (switches 0, 11, 13, and 14 up and all others down).

5. Lift and release the computer RESET/STOP switch and then the PR LOAD/EXEC switch. The system prints:

FILENAME?

6. Type ECLIPSE for the S/200 CPU or ZECLIPSE for the S/230 CPU and press the RETURN key. The system then prints:

MAPPED ECLIPSE MULTIGROUND RDOS REV x.xx.xx DATE(M/D/Y)?

The x.xx.xx indicates the current RDOS revision.

7. Type the actual date. For example, 5 11 78 is May 11, 1978. Press the RETURN key. The system then prints:

TIME(H:M:S)?

8. Type the actual time based on a 24-hour clock. For example 8 17 45 is 8:17 AM and 45 seconds while 20 17 45 is 8:17 PM and 45 seconds. Press the RETURN key. The system prints:

CALMA REVx CLI
R

The x indicates the current CLI revision and R is the CLI prompt.

The system is now ready to accept CLI commands so the system generation in Paragraph 4.5.2 or other CLI procedures can be performed.

NOTE

Do not perform the following steps unless a system with a NOVA computer is being started up.

9. Set all of the NOVA computer switch register down and then press and release the RESET switch.

10. Set the computer switch register to 000376₈ (switches 8 thru 14 up and all other switches down).

11. Press and release the computer EXAMINE switch.

12. Set the computer switch register to 060120₈ (switches 1, 2, 9, and 11 up and all other switches

down) to boot from the CENTURY disk. To boot from the CDC disk, set the switch register to 060126₈ (switches 1, 2, 9, 11, 13 and 14 up and all others down).

13. Press and release the computer DEPOSIT switch.

14. Set the computer switch register to 000377₈ (switches 8 thru 15 up and all other switches down).

15. Press and release the computer DEPOSIT NEXT switch.

16. Set the computer switch register to 000376₈ (switches 8 thru 14 up and all other switches down).

17. Press and release the computer RESET switch and then the START switch. The system prints:

    FILENAME?

18. Type 32KNOVA (or other applicable NOVA name) and press the RETURN key. The system then prints:

    NOVA RDOS REV x.xx.xx
    DATE (M/D/Y)?

    Where x.xx.xx indicates the current RDOS revision.

19. Perform steps 7 and 8 to complete the system start up procedure for NOVA computers.

4.5.2 System Generation Dialog

The system generation dialog is activated by a CLI command and can take three different forms depending upon the command used. These commands are as follows:

MSYSGEN is the filename for systems with ECLIPSE computers and multiground RDOS. The dialog is listed in Figure 4-5.

BSYSGEN is the filename for systems with ECLIPSE computers and without multiground RDOS. The dialog is listed in Figure 4-6.

SYSGEN is the name of the system directory that contains the system generation files and is also the filename for systems that use NOVA computers. The dialog is listed in Figure 4-7.

The following procedure describes only the **MSYSGEN** dialog since it contains most of the queries found in the other two. To use the system generation file, **SYSGEN** must be the current directory and **SYSGEN** is located in the primary partition (**DC0**). The first two steps of the procedure are used to ensure that **SYSGEN** is the current directory.

To generate an RDOS system, ensure that the last print out was the CLI prompt (R) and then perform the following steps:

1. Type **GDIR** and press the RETURN key to get the name of the current directory. The system prints:

    DC0 (or BD0 for CDC disks)
    R

    If the current directory is not DC0 (or BD0), type **DIR DC0** (or **DIR BD0**) and press the RETURN key. When the system prints an R continue to the next step.

2. Type **DIR SYSGEN** and press the RETURN key to make **SYSGEN** the current directory. The system prints an R.

3. Type **MSYSGEN** (plus arguments) and press the RETURN key to call the system generation routine. For global and local switches, refer to the **SYSGEN** description under CLI commands in Section III of this manual. The system prints:

    MULTIGROUND ECLIPSE SYSTEM REV x.xx.xx
    VALID ANSWERS ARE IN PARENTHESIS
    RESPOND ACCORDINGLY

    S/230 or C/330 MAP? ("0"=NO "1"=YES)

    The x.xx.xx indicates the current program revision.

NOTE

The system may skip some system generation queries depending upon the response typed to a previous query.

4-15

4. Type 1 if the system has a 256K or greater memory with memory protection option, or type 0 if the system has a 128K memory with memory protection option, and press the RETURN key. The system prints:

NO. OF ADDITIONAL BACKGROUNDS (0-14)

5. Type a number from 0 thru 14 and press the RETURN key to select the number of additional backgrounds needed for multiground operation. If unsure select 2 for a total of four grounds since the basic system has 1 foreground and 1 background. The system prints:

NO. BACKGROUNDS SWAPPABLE (0 OR 2-MAX)

6. Type 0 if no grounds are to be swapped, or type 2 so that two grounds can be swapped to disk if memory is full, and press the RETURN key. The system prints:

NO. BACKGROUND CHANNELS (15-63)

7. Type a number from 15 thru 63 and press the RETURN key to select the maximum number of hardware data I/O channels needed to execute programs in the main background. CLI runs in the main background and requires 15 channels. RDOS will not run a program that requests more channels than the number selected. If unsure, select 17. The system prints:

NO. FOREGROUND CHANNELS (0-63)

8. Type a number from 0 thru 63 and press the RETURN key to select the maximum number of hardware data I/O channels needed to execute programs in the foreground. RDOS will not run a program that requests more channels than the number selected. If unsure, select 17. The system prints:

NO. BG 03 CHANNELS (1-63)

9. Type a number from 1 thru 63 and press the RETURN key to select the number of hardware data I/O channels available to the first additional background. If unsure, select 17. The system repeats the last query for each additional background and following the last response, prints:

NO. SWAPPING BG CHANNELS (1-63)

10. Type a number from 1 thru 63 and press the RETURN key to select the number of channels to be swapped in the swapping backgrounds. If unsure, select 17. The system prints:

NO. OF CENTURY DISKS (0-2)

11. Type a number from 0 thru 2 and press the RETURN key to indicate the number of CENTURY disk controller boards in the system. The system prints:

NO. OF BIG DISKS (0-4)

12. Type a number from 0 thru 4 and press the RETURN key to indicate the number of CDC storage module disks (80MB or 300MB) in the system. The system prints:

DUAL PROCESSORS? ("0"=NO "1"=YES)

13. Type 0 and press the RETURN key for systems with one computer. If the system has two computers, type a 1 then press the RETURN key.
When a 1 response is typed the system prints:

OTHER CPU RDOS-IPB ("0") OR GDS ("1")?

Then, type a 0 and press the RETURN key to select the interprocessor bus (IPB) or type a 1 and press the RETURN key to select the CALMA CPU-CPU handler. The system then prints:

NO. OF STACKS (1-20)

14. Type a number from 1 thru 20 and press the RETURN key to indicate the number of I/O devices that will operate concurrently. One stack is required for each ground, one stack for spooling, and an additional stack for each task running concurrently. If unsure, select 3 for NOVA and 10 for ECLIPSE. The system prints:

NO. OF EXTRA CELLS (0-64)

15. Type a number from 0 thru 64 and press the RETURN key to select the number of extra cells (data buffers with 16 memory words) required. Two extra cells are recommended for each spooling device in the system such as a line printer or magnetic tape controller. If unsure, type 6. The system prints:

TUNING? ("0"=NO "1"=YES)

16. Type 0 and press the RETURN key if no tuning is desired. To request tuning, type 1 and press the RETURN key. When tuning is requested, RDOS records how often it needed a system stack, cell, or data buffer that was unavailable. This information is kept in a disk file with extension **.TU** and is used at a later time to reconfigure the RDOS system for more efficient operation.

If a 1 response is typed the system prints:

WITH("1") OR WITHOUT("0") OVERLAY REPORT

Then, type a 1 or 0 and press the RETURN key to indicate whether or not an overlay report is desired. The overlay report records the number of times each overlay is transferred from disk because it was not core resident. This information is appended to the tuning file and is used to determine if more buffer space should be allocated to the overlay. The system prints:

NO. OF EXTRA BUFFERS REQUIRED (0-63)

17. Type a number from 0 thru 63 and press the RETURN key to select the number of extra buffers required. Each extra buffer requires an additional 270 words and reduces the memory available for programs. Extra buffers are used to reduce the disk swapping time so that RDOS runs more efficiently. If unsure, select 6. The system prints:

NO. OF DISK DIRECTORY DCB'S (0-64)

18. Type a number from 0 thru 64 and press the RETURN key to select the number of system directories that can be initialized at the same time. If unsure, type 10. The system prints:

NO. OF CALMA STATIONS (0-8)

19. Type a number from 0 thru 8 and press the RETURN key to indicate the number of graphic display stations in the system. The system prints:

SYSTEM 6? ("0"=NO "1"=YES)

20. Type a 1 or 0 and press the RETURN key to indicate whether or not Calma System 6 stations are being used. The system prints:

NO. OF CALMA VMDS (0-8)

21. Type a number from 0 thru 8 and press the RETURN key to indicate the number of Vector Memory Displays in the system. The system prints:

NO. OF CONTROLLERS FOR MTA (0-2)

22. Type a number from 0 thru 2 and press the RETURN key to specify the number of magnetic tape controllers (not drives) in the system.

