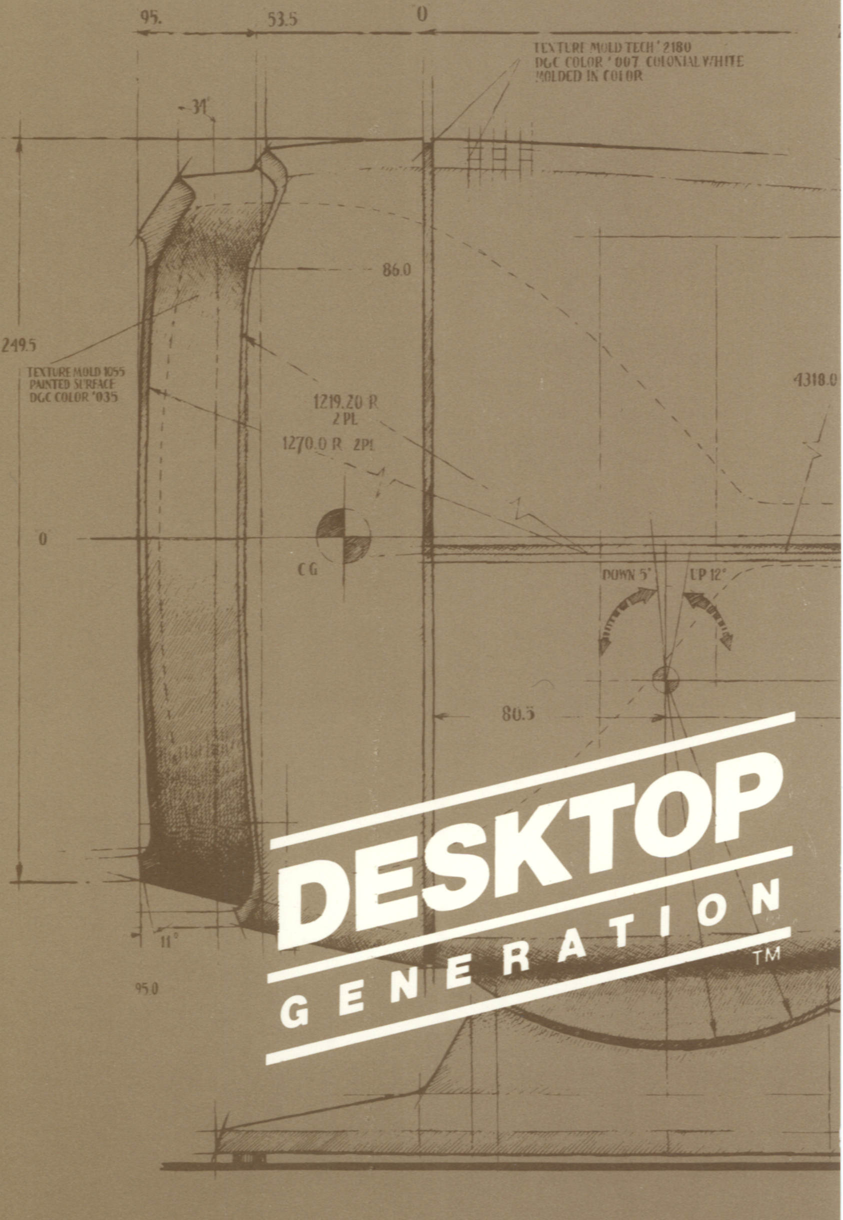
Data General



DESKTOP GENERATION™

Using DG/RDOS on DESKTOP GENERATION™ Systems

NOTICE

Using DG/RDOS on DESKTOP GENERATION™ Systems
069-000056-02

Revision History:	Effective with:
Original Release - September 1983	
First Revision - February 1984	
Second Revision - July 1984	DG/RDOS Rev. 1.10

Ordering No. 069-000056
©Data General Corporation, 1983, 1984
All Rights Reserved
Printed in the United States of America
Revision 02, July 1984

About this Book

DG/RDOS is Data General's (DG's) Real-time Disk Operating System for DESKTOP GENERATION™ computer systems.

In an hour or two, you can install DG/RDOS with support software. And in another hour, you can develop a working sense of DG/RDOS by using it. This book shows how — with enough information to get you started. Then, it lists other DG manuals that give more information.

Before you can start or run DG/RDOS (as shown in this book), you must have

- unpacked your DESKTOP GENERATION hardware;
- connected the components (computer, diskette, hard disk — if any — keyboard and display screen, printer, and user terminals, if any);
- plugged system components into appropriate ac outlets; and
- run a hardware test.

For Model 10 and 10/SP systems, these installation tasks and hardware operations are described in

Installing Model 10 and Model 10/SP Systems, part number 014-000901

Testing Model 10 and Model 10/SP Systems, 014-000902

Operating Model 10 and Model 10/SP Systems, 014-000900 (accompanies this manual for Model 10/SP systems).

For Model 20 and 30 systems, installation and hardware operations are covered in

Installing Model 20 and 30 Systems, part number 014-000904

Testing Model 20 and 30 Systems, 014-000905

Operating Model 20 and 30 Systems, 014-000903 (accompanies this manual for Model 20 and 30 systems).

What Do You Want to Do?

This book assumes that you want to learn about using (or installing) DG/RDOS or related software on your DESKTOP GENERATION system. To use the MS™-DOS* or CP/M®-86** operating system, see the appropriate book, listed later in this preface.

To install DG/RDOS software, read Chapter 1 (for background) and Chapter 2. Or, to update an existing DG/RDOS system or add hardware to it, see Chapter 8.

For general DG/RDOS operations, like creating files and directories, moving files, printing them, and backing them up for safekeeping, you'll need the command language (CLI), and other programs. Read Chapter 1 for background. Then, for the CLI, check Chapters 4 and 5. To format or copy diskettes, read Chapter 6. To back up files, see Chapter 7.

To write computer programs, you may need Chapter 1 for background. Then, for BASIC, read Chapter 10. For a language other than BASIC, you need a text editor (Chapter 9) and Chapter 10. To understand the examples, you must have some experience with the language.

To recover from an error condition, or whenever you don't know what to do, see Chapter 14.

To find the meaning of a computer-oriented word you don't know, see the Glossary.

* *MS-DOS is a trademark of Microsoft® Corporation.*

** *CP/M-86 is a registered trademark of Digital Research.*

How is this Book Organized?

Chapter 1	introduces the DG/RDOS software; describes the programs available with it; and offers some cautions and hints.
Chapter 2	tells how to install DG/RDOS — for the first time — on a blank hard disk or diskette. It then explains how to configure DG/RDOS for specific hardware and software.
Chapter 3	explains the steps you take to start up and shut down DG/RDOS on your DESKTOP GENERATION system.
Chapter 4	shows how to use the DG/RDOS command language (CLI) in a hands-on session.
Chapter 5	explains the file system, then describes common CLI commands and programs shipped with DG/RDOS.
Chapter 6	shows how to format and copy diskettes.
Chapter 7	tells how to back up your system's files for safekeeping and how to restore these files.
Chapter 8	explains how to handle hardware upgrades and software updates (new hardware, software, and software revisions acquired from DG).
Chapter 9	shows how to run the text editor named SPEED and describes other programs in the optional Development Kit.
Chapter 10	shows how to build computer programs in the BASIC, FORTRAN, and COBOL languages.
Chapter 11	explains how to use the DG/RDOS Sort/Merge program (CSSORT). This chapter may interest you if you have some programming experience.
Chapter 12	introduces the optional programs — like DG/BLAST and RJE80 — that enable your system to communicate with another computer system.
Chapter 13	outlines DG's optional graphics hardware and software available with DG/RDOS.

Chapter 14	describes error conditions, error messages, and how to recover from errors. It also gives the DG phone numbers to dial for help.
Glossary	defines pertinent terms, like *asynchronous line.* When you see a term you don't know, check the glossary.
Fast-Reference Summary Card	summarizes commands and programs; and startup and shutdown steps. It appears before the back cover.

For quick reference, insert and use the *tabbed dividers* packed with this book.

The Release Notice

The Release Notice is a document — packed with this manual — that describes product organization (files shipped, tips and techniques). You may want to check the Release Notice to ensure that you have all the documentation for your model of DG/RDOS.

Reader Please Note:

In this book, *system console* means specifically the operator's terminal — the one from which you bring up DG/RDOS. Other terminals on the system are called *user terminals.*

We use these conventions for command formats:

`COMMAND required` *`[optional]`* ...

Where	**Means**
`COMMAND`	You must type the entry as shown.
`required`	You must enter some argument, like a filename. Sometimes, we use: $\left\{ \begin{matrix} \text{required}_1 \\ \text{required}_2 \end{matrix} \right\}$ which means you must enter one of the arguments. Don't type the braces; they only set off the choice.
`[optional]`	You have the option of entering this argument. Don't type the brackets; they only set off what's optional.

...	You may repeat the preceding entry or entries. The explanation will tell you exactly what you may repeat.

Also, we use certain symbols in special ways:

↲	Press the NEW LINE key.
CR	Press the CR key (some programs require CR, not ↲).
□	Be sure to insert a space here. (We use this only where we must; normally, you can see where to put spaces.)

All numbers are decimal unless indicated otherwise.

We show commands in both UPPERCASE and lowercase; but you can type them in lowercase, UPPERCASE, or any combination. In examples, we use

```
THIS TYPEFACE TO SHOW YOUR ENTRY ↲
THIS TYPEFACE FOR SYSTEM QUERIES AND RESPONSES.
R
```

R is the DG/RDOS system CLI prompt. It's in italics because the system displays it.

If you have comments on this manual, please use the prepaid Remarks Form that appears after the Index.

Related Documentation

After you read the parts of this book you need, you may want to check other DG manuals, perhaps for MS-DOS and related software. A list of related DG manuals follows.

Backup Utilities

- *RDOS/DOS Backup Utilities*, 069-400022

BASIC Language

- *Business BASIC Commands, Statements, and Functions,* 093-705005
- *Business BASIC Subroutines, Utilities, and BASIC CLI*, 093-705006
- *Business BASIC System Management*, 093-705007
- *Business BASIC Technical Concepts,* 093-705004
- *Extended BASIC User's Manual*, 093-000065
- *Loading and Managing Extended BASIC*, 093-000119

CLI Command Line Interpreter

- *RDOS/DOS Command Line Interpreter User's Manual*, 069-400015

COBOL Language

- *Interactive COBOL Programmer's Reference Manual*, 069-705013
- *Interactive COBOL User's Guide (RDOS)*, 069-705014
- *Interactive COBOL Utilities (RDOS)*, 069-705020
- *IC/EDIT Interactive COBOL Editor*, 055-004

Communications

- *Generating, Running, and Using DG/XAP*™, 093-000352
- *Generating, Running, and Using DG/GATE*™, 093-000353
- *How to Use DG/BLAST*, 069-100006
- *RDOS RJE80 User's Manual*, 093-000164

CP/M-86

- *Using CP/M-86 on DESKTOP GENERATION*™ *Systems*, 069-100007

DG/RDOS Development Kit and Sysgen RDOS

- *How to Load and Generate RDOS*, 069-000013
- *Introduction to RDOS*, 069-000011
- *RDOS/DOS Assembly Language and Program Utilities*, 069-400019
- *RDOS/DOS Debugging Utilities*, 069-400020
- *RDOS/DOS Superedit Text Editor*, 069-400017 (SPEED Editor)
- *RDOS System Reference*, 093-400027

FORTRAN

- *FORTRAN IV User's Manual*, 093-000053
- *FORTRAN 5 Reference Manual*, 093-000085
- *FORTRAN 5 Programmer's Guide (RDOS)*, 093-000227
- *FORTRAN Commercial Subroutine Package*, 093-000107

Graphics

- *Model 10/SP Monitor and Keyboard User's Manual*, 014-000770

MS-DOS

- *Using MS-DOS on DESKTOP GENERATION Systems*, 069-100025

Contents

Tables

Table

Illustrations

Figure

Introducing DG/RDOS Software

1

Read this chapter when

- you want some background on the DG/RDOS system;
- you want to know what programs are included and available with DG/RDOS, and what these programs do.

The major sections in this chapter are

- What is the DG/RDOS Operating System?
- What Programs Are Available with DG/RDOS?
- DG/RDOS Files
- Cautions and Keyboard Control Characters
- If You Make a Mistake
- What Next?

What is the DG/RDOS Operating System?

An operating system is a large group of computer instructions that allows people to communicate with a computer. You type *commands* on a terminal keyboard; the operating system translates these for the computer; the computer does what the operating system directs, then tells the operating system it is done; and the operating system displays a suitable message on the terminal screen.

The operating system is the lowest level of computer software; it *supports* higher levels, like word processors and computer languages.

DG/RDOS is a general-purpose operating system that runs on DESKTOP GENERATION™ Model 10, 10/SP, 20, and 30 computers. DG/RDOS is based on DG's popular RDOS operating system — it is a pregenerated or *pregen* version of RDOS.

DG/RDOS can run two different programs at once, on up to five terminals. It supports up to 2 megabytes (Mbyte) of main memory and up to two up to two hard disk and minidiskette drives, for a total of more than 75 Mbyte of disk storage. It can run one or more printers or plotters, and communicate with other systems over telephone lines. And, on a Model 10 or 10/SP computer, DG/RDOS can run concurrently with MS-™DOS or CP/M®-86 software.

For all this, DG/RDOS and its CLI (Command Line Interpreter) run entirely in main memory, without needing to access a disk. So — in addition to its processing power — DG/RDOS is fast.

What Hardware Does DG/RDOS Support?

DG/RDOS requires

- a Model 10, 10/SP, 20, or 30 computer with at least 128 kilobytes (Kbytes) of memory;
- a one or two diskette drive for 360-Kbyte diskettes;
- a system console.

Over the minimum hardware, DG/RDOS can support

- a 16-color system console, on Model 10 or 10/SP systems;
- additional main memory, up to a total of 2 Mbytes;
- a second hard disk, for a total of 78 Mbytes of hard disk storage;
- a multiplexor (USAM, model 4463) with up to four lines to communicate with another system, a printer, a plotter, or user terminals;
- an additional terminal to run a second CLI in the foreground;
- a dot-matrix serial printer, model 4434;
- a letter-quality printer, model 4518;
- a color plotter, model 4435;
- a cartridge tape unit;

and other devices.

How Do I Install DG/RDOS?

Your desktop hardware must be connected and ready to run as described in the hardware manuals named in the preface.

Then, using the DG/RDOS diskettes, you'll format a disk(ette), install DG/RDOS on it, and configure DG/RDOS. The procedure depends on your disk hardware:

> with one diskette drive and no hard disk, you simply install something called an *emulator* on your system diskette; and configure the system;
>
> with two diskette drives or a hard disk, you format the diskette or disk, install DG/RDOS it; and configure the system.

Generally, you need not repeat the installations after doing it. You may need to format or configure again if you acquire new hardware or new software.

How Do I Work with DG/RDOS?

To some extent, the answer depends — again — on the kind of disks and amount of disk storage you have.

With one diskette drive and no hard disk, you can run DG/RDOS and one or two other programs. After starting DG/RDOS, you can remove the DG/RDOS diskette and insert another, perhaps with programs like MS-DOS and WordStar, or application programs designed for your needs. You're limited to the space on one diskette, so you can't really build programs or create a lot of big text files.

With two diskette drives and no hard disk, you can do everything described above — and run other programs quite easily. Also, you can copy files to diskette for transfer to other systems or for backup. And you *do* have the space to build small programs in BASIC.

With a hard disk, you can do everything described above, and much more. You can create whole libraries of text files, letters, invoices, and other things — using DG/RDOS directly or using systems that run in the 8086 processor (with a Model 10 or 10/SP). You can run software like FORMA-TEXT™ and COMPUCALC™, and build and run programs in high-level languages like ICOBOL. The disk is big and fast enough to run programs for several users — which means that more than one person can use the system at the same time. If desired, you can create programs for execution on other DESKTOP GENERATION systems.

Whatever hardware you have, startup is easy. It involves pressing one or two switches, typing a number, pressing ↲, and typing the date and time on the system console.

If your system has multiple terminals, you can type commands to run a program on other terminals. Then you can start *using* the system — for word processing, spreadsheet calculations, program development, or whatever you wish. If you choose to run another program, you'll then give commands to *that* program.

Eventually, you'll want to stop working with the computer. You can simply walk away; or you can shut the system down by typing commands on the system console.

That's all there is to it.

What Programs are Available with DG/RDOS?

With your DESKTOP GENERATION computer, you bought the DG/RDOS operating system. Many other software programs are available with DG/RDOS.

The programs shown in Table 1-1 are included with every DG/RDOS system. The names proceed alphabetically.

The programs shown in Table 1-2 are included in the optional DG/RDOS *Development Kit.*

The *products* shown in Table 1-3 are available as options with DG/RDOS on DESKTOP GENERATION systems.

Table 1-1 Programs included with DG/RDOS

Program Name	Comments
Command Line Interpreter (CLI)	The CLI is the main program you use to communicate with the DG/RDOS system. When DG/RDOS starts up, it runs the CLI on the system console. It can also run the CLI on a second terminal. The primary CLI chapters are 4 and 5.
Backup programs	It's prudent to copy your own data files and programs onto diskette or tape, in case something happens to the originals. Several backup programs are shipped with DG/RDOS. This book emphasizes the one named IMOVE.
Configuration program (CONFIG.SV)	This program tailors the operating system to your specific hardware: user terminals, communication lines, printers, and so on.
Disk formatter program (DKINIT.SV)	Any hard disk or diskette must be software formatted before it can be used to hold DG/RDOS. Also, you may want to use diskettes as directories. The DKINIT formatter can help with these things. (Hardware formatting, needed for non-DG diskettes only, is done by a program on the Customer Diagnostics diskette.) Both hardware and software formatting are described in Chapter 6.
Diskette copy program (FCOPY.SV)	Often, you may want to copy a diskette. The FCOPY program does this easily. It can copy a diskette even if you have only one diskette drive.
System	The DG/RDOS operating system manages devices, like terminals, disks and printers, for people. The system name is DGRDOS.SV.

Table 1-2 Some programs in the Development Kit

Program	Comments
Library File Editor (LFE)	A library file editor (LFE) allows you to create your own programming libraries.
Loader (linker) program (RLDR)	The RLDR program builds compiled programs (like FORTRAN programs) into executable files.
Macroassembler (MAC)	The macroassembler is needed for assembly language programming.
Text editor program (SPEED)	A text editor allows you to create and edit text, including computer programs.