A 1 is almost always typed for one controller. The system prints:

PRIMARY ("0") OR SECONDARY ("1")

This usually is set at the time the system is ordered (usually 0); type the applicable response and press the RETURN key.

For each controller specified, the system prints:

NO. OF DEVICES FOR CONTROLLER #1 (1-8)

Type a number from 1 thru 8 (normally 1) and press the RETURN key to indicate the number of tape drives connected to each controller. The system prints:

AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES)

23. Type 0 and press the RETURN key. The system prints:

LOGON? ("0"=NO "1"=YES)

24. Type 0 and press the RETURN key to prevent logon. To enable the logon feature, type 1 and press the RETURN key. When the Logon feature is selected, each time the system is booted through the computer switches, the program requires a user ID and password before CLI is allowed to run. The system prints:

RTC FREQ(1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ)

25. Type a number from 1 thru 5 and press the RETURN key to select the Real-Time Clock frequency. Normally a 4 is typed for ECLIPSE and a 1 is typed for NOVA.

program to be synchronized to the applicable line frequency. The system prints:

NO. OF PTR (0-2)

26. Type a number from 0 thru 2 and press the RETURN key to indicate the number of high speed paper tape readers in the system. The system prints:

NO. OF PTP (0-2)

27. Type a number from 0 thru 2 and press the RETURN key to indicate the number of high speed paper tape punches in the system. The system prints:

NO. OF LPT (0-2)

NOTE
CALMA systems only support one line printer at this time.

28. Type a 0 or 1 and press the RETURN key to indicate the number of line printers in the system. If a 1 response is typed, the system then prints:

SMART SPOOLER ("0"=NO "1"=YES)

If the smart spooler will not be used type 0 and press the RETURN key, and if the smart spooler will be used type 1 and press the RETURN key.

For each line printer indicated the system also prints:

COLUMN SIZE FOR DEVICE #1 (80 OR 132)

To respond to this query, type either 80 or 132 depending upon the column size for the line printer and then press the RETURN key.

The system then prints:

LOWER CASE PRINTER ("0"=NO "1"=YES)

29. Type a 1 or 0 and press the RETURN key to indicate whether or not the printer will print lower case (not all capital letters). The system then prints:

NO. OF CDR (0-2)

30. Type a number from 0 thru 2 and press the RETURN key to indicate the number of punched or mark-sense card readers in the system. The system prints:

NO. OF PLT (0-3)

31. Type a number from 0 thru 3 and press the RETURN key to indicate the number of plotters in the system. The system prints:

QTY? ("0"=NO "1"=YES)

32. Type a 0 or 1 (1 is normal for all ECLIPSE systems) to indicate whether or nor (respectively) the system is equipped with a 4060 asynchronous data multiplexer and then press the RETURN key. The system prints:

SIZE OF PRIMARY TTY (80 or 132)

33. Type 80 if the primary keyboard (TTY) is a Teletype, or type 132 if the primary keyboard is a Decwriter or other 132 character per column printer and then press the RETURN key. The system then prints:

SECOND TTY? ("0"=NO "1"=YES)

34. Type 0 if only one system console (TTY) will be used or type a 1 if the system uses a second TTY and then press the RETURN key.

If a 1 was typed the system prints:

SIZE OF SECOND TTY (80 or 132)

Type 80 or 132 as applicable and then press the RETURN key. The system then prints:

CORE DUMP FACILITY? ("0"=NO "1"=YES)

35. Type a 1 to allow core dump and press the RETURN key. The system then prints:

SYS00.SV LOADED BY CALMA RLDR REV 05.03 AT 08:45:19 05/11/78

The revision level, time, and date will vary.

Unless interrupted the system proceeds to print the load map and any load errors. Save the dialog just produced and the load map for future reference when making revisions to the system generation.

To interrupt the listing press the CTRL (Control) and A keys at the same time. When the listing is complete the system prints an R or, if interrupted, it prints INT and then R. System generation is complete and control is returned to CLI.

FILENAME? **ECLIPSE**

MAPPED ECLIPSE MULTIGROUND RDOS REV 6.01.05
DATE (M/D/Y) ? **5 11 78**
TIME (H:M:S) ? **8 17 45**

CALMA REV6 CLI
R
GDIR
DC0
R
DIR SYSGEN

R
MSYSGEN
MULTIGROUND ECLIPSE SYSGEN REV 6.01.05
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

S/230 OR C/330 MAP? ("0"=NO "1"=YES) **1**
NO. OF ADDITIONAL BACKGROUNDS (0-14) **4**
NO. BACKGROUNDS SWAPPABLE (0 OR 2-MAX) **2**
NO. BACKGROUND CHANNELS (15-63) **17**
NO. FOREGROUND CHANNELS (0-63)**17**
NO. BG 03 CHANNELS (1-63) **17**
NO. BG 04 CHANNELS (1-63) **17**
NO. SWAPPING BG CHANNELS (1-63) **17**
NO. OF CENTURY DISKS (0-2) **1**
NO. OF BIG DISKS (0-4) **2**
DUAL PROCESSORS? ("0"=NO "1"=YES) **1**
OTHER CPU RDOS-IPB("0") OR GDS("1")? **1**
NO. OF STACKS (1-20) **10**
NO. OF EXTRA CELLS (0-64) **6**
TUNING? ("0"=NO "1"=YES) **1**
WITH("1") OR WITHOUT("0") OVERLAY REPORT? **1**
NO. OF EXTRA BUFFERS REQUIRED (0-63) **6**
NO. OF DISK DIRECTORY DCB'S (0-64.) **10**
NO. OF CALMA STATIONS (0-8) **2**
SYSTEM 6? ("0"=NO "1"=YES) **0**
NO. OF CALMA VMDS (0-8) **2**
NO. OF CONTROLLERS FOR MTA (0-2) **1**
PRIMARY ("0") OR SECONDARY ("1")? **0**
NO. OF DEVICES FOR CONTROLLER #1 (1-8) **2**
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) **0**
LOGON? ("0"=NO "1"=YES) **1**
RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) **4**

(Continued on Sheet 2)

NOTE

Operator responses are shown in boldface
type and must be followed by pressing the
RETURN key.

Figure 4-5. Typical MSYSGEN Dialog for
Systems with Eclipse Computers
and Multiground RDOS
(Sheet 1 of 2)

```
NO. OF PTR (0-2) 0
NO. OF PTP (0-2) 0
NO. OF LPT (0-2) 2
SMART SPOOLER ("0"=NO "1"=YES) 1
COLUMN SIZE FOR DEVICE #1 (80 OR 132) 132
LOWER CASE PRINTER ("0"=NO "1"=YES) 0
NO. OF CDR (0-2) 0
NO. OF PLT (0-3) 1
QTY? ("0"=NO "1"=YES) 1
SIZE OF PRIMARY TTY (80 OR 132) 80
SECOND TTY? ("0"=NO "1"=YES) 0
CORE DUMP FACILITY? ("0"=NO "1"=YES) 1
SYS000.SV      LOADED BY CALMA RLDR REV 05.03 AT 08:45:49 05/11/78
   SYS00    000452
   ZINIT    000452
   ZPWRF    000452
   ZMAPZ    001041
   ZSCHE    001356
INT                      (Control A typed to interrupt listing.)
R
DIR DC0
R
RELEASE DC0

MASTER DEVICE RELEASED
```

Figure 4-5.   Typical MSYSGEN Dialog for Systems with Eclipse Computers and Multiground RDOS (Sheet 2 of 2)

BSYSGEN
ECLIPSE SYSGEN REV 6.01.05
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

S/230 OR C/330 MAP? ("0"=NO "1"=YES) **1**
NO. BACKGROUND CHANNELS (15-63) **15**
NO. FOREGROUND CHANNELS (0-63) **0**
NO. OF CENTURY DISKS (0-2) **1**
NO. OF BIG DISKS (0-4) **0**
DUAL PROCESSORS? ("0"=NO "1"=YES) **0**
NO. OF STACKS (1-20) **10**
NO. OF EXTRA CELLS (0-64) **6**
TUNING? ("0"=NO "1"=YES) **0**
NO. OF EXTRA BUFFERS REQUIRED (0-63) **6**
NO. OF DISK DIRECTORY DCB'S (0-64.) **10**
NO. OF CALMA STATIONS (0-8) **2**
SYSTEM 6? ("0"=NO "1"=YES) **0**
NO. OF CALMA VMDS (0-8) **0**
NO. OF CONTROLLERS FOR MTA (0-2) **1**
PRIMARY ("0") OR SECONDARY ("1")?   **0**
NO. OF DEVICES FOR CONTROLLER #1 (1-8) **1**
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) **0**
LOGON? ("0"=NO "1"=YES) **0**
RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) **4**
NO. OF PTR (0-2) **0**
NO. OF PTP (0-2) **0**
NO. OF LPT (0-2) **1**
SMART SPOOLER ("0"=NO "1"=YES) **1**
COLUMN SIZE FOR DEVICE #1 (80 OR 132) **132**
LOWER CASE PRINTER ("0"=NO "1"=YES) **0**
NO. OF CDR (0-2) **2**
NO. OF PLT (0-3) **1**
QTY? ("0"=NO "1"=YES) **1**
SIZE OF PRIMARY TTY (80 OR 132) **80**
SECOND TTY? ("0"=NO "1"=YES) **0**
CORE DUMP FACILITY? ("0"=NO "1"=YES) **1**
SYS000.SV   LOADED BY CALMA RLDR REV 05.03 AT 08:25:55 05/11/78
   SYS00   000452
   ZEINI   000452
   ZPWRF@
INT   (Control A typed to interrupt listing.)
R

> **NOTE**
>
> Operator responses are shown in boldface type and must be followed by pressing the RETURN key.