Table 1-3 Some other software products available with DG/RDOS (continues)

Product Name	Comments
BASIC	There are two BASICs available on DG/RDOS systems. Extended BASIC is a traditional BASIC, and Business BASIC is a business-oriented BASIC with screen handling and ISAM file capability. Using these BASICs is explained in Chapter 10.
COBOL	Interactive COBOL (ICOBOL) runs on all DG machines — from 32-bit ECLIPSE MV/10000 systems to DESKTOP GENERATION Model 10s. ICOBOL has its own text editor (IC/EDIT) and ISAM file management features. It is introduced in Chapter 10.

Table 1-3 Some other software products available with DG/RDOS (concluded)

Product Name	Comments
Communications products	Two products, DG/XAP™ and DG/BLAST, can move files over asynchronous communications lines between DG computer systems. DG/GATE™ allows a DESKTOP GENERATION system to emulate a terminal — and tie into into public electronic bulletin boards, which offer highly diverse services. RJE80 RJE80 emulates an IBM 2780/3780 terminal, allowing a desktop system to transmit or receive files from *any* system that also runs RJE80. The communications products are described in Chapter 12.
COMPUCALC Electronic Spreadsheet	COMPUCALC brings the ease and versatility of spreadsheet processing to DG/RDOS systems.
CP/M-86 operating system	CP/M-86, the popular operating system for microcomputers, and dependent programs like WordStar can run under DG/RDOS. Details are in *Using CP/M-86 on DESKTOP GENERATION Systems.*
FORMA-TEXT Word Processor	The FORMA-TEXT word processor is a powerful, multiuser word processing system.
FORTRAN	There are two FORTRANs available with desktop systems: FORTRAN 5 and FORTRAN IV. They are introduced in Chapter 10.
Graphics software	MS-DOS' GW BASIC includes some graphics capability.
MS-DOS operating system	MS-DOS, the operating system used by IBM's personal computer, and languages like GW BASIC can run under DG/RDOS. Details are in *Using MS-DOS on DESKTOP GENERATION Systems.*

DG/RDOS Files

A *file* is a collection of information stored under a name called a *filename*. This information can be *records*, like customer names, sales figures, and/or accounts receivable. It can also be the text of a report. Or, it can be a program: a series of computer instructions. The CLI and the DG/RDOS system itself are programs.

Files are stored on hard disks and diskettes. You'll be using them extensively. A file can be very large or very small: the largest file can store millions of bytes; the smallest, zero bytes.

Filenames

Each file is identified by a *filename*. DG/RDOS filenames can be from one to 10 of the following characters: upper- and lowercase letters, numbers, and dollar sign ($). You can add a period and a one- or two-character extension to the name, to further identify the file. For example,

```
ACCT$MAY.12
```

is a valid filename.

Some filename extensions have specific meanings to DG/RDOS. For example, executable program files (save files) have the extension

```
.SV
```

CLI macro filenames have the extension

```
.MC
```

And the names of directories that you create have the extension

```
.DR
```

A macro is one or more commands placed in a file for easy execution. Generally, the .SV, .MC, and .DR extensions are the only ones you need to know about. Extensions are further described in the *RDOS/DOS Command Line Interpreter User's Manual.*

Directory Files

A *directory* is a file that contains other files. The main advantage of directories is that they allow you to group files by *category.* You can create and use them at will to help organize your files. For example, a directory named LETTERS could contain all letters; a directory named REPORTS all reports, and so on.

DG/RDOS allows you to create two kinds of directories: secondary partitions, which are limited to the size you specify when you create them; and subdirectories, which can grow or shrink according to the files created in them. The disk (or diskette) that holds the DG/RDOS system is called the *master directory.*

You don't *have* to create directories, though — you can keep all your files in the master directory. This is a decision to make later on.

Cautions and Control Characters

This section gives some cautions and hints to help you as you create and run an DG/RDOS system. Simply read it; don't do anything yet.

Power Switches and Connections

While DG/RDOS is running, *don't* press the computer power switch to off, or disconnect or unplug any wires connected to the computer or terminals.

If power stops to the computer while DG/RDOS is running, system users may lose work, since the system disk or diskette hasn't been "closed" by normal shutdown. Also, data on a diskette can be damaged if you cut power while the diskette is inserted.

After DG/RDOS has been shut down normally, you can turn off power to the computer and user terminals (if any) if you want.

System Console

The system console is the terminal you use to start up, control, and shut down the DG/RDOS system. This terminal must be turned on and on line for startup, control, and shutdown.

When a Model 10 or 10/SP system is turned on, the computer runs tests and displays the results on the system console. Then, depending on your hardware, it may try to load a system from diskette; you will see either a FILENAME? question or a ! prompt:

```
!
```

While DG/RDOS or any program is running, *don't* type the break sequence (CMD and BREAK/ESC keys) on the system console, unless you want to stop everything. If a ! prompt appears on the system console while DG/RDOS is running, someone may have typed the break sequence on this console. DG/RDOS is stopped (halted). You can have it continue by typing

```
P ↲
```

next to the ! prompt. Accidental breaks can be annoying, but do no real harm as long as you know how to recover from them. Type P ↲ next to the ! prompt.

Keyboard Control and Command Sequences

There are several keyboard control sequences that govern terminal display, interrupt program execution, and so on. You will probably need one or more of these as you work with DG/RDOS. In any case, it's helpful to know about them.

To type a control sequence, first press the CMD or CTRL key; while you hold this key down, type the other character.

Table 1-4 lists the major control characters, special keys, and their functions.

Table 1-4 Control Characters and Special Keys

Key(s)	What it Does
\ (backslash)	The backslash key erases all characters you've typed, allowing you to start typing from scratch. It works in the CLI only. Pressing backslash is usually easier than pressing the DEL key many times.
CMD key	The CMD key and BREAK/ESC key create a break sequence that stops the computer. To have it continue, type P ⤶. The CMD and ERASE PAGE key terminate graphics mode (which can be turned on inadvertently if you type the wrong kind of file).
CTRL-C CTRL-A	A CTRL-C CTRL-A sequence interrupts execution of a CLI command or running program. You'll find this sequence useful.
CTRL-C CTRL-B	CTRL-C CTRL-B does the same thing as CTRL-C CTRL-A, and copies memory to a file named BREAK.SV or FBREAK.SV. Generally, avoid this sequence.
CTRL-C CTRL-F	When typed on the system console, CTRL-C CTRL-F terminates the foreground program (if a foreground program is running). A foreground program must be shut down before the system can be shut down.
CTRL-Q	CTRL-Q resumes terminal display after it has been stopped by CTRL-S.
CTRL-S	CTRL-S suspends terminal display. To continue display, type CTRL-Q. CTRL-S and CTRL-Q are useful when you want to read long files on a screen, or anytime the display is too fast to read.
DEL key	DEL erases the last character typed. In certain programs, DEL shows an underline (_) for each character erased.
NEW LINE key	Tells the computer to execute your command. Usually, when you type a command, nothing will happen until you press NEW LINE (shown as ⤶ in this book). Some DG/RDOS programs (like DKINIT) ignore ⤶ and you must press the CR key.

If You Make a Mistake

DG/RDOS and its companion software have good error messages and error recovery. Your mistakes (everyone makes them) will usually provoke an error message of the form

```
FILE DOES NOT EXIST: file
```

or

```
NO SUCH DIRECTORY
```

which enables you to identify the problem and retry. But if you make what appears to be a critical mistake, you can usually restart the program from the beginning without problems. At worst, you'll need to restart DG/RDOS.

If, at the system console, everything seems to have stopped, type CTRL-Q. For error situations you can't solve, go to Chapter 14, "Responding to Errors and Error Messages".

What Next?

To create your first DG/RDOS system, proceed to Chapter 2. If you already have a system running, go to the topic you want. Chapter 4 is a CLI session and Chapter 10 explains using BASIC and other computer languages.

Installing DG/RDOS for the First Time

2

Read this chapter when

- you have just connected and tested your DESKTOP GENERATION hardware and you want to install DG/RDOS on it;
- you want to install DG/RDOS on a blank hard disk;
- you want to change your system configuration (run CONFIG again for changes for new software or hardware).

This chapter tells you — step by step — how to install DG/RDOS software. The major sections are

- Your Software Diskettes
- Inserting a Diskette
- Installing DG/RDOS
- Installing Other DG Products
- Defining Your Configuration
- Backing Up the DG/RDOS System Files
- Shutting Down
- What Next?

Your Software Diskettes

Before you do anything else, get your DG/RDOS diskette(s), emulator diskette, and other software diskettes ready, in order, as shown by the diskette labels.

Each label has a UPC bar code and several lines of text. The first line gives the diskette part number, product model number, and diskette serial number. The second line gives the product name. The last line gives the diskette sequence number in the set and the product revision (REV). For example, for a DG/RDOS diskette:

```
                                 product   diskette serial
                 diskette        model     number (unique for
                 part number     number    each diskette)

                 082-000301-II
product name-    DG/RDOS System  30566T    nnnnnnnnnn
copyright-       COPYRIGHT 1976 ...................
copyright-       .................................
                 DISKETTE n OF   n   REV n.nn

                 diskette sequence   product
                 in set              revision
```

The names of most software products available with DG/RDOS start with DG/RDOS. The part of the name that follows DG/RDOS identifies each product.

Inserting a Diskette

This section tells you how to insert a diskette in the correct drive. If you already know how, skip to the next section.

With two drives, you will always insert diskettes supplied by DG in drive 0, the right drive. This drive is named DJ0. The left drive is named DJ1.

To insert a diskette, follow these steps. Practice, if you want, with a blank diskette (not a diskette with software supplied by DG).

- Make sure the red light above the drive is off.
- Turn the latch next to drive counterclockwise until the latch is vertical.

- Remove the diskette from its outer envelope. Don't try to remove the inner envelope — the diskette must remain in this.
- Hold the diskette by the edges and examine it. One side has a paper label and the other is blank. On each side, the envelope is cut away to expose part of the diskette surface. Just a reminder — *don't touch the diskette surface*. The oil on your finger could make that part of the diskette unreadable.

  One edge of the diskette has a small notch (about 1/4 x 1/4 inch). This is the *write-enable* notch. When this notch is uncovered, data can be written to the diskette. When the notch is covered (as with opaque tape), the diskette is write protected: it cannot be written to. *Do not write protect diskettes that you will use with DG/RDOS or to install DG/RDOS.*
- Hold the diskette with the write-enable notch up and your fingers on the label. Slide it into the drive as shown in Figure 2-1. The diskette should slide in smoothly and come to a firm stop.

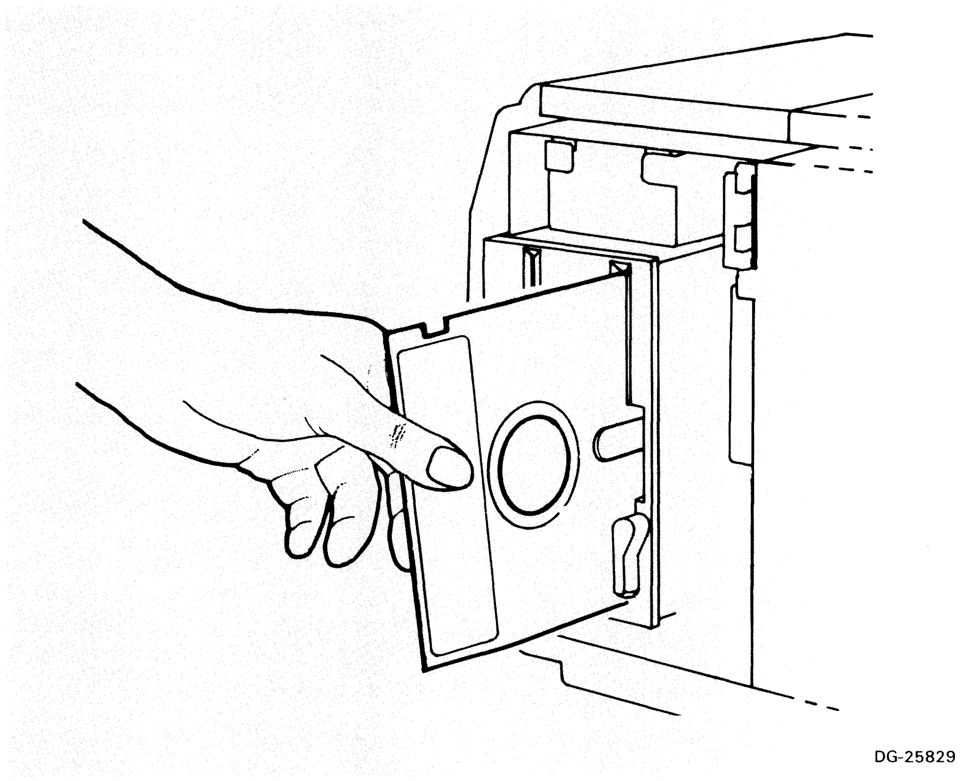


Figure 2-1 Inserting a diskette

- Turn the latch clockwise until it is horizontal. This locks the diskette in the drive.
- To remove a diskette, turn the latch counterclockwise to vertical, grasp the diskette, and pull it gently directly out.

Installing DG/RDOS

Just follow these steps. They will seem tedious, but you need to do them only once.

1. If the devices attached to your computer are not labeled with their names, you should label them. An extra diskette label or opaque tape will serve for the labels. The names are as follows:

   Diskette drives — If there are two, the right drive is DJ0; the left is DJ1. With only one drive, its name is DJ0.

   Hard disk — The drive attached to the computer and logic modules is named DE0. If you have a second drive (separate from the rest of the modules), its name is DE1.

   Tape — The cartridge tape drive name is MT0.

   Printer — The printer name is $LPT.

   Having your devices labeled with their names will make things much easier later on.

2. If the system console is off, turn it on: use the switch behind the screen, on the right.

   The screen may show a test message and beep.

3. Turn on power to the cartridge tape module (if any) and the second hard disk (if any).

   If you have model 10 or 10/SP, with a printer or terminal connected to its printer port, make sure the printer or terminal is turned on and is on line. If the device is a printer, make sure it has paper. When you power up a Model 10 or 10/SP with a device on the printer port, the computer tests the device. If the device fails the test — which would happen if it were off line — the computer may not complete its power-up test.

4. Turn computer power on, using the main power switch on the upper right of the computer module. At the top of the keyboard,

the ON LINE status light should glow. If it doesn't glow, press the CMD key, hold it down, and press the ON LINE key. This puts the terminal on line; the light should glow.

On a Model 10 or 10/SP, the screen will display

```
TESTING
        ABC...
TEST PASSED.
```

And on a Model 10 or 10/SP with a hard disk, the screen displays SELECT LOAD DEVICE... Then the screen may display the prompt

```
!
```

If nothing shows on the screen, check the brightness (contrast) control under the right front corner of the screen: put the control in a central position. If this doesn't help, make sure the system console is plugged in and connections are secure.

5. Make sure the system console is in ALPHA LOCK mode. If the ALPHA LOCK status light on the keyboard is off, press the ALPHA LOCK key to the right of the space bar.

6. If you have a Model 10 or 10/SP with a German, Swiss/German, or Swiss/French keyboard, you must now load a terminal emulator. Model 10s and 10/SPs with German, Swiss/German, and Swiss/French keyboards require an emulator for DG/RDOS installation. With this kind of machine and keyboard, continue with this step.

   With any other configuration, skip to step 7.

6a. The terminal emulator diskette you need is labeled "BOOTABLE D200 EMULATOR". Get the diskette and insert it in drive DJ0 as shown above.

6b. Next to the ! prompt, type 20H

```
! 20H
```

(If you get the error message ?!, use the *numeric keypad*, to the right of the main keypad, to type numbers. This keypad has only numbers and always produces numbers without relying on an emulator.)

This step loads a needed emulator program into the computer.

There are some test messages and beeps. Then, it displays

```
!
```

6c. Remove the "BOOTABLE..." diskette from drive DJ0 and replace it in the paper envelope. (If you typed the wrong thing, or nothing happens, type the break sequence, CMD and BREAK/ESC keys. Remove and reinsert the diskette as shown above. Then repeat this step.)

7. Now for a DG/RDOS diskette.

   With a 360-Kbyte diskette drive and hard disk, or with two 360-Kbyte diskette drives, get the DG/RDOS Stand-Alone Utilities diskette. This is the *second* diskette in the set.

   With *one* 360-Kbyte diskette drive and no hard disk, get the DG/RDOS System diskette.

8. Insert the DG/RDOS diskette in drive DJ0 as shown above.

You're ready to format the system diskette or disk.

Formatting the System Disk or Diskette

Before DG/RDOS can run from a disk or diskette, a formatter program named DKINIT must run on the disk or diskette. This software formatter writes needed information on the disk(ette). It also checks the surface for *bad blocks*, flawed areas that can't hold data.

If you want to use a blank diskette that didn't come from DG, you must hardware format this diskette before DKINIT can format it. Follow the steps under "Hardware Formatting Diskettes", in Chapter 6. Then return here.

Mistakes and Errors

The DKINIT formatter asks several questions, and you supply answers. After an answer, you must press the CR key (not ↓, NEW LINE).

If you type an incorrect answer to a question, and have not yet pressed CR, press the DEL key to erase the wrong characters. The program will display an underline (_) for each character erased.

To restart the program, type CTRL-A (press the CTRL key, hold it down, then press A). Return to step 13. To abort the program (kill execution and return to the ! prompt), type the break sequence (CMD and and BREAK/ESC keys) and return to step 9.

If the DKINIT formatter reports a disk or other error, find the error message in Chapter 14, the error chapter.

DKINIT Dialog

9. Now you will start a program from diskette. Next to the ! prompt, type 20H; for example

   ```
   ! 20H            (Type 20H)
   ```

   The diskette run light should glow, to indicate motor action. Wait 20 to 30 seconds. The screen displays

   ```
   FILENAME?
   ```
   or
   ```
   Filename?
   ```

   With a hard disk and/or two diskette drives, continue. With *one* diskette drive and no hard disk, skip all the way to section "Installing the Emulator," step 54.

   (If you see no FILENAME? question after a minute or so, the diskette may not be inserted correctly or you may have mistyped the number. Type the break sequence (CMD and BREAK/ESC keys). Remove the diskette from drive DJ0 and reinsert it, with write-enable slot up, as shown in Figure 2-1. Repeat step 9. If this doesn't help, consult Chapter 14, the error chapter.)

10. Type the filename DKINIT and press the ↲ key:

    ```
    DKINIT ↲
    ```

    Wait 20 seconds or so. DKINIT starts up and displays

    ```
    DISK INITIALIZER - REV x.xx

    DISK DRIVE MODEL NUMBER?
    ```

11. Remove the diskette from drive DJ0 and return it to the outer envelope.

12. With a hard disk, skip to the next step. With a two-diskette drive system, insert a blank diskette in the left drive, DJ1. This diskette must be hardware formatted. (All diskettes supplied by DG are hardware formatted.)

13. DKINIT wants the model of the disk to format. Model numbers are shown in Table 2-1.

Table 2-1 Disk and diskette formatting information

Description	Model Number	Disk Drive Unit Name	Device Code	Time per Pattern (Approx.)
Minidiskette, 360-Kbyte capacity	6268	DJ0 (right) DJ1 (left)	20	2 minutes
Hard disk, 15-Mbyte capacity	6271	DE0 (first) DE1 (second)	26	11 minutes
Hard disk, 39-Mbyte capacity	6301	DE0 (first) DE1 (second)	26	27 minutes

In Table 2-1, find the model number of the disk you want to format (diskette or hard disk) and type it and CR; for example,

`6271    CR`   (Press CR key after the model number.)

`DISK UNIT?`

14. For the first disk you format, the drive unit name is DE0 (hard disk) or DJ1 (diskette), as shown in Table 2-1. The unit name of the second hard disk (if you have one) is DE1. For example, type

`DE0    CR`   (Or `DJ1 CR` with a diskette-based system.)

The program now displays the disk(ette) type and asks

`COMMAND?`

15. You want a full format for your blank disk(ette), so type

```
FULL CR
```

```
COMMAND DESTROYS ANY PREVIOUS DG/RDOS DISK STRUCTURE.
DG/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) ?
```

16. During a full format, DKINIT checks the disk surface by writing a pattern on each disk block, then reading the block. If the pattern read differs from the pattern written, the block is unreliable: DKINIT notes it in a *bad block* list so that — later on — DG/RDOS will bypass the block.

    DKINIT warns you that the full format will destroy all data on the disk, and that an INIT/F command will be needed later. At this point, the disk(ette) you want to format is blank, so there's nothing on it to destroy. All you need do is specify the number of patterns to run.

    Each pattern takes some time to run, as shown in Table 2-1. Still, we recommend that you run all 5 patterns, since an undetected bad block can bring DG/RDOS down abnormally. Type the number of patterns you want to run; for example,

```
5 CR

*** PATTERN #1 (155555) ***
```

. (delay while pattern runs — be patient.)

```
*** PATTERN #2 (133333) ***
```

    DKINIT describes each pattern as starts the pattern. If it finds a bad block on any pattern, it identifies the block like this:

```
DISK ERROR - BAD BLOCK=000634
```

    Normally, there are few (or no) bad blocks on a new disk or diskette. If DKINIT displays a lot of ADDRESS ERROR messages, the diskette is not hardware formatted; stop DKINIT by typing CTRL-A and hardware format the diskette as shown in Chapter 6. Then return to step 9.

With a hard disk, five patterns will take at least 45 minutes. You might spend this time skimming the rest of this chapter — or other parts of the manual. Or, feel free to leave the system console if there are no messages for a minute or two. Eventually, DKINIT will display

```
*** ALL PATTERNS RUN ***

DO YOU WISH TO DECLARE ANY BLOCKS BAD
THAT ARE NOT ALREADY IN THE BAD BLOCK TABLE? NO CR
```

17. Answer NO and press CR

```
NO CR              (Press CR key)

DEFAULT REMAP AREA IS n BLOCKS LONG
IT NEEDS TO BE AT LEAST m BLOCKS LONG
REMAP AREA SIZE (TYPE RETURN FOR DEFAULT)?
```

18. Press the CR key for the default area size:

```
CR

REMAP AREA START BLOCK NUMBER (TYPE RETURN FOR DEFAULT)?
```

19. Press the CR key for the default block number:

```
CR

DEFAULT FRAME SIZE IS f
MIN IS 1 AND MAX IS ff

DISK FRAME SIZE (TYPE RETURN FOR DEFAULT)?
```

20. This question relates to the hash frame size (explained in the glossary). Press CR for the default:

```
CR

FULL DISK INIT COMPLETE
COMMAND?
```

You've finished formatting your disk or diskette.

If you have a second hard disk, and haven't formatted it, do so: type DISK↵ and return to step 14.

21. To terminate DKINIT, type

```
STOP CR
n
!
```

Now you can install DG/RDOS on your formatted disk(ette).

Installing DG/RDOS

Next, you'll run program MBOOT, bring up DG/RDOS, and move some files onto the disk(ette). The procedure varies, based on whether you have a hard disk or two diskette drives without a hard disk. Just follow the steps.

22. The DG/RDOS System diskette needs to be in drive 0. If it's not there, remove the diskette from drive 0 and insert the System diskette in drive 0.
23. The ! prompt is showing on the system console. (If not, type the break sequence — CMD and BREAK/ESC keys). Next to the ! prompt, type

```
! 20H
```

After a few seconds, it asks

```
FILENAME?
```

24. Type MBOOT and ↲ (NEW LINE key):

```
MBOOT ↲

MBOOT REV x.xx
BOOTSTRAP DEVICE SPECIFIER?
```

25. Type the name of the newly-formatted disk or diskette. The disk name is DE0; the diskette name is DJ1. For example

```
DE0↲            (Or DJ1↲ )

INSTALL BOOTSTRAP (Y OR N)?
```

26. Press

```
Y

DONE.

BOOTSTRAP DEVICE SPECIFIER?
```

27. This time, type the name of the DG/RDOS diskette:

    ```
    DJ0 ↵
    ```

    ```
    INSTALL BOOTSTRAP (Y OR N)?
    ```

28. This time, press

    ```
    N
    ```

    ```
    FILENAME?
    ```

29. The system console is asking FILENAME? For the DG/RDOS system, press

    ```
    ↵
    ```

    After 20 to 40 seconds, it displays

    ```
    DG/RDOS REV x.xx
    DATE (M/D/Y)?
    ```

    (If you see no message after a minute or so, perhaps you misinserted the diskette. Remove the system diskette from drive DJ0, reinsert it as shown in Figure 2-1. Type the break sequence, then type 20H and return to step 29.)

30. Type the date as numbers for month, day, and year. Insert a space or slash before the day and year. For example, for November 16, 1984, type

    ```
    11 16 84 ↵
    ```

    ```
    TIME (H:M:S)?
    ```

31. Type the time, based on a 24-hour clock, in hours, minutes, and seconds. Insert a space or colon before the minute and second numbers. (Actually, minutes and seconds are optional. If you omit them, the system uses 00:00.) For example, for 2:30 p.m., type

    ```
    14 30 ↵
    ```
    (pause)

    ```
    WELCOME TO THE DG DESKTOP GENERATION
    ```

    With a Model 20 or 30 computer, skip to step 33.

32. With a Model 10 or 10/SP, it says

    ```
    EMULATOR FILE NOT FOUND: xTERM$.EM
    DO YOU WISH TO LOAD THE EMULATOR (Y/N)?
    ```

    You'll load the emulator later. For now, answer N for No:

    ```
    N
    ```

33. After a delay of 20 seconds, it displays

    ```
    R
    ```

    R is the DG/RDOS CLI prompt. DG/RDOS is now running from the diskette.

    The next step is to type INIT/F to your disk(ette). If you have a hard disk, the name is DE0. With a two diskette drive system, the name is DJ1. For example,

    ```
    INIT/F DE0 ↵          (or INIT/F DJ1↵)
    ```

    ```
    CONFIRM?
    ```

34. The INIT/F command creates a new file directory on a disk(ette), effectively eliminating all files on it. Thus, DG/RDOS asks you to confirm the command. There aren't any files on the disk(ette) you're working on; and you must confirm to prepare this disk(ette) for DG/RDOS. Type

    ```
    Y
    ```

    It adds ES, executes the INIT/F command, and displays the prompt:

    ```
    R
    ```

35. If you have a second hard disk, type an INIT/F command to it (INIT/F DE1↵) and confirm (Y).

Moving Files onto Your System Diskette

36. The next steps are to copy files onto your system disk(ette). files on the DG/RDOS diskette(s) are described in Table 2-2 and Table 2-3.

    If you have a hard disk, you can copy all files from diskette. Skip to step 45.

    With a diskette-only system, all files may not fit on your system diskette. You must decide *which* files you want, using Tables 2-2 and 2-3. The System diskette is now in drive DJ0. A model 6268 diskette can store about 680 blocks, but we suggest that you leave free space on your system diskette (use 500 blocks or less), especially if you will want to load other software later. Don't copy any program you don't need onto your system diskette.

Table 2-2 Files on the DG/RDOS System diskette (continues)

Program Name	Disk Blocks Needed	Comments
BYE.MC	2	BYE.MC is a macro file to help shut down the system. We recommend it. The instructions in this book assume this file is on the system disk(ette).
CLI.SV	77	This is the Command Line Interpreter. It is an essential file.
CONFIG.SV	24	The CONFIG program allows you to tailor DG/RDOS systems. You *can* run DG/RDOS without it, but should copy it to the system disk(ette). The instructions in this book assume this file is on on the system disk(ette).

Table 2-2 Files on the DG/RDOS System diskette (continued)

Program Name	Disk Blocks Needed	Comments
DDUMP.- DLOAD.-	63	DDUMP.ER, DDUMP.OL, DDUMP.SV, DLOAD.OL, and DLOAD.SV are parts of the DDUMP/DLOAD backup and restore programs. They can copy disk-based material to and from diskettes quite flexibly. Disadvantages are that the diskettes must have been formatted with DKINIT, which takes time. Also, these programs don't use diskette space as efficiently as IMOVE (another backup program). Unless you know that you want DDUMP/DLOAD, we suggest you not copy their files to the system disk(ette). But if you want to do it, use the filenames given above.
DGRDOS.LM	41	This is the DG/RDOS system load map, needed if the system must be patched. You don't need it on a system disk(ette).
DGRDOS.SV	105	This is the DG/RDOS system file. It is an essential file.
FCOPY.SV	28	This program allows you to copy a diskette to another diskette, using only one diskette drive. FCOPY is invaluable if you have only one diskette drive. In any case, it's useful and you should copy it. We assume FCOPY.SV is on the system diskette.
FDUMP.SV FLOAD.SV	33	These are fast dump and fast load programs which require magnetic tape (cartridge tape). Unless you have a cartridge tape drive and a hard disk, it's fruitless to copy them to the system diskette.
IMOVE.SV	35	IMOVE is a backup and restore program that's versatile and efficient. If your system has a hard disk, we recommend that you use IMOVE for backup. (Another backup program is okay, however, if you know that you want it.) For a diskette-based system, it's easier to copy diskettes than use IMOVE; skip it.

Table 2-2 Files on the DG/RDOS System diskette (concluded)

Program Name	Disk Blocks Needed	Comments
LOADEM.SV	8	The LOADEM program can determine the correct emulator file for your system and load it onto the system disk(ette). LOADEM is essential for a Model 10 or 10/SP system disk(ette). We recommend LOADEM for *any* system.
MBOOT.SV	16	MBOOT.SV both installs a bootstrap root *and* helps start up systems. It is an essential file for a system disk(ette).
PATCH.SV ENPAT.SV	14 8	The PATCH program allows you to apply patches; the ENPAT program allows you to enter (create) patches. These programs are needed *only* if some correction or update is essential. In nearly all cases, you don't need them. See the Release Notice Installation Instructions for the last word on the need for installing them.
SEDIT.SV	20	This is a disk editor, used along with DGRDOS.LM if you need to patch the system. You don't need it on a system diskette.
SYS.SV	-	This is a link file to DGRDOS.SV. It allows you to bring up the system by pressing ↲ (instead of typing DGRDOS↲). We strongly recommend you copy it to your system diskette. The instructions in this book assume this file is on the system disk(ette).

Table 2-3 Files on the DG/RDOS Stand-Alone Utilities diskette (continues)

Program Name	Disk Blocks Needed	Comments
DKINIT.SV	51	The DKINIT software formatter prepares a disk(ette) for use with DG/RDOS. It is needed when you want to format a new disk(ette) as a directory (not needed for backup). Also, DKINIT has a PARTIAL command that can locate a bad block and remove it from use without destroying the file structure. For a diskette-only system, you may prefer to use DKINIT from a Stand-Alone Utilities diskette (not copy it to your system diskette). Whatever you choose, remember where DKINIT is, because you will probably want to use it in the future.
DO.SV	6	The DO utility executes a CLI macro (.MC) file, which extends the CLI's ability to handle macros by allowing you to specify arguments to macros. DO is useful if you plan to use CLI macros; otherwise, skip it.
MBOOT.SV	16	Same as MBOOT.SV above; needed on this diskette to start the stand-alone programs.
YBURST.SV	91	YBURST is a backup utility that copies disks to tape or other disks. Unless you have a cartridge tape drive, don't copy it.

Table 2-3 Files on the DG/RDOS Stand-Alone Utilities diskette (concluded)

Program Name	Disk Blocks Needed	Comments
YDBURST.SV	126	YDBURST is a backup utility that copies disks to diskettes. You might consider it if you have a hard disk. If you plan to use IMOVE (above), skip YDBURST.
YOWNER.SV	93	This program identifies the file that "owns" a disk block. It tells you which file is damaged if a new bad block develops. DKINIT can tell the number of the new bad block, but not the file that owns the block.) Knowing the name, you can delete and reload the file. Generally, on diskette-only systems, you won't want to copy YOWNER onto the system diskette. But you may need to use it later on.

Moving Files to Your System Diskette

37. With no hard disk, having decided on the files you want, copy them to the system diskette using MOVE commands of the form

    MOVE/V DJ1 filename ...

    This MOVE command tells DG/RDOS to move (copy) specific files from the DG/RDOS System diskette to DJ1, and verify the copy. To start, type

    ```
    MOVE/V  DJ1  BYE.MC  CLI.SV  CONFIG.SV   DGRDOS.SV ↲
    .
    . (The system verifies filenames copied. Eventually,
    . the R prompt returns.)

    R
    ```

38. Copy another group of files:

    ```
    MOVE/V  DJ1  FCOPY.SV  LOADEM.SV  MBOOT.SV  SYS.SV ↲
    .
    . (it verifies names copied)

    R
    ```

39. Release the System diskette:

    ```
    RELEASE DJO ↲
    ```

    and remove the System diskette from drive DJO.

40. If you want any programs from the DG/RDOS Stand-Alone Utilities diskette, take the following action (otherwise skip to the next step).

    Insert the Stand-Alone Utilities diskette in DJO. Type

    ```
    DIR DJO ↲             (type DIR DJO↲
    R
    MOVE/V    DJ1    desired-files ↲          (For example, DKINIT.SV.)
    .
    . (it verifies files copied)
    .
    R
    RELEASE DJO ↲
    ```

    Remove the Stand-Alone Utilities diskette from DJO and store.

41. Release your newly-built system diskette:

```
RELEASE DJ1 ⤶
R
```

Remove your system diskette from DJ1 and insert it in drive 0.

42. Type

```
DIR DJ0 ⤶
R
BYE ⤶

STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED
.
FILENAME?
```

43. Now you'll make sure that you can start DG/RDOS from your system diskette. Type the break sequence (CMD and BREAK/ESC keys), to produce the ! prompt.

```
!
```

44. To start from diskette, type 20H

```
! 20H

FILENAME?
```

After FILENAME appears, skip to step 54, in section "Installing the Emulator."

(If FILENAME doesn't appear after 20 seconds or so, you may have made a mistake with the diskette or number. Type the break sequence — CMD and BREAK/ESC. Then remove the diskette from DJ0 and reinsert it as shown in Figure 2-1. Repeat step 44.)

Moving Files to the Hard Disk

45. With a hard disk, you have enough space to put *all* supplied files on the disk. Type

```
MOVE/A/V/R    DE0 ↲
```

This MOVE command tells DG/RDOS to move (copy) all files from the system diskette to disk DE0 and verify each file copied. After 2 to 3 minutes, the R prompt returns:

```
.
.
R
```

46. Release the diskette:

```
RELEASE DJ0 ↲
R
```

47. Remove the DG/RDOS System diskette from drive DJ0 and put it in an envelope. Get the DG/RDOS Stand-Alone Utilities diskette (used earlier), and insert it in drive DJ0.

48. Type

```
DIR DJ0 ↲
R
```

(If the R prompt doesn't appear in 20 or so seconds, perhaps you misinserted the diskette. Remove the diskette from drive 0, reinsert it as shown in Figure 2-1, and type the DIR DJ0↲ command again.)

49. From Table 2-3, check the files you want and move them to the disk. We recommend DKINIT.SV and YOWNER.SV. For example, type

```
MOVE/A/V/R  DE0  DKINIT.SV  YOWNER.SV ↲
.
. (again, it verifies files copied.)
.
R
```

50. You've copied all the files. Start DG/RDOS from the hard disk using the following commands:

```
DE0:BYE ⤶
STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED

FILENAME?
```

51. You're done with the diskette in DJ0. Remove it from the drive and replace it in an envelope.

52. Now you'll make sure that you can start DG/RDOS from your system disk. Type the break sequence (CMD and BREAK/ESC keys), to produce the ! prompt.

53. To start from the hard disk, type 26H

```
! 26H

FILENAME?
```

(If FILENAME doesn't appear after 20 seconds or so, you may have mistyped the number. Type the break sequence — CMD and BREAK/ESC. Then repeat this step, 53.)

Installing the Emulator

Model 10 and 10/SP systems need a *terminal emulator* program. This program enables the system console to produce lowercase letters and otherwise emulate a DASHER D211 terminal. For *any* system, continue with the next step.

54. If you have a Model 20 or 30 system, skip to step 56.

55. Now, if an emulator is already in computer memory, you need turn power off and on so that a new emulator can be loaded. You can tell by the ALPHA LOCK light. Watch the light and press the ALPHA LOCK key several times. If the light goes off and on, this means an emulator is loaded; continue with this step.