**Figure 4-6. Typical BSYSGEN Dialog for
Systems With Eclipse Computers
and RDOS Without Multiground**

4-21

SYSGEN
NOVA SYSGEN REV 6.01.03
VALID ANSWERS ARE IN PARENTHESIS RESPOND ACCORDINGLY

CORE STORAGE (IN K WORDS 16-32) **32**
NO. OF CENTURY DISKS (0-2) **1**
NO. OF BIG DISKS (0-4) **0**
DUAL PROCESSORS? ("0"=NO "1"=YES) **0**
NO. OF STACKS (1-10) **3**
NO. OF EXTRA CELLS (0-64) **2**
TUNING? ("0"=NO "1"=YES) **0**
NO. OF EXTRA BUFFERS REQUIRED (0-63) **4**
NO. OF DISK DIRECTORY DCB'S (0-64.) **6**
NO. OF CALMA STATIONS (0-8) **2**
NO. OF CALMA VMDS (0-8) **2**
SYSTEM 6? ("0"=NO "1"=YES) **0**
NO. OF CONTROLLERS FOR MTA (0-2) **1**
PRIMARY ("0") OR SECONDARY ("1")? **0**
NO. OF DEVICES FOR CONTROLLER #1 (1-8) **2**
AUTO RESTART ON POWER FAIL? ("0"=NO "1"=YES) **1**
LOGON? ("0"=NO "1"=YES) **0**
RTC FREQ (1=10HZ 2=50HZ 3=60HZ 4=100HZ 5=1000HZ) **1**
NO. OF PTR (0-2) **0**
NO. OF PTP (0-2) **0**
NO. OF LPT (0-2) **0**
NO. OF CDR (0-2) **0**
NO. OF PLT (0-3) **1**
QTY? ("0"=NO "1"=YES) **0**
SIZE OF PRIMARY TTY (80 OR 132) **80**
SECOND TTY? ("0"=NO "1"=YES) **0**
CORE DUMP FACILITY? ("0"=NO "1"=YES) **1**
SYS000.SV   LOADED BY CALMA RLDR REV 05.03 AT 08:29:37 05/11/78
   SYS00   000452
   UINIT   000452
   CLIBT   000452
   TMIN   004535
   NSAC3   004631
   CLIEN   004631
   UINI1   004631
   UINI2   X
INT   (Control A typed to interrupt listing.)
R

> **NOTE**
>
> Operator responses are shown in boldface type and must be followed by pressing the RETURN key.

Figure 4-7.   Typical SYSGEN Dialog for Systems with NOVA Computers and RDOS without Multiground

APPENDIX A

CALMA RDOS EDITORS

The Calma version of RDOS includes five utilities for editing that vary in complexity and purpose. Table A1-1 lists the five editors and their uses. The most commonly used editors are described in this appendix.

This appendix is divided into two main sections as follows:

Section A1   contains the Text Editor with CALMA enhancements and the Multiuser Text Editor (**EDIT**).

Section A2   contains the CALMA station editor (**RDEDIT**).

Section A1 contains a reproduction of a Data General manual with CALMA enhancements included. Section A2 contains CALMA editing instructions that were reproduced from a previous document. In these sections, the data retains the same chapter and paragraph numbers as the original documents. To avoid confusion, page numbers were changed by adding the Appendix Section number as a prefix. Stay within one section of this appendix during editing.

Table A1-1. CALMA RDOS Editors

NAME OF EDITOR	CLI COMMAND TO INVOKE EDITOR	EDIT COMMAND PROMPT	COMMAND TO RETURN TO CLI	USE OF EDITOR AND COMMENTS
Text Editor	**EDIT**	*	↑D$$	Data General text editor to create, update, and modify ASCII text. Also, includes CALMA enhancements for easier editing. See Appendix Section A1.
Octal Editor	**OEDIT**	. (period)	$Z	Data General octal editor to examine and modify octal, decimal, or ASCII in user files. Refer to the Data General Octal Editor Manual, Part No. 093-000084 for more detail.
CALMA Station Editor	**RDEDIT**	*	↑D$$	Editor for ASCII files from CALMA stations. See Appendix Section A2.
SEDIT	**SEDIT**	. (period)	$Z	Data General Save File Editor which follows the syntax of the RDOS Debugger. Refer to the Data General Save File Editor Manual.
SUPEREDIT	**SPEED** (for Eclipse) **NSPEED** (for Nova)	!	H$$	Data General program for multi-buffer, multiple I/O file ASCII text editor. Refer to the Data General SUPEREDIT Manual.
NOTE The up arrow (↑) is echoed when the CTRL (control) key is pressed and the dollar sign ($) is echoed when the ESC (escape) key is pressed.				

APPENDIX A

SECTION A1

TEXT EDITOR
USER'S MANUAL

INTRODUCTION

The purpose of the DGC Text Editor is to create, update and modify ASCII text. This is done by the use of simple commands.

The commands used in the Editor are divided into groups, those that input and output the contents of the edit buffer and those that modify the contents of the buffer. The edit buffer is an area of memory in which input is stored while being modified. Input commands bring the text into the buffer, modification commands are used to change the contents of the buffer, and output commands transfer the updated files to the output device.

The command structure is versatile enough to allow the modification of a single character, several characters, or a whole line.

This manual incorporates the CALMA enhancements to increase the edit capabilities and decrease the probability of accidentally destroying a file when using CALMA programs. No attempt has been made to reorganize or change existing data.

TABLE OF CONTENTS

CHAPTER 2   Commands (Continued)

CHAPTER 1

CONCEPTS

EDITING PROCESS

The editing process consists of the following steps:

1. Call or load the Editor
2. Specify I/O devices or files.
3. Bring a page into the buffer from the input device or file.
4. Modify the page.
5. Transfer the page from the buffer to the output device or file.

When the Text Editor is loaded (step 1) it prints a prompt ("*") on the input console. The user must then specify the I/O devices or files (step 2). Another prompt is issued and an input command is issued by the user (step 3). Whenever a prompt appears on the console, it signifies the Editor is ready to take another command. To modify the page the user issues commands from the console (step 4). Step 5 is accomplished by issuing output commands.

If the capacity of the buffer is exceeded, while reading input into the edit buffer, an error message is printed on the console. Error messages are described later in the manual.

INPUT

The DGC Text Editor takes a continuous string of characters as input. These characters form lines, where a line is a string of characters up to and including a carriage return (<cr>). A page of input is a string of characters up to but not including a form feed.

While using the DGC Text Editor in the Real Time Disk Operating System (SOS), input is in the form of an input file or any input device supported by the operating system. When editing in Stand-alone Operation, input is in the form of paper tape.

OUTPUT

Output is the information that is transferred from the buffer to the output device or file. RDOS output disk files are updated after every page. This saves a part of the edited file in the event of a system failure.

CHARACTER POINTER

The DGC Text Editor maintains an implicit pointer called the Character Pointer (CP). This pointer resides between two characters and is used to facilitate locating the exact position for modification or examination. For example:

ABCDF
↑
CP

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                                │
│   ┌──────────────┐         ┌──────────────┐         ┌──────────────┐           │
│   │    INPUT      │──────▶ │     EDIT      │──────▶ │    OUTPUT     │           │
│   │  DEVICE/FILE  │         │    BUFFER     │         │  DEVICE/FILE  │          │
│   └──────────────┘         └──────────────┘         └──────────────┘           │
│                                                                                │
└──────────────────────────────────────────────────────────────────────────────┘
```
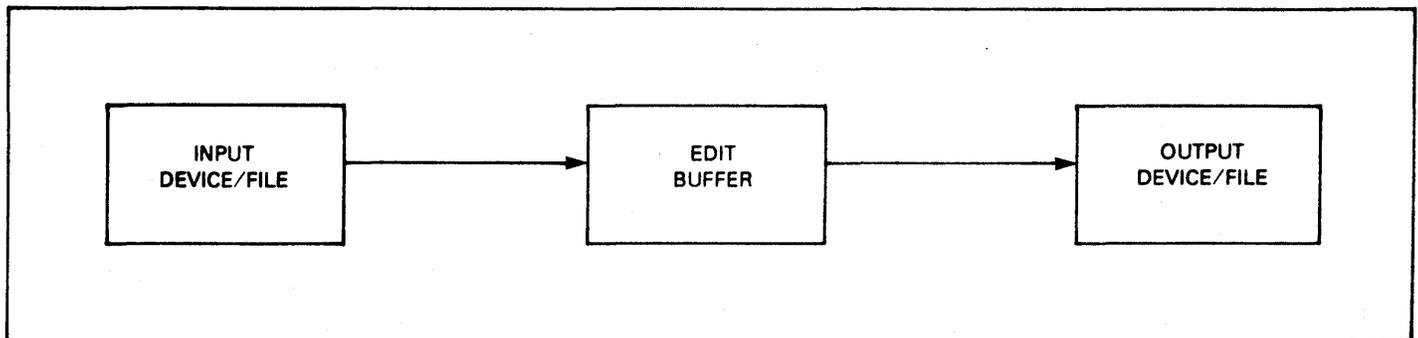
Figure 1-1.   Editor I/O

Any modifications performed are placed at the current location of the CP (i.e., insertion of an E produces, ABCDEF).

There is also an implicit line numbering system that is maintained by the Text Editor. This numbering system starts with the number 1 and is continually updated as lines are inserted and deleted.

TABBING

The Editor is equipped to simulate tabs with spaces. This feature is usually used if the console being used does not have a tab key. If tabs are simulated, the pre-defined tab positions occur at 1, 9, 17, 25 etc. The Editor is initialized to simulate tabs. It may be complemented, by typing a CTRL P (↑P), in a command string. Each occurrence of ↑P causes the tabbing feature to be complemented, thus ↑P↑P has no effect.

MODE

The Editor can operate in one of the following two modes:

Page Mode   File input is page oriented.

Window Mode   File input deals with a set number of lines of input (RDOS only).

At any instant the Editor is in one of the following two states:

Command Mode   The Editor is ready to accept a command.

Execution Mode   Commands issued to the Editor are being carried out.

RDOS NUMBER REGISTER

The Editor maintains a register represented by "#", which may be used to modify text when operating in the Real Time Disk Operating System. The commands for the manipulation of this register are found under Special Editing Commands Used in RDOS in Chapter 2.

PARITY CHECK

Some of the input commands can check parity. A parity bit is a binary bit that indicates the total number of binary "1" digits in a character. When the total number of "1" bits, including the parity bit, is even, the system is an even parity system.

When parity is checked, the parity bit is looked at to see if it indicates odd or even parity. If the parity bit is 0 then it has odd parity and an error message is printed on the teletype.

NOTATION CONVENTIONS

Console Representation	Results from
↑letter	Pressing CTRL and some letter e.g., CTRL E.
$	Pressing ESC.

Notation Conventions Used in Formats in the Manual

NOTATION	MEANING
<cr>	Represents pressing RETURN on console.
[ ]	Enclose optional arguments to a command.
variable	An argument in lower case and boldface is a variable for which the user substitutes the appropriate literal.
filename	The variable argument **filename** is usually replaced with the name of a file such as FILE 1. However, under RDOS, **filename** may represent the format:

[primary partition:] [sub-partition:] [sub-directory:] filename [extension]

e.g., DP1:MYDIR:FILE1.LS

primary partition — filename — sub-directory — extension

(RDOS only)

CHAPTER 2

COMMANDS

Once the user loads the Editor and the Editor issues a prompt the user is in command mode. It is now possible to issue input, modification, and output commands. These commands are described in this Chapter.

COMMAND STRUCTURE

ESC Key

The ESC (escape) character is represented by a $ on the output console. It is used for two purposes:

1. To signal the end of a command or command string. This is done by typing two consecutive escapes.
2. To delimit a string from the next command character or from another string. This is done by typing a single escape.

Single Command

A single command has the format:

$$[n] \textbf{ code } [\textbf{string } \$]$$

where: n is an optional integer in the range $-2074<n<+2074$ in SOS and in the range $-2^{15} \leq n \leq +2^{15} -1$,
code is a letter, two letters, or special graphic
string is an optional string of characters terminated by an ESC.

The only exceptions to this are the change and UR commands which have two strings following the code:

$$\textbf{Cstring\$string\$}$$
$$\textbf{URstring\$string\$}$$

Command String

A command string is two or more commands stacked for sequential execution by the Editor. Each single command may be delimited from the next by a single escape. The format is:

where: [n] **code** [string$] [n] **code** [string$] [n] **code** ↑ [string$] ...

n is an optional integer in the range -2074 n $+2074$ in SOS and in the range $-2^{15} \leq n \leq +2^{15} -1$,
code is a letter, two letters, or special graphic
string is an optional string of characters terminated by an ESC.

A command string may be carried over to the next line by the use of a carriage return between two commands. Two consecutive ESC's terminate the command string.

Deleting a Command Character

The RUBOUT key is used to delete characters in a command. It deletes the last typed character from right ot left, each time it is depressed. When a character is deleted, it is repeated on the output console. For example:

        THE ABSSBA VALUE
               ↑↑↑
               rubouts

produces:

        THE VALUE

When the RUBOUT key is used to delete all the characters in a command line a new prompt is issued by the Editor.

INPUT COMMANDS
---↑Y

The command to read input into the buffer until a form feed or window width is encountered is:

        ↑Y$$

In page mode, the form feed is read but not stored in the edit buffer. The CP is positioned at the beginning of the buffer and a prompt is issued on the output console.

If the capacity of the buffer is exceeded the resulting error message is:

        TEXT BUFFER FULL WHILE READING

The buffer must then either be output or sections deleted from it before the rest of the page can be read in.

If parity is checked and a read fails parity, the message printed on the teletype is:

PARITY ERROR IN LINE **n**

For example, to bring the first 5 lines of the Sample Program (Figure 2-1) into the edit buffer, issue the command:

↑Y$$

The command to yank in a specified number of lines is:

n↑Y$

where: **n** is the number of lines to be read into the edit buffer. This command is not applicable in the Stand-alone Editor.

If the next page has fewer than **n** lines, the Editor will stop at the form feed.

In the Window mode both Y or **n**Y brings in the window width. A form feed is read in as just another character.

--- A

The command to append a page or window width of input to the material already contained in the edit buffer, is:

A$

The Text Editor appends a page of input to the present contents of the buffer and returns a prompt on the console. The CP is located at the beginning of the appended page.

If the buffer is already full or its capacity is exceeded while appending, the resulting message is:

TEXT BUFFER FULL WHILE READING

In either case the buffer must be output or sections deleted before the rest of the page can be appended.

For example, if the contents of the second buffer of the Sample Program (Figure 2-1) is to be appended to the first buffer to form one buffer the command issued is:

A$$

--- GR

The command to get for reading (input) a specified file is:

GRinputfilename$

inputfilename is the file to be input and must be an existing file. When this command is issued the input file, if any, is closed and the specified file is opened.

NOTE
This command does not read the first page of inputfilename into the buffer, the user must issue a command in order to bring the first page into the buffer.

If the file does not exist the resulting error message is:

NO SUCH FILE

If the file is read-protected the resulting error message is:

FILE CAN'T BE USED FOR INPUT

For example, when the command:

GRSAMPLE$$

is issued, the Editor opens the file SAMPLE for examination and/or modification.

--- UN

UNfilename$

is used to create a new file. This command creates a file with the specified name, makes it the input file, and makes it permanent for the duration of its editing. It also creates and opens **filename**.SC for output. A **filename** with suffix .SC or a file with the same name as a device name cannot be used. The attributes of the input file are restored only when a UE, US, UC or UH is issued.

For example, to create a permanent file named SAMPLE and open SAMPLE.SC for output, issue:

UNSAMPLE$$

--- UY

UYfilename$

is used to edit an existing file. This command makes the specified file permanent for the duration of its editing, opens it for input and yanks a page. It also creates and opens **filename**.SC for output. A **filename** with the suffix .SC or a file with the same name as a device name cannot be used. The attributes of the input file are restored only when a UE, US, UC or UH is issued.

For example, to change the attributes of SAMPLE to permanent, open it for input and yank the first page into the edit buffer, issue:

UYSAMPLE$$

In addition it creates and opens SAMPLE.SC for output.

NOTE
After opening a file with UY or UN a GRfilename can be used to read in from other files. At the end when a UE, UC, or US is issued, the original input **filename** will still be used for renaming purposes.

When the **filename** is given immediately after the Edit command in the CLI, it is equivalent to UY**filename**. For example,