    If, when you press ALPHA LOCK, the light remains on, no emulator is loaded. Skip to step 56.

55a. Turn computer power off, then on, using the switch on the upper right corner of the unit.

55b. The system will run through its self-test patterns and characters. Then it may display the ! prompt. If so, type 26H

```
! 26H
```

Or it may display LOADING FROM DISKETTE and continue to the next step.

56. The system console is asking

```
FILENAME?
```
or
```
Filename?
```

To specify DG/RDOS, press the ↵ key:

```
↵
```

After 10 to 30 seconds, it says

```
DG/RDOS REV x.xx
DATE (M/D/Y)?
```

57. Type the date as numbers for month, day, and year, with a space or slash before the day and year, as before. For example, on November 16, 1984, type

```
11 16 84 ↵

TIME (H:M:S)?
```

58. Type the time, based on a 24-hour clock, in hours, minutes, and seconds. Insert a space or colon before the minute and second numbers, as before. For example, at 2:55 p.m., type

```
14 55 ↵            (For 2:55 pm)
```

After a pause, it displays

```
WELCOME TO THE DG DESKTOP GENERATION
```

59. With a Model 20 or 30 computer, it displays R. Skip to step 65.

On a Model 10 or 10/SP, as before, it says

```
EMULATOR FILE NOT FOUND: xTERM$.EM
DO YOU WISH TO LOAD THE EMULATOR (Y/N)?
```

60. You want to load the emulator, so answer Y for yes:

```
Y

INSERT EMULATOR DISKETTE IN DRIVE 0
TYPE C TO CONTINUE ...
```

61. If there is a diskette in drive DJ0, remove it.

    Get the terminal emulator diskette supplied with the DG/RDOS diskettes, and insert it in drive DJ0.

62. Type `C`

    The program displays

```
EMULATOR IS BEING LOADED ...
.
Emulator load completed.
```
(Note the lowercase letters.)

    With a hard disk, skip the next step.

63. Without a hard disk, the program asks

```
Insert system diskette in drive 0
TYPE C TO CONTINUE...
```

    Remove the diskette from drive DJ0. Insert your system diskette in drive DJ0. Type C.

64. The program displays

```
xD211yyyy.TX is being copied to the master directory
R
```

65. You've installed DG/RDOS! Copies of essential files are on your system disk(ette). The R prompt means that DG/RDOS is running from this disk(ette). Congratulations!

    A program named LOADEM will load the terminal emulator file — if needed — in the future. In the emulator filename, the letter x is M for monochrome or C for color; the letters yyyy stand for a language. A typical emulator filename is MD211AMUK.TX, the monochrome (M) American-United Kingdom (AMUK) emulator.

    You're done with the DG/RDOS software and emulator diskettes. You might need them again if you ever want to build a system disk(ette) from scratch.

    Sections that following this one show how to install other software and configure DG/RDOS for your hardware and software. Figure 2-2 summarizes the steps to install DG/RDOS.

Checking the Emulator

DG/RDOS is up and running. To test for the correct emulator, turn ALPHA LOCK off (press the ALPHA LOCK key). Then type a lowercase letter (like q). If you can make q appear in lowercase, the emulator is loaded and working correctly. Continue to the next section. (If the letter won't appear in lowercase, type BOOT SYS) and return to "Installing the Emulator," above.

1. If the disk and tape drives aren't labeled with their names, label them (DJ0, DE0, and so on).
2. Turn on system console (if off).
3. Turn on power cartridge tape (if any) and second hard disk (if any). On Model 10 or 10/SP with a printer on printer port, make sure printer is on line and has paper.
4. Turn on computer.

```
TESTING...          (Model 10 and 10/SP only)
ABC....
```

5. Check system console ALPHA LOCK status light; it should be on.
6. If you have a Model 10 or 10/SP with a German, Swiss/German, or Swiss/French keyboard, continue with this step. With other hardware, skip to step 7.

6a. Find diskette labeled "BOOTABLE D200 EMULATOR" and insert it in drive DJ0.

6b.
```
! 20H          (Type 20H)
. (messages)
!
```

6c. Remove the "BOOTABLE" diskette from drive DJ0.

7. With a diskette drive and hard disk, and/or with two diskette drives, get the DG/RDOS *Stand-Alone Utilities* diskette.

   With other hardware (no hard disk, single diskette drive system, and so on), get the DG/RDOS *System* diskette.
8. Insert the DG/RDOS diskette in drive DJ0 as shown above.

Formatting the System Disk or Diskette

9.
```
! 20H          (Type 20H)
```

(20 or 30 seconds pass.)

```
FILENAME?
```

DG-26407

Figure 2-2 DG/RDOS installation summary (continues)

10. `DKINIT ↲` (Type DKINIT and press NEW LINE key.)

    (20 seconds or so pass.)

```
DISK INITIALIZER - REV x.xx
DISK DRIVE MODEL NUMBER?
```

11. Remove diskette from drive DJ0 and return it to an envelope.
12. With a hard disk, skip to the next step. With a diskette-based system, insert a blank diskette in the left drive, DJ1.
13. `DISK DRIVE MODEL NUMBER? n CR` (Type number n and press CR key. Number is 6271 for 15-Mbyte, 6301 for 39-Mbyte; see Table 2-1.)
14. `DISK UNIT? DE0 CR` (For diskette, type `DJ1 CR`)
15. `COMMAND? FULL CR`

```
COMMAND DESTROYS ANY PREVIOUS DG/RDOS DISK STRUCTURE.
.
```

16. `NUMBER OF PATTERNS TO RUN (1-5) ? 5 CR`

```
*** PATTERN #1 (155555) ***
```

    . (delay while patterns run — be patient.)

```
*** ALL PATTERNS RUN ***

DO YOU WISH TO DECLARE ANY BLOCKS BAD
```

17. `THAT ARE NOT ALREADY IN THE BAD BLOCK TABLE? NO CR`

```
DEFAULT REMAP AREA IS n BLOCKS LONG
IT NEEDS TO BE AT LEAST m BLOCKS LONG
```

18. `REMAP AREA SIZE (TYPE RETURN FOR DEFAULT)? CR`
19. `REMAP AREA START BLOCK NUMBER (TYPE RETURN FOR DEFAULT)? CR`

```
DEFAULT FRAME SIZE IS f
MIN IS 1 AND MAX IS ff
```

DG-26407

Figure 2-2 DG/RDOS installation summary (continued)

20. *DISK FRAME SIZE (TYPE RETURN FOR DEFAULT)?* CR

    FULL DISK INIT COMPLETE

21. *COMMAND?* STOP CR

    For a second hard disk, type DISK and return to step 14.

Installing DG/RDOS

22. Make sure the DG/RDOS System diskette is in drive DJ0. Remove any other diskette to insert it if needed.

23. *!* 20H (Type 20H)

24. *FILENAME?* MBOOT ↵ (Type MBOOT and press NEW LINE key.)

    MBOOT REV x.xx

25. *BOOTSTRAP DEVICE SPECIFIER?* DE0↵ (or, for diskette, DJ1↵)

26. *INSTALL BOOTSTRAP (Y OR N)?* Y

    DONE.

27. *BOOTSTRAP DEVICE SPECIFIER?* DJ0 ↵

28. *INSTALL BOOTSTRAP (Y OR N)?* N

29. *FILENAME?* ↵ (Press NEW LINE key.)
    (20 to 40 seconds pass.)

    DG/RDOS REV x.xx

30. *DATE (M/D/Y)?* 11 16 84 ↵ (Type current date.)

31. *TIME (H:M:S)?* 14 30 ↵ (Current time, 24-hour clock.)

    WELCOME TO THE DG DESKTOP GENERATION

    With a Model 20 or 30 computer, skip to step 33.

    EMULATOR FILE NOT FOUND: xTERM$.EM

DG-26407

Figure 2-2 DG/RDOS installation summary (continued)

32. *DO YOU WISH TO LOAD THE EMULATOR (Y/N)?* N

    (20 seconds or so pass.)

    R

33. `INIT/F DE0 ↵` (Without hard disk, type `INIT/F DJ1↵`)

34. *CONFIRM?* Y *ES*
    R

35. For a second hard disk, type `INIT/F DE1↵` and confirm (Y).

Moving Files onto Your System Diskette

36. The next steps involve coping files onto your system disk(ette). For a hard disk, skip to step 45. For a diskette-based system, decide on the files you want, using Tables 2-2 and 2-3 (suggestions next).

37. Move copies of the files; for example

```
MOVE/V   DJ1   BYE.MC   CLI.SV   CONFIG.SV   DGRDOS.SV ↵
```

    .
    . (It verifies filenames copied.)
    .
    R

38. Move copies of more files:

```
MOVE/V   DJ1   FCOPY.SV   LOADEM.SV   MBOOT.SV   SYS.SV↵
```

    .
    . (It verifies names.)

    R

39. `RELEASE DJ0 ↵`
    R

    Remove the System diskette from drive DJ0.

41. `RELEASE DJ1 ↵`
    R

    Move your system from drive DJ1 to DJ0.

DG-26407

Figure 2-2 DG/RDOS installation summary

42.
```
DIR DJO ↲
R
BYE ↲

STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED

FILENAME?
```

43. Type the break sequence (CMD and BREAK/ESC keys).

44.
```
! 20H

FILENAME?
```

Skip to step 54.

Moving Files to the Hard Disk

45.
```
MOVE/A/V/R      DEO ↲
.
```
. (It verifies filenames copied.)
```
.
R
```

46.
```
RELEASE DJO ↲
R
```

47. Remove the diskette from drive DJO. Get the DG/RDOS Stand-Alone Utilities diskette and insert it in drive DJO.

48.
```
DIR DJO ↲
R
```

49. Move the files you want (Table 2-3) to the hard disk. For example
```
MOVE/A/V/R   DEO   DKINIT.SV   YOWNER.SV ↲
.
```
. (It verifies.)
```
.
R
```

50.
```
DEO:BYE ↲
STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED
FILENAME?
```

DG-26407

Figure 2-2 DG/RDOS installation summary

51. Remove diskette from drive DJ0 and replace it in an envelope.
52. Type the break sequence (CMD and BREAK/ESC keys).
53. *!* 26H (Type 26H)
    FILENAME?

Installing the Emulator

54. If you have a Model 20 or 30 system, skip to step 56.
55. Check if an emulator is already loaded: press ALPHA LOCK key several times. If ALPHA LOCK light goes off and on, an emulator is loaded; continue with this step.

    If, when you press ALPHA LOCK, the light remains on, no emulator is loaded. Skip to step 56.

55a. Turn computer power off, then on.

55b. System does its self test. Then

! 26H (Type 26H — displayed with hard disk only.)

Or it may display LOADING FROM DISKETTE; if so, continue.

56. The system console is asking

FILENAME?
or
Filename?

⤶ (Press ⤶ key.)

(10 to 30 seconds pass.)

DG/RDOS REV x.xx

57. *DATE (M/D/Y)?* 11 16 84 ⤶ (Current date)
58. *TIME (H:M:S)?* 14 55 ⤶ (Current time, 24-hour clock)

WELCOME TO THE DG DESKTOP GENERATION

DG-26407

Figure 2-2 DG/RDOS installation summary

59. With a Model 20 or 30 computer, skip to step 65.

```
EMULATOR FILE NOT FOUND: xTERM$.EM
```

60. `DO YOU WISH TO LOAD THE EMULATOR (Y/N)?` Y

```
INSERT EMULATOR DISKETTE IN DRIVE 0
TYPE C TO CONTINUE ...
```

61. If there is a diskette in drive DJ0, remove it. Get the emulator diskette supplied with DG/RDOS and insert in in DJ0.

62. Type C

```
EMULATOR IS BEING LOADED ...

.
Emulator load completed.
```

(Note the lowercase letters.)

With a hard disk, skip the next step.

63. Without a hard disk, the program asks

```
Insert system diskette in drive 0
TYPE C TO CONTINUE...
```

64. Remove the diskette from drive DJ0. Insert your system diskette in drive DJ0. Type C.

The program displays

```
xD211yyyy.TX is being copied to the master directory
R
```

65. You've installed DG/RDOS! Congratulations. Shut down (DIR %MDIR%↵; BYE↵) if desired.

DG-26407

Figure 2-2 DG/RDOS installation summary

Installing Other Software Products

Now — before you configure DG/RDOS — you should install any other DG software products you acquired with DG/RDOS. You'll want to do this sooner or later — and doing it now will give any special product information needed for configuration.

Available system software includes the DG/RDOS Development Kit and DG/RDOS Sysgen files. Other products are described below.

Installing the DG/RDOS Development Kit

If you received the Development Kit, move all its files to the disk, directory DE0, as follows. (You can put the files anywhere you want, but they will be most accessible in directory DE0.)

- Get the Development Kit diskette and insert it in drive DJ0.
- Type the following commands:

```
DIR DJ0 ⤶
R
MOVE/V/R     DE0 ⤶

.
. (it verifies files moved.)
.
R

RELEASE DJ0 ⤶
```

- Remove the Development Kit diskette from DJ0 and store it safely.

The Development Kit programs, described in Chapter 9, are installed.

Installing the DG/RDOS Sysgen Files

The Sysgen files allow you to generate RDOS systems for some devices that DG/RDOS (a *pregenerated* system) doesn't allow. For example, you can generate a system for configurations other than those supported by DG/RDOS. To install the Sysgen files, follow these steps:

- Get the DG/RDOS Sysgen diskette and insert it in drive DJ0.
- Type the following commands:

```
CDIR DE0:DGRDOSn ⤶          (n indicates the revision. For example,
                             use the name DGRDOS110 for revision
                             1.10.)
R
INIT DE0:DGRDOSn ⤶
R
DIR DJ0 ⤶
R
MOVE/V/R    DGRDOSn ⤶
.
. (it verifies files moved.)
.
R
RELEASE DJ0 ⤶
R
```

- Remove the Sysgen diskette from DJ0 and store it safely.

The RDOS system generation files are installed. You can use them, from the DGRDOSn directory, as described in the manual *How to Load and Generate Your RDOS System.*

Other Software Products

Software available with DG/RDOS includes MS-DOS or CP/M-86 and related software (Model 10 and 10/SP only), Interactive COBOL, COMPUCALC, FORMA-TEXT, Business and Extended BASIC, FORTRAN IV, and more.

Installation instructions vary with the software. For specifics, read the product Release Notice — section "Installation Instructions." Generally, the installation procedure goes something like this:

- Read the product Release Notice, "Installation Instructions"
- Discover how much disk space you have left on the disk(ette) where you plan to install the product. Do this by typing

```
DIR directory          (for example, DIR DE0↵ or DIR DJ1↵)

DISK ↵
LEFT: n USED: n MAX CONTIGUOUS: n
R
```

- Check disk space figures for the product if any; some base figures are in Table 2-5. Decide whether the product will fit, or — on a diskette-only system — whether you need to put it on its own diskette.
- Create a subdirectory for the product, preferably with the product name, using the CDIR command; then initialize the directory (INIT). For some products, you should skip this (as specified by the Release Notice).

```
CDIR product ↵         (example names: MSDOS, DGGATE, FORT4)
R
INIT product ↵
R
```

- Insert the product diskette in a diskette drive. With a hard disk, use DJ0; with two diskette drives, use DJ1. If there is more than one product diskette, use the first one.
- Make the product diskette the current directory with the DIR command:

```
DIR diskette ↵          (for example, DJ0↵)
R
```

- Move the product files to the destination directory, using the MOVE command or a macro supplied with the product. For example,

```
MOVE/V/A    product ↲          (For example, FORTRAN IV, MOVE/V/A FORT4↲)
.
. (the CLI verifies filenames copied)
.
R
```

  The R prompt may return, or you may be prompted to insert another diskette. If the R prompt returns, type

```
RELEASE diskette ↲           (for example, RELEASE DJ0↲)
R
```

- If there are more product diskettes, or if you are prompted to insert another, repeat the previous step until you see the R prompt or the installation macro tells you that you're done.
- Remove the product diskette and store all product diskettes safely.
- Take another reading on disk space by typing

```
DIR product ↲
R
DISK ↲
LEFT: ... ... ...
```

  You may want to note the amount of disk space used by the product.

- Follow any special configuration steps given in the Release Notice. For some products (like FORTRAN), you may need to create one or more link files to programs on the master directory. This is true for the Relocatable Loader files (RLDR.SV and RLDR.OL).

To *use* the product software, see the product manual and the Release Notice. You may need only get into the product directory (DIR command), then type the product name.

This manual has more information on using the following products: CLI and DG/RDOS, Chapters 4 and 5; backup products, Chapter 7; SPEED text editor and Development Kit, Chapter 9; FORTRAN and BASIC languages, Chapter 10; communications products, Chapter 12, and so on.

For other products, see the product manual.

Defining the System Configuration

The DG/RDOS system originally supplied by DG will run on any DESKTOP GENERATION computer, but it does not support important things like foreground/background (two program) operation, a printer, or communication lines. It can't run certain applications software (like multiuser ICOBOL).

To enable all but minimal features, you must configure the supplied system for your hardware and software. The CONFIG program does this. You can also use CONFIG to change a tailored system (for example, if you acquire new hardware or want to change an existing DG/RDOS system). And, you can configure systems to run on other, different DESKTOP GENERATION computers, not just your own.

Since you can run CONFIG at any time (not just when your system is new), we will stop numbering the steps here. But you will still be able to follow the directions. Instead of a number, we'll use a bullet (•) to identify each parameter you can specify with CONFIG.

CONFIG works by changing values in the DG/RDOS system file on disk. Thus, any values you change won't take effect until you leave CONFIG, shut down DG/RDOS, and start up the newly-configured system.

CONFIG has several pages of screen menus. Each menu lists several system parameters. Each parameter has a default value, which will be used if you don't specifically change the parameter value.

If you default a value, the default will be the value next time you run CONFIG on that system. For example, if you run CONFIG on DGRDOS.SV and change the memory parameter from 128 to 768, the next time you run CONFIG on DGRDOS.SV the memory default will be 768.

Creating Copies of DG/RDOS Systems

For your first system, you don't need to copy the DG/RDOS system: skip the rest of this section.

You can run CONFIG on the system supplied with DG/RDOS (DGRDOS.SV) at will. But if you want to configure for another system — for example, a system specifically designed to run multiuser Business BASIC or run on a different computer — you'll want to copy DGRDOS.SV to another file and work with the copy. The new system name must have the .SV extension.

For example, suppose your computer has a terminal attached to the printer port (this is called a *foreground console*); a magnetic tape, a USAM multiplexor, and 4 communication lines. You've already configured the supplied system (DGRDOS.SV) for this hardware. You want to create a DG/RDOS system for a computer without tape, with a *printer* on its printer port, and no USAM. You want this system to use as little memory as possible. You might use the name SMALLSYS.SV for the system. You'd copy it using the following MOVE command

```
MOVE/V    %GDIR%    DGRDOS.SV/S    SMALLSYS.SV ⤶
R
```

Next, you'd fire up CONFIG and configure SMALLSYS.SV as desired. Then, you could create a system diskette (via steps 9 through 34, shown earlier), copy SMALLSYS.SV, MBOOT.SV and CLI.SV to the diskette with the MOVE command, take the diskette to the smaller system, and start and run it there. The system file must be in the master directory DE0 or DJ0, not in a subdirectory, to run.

Configuration Parameters

CONFIG has the following parameters and defaults for DG/RDOS revision 1.10 (summarized in Table 2-4). The original DG default values follow the colon.

Table 2-4 CONFIG parameters and defaults

- *Enter name of system to configure:* DGRDOS.SV

Page 1

- *Memory size in Kb (128-2048) : 128*
- *Number of FG/BG sharable pages (0-255) : 0*
- *Number of BG I/O channels (16-255): 40*
- *Number of FG I/O channels (0-255) : 0*
- *Number of stacks (1-10): 3*
- *Number of system buffers (6-63): 6*
- *Number of cells (9-64): 12*
- *Number of directories accessible at one time (1-64) : 10*

Page 2

- *8 bit character support for the master console ($TTO) ? (Y/N) : N*
- *Secondary console/Printer port? (Y/N): Y*
- *8 bit character support for the secondary console ($TTO1)? (Y/N): N*
- *Printer ($LPT)? (0=NONE, 1=$TTO, 2=$TTO1, 3=USAM line 0,*
  4=USAM line 1, 5=USAM line 2, 6=USAM line 3) : 0
- *Plotter ($PLT)? (0=NONE, 1=$TTO, 2=$TTO1, 3-USAM line 0,*
  4=USAM line 1, 5=USAM line 2, 6=USAM line 3) : 0
- *Magnetic tape drive ? (Y/N) :*
- *USAM ? (Y/N) : N*

Page 3

- *Break character for USAM : 1*
  (Enter decimal ASCII character code, or "128" for none)
- *Number of lines on USAM (1-4): 4*

Pages 4-7

- *Line number (0-3) : 0 (on page 4) 1 (page 5) 2 (page 6) 3 (page 7)*
- *Line speed (in bits/second) for USAM from table below : 8*
  0=3600, 1=19200, 2=50, 3=75, 4=134.5, 5=2000, 6=600, 7=2400,
  8=9600, 9=4800, 10=1800, 11=1200, 12=7200, 13=300, 14=150, 15=110
- *Number of stop bits (1-2) : 1*
- *Number of data bits (5-8) : 7*
- *Parity (0=NONE,1=ODD,2=EVEN) : 2*
- *Modem (Y/N) : N*
- *Loopback mode (Y/N) : N*
- *Data Set Ready (DSR) : 0*
- *Request To Send (RTS) : 1*
- *Data Terminal Ready (DTR) : 1*

Broadly speaking, these divide into the following major categories:

second program capability (memory and FG/BG figures);

memory for DG/RDOS use (stacks, cells, buffers);

devices (foreground terminal, printer, plotter, tape drive); and

communications lines (USAM, USAM line, and modem questions).

Unless you have the absolute minimum hardware and software, these categories are important. Table 2-5 shows some popular DG/RDOS software products and their ballpark requirements in terms of users, memory, and approximate disk space.

Table 2-5 Program Requirements, Users and Memory (continues)

	Memory (Kbytes and Kwords [Kw])			
Product No. of Users	**Recom-mended**	**Max.**	**No. of Channels**	**Disk Storage (512-byte blocks)**
Business BASIC 1-4 users	256 Kbytes	512 Kbytes	21 for first user. Add 16 per additional user.	n/a
COMPUCALC 1 user	64 [32 Kw]	64 [32 Kw]	16	128, plus spreadsheets. A 7 x 20 spreadsheet uses 5; a 14 x 60 spreadsheet uses 29
CP/M-86 1 user	86 [43 Kw]	n/a	16	355
DG/BLAST 1 user	64 [32 Kw]	64 [32 Kw]	16	53
DG/RDOS Pregen 1-2 users	varies (CONFIG)	varies (CONFIG)	16 (CLI)	875
Extended BASIC 1-4 users	256 Kbyte	512 Kbyte	5 plus 1 for each open file	n/a

Table 2 -5 Program Requirements, Users and Memory (concluded)

	Memory (Kbytes and Kwords [Kw])			
Product No. of Users	**Recom-mended**	**Max.**	**No. of Channels**	**Disk Storage (512-byte blocks)**
DG/GATE 1 user	64 [32 Kw]	64 [32 Kw]	16	108
FORMA-TEXT with spelling 1-4 users	62 [31 Kw]	62 [31 Kw]	14	2,605 plus documents.
	72 [36 Kw]	72 [36 Kw]	20	A one-page document uses 4 to 8 blocks;
	82 [41 Kw]	82 [41 Kw]	27	an average letter uses
	47 [47 Kw]	94 [47 Kw]	32	6 blocks
Interactive COBOL Develop-ment Runtime	Program FLEXSTATS gives more in-formation			1110 plus user pro-grams and files
MS-DOS and utilities 1 user	92 [46 Kw]			644
Multi-plan	n/a	n/a	16	n/a
WordStar/ SpellStar 1 user	n/a	n/a	16	n/a

Running CONFIG

Program CONFIG.SV should have been copied into the master directory (DE0 for disk; DJ0 for diskette) when DG/RDOS was installed. The steps are given in the previous section.) Unless CONFIG was moved to a different directory, you can run CONFIG from the master directory. To get to the master directory, type DIR %MDIR%↵.

On a Model 10 or 10/SP, CONFIG needs to have the emulator loaded. Test this by looking at the ALPHA LOCK status light and pressing the ALPHA LOCK key twice. If the light goes off and on, the emulator is loaded. If the light doesn't go off, the emulator isn't loaded. Load it as described in the previous section, "Installing the Emulator."

CONFIG accepts both UPPER- and lowercase letters, so, if you have ALPHA LOCK on, you can turn it off (press the ALPHA LOCK key).

The form of the CONFIG command is

```
CONFIG [system-name] [dialog-filename/V]
```

where

system-name	is the name of the system you want to configure. If you don't specify this here, CONFIG will ask for it. Giving the name in the command line just saves a step later.
dialog-filename/V	tells CONFIG to record the parameters you select in a dialog file, from which you can read them easily later, without having to run CONFIG again. This record can be quite handy if you configure a system often. You might even put the date in the dialog filename. (If disk space is tight, as on a diskette-based system, skip the dialog file.) For example, for a dialog file on July 28, you might type `CONFIG DGRDOS0728.CF/V ↵` (The .CF is shorthand for CONFIG.)

For example

```
CONFIG DGRDOS1228.CF/V ↵
```

This starts CONFIG with a dialog file named DGRDOS1228.CF. If you omit the name of the DG/RDOS system, CONFIG asks for it. The question is

```
Enter name of system to configure: DGRDOS.SV
```

The default value is displayed dim, after the colon. You must specify a name to continue.

Any values you change with CONFIG won't be written to the system file until you tell CONFIG specifically to do it — thus you can experiment as desired, without fear of making mistakes. So, if you want practice, just press ↵ to choose the default value, DGRDOS.SV. For example, press

```
↵
```

To edit a different system file, type its name and ↵. You can omit the .SV extension if you want. CONFIG checks for the existence of the file, then; if it exists, CONFIG displays page 1 of system parameters. Page 1 looks like this:

```
          DGRDOS System Configuration Program        REV:  x.xx
    System name: xxxxx.SV      Page 1       FUNCTION-KEY-1-FOR-COMMANDS
    --------- USE ARROWS (UP,DOWN,LEFT,RIGHT) TO MOVE CURSOR -----------

      Memory size in Kb (128-2048) :  128

      Number of FG/BG sharable pages (0-255) :  0

      Number of BG I/O channels (16-255):   40

      Number of FG I/O channels (0-255) :  0

      Number of stacks (1-10):  3

      Number of system buffers ( 6-63):  6

      Number of cells ( 9-64): 12

      Number of directories accessible at one time (1-64) :  10
   ________________________________________

     (1) Go to next page         (4) Size the system
     (2) Edit parameters         (5) Exit with/without change
     (3) Go to previous page     ENTER YOUR CHOICE:  1↵
```

The screen header gives the program name, revision number, system name page number and some messages, as shown above.

The parameters follow in the middle of the screen.

At the bottom are the action choices. The default is 1, go to next page. To edit values, you must type 2↵. After typing 2↵, you can use the uparrow and downarrow keys (in the cursor control keypad on the keyboard) to move from parameter to parameter.

You can display the action choices at any time by pressing function key 1 (the left key in the top row).

Try it: type 2↵, then press downarrow a few times. Eventually you'll arrive at the action choices again. Choose 1↵ and check out later pages. You can't go farther than page 2 without specifying a USAM. Then return to screen 1 via choice 3↵. Finally, check the amount of memory the system will consume by typing 4↵. (This memory figure is in 16-bit words. Multiply it by two to get the number of bytes.)

To summarize the action choices:

Go to next page and *Go to previous page*, choices 1 and 3, allow you to change pages.

Edit parameters, choice 2, allows you to edits values on the current page. You can position the cursor with the uparrow and downarrow keys and type new values at will. The valid range of answers appears in parentheses before the colon. The default value appears *after* the colon.

Size the system, choice 4. This tells you how much memory, in two-byte words, the system will occupy as currently configured. The system must consume less than 31 K words (62 Kbytes) to run. Check this as often as you please.

Exit with/without change, choice 5. This exits with or without change. If you type 5↵, CONFIG asks

```
Do you want to install changes in the system (Y/N):
```

If you type N↵, CONFIG makes no changes and terminates. It does not write CONFIG dialog to the dialog file, if you specified one with the name/V switch.

If you answer Y↵, CONFIG computes and displays the memory requirements of the configured DG/RDOS system. Then it writes the new values to the system file (and dialog to any log file), and terminates. The system changes aren't effective until you start up the newly-configured system.

Parameters on Page 1 (Memory, Stacks, Cells, Buffers)

The first parameter on page 1 is memory size:

- *Memory size in Kb (128-2048) : n*

Generally, this value should show the amount of memory in the computer on which the new system will run. (Kb means Kilobyte, abbreviated Kbyte; a Kbyte is 1,024 bytes.) For Model 10 and 10/SP systems, valid answers are 128 to 768 in steps of 128: 128, 256, 384, 512, 640, or 768. If you don't know the amount of memory in your machine, you can run the Customer Diagnostics diskette or check the packing list for your hardware. For example, after positioning with the up- and downarrow keys, you might type

```
768 ↓
```

- *Number of FG/BG sharable pages (0-255) : 0*

BG and FG mean background and foreground program memory. At startup, all memory is allocated to the background program, the CLI. If you have enough memory, you can allot some of it to the foreground, then run a program in the foreground.

Some DG applications products need foreground/background sharable pages. FORMA-TEXT and COMPUCALC don't require sharable pages; neither do most other application products. MS-DOS and CP/M-86 must execute in the background only, thus have no need for sharable pages.

If you will be running an application program in the foreground, and think sharable pages might be needed, consult the product manual or Release Notice for installation/CONFIG information. Generally, this value can stay 0.

- *Number of BG I/O channels (16-255): 40*
- *Number of FG I/O channels (0-255) : 0*

A *channel* is a memory area that holds information about a file (or device); each open file needs at least one channel.

For any program you plan to run, the number of channels the program needs must be available in the ground where you plan to run it. For example, if program MYPROG requires 25 channels and you want to be able to run MYPROG in either ground, the BG value and the FG value must each be at least 25. The CLI requires 16 channels; if you want to run a foreground CLI, make sure the value for FG channels is at least 16. Table 2-5, earlier, shows the number of channels needed by some popular products.

Each channel reserved by CONFIG consumes about 74 bytes of memory, whether or not the program actually uses the channel. This memory is lost for other uses. So, if a program you want to run uses a lot of channels, you will probably not want to reserve them in both grounds. Choose a ground — background or foreground — for the program to run in, and specify enough channels for that ground. A program in either ground can run on multiple terminals.

For BG channels, change the value if a program you want to run in the background (like MS-DOS) needs more channels.

For FG channels, you must specify at least 16 channels to run the CLI — more for some other programs (Table 2-5).

(Remember that — after you type an answer to CONFIG — you can press function key 1 to move to the bottom of the page, then type 4↵ to have CONFIG size the system for you.)

- `Number of stacks (1-10): 3`

DG/RDOS uses stacks for system calls, disk I/O, and printer operations. You need 1 stack for the background program, 1 stack for the printer (if any), and 1 stack for the foreground program (if you will run a a program in the foreground).

If your system has a printer and more than one terminal, we recommend 4 or 5 stacks. Stacks use relatively little memory — but for every stack over three, CONFIG reserves two extra system buffers. For single-terminal systems, the original default of 3 is fine.

- *Number of system buffers (6-63): 6*

DG/RDOS uses system buffers for temporary storage of disk-based data. Each buffer consumes 540 bytes. We recommend the default value (twice the number of stacks). Unless you know that you need a nonstandard number of buffers, leave the default value unchanged.

- *Number of cells (9-64): 12*

DG/RDOS uses cells in conjunction with stacks and buffers. They are modest (32 bytes each). Generally, you don't need to change the number of cells. Leave it unchanged.

- *Number of directories accessible at one time (1-64) : 10*

The system disk(ette) is a directory (the master directory). On it, and other disk(ette)s, you can create subordinate directories, called secondary partitions and subdirectories. Users and programs can address any of these subordinate directories as if it were a self-contained disk drive.

Directories are a useful part of DG/RDOS. DGs BASICs and other software rely on them. This value limits the number of directories that can be *initialized* at one time; it doesn't limit the number of directories that can be *created.*

Generally, you don't need to change the original default value (10).

Page 2 Parameters (Devices, USAM)

The CONFIG page 2 parameters are

- *8 bit character support for the master console ($TTO) ? (Y/N) : N*

Normally, DG/RDOS provides 7-bit character support, which includes the entire US ASCII character set (on US terminals). The 7-bit support also allows special native characters on non-US terminals to display as expected.

On the other hand, all terminals supplied with DESKTOP GENERATION systems can handle 8-bit characters — providing an additional 128 characters (like the UK currency symbol), called the international character set.

For 8-bit character handling on the system console, the value of this parameter must be Y. Also, on Model 20 and 30 systems, certain DIP switches behind the terminal must be set to allow 8-bit operation. Specifically, the mode DIP switch must be set to the 8 bit position, and the parity DIP switches must be set to "none" (two switches).

Leave the value alone or change it, as desired.

- *Secondary console/Printer port? (Y/N) : Y*

DESKTOP GENERATION systems can have a second terminal or printer that works directly with the computer (not through a multiplexor). A terminal connected this way can run a second CLI in the foreground. A printer connected this way can spool (store print requests temporarily on disk until printing occurs). Thus, any terminal or printer connected to the secondary console/printer port interface has an advantage over a terminal or printer connected to a USAM line.

On a Model 10 or 10/SP, the CPU has a built-in printer port. A Model 20 or 30 can have an optional secondary console board.

If this DG/RDOS system will run on a Model 10 or 10/SP with a terminal *or* printer connected to the printer port, the value of this item must be Y. Or, if this system will run on a Model 20 or 30 with a secondary console board, the value must be Y.

Generally, for a terminal on the secondary console/printer port to work, its switches must be set fo 9600 baud, mark parity (unless you need 8-bit characters), and DG mode. The correct DIP switch settings on the back of the terminal (HOST group) are as follows:

Switch number:	1	2	3	4	5	6	7	8
Position:	Up	Up	Up	Down	Up	Down	Down	Down

Generally, for a printer on the secondary console/printer port to work, its firmware parameters must be like those on a terminal: 9600 baud and mark parity (unless you need 8-bit characters).

8 bit character support for the secondary console ($TTO1)? (Y/N) : N

DASHER D211, D410, and D460 terminals support 8-bit operations. For a D210, the value must be N. The default, 7-bit support, is suitable for most applications. But if you want 8-bit character support, and the printer or terminal can handle 8-bit characters, make sure the value is Y. For 8-bit support, the device switch settings must be set as shown above, *except* that they must indicate no parity and 8-bit mode (for a terminal, this means switches 5 and 6 must be down; switch 7 must be up).

- *Printer ($LPT)? (0=NONE, 1=$TTO, 2=$TTO1, 3=USAM line 0,*

  4=USAM line 1, 5=USAM line 2, 6=USAM line 3) : 0

On a Model 10 or 10/SP, a printer can be on the CPU printer port (choice 2, $TTO1) or on a USAM line if your system has a model 4463 USAM card. If you don't have a USAM, the printer *must* be on the printer port.

On a Model 20 or 30 system, the printer can be on the optional secondary console board (choice 2, $TTO1), or a USAM line.

If you set up your own hardware (as described in the *Installing* manual), you know where the printer is attached. If you don't know, check the Configuration Sheet at the end of the *Installing* manual, may help, if someone has filled it in.

If you have no printer, the value for this parameter should be 0 (none).

According to the printer value you want for this DG/RDOS system, change this parameter or leave it as is.

Even if a system has no printer, people using it have *some* printing options. Printer output is always sent to a file named $LPT — even on a system without a printer, from which the file can be taken to a system *with* a printer. Doing this is described later, in Chapter 5, under pertinent commands.

- *Plotter ($PLT)? (0=NONE, 1=$TTO, 2=$TTO1, 3-USAM line 0,*

  4=USAM line 1, 5=USAM line 2, 6=USAM line 3) : 0

As with the printer, on a Model 10 or 10/SP, a plotter can be on the printer port ($TTO1) or a USAM line. On a Model 20 or 30, it can be on the secondary console ($TTO1) or a USAM line.