```
EDIT SAMPLE <cr>
is equivalent to
EDIT <cr>
*UYSAMPLE$$
```

CHARACTER POINTER COMMANDS

--- B

The command to move the CP to the beginning of the buffer is:

B$

There is no argument used with this command. If one is given, it will be ignored by the Text Editor.

For example, to position the CP right before the period in line 1 of the Sample Program (Figure 2-1), issue the command:

B$$

--- nJ

The command to position the CP before the first character in line **n** from the beginning of the buffer is:

nJ$$

where: **n** is the number of the line the CP is to jump to.

For example, the CP is positioned at the end of line 3 in the Sample Program (Figure 2-1), and the user wants it positioned at the beginning of line 5, the command issued is:

5J$$

--- nL

The command to move the CP **n** lines from its present position is:
nL$

where: **n** is the number of lines the CP is to be moved.

For example, the CP is positioned at the beginning of line 3 in the Sample Program (Figure 2-1), and the user wants it positioned at the beginning of line 5, the command issued is:

2L$$

The command to move the CP to the beginning of the line on which it is located is:

L$

--- nM

The command to move the CP **n** characters to the right or left is:

nM$

where: **n>0** The CP is moved **n** characters to the right
n>0 The CP is moved **n** characters to the left

For example, the CP is positioned before the T of TEMPORARY in line 5, of the Sample Program (Figure 21), the command to position it before the Y in line 5 is:

8M$$

--- Z

To move the CP to the end of the buffer, issue the command:

Z$

An example of this would be if the CP were positioned anywhere on line 2 of the Sample Program (Figure 2-1), the command to position it after the <cr> in line 5 is:

Z$$

BUFFER EXAMINATION COMMANDS

There are two commands to examine the buffer: T and nT. Neither of these commands have any effect on the CP.

The command to type the entire buffer on the console is:

T$

The command to examine a certain number of lines of the buffer is:

nT$

The command to examine a certain number of lines of the buffer is:

nT$

where: n is the number of lines to be examined from the present position of the CP.

MODIFICATION COMMANDS

--- S

The command to search for a character in the current buffer is:

Sstring$

The Editor searches for the string from the present CP position. The CP is then positioned immediately after the last character of the first occurrence of the string.

When the end of the buffer is encountered and the string has not been found the console prints the message:

STR NOT FOUND

The CP is then positioned at the beginning of the buffer and a new prompt is issued.

For example, the CP is located at the beginning of the Sample Program (Figure 2-1), and the user issues the command:

SSAVE$$

The CP is positioned directly after the E in SAVE in line 1.

--- N and ↑Q

The command to continue searching for a string throughout the remainder of the input file is:

Nstring$

The Editor then searches through the current buffer for the string. If it is not found the buffer is output, the next page is yanked in, and the search is continued. This process is repeated until the string is found. When the Editor encounters the string the CP is positioned immediately after the last character of the string.

If the end of the program is encountered and the string is not found the message printed on the console is:

STR NOT FOUND

The CP is now positioned at the start of an empty buffer.

For example, the CP is positioned before the T in TEMPORARY on line 5 of the Sample Program (Figure 2-1), and the user issues the command:

N.MPYA$$

The Editor searches throught the current buffer, punches it, yanks the next page and continues the search until the string is encountered. The CP is then positioned after the A in .MPYA.

Another way to perform a continuous search for a string is the command:

↑Qstring$

This is the same as the N command except the pages before the page in which the string is found are not punched.

--- C

The change command has the format:

Cstring$string2$

This searches for **string**₁, deletes it and inserts **string**₂. The CP is positioned after the last character of **string**₂.

When the command:

Cstring$$

is issued the Editor searches for the string and deletes it.

If the string is not found, the message printed on the console is:

STR NOT FOUND

For example, the CP is positioned after the R in STORAGE on line 5 of the Sample Program (Figure 2-1).

CAGE$ED$$

produces:

(5)   .BC11   .BLK 3   ;TEMPORARY STORAGE

--- I

The command to insert the string at the present CP position is:

Istring$

The CP is positioned directly after the last character of the string.

For example, the CP is positioned after the N of RETURN in line 1 of the Sample Program (Figure 2-1).

IS$$

produces

(1)   .CC03:   0   ;SAVE RETURNS

To insert an ASCII character into the text, using the decimal representation, issue the command:

nI$$

where: **n** is the decimal representation of an ASCII character.

To insert an ASCII character into the text, using the octal representation, issue the command:

"nI

where: **n** is the octal representation of an ASCII character.

For example, the CP is positioned directly after the N of RETURN in line 1 of the Sample Program (Figure 2-1).

83I$$

produces:

(1)   .CC03:   0   ;SAVE RETURNS

whereas

"123I$$

produces:

(1) .CC03:   0   ;SAVE RETURNS

NOTE
The **nI** command is primarily used to insert lower case ASCII characters into a text, using an upper-case keyboard.

--- **nD**

The command to delete a number of characters before or after the present position of the CP is:

nD$

where: **n>0**   Deletes **n** characters to the right of the CP.
         n<0   Deletes **n** characters to the left of the CP.

The CP is positioned immediately after the deleted character.

For example, the CP is positioned after the E in RESULTS in line 2 of the Sample Program (Figure 2-1).

-2D$$

produces:

(2)   .BC10:   ,BLK 4   ;SAVE ABS(U), SULTS OF MULTIPLY, AS RE-

whereas:

    2D$$

produces:

    (2) .BC10 .BLK4 ;SAVE ABS(U) ,
    RELTS OF MULTIPLY, AS RE-

--- nK

The command to delete whole lines from the buffer is:

    nK$
where: **n>0** Deletes **n** lines forward from the present CP
                 position
        n<0 Deletes **n** lines backwards plus any charac-
                 ters to the left of the CP in the current line
                 from the present position of the CP.

The CP is positioned after the last deleted character.

For example, the CP is positioned at the beginning of line 3.
To kill lines 3 and 4 issue the command:

    2K$$

--- Macro Command

A macro command is a command which contains an
arbitrary command string. The DGC Text Editor provides
for the definition and execution of a macro command.

The command to define the contents of the macro
command is:

    XMcommand$_1$$...command$_n$$

where **command** has the format:

    [n] code [string$]

The macro definition is terminated by a double escape;
therefore it must be the last command in a command string.
Once the macro command is defined, it is retained by the
Editor and can be called for future execution by means of a
single code. The Editor allows for the handling of only one
macro definition at a time.

Execution of the macro command is accomplished by
issuing the command:

    nX$

where: **n** is the number of times the macro command is to
        be executed.

The command to delete a macro command is:

    XD

If the macro command is undefined or the macro is
recursive, (e.g., XMX$$ 1000X$$) the resulting error
message printed on the console is:

    MACRO ERROR
For example, the CP is positioned at the very beginning of
the buffer containing the first 5 lines of the Sample Program
(Figure 2-1), the command:

    XMCAEII$$
    1X$$

produces:

    (1) .CC03: 0 ;SAVE RETURN

The command to type out the contents of the current
macro is:

    X?$

RDOS FILE COMMANDS

--- UD

The command to delete a specific file is:

    UDfilename$

where: **filename** is the name of the file to be deleted.

An example of this is:

    UDSAMPLE$$

This deletes the Sample Program (Figure 2-1).

--- UR

The command to rename a file is:

    URfilename$_1$$filename$_2$$

where: **filename₁** is the current name of the file.
filename₂ is the name it is to be changed to.

For example,

    URSAMPLE$PROGRAM$$

changes the name of the Sample Program (Figure 2-1) to PROGRAM.

--- UZ

The command to change the attributes of a file to nonpermanent is:

    UZfilename$$

where: **filename** is the name of the file to be changed.

For example,

    UZSAMPLE$$

changes the attributes of SAMPLE to non-permanent.

--- U:directory name

The command to append a directory name to all file names. For example, while operating in directory DC0 with directory STUFF initialized, a series of files in STUFF are available for edit when the following command is used:

    U:STUFF$$

The Editor then uses the name STUFF:PROG1 to attempt to open the file for reading if a GRPROG1$$ is issued.