If you have a plotter, make sure this parameter describes the line it's on. If you don't have a plotter, the answer must be 0. With no plotter, DG/RDOS will write output of PLOT command to file $PLT, which you can later copy to diskette and take to a system that has a plotter.

- `Magnetic tape drive ? (Y/N) :`

If you want this DG/RDOS system to support a cartridge tape drive, the value of this parameter must be Y. Otherwise, it should be N. (If you specify a tape for the system, you might want to move down to the bottom of the page and type 4⏎ to size memory.)

- `USAM ? (Y/N) : N`

This parameter pertains to DG's (optional) USAM multiplexor for DESKTOP GENERATION systems (model 4463 universal synchronous/asynchronous multiplexor). A system can support no USAM, a one-line USAM, or a four-line USAM.

A USAM line can be used for communications, or attached to a user terminal, a printer, a plotter, a mouse, or a data tablet. "Communications" means interaction with another system — perhaps via a modem and telephone line, or interaction with a remote terminal over a modem.

If you don't want this DG/RDOS system to support a USAM, the value for *USAM* should be N. If the value is N, this is the last CONFIG parameter. You've finished configuring this DG/RDOS system. You might want to review earlier pages before typing 5⏎, then Y⏎ to change the system. Then start up the new system as described in the next chapter. Sample CONFIG parameters are shown later in *this* chapter, Figures 2-3, 2-4, and 2-5.

If you want the new DG/RDOS to support a USAM, the value for USAM should be Y. But multiuser Business BASIC systems will require you to answer N. See the *Managing Business BASIC* manual for more detail.

If the new system is to support a USAM, a USAM break character and each USAM line must be defined (or defaulted). So, if you change the USAM value to Y, or it's already Y, please check the next CONFIG pages for the correct values. Use action choice 1⏎.

Parameters on Page 3 (USAM Specifications)

The first parameter on page 3 is

- *Break character for USAM : 1*

  (Enter decimal ASCII character code, or "128" for none)

A break character is a key sequence that, when typed on the keyboard, stops program execution. There are two break characters for terminals on USAM lines. One is CTRL-C, which you cannot change; it's always a break character for USAM lines. The second break character — the subject of this question — is CTRL-A by default. (The ASCII value of CTRL-A is 1, which shows up as the original default.)

If any of your USAM terminals will run Interactive COBOL or BASIC (Extended or Business), the break character should be ESC. This is true because ICOBOL and BASIC users expect to be able to interrupt program execution by pressing the ESC key. The ASCII value of ESC is 27 (decimal).

Thus, if ICOBOL or BASIC will run on any USAM-controlled terminal, the value of this parameter must be 27. Otherwise, generally, the original default of 1 (CTRL-A) is okay. But if you want to define your own second break character (perhaps for your own programs or programs not supplied by DG), then make sure the value in this parameter is the (decimal ASCII) value you want. The ASCII values of all characters appear in Appendix A.

If you define ESC or your own special character as a break character, it replaces CTRL-A. CTRL-C remains a USAM break character. The key sequence for terminals not on a multiplexor is CTRL-C CTRL-A, and you can't change this.

- *Number of lines on USAM (1-4): 4*

This value describes the number of USAM lines to be supported by this DG/RDOS system. For a one-line USAM, make sure it's 1; for a four-line USAM, make sure it's 4. For example, 4↵.

USAM lines are numbered from 0. The first is line 0, the second (if any) is line 1, the third is line 2 and the fourth is line 3.

The parameters on CONFIG's next page describe line 0. For a four-line USAM, the pages following the next describe line 1, 2, and 3. You should check the line values on the next page. Please do so.

Parameters on Pages 4 through 7 (USAM Lines)

As you check and confirm values for each line, go to the next page, until you've checked and confirmed all values.

- *Line number (0-3) : n*

Number *n* is 0 (page 4), 1 (page 5), 2 (page 6), or 3 (page 7). If you change the value, CONFIG will display the appropriate page for the line.

- *Line speed (in bits/second) for USAM from table below :*
  0=3600, 1=19200, 2=50, 3=75, 4=134.5, 5=2000, 6=600,
  7=2400, 8=9600, 9=4800, 10=1800, 11=1200, 12=7200, 13=300,
  14=150, 15=110

Line speed is the rate at which a line or modem can transfer data, in bits per second (also called baud). By default, each character requires 10 bits — producing a standard line speed for each type of line, as follows. *Note* that the line that connects a DESKTOP system to another DG computer system must be line 0.

Line connects to terminal, printer, mouse, data tablet, or plotter	Line connects to modem - remote terminal	Line connects to another DG system
9600	1200 (newer modem) or 300 (older modem); see the modem for the number.	4800

Make sure the line speed parameter for this USAM line is correct (8 for 9600, and so on).

- *Number of stop bits (1-2) : 1*

On any line, the standard device uses 1 stop bit (default) and 1 start bit. Unless you have a specific reason for selecting a nonstandard number of stop bits, leave this value at its original number (1).

- *Number of data bits (5-8) : 7*

On any USAM line, a terminal or printer will work normally with 7 data bits per character and even parity; 7 data bits is the original DG default. But, for 8-bit character handling on this line, the value must be 8. (For a DASHER D210 terminal, use 7.)

If you're considering 8-bit character handling, read the following paragraphs for some background.

DG DASHER model D211, D220, D410, and D460 terminals can send and display 8-bit characters. The main advantage of 8-bit character handling is the ability to read and display characters with values above 177 octal — which includes many characters in the international character sets (like the UK currency symbol) and other special characters. In 7-bit mode, the high bit is ignored — which means the console can't see codes above 177 octal.

One disadvantage of 8-bit character handling is that, on a terminal with a non-US keyboard, some keys send different codes in 7-bit mode from 8-bit mode. In 7-bit mode, a terminal's primary character set is based on its native keyboard, but in 8-bit mode, the primary character set is U.S. This means, for example, that on a French keyboard the C Cedilla key will produce C with a cedilla in 7-bit mode, but will produce ^\ ASCII 34, using the U.S. character set, in 8-bit mode.

If there is a multifunction printer on this line, or if you want 8-bit character handling on the terminal on this line, make sure the value is 8. For 8-bit handling to work, device switch settings must allow it (shown below). Also, you must specify *no parity* for this line, next.

- *Parity (0=NONE, 1=ODD, 2=EVEN) : 2*

Parity is a way of checking for correct transmission of a character. For 7-bit characters, the value of this should be 2 (EVEN parity). For 8-bit characters, the value should be 0 (no parity).

For a terminal on this line, the DIP switches on the back of the terminal (HOST group) must be set as follows:

Switch number:	1	2	3	4	5	6	7	8
Position for 7 data bits:	Up	Up	Up	Down	Up	Up	Down	Down
Position for 8 data bits:	Up	Up	Up	Down	Down	Up	Up	Down

If the device settings are wrong for the data bits and parity you specify to CONFIG (or default), the device probably won't work. You may see strange looking characters or blocks. Keystrokes may not echo properly.

Modem (Y/N) : N

A terminal can connect directly to the computer, or it can connect remotely via a modem. There are actually two modems: the computer connects to one modem, which is attached to a telephone. The remote terminal has another modem connected to a telephone. The person at the remote terminal can dial up the computer — allowing him/her to work at home, for example.

If this USAM line will be connected to a modem, make sure the value is Y. Otherwise, the value should be N.

For each USAM line connected to a modem, a *Ring Indicator* switch on the USAM card must be set on. Doing this is described in the *Installing* manual, the chapter on card installation.

- *Loopback mode (Y/N) : N*

Loopback mode allows your system to both send and receive messages over a communications line. It is quite useful when you want to check out communications hardware and software (for example, try out DG/BLAST). However, a line in loopback mode can't communicate with the other system. So, you'll generally want to use loopback mode to test things after you install hardware and software. Then, if everything works properly on your system, run CONFIG again, change the value to N, leave CONFIG, and start up the newly-configured system.

Leave the value as is, or change it, based on your desire for this line.

- *Data Set Ready (DSR) : 0*
- *Request To Send (RTS) : 1*
- *Data Terminal Ready (DTR) : 1*

These three items refer to modem line status signals. If you don't have a modem on the line, leave the original values as they are.

Generally, unless you know that the modem on this line requires a different initial setting for DSR, RTS, and DTR, keep the original defaults. If the default values don't work, consult your modem manual for the required initial values.

Proceeding

The modem status parameters are the last ones on this page. If there are more pages, this means you should review and possible change the values for at least one more USAM line (unless it's not connected to anything).

If you've finished defining all the USAM lines on which the system will run devices, you're done. Congratulations! Type

```
5 ↵
```

and CONFIG asks

```
Make changes to system file (Y/N)
```

If you type N↵, CONFIG will make no changes and record no dialog. But to configure the system, you must type

```
Y ↵
```

CONFIG now describes the amount of memory needed by the system in words (multiply by 2 to get bytes), and the dialog file, if any:

```
System size in words: n
Writing dialogue to file x
```

If the system is too large, CONFIG says so. You must make it smaller, perhaps by specifying a smaller number of stacks (if you specified more than 3 stacks) or channels. If the system isn't too large, CONFIG makes changes to the system file and creates the log file. Then, the CLI prompt returns to the system console:

```
R
```

The new system is ready to roll. If you built the system for your current hardware, or for a system that has less hardware than the current system, you should now try it out.

Testing the Configured System

To run, a DG/RDOS system file must be in the master directory (like DE0 or DJ0); it cannot be in a subdirectory. Assuming this is true, you can check the system as follows:

```
R
BOOT system ⤶            (system is the name of the system you just
                         configured; for example, DGRDOS.SV; you can omit
                         the .SV extension).

MASTER DEVICE RELEASED

DG/RDOS x.xx

DATE (M/D/Y)? 11 5 84⤶       (Type current date.)

TIME (H:M:S)? 16 45⤶       (Type current time, 24-hour clock.)

WELCOME TO THE DG DESKTOP GENERATION
Emulator is already loaded         (Displayed on model 10, 10/SP only)

R
```

Your configured DG/RDOS is up and running. Now can install other software (if needed) as described next; or you can try to use the device(s), communications lines, and software you just configured.

If the system is configured for hardware you don't have, put it on a a system diskette, take it to the other system, and try as described above. Making a system diskette is sketched earlier in this CONFIG section, under "Creating Copies of DG/RDOS Systems."

Summary figures of CONFIG parameters for different DG/RDOS systems follow in Figures 2-3 and 2-4. These were taken from actual CONFIG dialog files.

```
               DGRDOS System Configuration Program     REV:  1.10
System name: DGRDOS.SV          Page  1       FUNCTION-KEY-1 FOR COMMANDS
----------- USE ARROWS (UP,DOWN,LEFT,RIGHT) TO MOVE CURSOR ---------------

     Memory size in Kb (128-2048) :  768

     Number of FG/BG sharable pages (0-255) :    0

     Number of BG I/O channels (16-255):   40

     Number of FG I/O channels (0-255) :   40

     Number of stacks (1-10):  3

     Number of system buffers ( 6-63):  6

     Number of cells ( 9-64): 12

     Number of directories accessible at one time (1-64) :   10
```

Figure 2-3 CONFIG values for a DG/RDOS system without a USAM multiplexor (continues)

```
8 bit character support for the master console ($TTO) ? (Y/N) :  N

Secondary console/Printer port? (Y/N) :  Y

8 bit character support for the secondary console ($TTO1)? (Y/N) :  N

Printer ($LPT)? (0=NONE, 1=$TTO, 2=$TTO1, 3=USAM line 0,
                4=USAM line 1, 5=USAM line 2, 6=USAM line 3) :  2

Plotter ($PLT)? (0=NONE, 1=$TTO, 2=$TTO1, 3=USAM line 0,
                4=USAM line 1, 5=USAM line 2, 6=USAM line 3) :  0

Magnetic tape drive ? (Y/N) :  N

USAM ? (Y/N) :  N
```

Figure 2-3 CONFIG values for a DG/RDOS system without a USAM multiplexor (concluded)

```
                DGRDOS System Configuration Program     REV:  1.10
System name: EBASIC$SYS.SV          Page  1        FUNCTION-KEY-1 FOR COMMANDS
----------- USE ARROWS (UP,DOWN,LEFT,RIGHT) TO MOVE CURSOR ---------------


    Memory size in Kb (128-2048) :  256

    Number of FG/BG sharable pages (0-255) :    0

    Number of BG I/O channels (16-255):   50

    Number of FG I/O channels (0-255) :   16

    Number of stacks (1-10):  5

    Number of system buffers (10-63): 10

    Number of cells (15-64): 15

    Number of directories accessible at one time (1-64) :   10
```

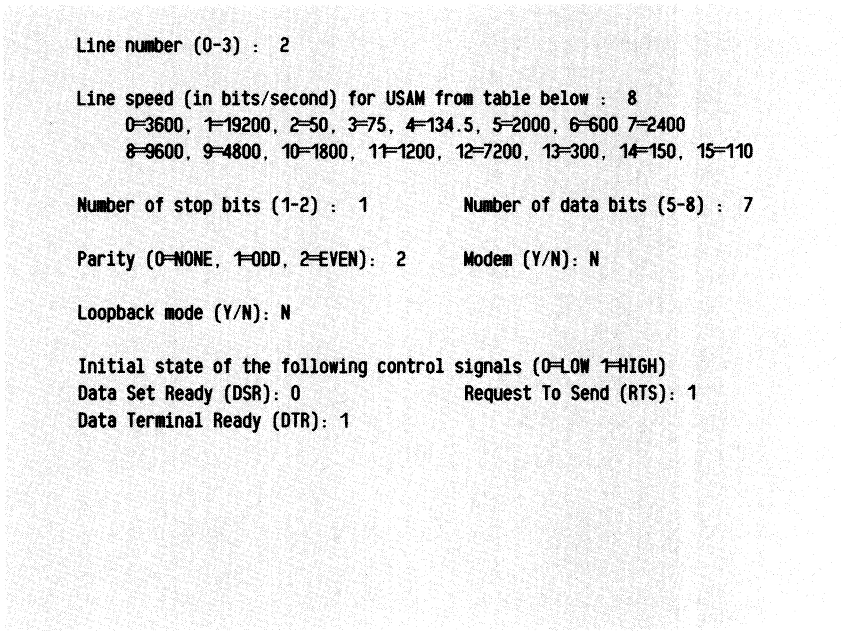
Figure 2-4 CONFIG values for a DG/RDOS system with a printer on the printer port and 4 USAM lines. (continues)

```
8 bit character support for the master console ($TTO) ? (Y/N) :  N

Secondary console/Printer port? (Y/N) :  Y

8 bit character support for the secondary console ($TTO1)? (Y/N) :  N

Printer ($LPT)? (0=NONE, 1=$TTO, 2=$TTO1, 3=USAM line 0,
                 4=USAM line 1, 5=USAM line 2, 6=USAM line 3) :  2

Plotter ($PLT)? (0=NONE, 1=$TTO, 2=$TTO1, 3=USAM line 0,
                 4=USAM line 1, 5=USAM line 2, 6=USAM line 3) :  0

Magnetic tape drive ? (Y/N) :  N

USAM ? (Y/N) :  Y

Break character for USAM :  27
     (Enter decimal ASCII character code, or "128" for none)

Number of lines on USAM (1-4) :  4
```

Figure 2-4 CONFIG values for a DG/RDOS system with a printer on the printer port and 4 USAM lines. (continued)

```
Line number (0-3) :  0

Line speed (in bits/second) for USAM from table below :  8
     0=3600, 1=19200, 2=50, 3=75, 4=134.5, 5=2000, 6=600 7=2400
     8=9600, 9=4800, 10=1800, 11=1200, 12=7200, 13=300, 14=150, 15=110

Number of stop bits (1-2) :  1          Number of data bits (5-8) :  7

Parity (0=NONE, 1=ODD, 2=EVEN):  2      Modem (Y/N) :  N

Loopback mode (Y/N): N

Initial state of the following control signals (0=LOW 1=HIGH)
Data Set Ready (DSR): 0                 Request To Send (RTS): 1
Data Terminal Ready (DTR): 1

Line number (0-3) :  1

Line speed (in bits/second) for USAM from table below :  8
     0=3600, 1=19200, 2=50, 3=75, 4=134.5, 5=2000, 6=600 7=2400
     8=9600, 9=4800, 10=1800, 11=1200, 12=7200, 13=300, 14=150, 15=110

Number of stop bits (1-2) :  1          Number of data bits (5-8) :  7

Parity (0=NONE, 1=ODD, 2=EVEN):  2      Modem (Y/N): N

Loopback mode (Y/N): N

Initial state of the following control signals (0=LOW 1=HIGH)
Data Set Ready (DSR): 0                 Request To Send (RTS): 1
Data Terminal Ready (DTR): 1
```

Figure 2-4 CONFIG values for a DG/RDOS system with a printer on the printer port and 4 USAM lines. (continued)



```
Line number (0-3) :  2

Line speed (in bits/second) for USAM from table below :  8
     0=3600, 1=19200, 2=50, 3=75, 4=134.5, 5=2000, 6=600 7=2400
     8=9600, 9=4800, 10=1800, 11=1200, 12=7200, 13=300, 14=150, 15=110

Number of stop bits (1-2) :  1         Number of data bits (5-8) :  7

Parity (0=NONE, 1=ODD, 2=EVEN):  2     Modem (Y/N): N

Loopback mode (Y/N): N

Initial state of the following control signals (0=LOW 1=HIGH)
Data Set Ready (DSR): 0                Request To Send (RTS): 1
Data Terminal Ready (DTR): 1
```

Figure 2-4 CONFIG values for a DG/RDOS system with a printer on the printer port and 4 USAM lines. (continued)

```
Line number (0-3) :  3

Line speed (in bits/second) for USAM from table below :  8
     0=3600, 1=19200, 2=50, 3=75, 4=134.5, 5=2000, 6=600 7=2400
     8=9600, 9=4800, 10=1800, 11=1200, 12=7200, 13=300, 14=150, 15=110

Number of stop bits (1-2) :  1          Number of data bits (5-8) :  7

Parity (0=NONE, 1=ODD, 2=EVEN):  2      Modem (Y/N): N

Loopback mode (Y/N): N

Initial state of the following control signals (0=LOW 1=HIGH)
Data Set Ready (DSR): 0                 Request To Send (RTS): 1
Data Terminal Ready (DTR): 1
```

Figure 2-4 CONFIG values for a DG/RDOS system with a printer on the printer port and 4 USAM lines. (concluded)

Backing Up the DG/RDOS System Files

Next, you should make a system backup diskette of essential DG/RDOS system files — especially your configured system files. This may seem like extra effort after all you've done, but losing files from your configured disk(ette) — without backup — would be worse.