SPECIAL EDITING COMMANDS

COMMAND	MEANING
:	Print the number of lines currently in the edit buffer.
.	Print the number of the line the CP is pointing to.
=	Print the number of characters contained in the edit buffer.
↑Istring$	Insert **string** with tabulation.
↑A	Terminates an EDIT command but does not cause an exit from the Editor.
↑Z	In a search string matches all characters (i.e., .BC↑Z searches for any occurences of .BC with any character that follows it.)

COMMAND	MEANING
U?	Type out the I/O file names.
CANCEL (↑X)	Delete the current line of the command line. When CANCEL is issued in the first line, delete it and restart the command. This can be repeated to delete multiple lines.
n#←	Reset the register to **n**, where **n** may be from 1 to 5 digits in the range 0 n 65,535.
#+	Increment the register by 1.
#-	Decrement the register by 1.
#?	Type out the contents of the register.
#0	Output the register as a five digit unsigned integer to the text buffer at the current location of the CP.
n#!...$!...$	When the register does not match the number argument, skip the command characters up to the next escape followed by an exclamation.
↑C	The ↑C is an RDOS break character. Once it is issued it terminates the Editor and returns command to the CLI.
n<>	Angle brackets are used to iterate an enclosed command n times. If n=0 the operation contines indefinately or until a failure occurs.

Examples:
To change the next 64 lines to double spacing;

    64<C <cr>
    $ <cr>
    <cr>
    $>$$

To insert COMMENT; until the Editor runs out of memory space;

    <ICOMMENT; <cr>
    $>$$

WINDOW MODE COMMANDS

nW	If currently in the page mode, this sets the mode to window mode with a width of n lines. If in the window mode it resets Editor to page mode.
W?	Displays the page/window mode status.
W	Resets the mode to page mode from window mode. If the Editor is already in page mode the resulting error message is:

    ILLEGAL WINDOW WIDTH

OUTPUT COMMANDS

--- E

The command to output the contents of the buffer as well as the remainder of the input file to the output file is:

    E$

For example, if the user makes a correction to the first line of the Sample Program (Figure 2-1), and wishes to copy, as is, the rest of the file, issue the command:

    E$$

--- F

The command issued to output a form feed to the output file is:

    $F

When an argument is given:

    nF$$

n inches of leader is output. If n is greater than 100, only 100 inches of leader is output.

Neither F nor nF has any effect on the CP.

--- P and PW

The command to output the entire buffer to the output device ending with a form feed is:

    P$

If a numeric argument precedes the P command:

    nP$

then starting from the CP, n lines will be output ending with a form feed. When operation in window mode there is no form feed. When n is greater than the number of lines contained in the buffer, punching will stop at the end of the buffer.

The command issued to eliminate the ending form feed in the above commands is either:

    PW or
    nPW

None of the Punch Commands have any effect on the CP.

--- nR

The command to output n pages and read in n pages is:

    nR$

This is equivalent to PY...PnYn. If the n argument is not given then one page is output and one page is read into the buffer.

For example:

    R$$

outputs the contents of the first buffer of the Sample Program (Figure 2-1) and reads in the contents of the second buffer.

--- GW

The command to specify an output file is:

    GWoutputfilename$

This will get for writing (output) the specified file **outputfilename**. The **outputfilename** cannot already exist.

The Editor issues the following error messages:

    OUTPUT FILE ALREADY ACTIVE
        if the output file is active and has not been closed

    FILE NAME IN USE
        if the file name specified already exists

    FILE CAN'T BE USED FOR OUTPUT
        if the file is write-protected

    ILLEGAL FILE NAME
        if the specified file name does not conform to legal
        RDOS file name specifications.

For example,

    GRSAMPLE$GWPROGRAM$$

Opens the input file SAMPLE for reading and the output file PROGRAM for writing.

~ GC

The command issued to close the input and output file upon completion of the editing of an input file to produce a new output file is:

GC$

WARNING
If return to the operating system takes place without closing the output file, it has a byte count of zero.

When an output command is issued and no output file has been specified the resulting error message is:

NO OUTPUT FILE

To include the remaining input in the output file a P or E command must be issued before the GC command. The GC command does not force the writing of the last output page.

NOTE
Multiple files may be appended to produce one output file. This is accomplished by successive use of the GR command without declaring a new output file.

--- !D

A return can be made to the CLI, upon completion of the editing by issuing this command:

!D$$

--- GO

The command to close the current output file, opened by GW, and open a new output file is:

GOfilename$$

where: filename is the file to be opened.

For example,

GRSAMPLE$GWEXAMPLE$$
.
.
.
GOPROGRAM$$

closes EXAMPLE and opens PROGRAM

--- UE

The command to copy the remaining input file, then close the input and output files, delete the input file and rename the output file with the input file name is:

UE$

For example,

UYSAMPLE$$
.
.
.
UE$$

copies the rest of SAMPLE, closes SAMPLE and SAMPLE.SC, deletes SAMPLE and renames SAMPLE.SC to SAMPLE.

--- US

The command to copy the input file until its end and close the input and output files is:

US$

This also renames the input file name to **filename**.BU, renames the output file to the input file name, restores the attributes of the input file and deletes the previous file.BU, if any.

For example,

UYSAMPLE$$
.
.
.
US$$

copies SAMPLE until its finish, closes SAMPLE and SAMPLE.SC, renames SAMPLE to SAMPLE.BU and SAMPLE.SC to SAMPLE and restores the attributes to SAMPLE.

--- UC

The command to close input and output files, restore the input file's attributes and rename the output file with a specified name is:

UCfilename$

where: **filename** is the specified name the output file is to be changed to.

For example,

GRSAMPLE$GWPROGRAM$$
.
.
.

UCEXAMPLE$$

closes SAMPLE and PROGRAM, restores the attributes to SAMPLE and renames PROGRAM to EXAMPLE.

--- UH

The command to close input and output files opened by UN or UY is:

UH$

CONTENTS OF FIRST BUFFER
(1)	.CC03:	0	;SAVE RETURN \<cr\>
(2)	.BC10:	.BLK 4	;SAVE ABS(U), RESULTS OF MULTIPLY \<cr\>
(3)			;AS REMAINDER OF DIVIDE \<cr\>
(4)	.BC11:		;SAVE ABS(V) \<cr\>
(5)	.BC11:	.BLK 3	;TEMPORARY STORAGE \<cr\>

CONTENTS OF SECOND BUFFER
(6)	.BC13:	0 \<cr\>
(7)	.BC14:	0 \<cr\>
(8)	.BC20:	42 \<cr\>
(9)	.BC30:	.MPYU \<cr\>
(10)	.BC31:	.MPYA \<cr\>
(11)	.BC32:	.BC10 \<cr\>

filename: SAMPLE

Figure 2-1. SAMPLE PROGRAM

CHAPTER 3

ERROR MESSAGES

During the editing process various errors may occur; the outputed error messages and their meanings are shown below. Where error messages are only output in a given operating environment, the appropriate environment(s) are listed in parentheses. Other error messages are applicable in all operating environments.

ERROR MESSAGE	MEANING
COMMAND BUFFER FULL; EXECUTING COMMAND. TEXT BUFFER FULL DURING INSERT	Command string exceeds capacity of edit buffer.
TEXT BUFFER FULL WHILE READING	Attempting to append a page when the buffer is full. During a read or append the buffer capacity is exceeded. A partial page has been read in or appended.
FILE CAN'T BE USED FOR INPUT	Attempt to read a read-protected file. (RDOS, SOS)
FILE CAN'T BE USED FOR OUTPUT	Attempt to write to a write-protected file. (RDOS, SOS)
FILE NAME IN USE	Attempt to create an output file when that file name already exists in the file directory. (RDOS)
ILLEGAL FILE NAME	File name does not conform to a legal operating system file name. (RDOS, SOS)
ILLEGAL WINDOW WIDTH	Editor is already in page mode when a W command was issued. (RDOS)
MACRO ERROR	Undefined or recursive macro.
NO OUTPUT FILE	Attempt to issue output command, without first specifying an output file. (RDOS, SOS)
NO SUCH FILE	Attempt to specify an input file which doesn't exist. (RDOS, SOS)
OUTPUT ALREADY ACTIVE	Attempt to get for writing an output file which has not been closed, and is still active. (RDOS, SOS)
PARITY ERROR IN LINE n	During a read, a parity error occurred in line n. When examined, the character in error will be replaced by " ".
STR NOT FOUND	Unsuccessful string search.
??command string	Editor cannot understand or cannot execute command. It outputs the remainder of the string to which the message refers. If this message occurs while using one of the operating systems and the command buffer contained at least 256 characters when this message occured, the following message will be printed, and the user may follow the procedure outlined to recover his command buffer:
SAVE COMMAND BUFFER YES (1) OR NO (2) ?	The command buffer contained at least 256 characters when either an illegal command or an RDOS interrupt was detected. If the user responds with anything other than a 1 <cr> then the command buffer is deleted and the prompt character "*" is reissued to await a new command.

If the user responds with
1 <cr>
the message:
ENTER FILE NAME
is printed instructing the user to specify the file name in the following manner:
xxx <cr>
where xxx is any valid output file. The contents of the command buffer will then be written to the specified file. When this operation is complete the Editor issues the prompt character "*" to await a new command.

If the specified filename is not valid output file, the message:
ENTER FILENAME
will be printed again.

APPENDIX A1-A

COMMAND SUMMARY

COMMAND	MEANING	PAGE
↑P	Change tab simulation (initially simulated).	1-2
RUBOUT	Cancel and echo previous character entered.	2-1
.	Display the line number the CP is pointing to.	2-7
:	Display the number of lines in the buffer.	2-7
=	Display number of characters in the buffer.	2-7
Istring$	Insert with tabulation in front of CP.	2-7
A	Append a page, or windows to buffer.	2-2
B	Move CP to beginning of the buffer.	2-3
Cstring$string$	Change **string**, set CP.	2-5
nD	Delete **n** characters ahead of CP.	2-5
E	Copy contents of input file to output file.	2-8
F	Issue a form feed to output file.	2-8
nF	Output **n** inches of leader.	2-8
Istring$	Insert **string** in front of CP.	2-5
nI	Insert character using the decimal representation.	2-5
"nI	Insert character using octal representation.	2-5
nJ	Jump CP to line **n**.	2-3
nK	Delete **n** lines ahead of CP.	2-6
L	Move CP to beginning of the line.	2-3
nL	Move CP **n** lines from CP.	2-3
nM	Move CP **n** characters.	2-3
Nstring$	Perform a continous search for **string**, starting at CP, punch pages.	2-4
P	Punch the buffer ending with a form feed. In window mode, just output the buffer.	2-8
nP	Starting at CP, punch **n** lines ending with a form feed in window mode, just output **n** lines.	2-8
PW	Punch the buffer.	2-8
nPW	Punch **n** lines, starting at CP.	2-8
↑Qstring$	Perform a continuous search for **string**, from CP.	2-4
nR	Punch **n** buffers and read **n** pages or window frames.	2-8
Sstring$	Search for string, beginning at CP.	2-4
T	Type buffer.	2-4
nT	Type **n** lines of buffer, starting at CP.	2-4
XM	Create a macro.	2-6
nX	Execute a macro **n** times.	2-6
XD	Delete a macro command.	2-6
X?	Types out contents of current macro.	2-6
↑Y	Yank a page or window frame.	2-1
nY	Yank **n** lines.	2-2
Z	Move CP to end of buffer.	2-4

COMMAND	MEANING	PAGE
n#←	Resets the number register to **n**.	2-7
#+	Increments the register by 1.	2-7
#−	Decrements the register by 1.	2-7
#?	Types the contents of the register.	2-7
#0	Outputs the register to the text in front of CP.	2-7
n#!...$!...$	If the register doesn't match **n**, skip the command characters up to the next exclamation escape.	2-7
↑A	Terminates an EDIT command.	2-7
↑C	Terminates the Editor.	2-7
↑D	Return to CLI.	2-9
↑Z	Search for string matching all characters.	2-7
CANCEL (↑X)	Scraps the current line of the command.	2-7
GC	Closes the input file and the output file without data transfer.	2-9
GRfilename$	Gets for Reading, closing the previous input file if any.	2-2
GOfilename$	Closes current output file and opens new output file.	2-9
GWfilename$	Creates and opens a new file for writing.	2-8
n<>	Iterate enclosed command n times.	2-7
U?	Types I/O file names.	2-7
UCfilename$	Closes I/O files, restores input file's attributes and renames the output file.	2-10
UDfilename$	Deletes a file.	2-6
U:directory name	Appends a directory name to all files.	2-7
UE	Punches remaining input file, closes I/O files, deletes input file, and renames output file according to original input.	2-9
UH	Closes I/O files, resets attributes.	2-10
UNfilename$	Creates a permanent file and opens it for input; opens **filename**.SC for output.	2-2
URfilename$	Renames a file.	2-6
US	Punches remaining input file, closes I/O files, restores the attributes of the input file, changes its name extension to .BU, and rename the output according to original input.	2-9
UYfilename$	Makes **filename** permanent, opens it for input and yanks in the first page or window frame. Creates and opens **filename**.SC for output.	2-3
UZfilename$	Removes the permanent attributes of a file, so it can be renamed or deleted.	2-7
W	Resets mode.	2-7
W?	Displays mode status.	2-7
nW	Set window mode with a width of **n** lines.	2-7
GC	Closes input file, and output file.	2-9
GR	Get for Reading, closing previous input file if any.	2-2
GW	Get for Writing.	2-8

A1

Text Editor Index

APPENDIX A

SECTION A2

CALMA STATION EDITOR
(RDEDIT)
USER'S MANUAL

1.1 DESCRIPTION

This editor allows character string oriented editing of ASCII RDOS files. It supports the station keyboard for input and the CRT and TTY for output. It is basically the Data General Editor extended and modified by CALMA.

1.2 CONVENTIONS

Upper case text in the operating instructions must be typed exactly as shown. Lower case text is to be replaced with the appropriate argument(s).

↑ letter means holding down the control key and typing the letter on the TTY or typing a special key with that function on the station keyboard or typing SSB special string begin (SSB) then the letter on the station keyboard.

The following special characters will be referred to use the convention <...>.

<RUBOUT>	Rubout (or BS on station keyboard)
<CR>	↑M Carriage Return
<LF>	↑J Line Feed
<FF>	↑L Form Feed
<TAB>	↑I Horizontal Tab

$ is the printable echo for ESCape (COORD ENTER on station keyboard).

±num means any decimal number, negative or positive.

str means an arbitrary string of 0 or more characters, excluding $, ↑C, ↑G, ↑K.

CP	Character Pointer
MP	Merge Character Pointer

1.3 CLI COMMAND

To run RDEDIT the following CLI command is used:

    RDEDIT [n/S] [filename]

where: n is the number of the station, the default is 0; the first station.
filename is used to execute a GBfilename$↑Y$$ upon entering RDEDIT.

1.4 MESSAGES

The "*" is a prompt indicating the editor is ready to accept command input.

The message "LONG STRING NOT INSERTED - TYPE 'I' or '$' or '↑C↑C' is written to indicate that a string greater than 64 characters long was typed in that did not start with an I or an ↑I (tab). Typing an I will cause the complete message to be executed as if preceded by an "I", i.e., forced to an insert. If the line is to be executed as it was typed, input ESC ($). The ↑C↑C input effectively erases the complete line.

2.1 COMMANDS

Commands are of the form:

CM
num CM
±num CM   where CM is the command keyword
CM str$ string of one or two characters
CM str$ str$

Commands may be placed one after another forming a command string. No command in the string is executed until the command string is terminated by two $'s (including any needed to form part of the last command in the string). To delete the entire command string, type ↑C (or ↑G). To delete the current line (back up to last <CR>), type ↑K. To delete the last character of the command string, type <RUBOUT>, this will cause the deleted character to be typed. To exit to RDOS type ↑D $$.

Line feeds are always ignored on input from both the disk and the keyboard, and are not echoed.

All command characters may be typed in lower case with the same effect; argument strings must be in upper or lower case depending on the effect desired.

The cursor on the CRT indicates the current position to be typed next during input.

The lights on the station keyboard are used to indicate the current input status.

ACK means RDEDIT is waiting for input.

NAK means last command was in error.

INPUT LOST means TYPEIN was lost.

X OFF SCALE means software lower case is active.

3.1 DISK I/O SELECTION

GR filename $
Select the RDOS file (filename) for input. Position the file pointer to the beginning of the file. If the extension is omitted, .SR is assumed for commands GR, GB, and GM.

GW filename $
Select the RDOS file (filename) for output. If a previous output was selected by a command GB or GW, it must be closed with the GC command before a new output file may be selected.

GC
Close the output file and enter into the RDOS directory.

GB filename $
Edit back up the RDOS file (filename). This appears as GR and GW, but does not cause an error because the file already exists. When GC is done, any existing filename .BK is deleted, the input file is renamed to filename .BK, and the output file is entered into the directory using the name (filename).

GM filename $
Select the RDOS file (filename) as the input merge file. Position MP to the beginning of the file.

↑Y
Yank - clear the text buffer, and read in from input file until <FF> is encountered or until the buffer is full. Position the input file PTR past <FF> or last read. Position the CP a the beginning of the text buffer.

A
Append - Insert a <FF> at the end of the buffer and read in from input file appending to end of text buffer until <FF> or until buffer fill. Position input file PTR past <FF> or last read. The CP is not moved.