Backing Up System Files from a Hard Disk

If you have a hard disk, back up the system files as follows:

1. Get a blank or scratch diskette. If this isn't hardware formatted, format it as described in Chapter 6.
2. Use FCOPY to make a duplicate of the DG/RDOS system diskette; for example, FCOPY/D/V DJ0 DJ0⤶. Chapter 6, section "Copying Diskettes", describes using FCOPY with one drive or two drives. Be sure to choose the verify option (FCOPY/V). If FCOPY finds any bad blocks on the destination diskette, use another diskette.
3. Check the duplicate diskette as shown in the FCOPY session, and label it; for example, "DG/RDOS System Backup," with the date.
4. Remove the original DG diskette from DJ0.
5. Insert the duplicate diskette in drive DJ0. Then type the following commands:

```
INIT DJ0 ⤶
R
MOVE/V/R    DJ0    DGRDOS.SV    -.TX    -.EM ⤶
.
. (it verifies files moved)
.
R
RELEASE DJ0 ⤶
R
```

   This produces a diskette with a copy of your configured system, from which you can start DG/RDOS, or copy system files, if needed in the future.

   If your configured system has a name other than DGRDOS.SV, type that name instead of DGRDOS.SV.

5. Remove the duplicate diskette from drive DJ0 and store it safely.

Backing Up System Files — Diskette-Based Systems

With a diskette-based system, you should copy your configured system diskette — *and* the DG-supplied Stand-Alone Utilities diskette, if you received this latter diskette.

For every diskette you want to copy, follow these steps:

1. Get enough blank or scratch diskettes to serve for copies. If these aren't hardware formatted, format them as described in Chapter 6.
2. Use FCOPY to make a duplicate of your DG/RDOS system diskette; for example, FCOPY/D/V DJ0 DJ1↲. Chapter 6, section "Copying Diskettes", describes using FCOPY with one drive or two drives. Be sure to choose the verify option (FCOPY/V). If FCOPY finds any bad blocks on the destination diskette, use another diskette.
3. Check the new duplicate diskettes as shown in the FCOPY session, and label them. For example, you could label the system diskette copy "DG/RDOS System Backup," with the date.
4. Replace the duplicate and original in outer envelopes and store the original safely.
5. Use the duplicate instead of the original to run DG/RDOS, and/or when you need to use a stand-alone program like DKINIT or YOWNER.

Shutting Down

For neatness, shut down the system. You can do this only from the system console (not a user terminal). Type

```
DIR %MDIR% ↲
R
BYE ↲

STARTING SYSTEM SHUTDOWN

MASTER DEVICE RELEASED
Filename?
```

Type the break sequence (CMD and BREAK/ESC keys).

```
!
```

Congratulations! You have installed DG/RDOS, configured it, and tested it. DG/RDOS and all needed support software are complete on the system disk(ette). You may also have installed other needed support software.

You can bring DG/RDOS up again by typing nnH, ⤶, the date and the time. And you can bring up a *foreground* program using commands explained in Chapter 3. Using DG/RDOS is much easier than installing and configuring it.

CAUTION *Orderly shutdown is very important. Never turn computer power off when DG/RDOS is running. Turning power off while DG/RDOS is up produces an abnormal (incomplete) shutdown. System users may lose some of the work they've done on the system.*

What Next?

If you want to stop for a while, fine. Later, you can bring the system back up as described in the next chapter.

If this is your first system, you may want to check startup/shutdown steps, described in the next chapter; or you may want to try the CLI session described in Chapter 4; or get some system background (Chapter 5); or try a generic operating system like MS-DOS (see the manual described in the preface).

If you are rebuilding/restoring a system, the next steps are to restore information from backup diskettes (Chapter 7).

Starting Up and Shutting Down 3

Read this chapter when

- you want to start up your computer or DG/RDOS system;
- you want to shut down your DG/RDOS system or computer;
- your computer system has stopped unexpectedly or power has failed.

This chapter gives the details on system startup and normal shutdown. It also touches on abnormal shutdown. The major sections are

- The Break Sequence
- Startup
- Normal Shutdown
- Abnormal Shutdown or Power Failure
- What Next?

The Break Sequence

The break sequence stops the computer, freezing all system activity, and gives control of the system console to a loader program (! prompt.) Generally, you should avoid typing the break sequence unless DG/RDOS is deadlocked (frozen, with no response at any terminal).

The break sequence is the CMD and BREAK/ESC keys. If someone types the the break sequence at the system console, the computer halts and displays a ! prompt. DG/RDOS and all programs it is running are suspended. To have the computer proceed, type P↵. The P↵ command tells the computer hardware to proceed, and system activity resumes.

Accidental break sequences can be disconcerting and annoying. But they are not disastrous if you know how to handle them.

Startup

Startup includes turning power on (if off), starting up a DG/RDOS operating system, and bringing up a foreground program (if desired). It assumes that DG/RDOS was shut down normally, not abnormally (shown by five 6-digit numbers displayed on the system console), or by a power failure. Abnormal shutdowns and power failures are described later in this chapter.

On Model 10 and 10/SP computers, when you turn power on, the system console has limited capability. For example, it can display only UPPERCASE letters. To give the system console full capability, a DG/RDOS program called LOADEM loads a terminal emulator program from the system disk(ette) at startup.

If the system console remains limited — for example, it can't produce lowercase letters — after DG/RDOS is up, then the emulator program file may be missing from the disk(ette).

Both a link entry file and emulator file are needed for LOADEM to load the emulator. The link entry filename is MTERM$.EM (monochrome monitor) or CTERM$.EM (color monitor). The emulator filename varies according to the native language of keyboard. For terminals with U.S. or U.K keyboards, the emulators filename is MD211AMUK.TX (monochrome) or CD211AMUK.TX (color). You can check for these files from DG/RDOS by typing

```
LIST/E/A    *TERM$.EM    *D211-.TX ↵
```

If two filenames, form xTERM$.EM and xD211yyyy.TX don't show up, you must install the emulator program on the system disk(ette) as shown in Chapter 2, "Installing the Emulator." Type BYE↵ to DG/RDOS before returning to Chapter 2.

If You Make a Mistake

While bringing the system up, you can recover from typing errors by pressing the DEL key until you've eliminated the wrong characters; then type the correct ones. Usually, DEL erases characters from the screen, so you can see what's happening.

On Model 20 and 30 systems, before DG/RDOS starts up, the DEL key does not erase characters; instead, it displays as underscore (_) for each character erased. For example,

36__

indicates that someone pressed the keys 3, 6, DEL, DEL. The 36 is erased and you can type the desired number.

If you type a wrong number and pressing the DEL key has no effect, this probably means that you completed the command and that the computer is trying to act on it. Type the break sequence (CMD and BREAK/ESC keys) to regain the ! prompt. Then type the command again.

To start up DG/RDOS, follow these steps:

1. If your system doesn't have a hard disk, insert the DG/RDOS system diskette in drive DJ0 (the right drive).

2. On the printer (if you have one), make sure printer power is on, and that the printer is on line. Use the printer switches. The printer should have paper.

   On a model 10 or 10/SP, if a printer or terminal is connected to the printer port, the computer tests the printer/terminal. If the device fails the test — which would happen if it were off line — the computer may not complete its power-up test.

   If you must use your system, and a device on the printer port won't pass the test, turn off power to all devices and disconnect the printer port device at the cable connection near the device. Then return to step 1.

3. Turn on power to the tape module (if any) and the second hard disk drive (if any).

4. If the system console is turned off, turn it on. Use the switch behind the screen, on the right. The system console may show a self-test message and beep, or it may do nothing.

5. Turn on power to the computer unit. Use the main power switch on the upper right of the computer module.

   On to a Model 10 or 10/SP, the computer runs tests. The screen fills with shifting patterns. Then, it clears and displays

   ```
   TESTING...
   ```

   For every test passed, the screen displays a letter or number. Eventually, it shows

   ```
   ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
   BLINK DIM REVERSE UNDERLINE ALL NORMAL

   TEST PASSED.
   ```

   On a Model 20 or 30, it displays the ! prompt:

   ```
   !
   ```

   (If the screen displays nothing, check the brightness (contrast) control under the right front corner of the mnitor. Set this contrast control in a central position. If this doesn't help, sure that everything is plugged in and that connections are secure. Turn computer power off, check that other devices are turned ON, and turn power on again. If this doesn't help, consult the hardware *Testing...* manual.)

   Without a hard disk, the system does the next step for you. It displays LOADING FROM DISKETTE and you can skip to step 7.

   If your system has a working hard disk, it displays

   ```
   SELECT LOAD DEVICE...
   !
   ```

6. Follow the advice. Type 26H for the hard disk or 20H for a system diskette in drive DJ0. For example,

   `26H` (or `20H` for diskette)

7. The system console displays

   ```
   FILENAME?          (Model 10 and 10/SP)
   or
   Filename?          (Other systems)
   ```

   It's asking for the filename of the system or program to run.

   (If, with a European keyboard, you get a ?! message, use the numeric keypad on the far right to type the number.)

   (If the Filename? doesn't appear after 30 seconds or so, perhaps you mistyped the number. Type the break sequence (CMD and BREAK/ESC keys) and retry step 6. If this doesn't help — starting from diskette — maybe the system diskette isn't inserted properly. Type the break sequence again. Then, remove the diskette from DJ0 and check the label to make sure it's a DG/RDOS system diskette. If it is a system diskette, turn power off, insert the diskette in DJ0, and return to step 5.)

8. With the Filename? question, you've ready to start DG/RDOS or a program like DKINIT.

   Whenever the system console asks Filename?, you can specify a DG/RDOS system or program like DKINIT by typing its name and ↲. If the system is named DGRDOS.SV (as is true for the supplied system), you can press ↲ to specify its name. For example, type

   ↲

   (If you see the message FILE DOES NOT EXIST, you may have made a mistake typing the name. Try again.)

   The bootstrap program, MBOOT.SV, reads the system into computer memory and passes control to it. From diskette, this takes awhile. Then, the DG/RDOS system asks

   ```
   DG/RDOS REV x.xx
   DATE (M/D/Y)?
   ```

9. It's very important to give the correct date and time. Every file that you — and other users — create while the system is up will have a creation date/time based on the date and time you enter. A file's creation date is vital information.

   If you type the wrong date or time here, continue until you see the CLI R prompt. Then correct the date or time using the SDAY or STOD command, Chapter 5. The system date and time can be set only from the system console (not a foreground or user terminal).

   Type the date as numbers for month, day, and year. Insert a space or slash before the day and year. For example, for December 14, 1984, type

   ```
   12 14 84 ↵
   ```

   (If, with a European keyboard, you get an ILLEGAL CHARACTER message, use the numeric keypad on the far right to type the date and time.)

   ```
   TIME (H:M:S)?
   ```

10. Type the time, based on a 24-hour clock, in hours, minutes, and seconds. (Minutes and seconds are optional. If you omit them, the system uses 00:00.) Insert a space or colon before the minute and second numbers. For example, for 2:30 p.m., type

    ```
    14 30 ↵
    ```

    The DG/RDOS system now runs the LOADEM program, which displays

    ```
    WELCOME TO THE DESKTOP GENERATION
    ```

    And on a Model 10 or 10/SP, displays either

    ```
    LOADING THE EMULATOR
    ```

    ```
    Emulator load completed
    ```
    (On cold start, from power off)

    or

    ```
    Emulator is already loaded
    ```
    (On warm start, from power on)

    ```
    R
    ```

    DG/RDOS is up; the CLI shows its R prompt, awaiting a command. You might want to adjust screen brightness for comfort here, using the brightness control under the panel.

Warm Start

You don't need to start DG/RDOS with power off. Startup has fewer steps if power has remained on to the computer since shutdown.

- If the system console shows a Filename? prompt, start at step 8.
- If the system console shows a ! prompt, start at step 6.

Starting Stand-alone Programs

The procedure for starting any stand-alone program (like DKINIT) is just like that for DG/RDOS, up to step 8. At step 8 (Filename?), you type the name of the stand-alone program, then work with that program instead of DG/RDOS. For example,

```
Filename? DKINIT ⤶
DISK INITIALIZER REV x.xx

DISK DRIVE MODEL NUMBER?
```

Running Programs from the CLI

The CLI, your interface to the operating system and other programs, is running on the system console.

Often, you'll want to use the CLI to run other programs, like COMPUCALC. This is easy. But, before you do it DG/RDOS must be aware of the directory that that holds the program, so that it can find the program. If the program you want is in the master directory, DG/RDOS is already aware of the directory.

If the program file is on a different disk or diskette, make sure the disk(ette) is available (insert if needed), and type DIR directory-name⤶. Details on using directories appear in Chapters 4 and 5; and details on *using* other software products appear in the product manual; but the method above is the easiest and most common.)

The Two Program Grounds

A program you run from the CLI on the system console is called a *background* program. Systems with at least 256 Kbytes of memory can run a second program — called the *foreground* program — simultaneously.

Running a foreground program is most useful if you have one or more user terminals to let people use the program. But, even with only the system console, the foreground is useful for things like compilations, sorts, and backups.

If you don't want to run two programs simultaneously, the background program will be the only program: forget about the foreground and proceed to do what you want (other program, DG/RDOS operations, and so on). Skip the next section.

Starting a Foreground Program

If your system has one or more user terminals, and people will be using them, each one should be turned on and be on line.

And, if the program you want is not immediately accessible, you must make its directory accessible (via the DIR command) as above.

Then you can allot some memory to the foreground. (At startup, the background has all memory.)

The SMEM command allots memory; to check the total amount, use the GMEM command. Finally, to start the program in the foreground, use the EXFG command. For example,

```
R
GMEM ↵                      (Check memory allotment.)
BG: 84 FG: 0                (All memory pages is allotted to back-
                            ground.)
R
SMEM 32 ↵                   (Give the background 32 pages.)
R
GMEM ↵                      (Check allotment again.)
BG: 32 FG: 52               (Background has 32 pages, foreground
                            the rest.)
R
EXFG/E MYPROGRAM ↵          (Start up foreground program, equal
                            priority.)
R
```

The GMEM figure indicates the number of 2,048-byte memory pages in background (BG) and foreground (FG). This reflects *all* memory, less memory occupied by DG/RDOS and the terminal emulator (Model 10 and 10/SP only).

Some DG products have minimum memory requirements; and some, like BASIC, may have maximums (they cannot run with *more* than 512 Kbytes). Check Chapter 2, Table 2-5, or the product Release Notice for specifics.

After the EXFG command sequence, the foreground program is running. The foreground terminal or user terminals (if any) may display the foreground program's prompt. People can use the program via these terminals.

Starting a Program in the Background

To run two non-CLI programs, you must start the foreground program first, then start the new background program.

If the program you want is not immediately accessible, you must make its directory accessible (via the DIR command) as above.

Then you need only type the program name and ⮐. For example,

```
R
MSDOS ⮐
.
.
```

Blank Lines for Your Own Startup Commands

There are many courses you can chart with your DG/RDOS and supporting software. We can't cover all of them, and suggest that you write here the commands used to bring up the software you run with DG/RDOS. Use the following blank lines for the commands.

(One way to ease the process of bringing it all up is to write a CLI macro to initialize or DIR to directories, set memory, and execute programs as desired. If you created such a macro — perhaps called UP.MC — someone would need only to type UP⮐ to bring it up. An UP macro can be a great boon to system operation.)

System 1	**System 2**
System name: ________________	System name: ______________________
______________________________	____________________________________
______________________________	____________________________________
______________________________	____________________________________
______________________________	____________________________________
______________________________	____________________________________
______________________________	____________________________________
______________________________	____________________________________

Cautions

Please observe these cautions while DG/RDOS is running.

- Don't turn computer power off when DG/RDOS is running. First, type BYE ↵ and wait for the system console to display MASTER DEVICE RELEASED, then Filename?. Turning power off while DG/RDOS has one or more files open produces an abnormal shutdown. Anyone using the system may lose a lot of work. And the files that were open may not be internally consistent (since they weren't closed normally).
- Don't remove an initialized diskette from its drive without releasing it. If this happens, reinsert the diskette in the original drive and type RELEASE diskette-name↵; for example, RELEASE DJ0↵.
- Don't turn power off while a diskette is in its drive.

Startup Summary

Your DG/RDOS system is up and running.

Figure 3-1, next, summarizes startup.

Warm start: If the system console displays *Filename?*, start at step 8. If it shows a *!* prompt, start at step 6.

1. With no hard disk, insert the DG/RDOS system diskette in DJ0.
2. If you have a printer, make sure it is on and on-line.
3. Power up tape drive (if any) and second hard disk (if any).
4. If the system console is turned off, turn it on.
5. Turn on power to the computer unit.

   A Model 10 or 10/SP computer runs tests. Any computer without a hard disk loads from diskette; skip to step 7.

   SELECT LOAD DEVICE...

6. *!* 26H (Type 26 or, for diskette, 20H.)
7. *Filename?*
8. ↵ (Press ↵ or type program name and ↵.)

   DG/RDOS REV x.xx

9. *DATE (M/D/Y)?* 12 14 84↵ (Type current date.)
10. *TIME (H:M:S)?* 14 30 ↵ (Type correct time, 24-hour clock.)

    . (emulator messages on Model 10 or 10/SP)

    .

    R

DG-26404

Figure 3-1 System startup summary

Normal Shutdown

Normal shutdown means orderly shutdown from an active DG/RDOS system to the ! prompt; and, optionally, to turning off computer power.