AW
Same as A, but no <FF> is inserted in the buffer.

num P
Punch - output the text buffer to output file. If num = 0 or is omitted, output the entire buffer. If num > 0, then output num lines from CP; in no case going beyond the end of the buffer. After outputting text, a <FF> is output.

num PW
Same as P, but no <FF> is output at the end.

num R
If num = 0, 1 or is omitted, perform P↑Y. If num > 1, perform P↑Y num times.

E
End - perform R commands until all of the input file is copied to output.

NOTE
All G commands are also typed on the TTY to provide a record of disk I/O.

4.1 CP POSITIONING COMMANDS

B
Beginning - position CP to beginning of buffer.

Z
Position CP to end of buffer.

num J
Position CP to beginning of the (num)th line.

±num L
If num = 0 or is omitted, position CP to beginning of current line; otherwise, move CP num lines forward if num > 0 or backwards if < 0. If CP is moved off beginning or end of buffer, then CP is positioned to the beginning or end of the buffer.

±num M
Position CP by num characters. If off beginning or end, then CP is positioned to beginning or end of the buffer.

5.1 INSERT/DELETE COMMANDS

num I
Insert the ASCII character represented by num at CP; space CP past the inserted character.

Istr$
Insert str at CP; space CP past str inserted.

<TAB>str$
Insert <TAB> str at CP; space CP past str inserted.

±num D
Delete num characters from CP; forward if num>0, backwards if num<0.

±num K
Delete num lines from CP; forward if num>0, backwards if num<0.

6.1 TYPEOUT COMMANDS

±num T
If num = 0, or is omitted, type out entire buffer to CRT.
If num>0 type out num lines starting at CP.
If num<0 type out num lines before the character at CP.

=
Type number of characters in buffer.

:
Type number of lines in buffer.

.
Type number of lines containing CP.

±num 0
Like T except output is directed to TTY.

7.1 SEARCH COMMANDS

Sstr$
Search - search buffer starting at CP for the string str. If found then space CP past the first occurrence of the string. If not found, the CP is unchanged and error results (except when followed by ; in interation brackets see ;).

Nstr$
Same as S except if search fails, perform R command and continue.

↑Qstr$
Same as S except if search fails, perform ↑Y command and continue.

Cstr1$str2$
Change - like Sstr$ except if search succeeds delete str1 and insert str2 in its place.

8.1 MACRO COMMANDS

XMstr$
Define macro as str.

XD
Delete macro definition.

numX
Execute defined macro num times.

9.1 MERGE FILE COMMANDS

±numUT
Type num lines from MP in merge file, forward if num<0, backwards if num>0.

±numUL
Position MP by num lines. If the resulting MP is before the beginning, set MP to beginning of file; if result is after end, set MP to end of file.

numUI
Insert num lines from merge file at MP into buffer at CP. Position MP and CP past inserted text.

UA
Insert from merge file at MP into buffer at CP up until the next <FF>. Position CP past inserted text and MP past the <FF>.

UNstr$
Search for str in merge file starting at MP. If search succeeds, space MP past first occurrence of str. If search fails, MP is unchanged and error results.

UB
Position MP to beginning of file.

UZ
Position MP to end of file.

±numUO
Like UT, except output is directed to TTY.

10.1 ITERATION COMMANDS

num<str>
If num = 0 or is omitted, perform the command(s) in str indefinitely until one of the following occurs: an error occurs, the ; or ↑ is executed, a ↑C is typed. If num>0, then perform the commands in str num times, or until one of the conditions above occurs. Up to 5 levels of <...>, may be nested.

search command;
If inside of interation brackets and the search command fails, continue command execution after the next following >. Search commands are C and S. If encountered outside of interation brackets it is an error.

Exit from iteration and return to command input mode. If encountered outside of iteration brackets, it simply ends the command execution.

11.1 CRT SCROLLING

On input, if typing goes off the bottom of the page, the last 20 lines will be rewritten at the top of the screen after it is erased. Any type in during the erase period will not be lost; therefore, the programmer need not stop typing during the rewrite.

All type out including echoing of input is saved for CRT scrolling, except the special characters ↑E and ↑G.

If ↑E is typed during command input, the last 10 lines typed as well as any characters typed so far in the current line are rewritten at the top of the screen after it is erased. There is no effect on the command string being collected.

If ↑G is typed during command input, the screen will be erased, and the current command string deleted. No type in is lost after ↑G and before "*" is typed.

11.2 WINDOWING

After any command string is executed that changes the contents of the text buffer or moves the buffer pointer (CP), the new contents of the buffer will be typed for several lines on either side of the CP; windowing is not done if the command string contains a type (T) command. In the windowing typeout, the "↑" indicates the position of the CP. The nW command sets the window width to ±n lines; a total of 2n lines are typed unless the CP is less than n lines from the beginning or end of the buffer.

12.1 MISCELLANEOUS

↑C will cleanly abort any command. If the E or num R command is interrupted, then the buffer will be left unpunched after the last ↑Y. Any typein between ↑C and the appearance of the "*" prompt will be ignored even if it is echoed.

↑D exits and returns to RDOS.

Figure 2-1 shows where the special characters necessary for editor operation are located on the station keyboard.
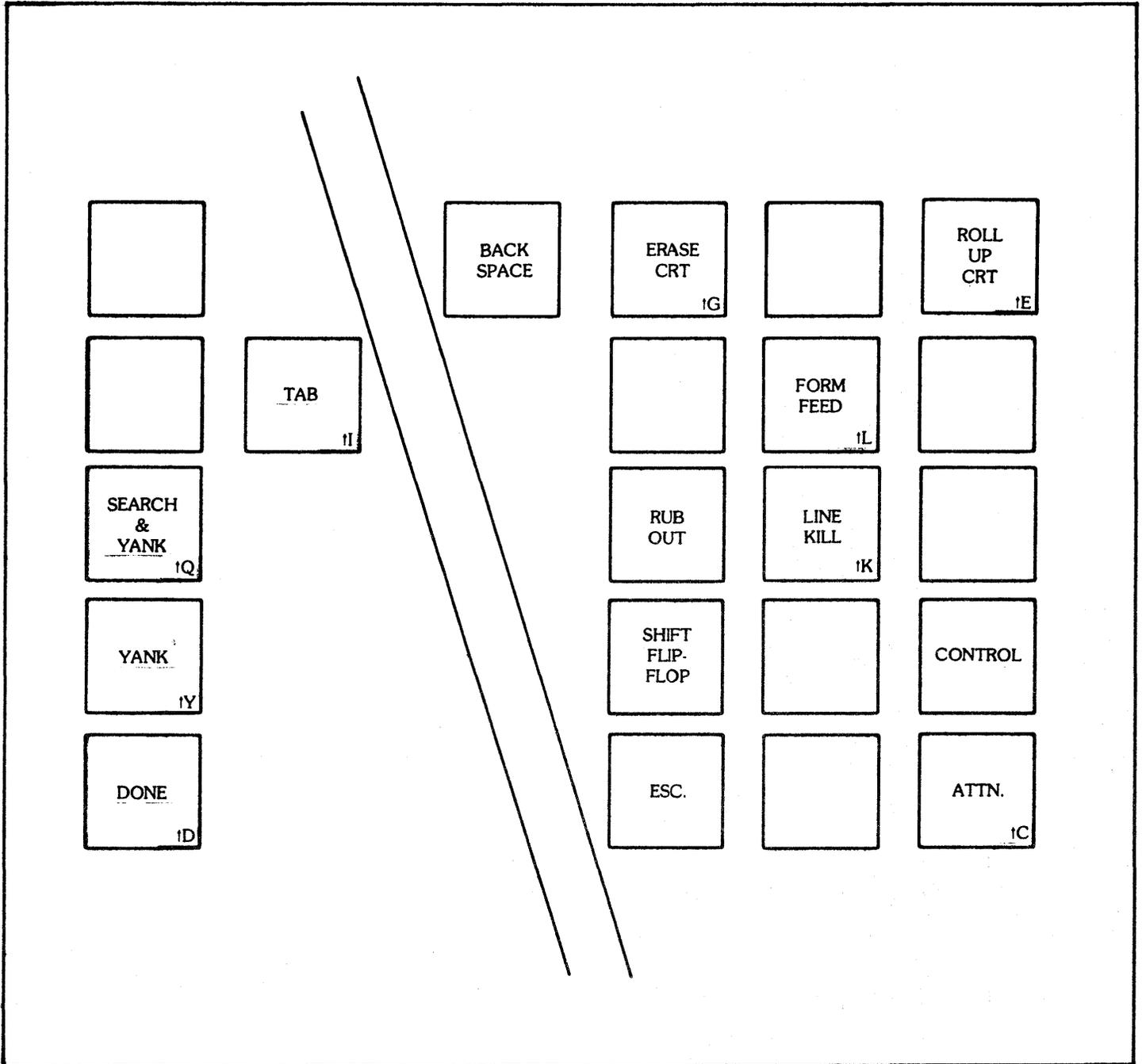
Figure A2-1.   CALMA Keyboard Special Editor Characters

13.1 SPECIAL INPUT CONTROL CHARACTERS

↑C	Abort current command and return to input mode; also reset lower case mode.
↑E (View Again)	Roll up screen and retype 10 lines.
↑G (View Next)	Erase screen and flush any command input so far.
↑K	Delete current line of command input up to, but not including, the last <cr>.

SHIFT F-F	Complement the lower case mode.

NOTE

In lower case mode all alphabetic input input is interpreted as being lower case.

RUBOUT/BS	Delete last character in buffer and type it to CRT.
CONTROL	Interpret next input character as a control character (shown as ↑ char).

13.2 COMMAND SUMMARY

(All of these may appear in the command string.)

COMMAND	FORM	FUNCTION
<>	n<str>	n>0 repeat str n times or until error. n=0 repeat string until error.
A	A	Append a page to the edit buffer.
AW	AW	Append without <FF>.
B	B	Position CP to beginning of the edit buffer.
C	Cstring1$string2$	Change string1 to string2.
D	nD	Delete n characters starting at CP.
↑D	↑D	Exit and return to RDOS.
E	E	<Punch, yank> to end-of-file.
GB	GBfilename$	Open filename as current input file rename to filename.BK and open filename.SO as output file.
GC	GC	Close the output file.
GM	GMfilename$	Open filename as the current merge file and position MP to first line.
GR	GRfilename$	Open filename as the current input file and close any previous input files.
GW	GWfilename$	Open filename as the current output file.

COMMAND	FORM	FUNCTION
I	nI	Mask n to seven bits and insert at CP.
	Istring$	Insert string starting at CP.
↑I	↑Istring$	Tab then insert string starting at CP.
J	nJ	Jump to beginning of line n.
K	nK	Delete (KILL) n lines from current CP.
L	nL	Position CP n lines from current CP.
M	nM	Position (move) CP n characters from current CP.
N	Nstring$	Search for string; if not found, punch read, and continue search.
O	nO	Output n lines to line printer.
P	P	Punch entire edit buffer followed by a form feed.
	nP	Punch n lines from current CP followed by a form feed.
PW	PW	Punch entire edit buffer.
	nPW	Punch n lines from current CP.
↑Q	↑Qstring$	Search for string; if not found, yank and continue search.
R	nR	Perform PY n times.
S	Sstring$	Search for string.
↑T	n↑T	Set <TAB> stops to n (1≤n≤15).
T	T	Type entire edit buffer.
	nT	Type n lines from current CP.
UA	UA	Insert from merge file at MP into buffer at CP.
UB	UB	Position MP to start of file.
UI	nUI	Insert n lines from merge file at MP into buffer at CP.
UL	nUL	Position MP by n lines.
UO	nUO	Type n lines from MP to TTY.
UT	nUT	Type n lines from MP to CRT.
UN	UNstring$	Search merge file for "string"; position MP past "string".

COMMAND	FORM	FUNCTION
UZ	UZ	Position MP to end of file.
W	nW	Set window width to n (n=0, turn off windowing).
X	nX	Execute the define macro n times.
XD	XD	Delete the current macro definition.
SM	SMstring$	Define string to be a macro command.
↑Y	↑Y	Yank a page into the edit buffer.
Z	Z	Position CP to end of edit buffer.
=	=	Print number of characters in edit buffer.
:	:	Print number of lines in edit buffer.
.	.	Print line number wher CP resides.
<>	n<command>	Execute command n times (0≤n≤5).
;	Sstring$ Cstring1$string2$	If either search command fails inside of iteration brackets, start command again past next right iteration bracket.
"	"n	Interpret n as an octal number.
↑	↑	Exit from iteration.