CAUTION *It's very important to shut DG/RDOS down normally. Never turn computer power off when DG/RDOS is running. Doing this produces an abnormal shutdown and may cause problems upon restarting. Other system users — if any — may lose work. Files left open may have inconsistent information in them.*

1. If any program other than the CLI is running on the system console, stop it normally, using the standard command or menu sequence for that program. The CLI R prompt shows on the system console.
2. If you know there is no foreground program running, skip to step 5. You can check by typing FGND⏎ on the system console; it will tell you whether or not a foreground program is running.
3. If anyone is using the foreground program (for example, ICOBOL), ask him/her to log off. If you shut down while other users are active, they may lose some of the work they've done.
4. From the system console, terminate the foreground by typing CTRL-C CTRL-F. It will display FG TERM, meaning that the foreground program is terminated. For example,

   ```
   CTRL-C CTRL-F
   ```

   FG TERM (It does not display the R prompt here.)
5. Get into the master directory using the DIR command. The master directory name is usually DE0 (hard disk) or DJ0 (diskette). Or, you can use the CLI variable %MDIR%, which contains the name of the master directory. For example

   `DIR DE0 ⏎` (Or `DIR DJ0⏎` or `DIR %MDIR%⏎`)

6. Shut down via the BYE macro (supplied by DG to simplify shutdown). For example,

```
BYE ↵

STARTING SYSTEM SHUTDOWN

MASTER DEVICE RELEASED

Filename?
```

   The message MASTER DEVICE RELEASED means that DG/RDOS has closed all directories and terminated normally. You can — if you want — start another system now.

7. To return to the ! prompt, which allows you to start up from either disk or diskette (20H or 26H), type the break sequence (CMD and BREAK/ESC keys). For example,

```
CMD BREAK/ESC
nnnnnn
!
```

   DG/RDOS shutdown is complete.

After shutdown, you can leave everything as is, you can cut power to the computer, or you can bring up DG/RDOS or a stand-alone program.

You might decide to cut power if you knew the system would be unused for a while, like over the weekend. If you *do* decide to cut power, remove any diskettes from their drives before you do it (turning power off while diskettes are inserted might affect data on the diskettes).

Remove the cartridge tape from the tape module (if any) and turn off the module's power. Cut power to the second hard disk (if any).

Turn off the printer, if any, and user terminals. Turn off the computer unit using the main power switch. Finally, turn off the system console.

Figure 3-2 summarizes normal shutdown.

1. If any non-CLI program is running on the system console, stop the program normally.

   R

2. Check for a foreground program:

   ```
   FGND ↓
   NO FOREGROUND PROGRAM RUNNING
   or
   FOREGROUND PROGRAM RUNNING
   ```

   If no foreground program is running, skip to step 5.

3. Warn foreground users of impending shutdown.

4. `CTRL-C CTRL-F` (CTRL-C CTRL-F terminates the foreground program.)

   FG TERM

5. `DIR DEO ↓` (Or `DIR DJO↓` or `DIR %MDIR%↓`)

6. `BYE ↓`

   STARTING SYSTEM SHUTDOWN

   MASTER DEVICE RELEASED

   Filename?

7. Type the break sequence (CMD and BREAK/ESC keys).
   !

   Power down devices (computer and system console last) if desired.

DG-26405

Figure 3-2 System shutdown summary

Abnormal Shutdown or Power Failure

An abnormal shutdown can result from a serious error condition (panic or hang), hardware failure, and/or power failure.

There are certain kinds of errors that DG/RDOS can recognize, but cannot fix. Such an error may involve hardware or software. In response, DG/RDOS *panics* (stops running and enters its panic routine). The system console displays five 6-digit numbers:

```
nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn
```

Panics are often caused by inconsistencies in hardware or software.

A *hang* also results from a serious error, but there is no panic message on the system console. Instead, DG/RDOS seems frozen. There's no response to commands you type at the system console.

If the system seems hung, try typing CTRL-C CTRL-A, then CTRL-C CTRL-B, a few times. If this produces the R prompt, you've recovered. On a user terminal, try typing CTRL-Q to enable output suspended by CTRL-S. If neither of these work, perhaps you removed an initialized diskette without releasing it; check for a red status light on an empty diskette drive. If you did remove an initialized diskette without releasing it, find it and reinsert it; and try CTRL-C CTRL-A again; these steps should produce the R prompt.

If none of the steps above works, the system is hung. You must abort processing, as shown next.

Restarting after a Panic or Hang

The ! prompt may or may not show on the system console. If not, type the break sequence (CMD and BREAK/ESC). To break a hang, you *must* type the break sequence.

Then, next to the !, type I:

```
! I
!
```

Now, warm start DG/RDOS as shown earlier in this chapter (20H or 26H, ↵, and so on).

When you see the CLI prompt, type

```
CLEAR/A/V/D ↵
CLEARED SYS.DR
.
R
```

After clearing the master directory, DIR to each directory that was initialized at the abnormal shutdown, and repeat the CLEAR/A/V/D command.

Note all filenames that CLEAR displays (aside from SYS.DR). Especially, note any of your own application filenames displayed.

Any file whose name appears after you type CLEAR/A/V/D had a nonzero use count, which means it was open when the system went down. Since the file wasn't closed normally, it may not contain current information; in fact, its data may be inconsistent. Keep an eye on any important user file whose name is displayed from CLEAR/A/V/D. In worst case, you may have to delete such a file and reload it from backup media.

The CLEAR command simply zeros the use count of each open file, allowing people and programs to access, rename, or delete the file. CLEAR does not check or restore file consistency and integrity.

Power Failures

If power fails to the computer unit, it will stop functioning until power returns.

Turn off power to the computer unit and all devices. When power returns, cold start as shown earlier in this chapter.

When you see the CLI prompt, type the CLEAR command as described in the preceding section.

Persistent Problems

If DG/RDOS panics or hangs repeatedly without apparent cause, run diagnostics. Use the Customer Diagnostics Diskette as described in Chapter 3 of the hardware *Testing* manual). If the diagnostics find hardware problems, call your support organization (described in Chapter 14, the error chapter).

If the diagnostics find no errors, or if DG/RDOS panics when you try to access a specific subdirectory or file, the problem may be a new bad block. (As time passes, the magnetic qualities of media may change — especially with diskettes — and one or more disk blocks may lose its ability to retain information.)

To check for, and recover from, a new bad block, see "About New Bad Blocks" in Chapter 14, the error chapter.

What Next?

This chapter described the break sequence, startup, normal shutdown, and abnormal shutdown. Now that you know how to start up and shut down your system, you may want to learn more about the CLI. Try the CLI session in the next chapter. Or, learn about formatting diskettes (Chapter 6).

Using the CLI: A Session with DG/RDOS 4

Read this chapter when

- you want to learn how to use the CLI;
- you want hands-on experience with major features of DG/RDOS.

You don't need this whole chapter to use MS-DOS or a language like BASIC. It's here to show you how to use a terminal, and to give you a sense of the CLI's commands and DG/RDOS features.

This chapter leads you through a sample session with the CLI. It's for the beginner and assumes no experience with DG/RDOS — so if you have built a DG/RDOS system or know Data General's RDOS or DOS systems, some material will be familiar. The major sections are

- About the Terminal
- In the CLI
- Typing Mistakes
- Files and Directories
- Printing Files
- Taking a Break
- Running Other Programs
- Backup for Your Files
- What Next?

Just follow the steps described. The session will take between 30 minutes and an hour or so, but don't hurry.

Typing Mistakes

If you type a faulty command and press ↲ without realizing that you made a mistake, the CLI will display an error message and the R prompt. Usually the message is

```
FILE DOES NOT EXIST: faulty-entry.SV
R
```

If you want to change a line before pressing ↲, press the DEL key to erase characters from right to left. To erase the whole line, press the backslash (\) key.

To stop execution of any CLI command or utility program (except an editor), type CTRL-C CTRL-A. The CTRL and C keys, then CTRL and A keys, halt the command or utility and return the R prompt.

You can interrupt display to your terminal at any time by typing CTRL-S. The text won't be lost; you can continue it from the point of CTRL-S by typing CTRL-Q. You can use CTRL-S and CTRL-Q intermittently to read a long file.

If you see an error message that isn't mentioned in the text of this chapter, and the message text doesn't explain the problem, you can look up the message in Chapter 14, the error chapter.

Before you can proceed, DG/RDOS must be running on your DESKTOP GENERATION system. If it's not running, start it as described in Chapter 3.

About the Terminal

Use the system console. Later in the CLI session, you will try another terminal, if your system has one.

Before proceding, check the screen brightness for comfort. The brightness control is beneath the lower right corner of the terminal. (Another essential — the ON/OFF switch — is behind the terminal cabinet, on the right.)

ON LINE and ALPHA LOCK Keys

The ON LINE key, in conjunction with the CMD key, takes a terminal on or off line. (But it has no effect on Model 10 or 10/SP system consoles.) Your terminal should be on line. Confirm this by checking the ON LINE light, in the row of status lights above the main keyboard. ON LINE should be lit.

The ALPHA LOCK status light should be off. If this light is on, all letters you type will be UPPERCASE. If off, letters will be lowercase unless you press SHIFT. Lowercase letters are more readable — and DG/RDOS accepts either case. So, if the ALPHA LOCK light is lit, press the ALPHA LOCK key. This turns the light off.

In the CLI

The R prompt means that the CLI, Command Line Interpreter, is ready for a command. The CLI has many system control commands; it can also execute other programs.

The Keyboard

At the top, in a line, are the status lights: ON LINE, ALPHA LOCK, and so on.

Below the status lights, there are four groups of unmarked keys. These *function keys* are shorthand commands for some products, identified by plastic *templates* shipped with the product.

Below the function keys is the main keypad, which resembles the one on a typewriter. Numbers 1 through 0 are at the top, letters below, and space bar at the bottom. There are some extra characters and some extra keys. Some important extra keys are ON LINE (lower left) and CTRL middle left), NEW LINE (middle right, shown here as ↓), and CMD (below NEW LINE). The CR key, shown in this book as CR, differs from NEW LINE (explained as needed in this book).

The SHIFT keys work the same way they do on a typewriter. ALPHA LOCK tells the terminal to print all letters in uppercase. ALPHA LOCK affects only letter keys — no others. (Zero and uppercase O (0 and O) and number 1 and lowercase l are *different characters.*) For example, FO and F0 are different combinations; so are lx and 1x.

To the right of the alphanumeric keypad, you'll see two other keypads. The numeric keys on the far right are simply that — numeric keys. The pad between the main and numeric keypads is for cursor control; it includes arrow keys (which you may have used for CONFIG in Chapter 2) and a HOME and ERASE PAGE key.

Typing Mistakes

If you type a faulty command and press ↓ without realizing that you made a mistake, the CLI will display an error message and the R prompt. Usually the message is

```
FILE DOES NOT EXIST: faulty-entry.SV
R
```

If you want to change a line before pressing ↓, press the DEL key to erase characters from right to left. To erase the whole line, press the backslash (\) key.

To stop execution of any CLI command or utility program (except an editor), type CTRL-C CTRL-A. The CTRL and C keys, then CTRL and A keys, halt the command or utility and return the R prompt.

You can interrupt display to your terminal at any time by typing CTRL-S. The text won't be lost; you can continue it from the point of CTRL-S by typing CTRL-Q. You can use CTRL-S and CTRL-Q intermittently to read a long file.

If you see an error message that isn't mentioned in the text of this chapter, and the message text doesn't explain the problem, you can look up the message in Chapter 14, the error chapter.

Before you can proceed, DG/RDOS must be running on your DESKTOP GENERATION system. If it's not running, start it as described in Chapter 3.

Files and Directories

To record your dialog with the CLI, start a log file by typing

```
log/h ⤶                  (Specific steps are in lowercase text because it's
                          easier to read.)

R
```

LOG is a CLI command, and /H is a *switch* that changes the way the command works. The LOG command creates a log file, opens it, and starts recording dialog in it. The /H switch adds a header, with directory and date information, to the log file. The log file name is LOG.CM. Later on, to close it, you'll type ENDLOG⤶.

Master Directory Name

Next, you should discover your master directory name. Type

```
mdir ⤶
DxO
R
```

The CLI displays the master directory name, DE0 for a hard disk or DJ0 for a diskette. This session shows the master directory name as Dx0, so you should mentally substitute your master directory name for Dx0 as it appears here.

Creating Some Files

The CLI has several file-creating commands. Try the command CRAND:

```
ctand myfile ⤶
FILE DOES NOT EXIST: CTAND.SV
R
```

The FILE DOES NOT EXIST is an error message. Try again:

```
crand myfile ⤶
R
```

Typing mistakes happen fairly often — ans usually produce an error message. Here, the system didn't understood the command `ctand`, but did understand, and obey, `crand`. You now have an empty disk file named MYFILE.

MYFILE was an *argument* to your command; it told the CLI what to name the file. Some CLI commands require arguments, others don't. To separate a command and its arguments, insert at least one space. You can use more than one space, one or more tabs, or combinations as desired.

You can verify MYFILE's existence with the LIST command (one of the most useful commands):

```
list myfile ⤶
MYFILE. 0 D
R
```

The number and letter mean

- the size of the file in bytes (here, 0);
- the organization of the file (D means random);

For more information, add the /E switch to LIST:

```
list/e myfile ⤶
MYFILE 0 D 12/14/84 13:46 12/14/84 [004344] 0
R
```

The /E switch tells the CLI to display everything about the file, including

- the file size (still 0);
- the file organization (still D);
- the date and time the file was created or last modified (12/14/84);
- the date the file was last opened (12/14/84);
- the starting address of the file (first disk block in the file, 4344);
- the file use count (number of times it was opened, less number of times it was closed. 0 is normal for a file not in use).

Obviously, the time, date, and file address will differ for your own MYFILE.

Now, remove MYFILE and list it again:

```
delete/v    myfile ↲
DELETED MYFILE
R
list/e    myfile ↲
R
```

DELETE does pretty much what you'd expect: removes the file. The /V switch tells the CLI to verify the deletion, which it did by saying DELETED MYFILE. The LIST command displays no name, which means that the file was, in fact, deleted.

Now, create a different version of MYFILE, using the XFER command to insert text in it:

```
xfer/a    $tti    myfile ↲
Greetings from MYFILE. ↲              (Type this text and ↲.)
↓                                     (Press downarrow key on cursor
                                      keypad.)
R
```

You've just transferred (XFER) the text string Greetings from MYFILE into a file MYFILE, creating MYFILE at the same time. The XFER command transfers the contents of one file (the terminal input file, $TTI) to another file (disk file MYFILE). The /A switch specifies ASCII transfer. The downarrow key terminates the text insert.

Type, then list the new MYFILE:

```
type myfile ↲
GREETINGS FROM MYFILE.
R
list/e    myfile ↲


MYFILE    23        12/14/84  13:54  12/14/84  [004344] 0
R
```

The new MYFILE has 23 bytes and is organized sequentially (no letter follows the number). Your MYFILE might have more or fewer bytes, depending on the number of spaces you inserted. The TYPE command displays the contents of a file.

```
xfer/a    $tti    typeit.mc ⤶               (Create another file, text trans-
                                             fer.)
message myfile contains ⤶                   (Insert text.)
type myfile ⤶                               (Insert a command.)
↓                                           (Press downarrow key.)
R
```

One of the great features of the CLI is its custom command groups called CLI macros. If a filename consists of name and .MC, when you type the name⤶, the CLI will try to execute all commands in the file. Try it:

```
typeit ⤶

MYFILE CONTAINS
GREETINGS FROM MYFILE.
R
```

You created TYPEIT.MC to display "MYFILE contains", and then type MYFILE; and it does so. The MESSAGE command simply writes the characters that follow it; it's useful to display text on the screen.

Now that you have created two files, check them:

```
list/e    myfile    typeit.mc ⤶


MYFILE     23       12/14/84  13:54  12/14/84  [004344] 0
TYPEIT.MC  36       12/14/84  14:06  12/14/84  [004347] 0
R
```

Add a little text to TYPEIT.MC:

```
xfer/a    $tti    typeit.mc ⤶                          (Text transfer.)
FILE ALREADY EXISTS: TYPEIT.MC                         (Error message.)
R

xfer/a/b    $tti    typeit.mc ⤶       (Text transfer, add /B switch.)

message How many blocks are left on the disk? ⤶       (Insert text...)
disk
message What's the current directory? ⤶
gdir ⤶
↓         (Downarrow key.)
```

The XFER/A command tells the CLI to *create* a file for text insert. Here, this produced an error message since the file already existed. Adding the /B switch (XFER/A/B) tells the CLI to open the *existing* file and add text to it.

One problem with adding text this way is that you can't edit it. If you make a mistake after you terminate the line with ⤶, you can't back up and fix it. You must either live with the mistake or delete the file and recreate it from the beginning. Having typed the new text, try the improved TYPEIT.MC:

```
typeit ⤶


MYFILE CONTAINS
GREETINGS FROM MYFILE
HOW MANY BLOCKS ARE LEFT ON THIS DISK?
LEFT: n     USED: n     MAX. CONTIGUOUS: n
WHAT'S THE CURRENT DIRECTORY?
DxO
R
```

TYPEIT.MC is larger now, as you can see:

```
list/e    typeit.mc ⤶


TYPEIT.MC 131      12/14/84  14:06  12/14/84  [003347] 0
R
```

TYPEIT.MC now contains a MYFILE command, four MESSAGE commands, a DISK command, and a GDIR command. Each would work separately, but you've combined them.

For some perspective, type

```
list/e/s ⤶
```

(a moment or two passes.)

```
BOOT.SV    7680 SD   6/30/83   9:27    1/27/84  [003347] 0
  BYE.MC        56     7/21/83   8:38   2/17/84  [004136] 0
  CLI.SV    38400 SD   9/07/83  11:58   1/27/84  [003732] 0
  CONFIG.SV 12288 SD  10/13/83  12:31   1/27/84  [004325] 0
  DGRDOS.SV 55296 SD   1/09/84   9:06   1/27/84  [003360] 0
  LOADEM.SV 22016 SD   .          .       .        .      .
  LOG.CM     8644 D    12/14/83  13:40  12/14/84 [003155] 0
     .         .         .        .       .        .
  TYPEIT.MC   122 D    12/14/83  14:06  12/14/84 [004447] 0
```

```
CTRL-C CTRL-A
```
(Type CTRL-C CTRL-A)

```
-INT
R
```

LIST *without* an argument describes all nonpermanent files in the current directory. (The /S switch specifies an alphabetic sort.) To abort the command, you typed CTRL-C CTRL-A.

This is a good time to try some other CTRL sequences: CTRL-S to suspend display, and CTRL-Q to resume it. Repeat the LIST command, type CTRL-S then CTRL-Q; then try CTRL-C CTRL-A again:

```
list/e/s ⤶
.
.
CTRL-S
CTRL-Q
.
.
CTRL-C CTRL-A
```

CTRL-S (suspends display.)
CTRL-Q (restores display.)
CTRL-C CTRL-A (aborts command.)

```
-INT
R
```

CTRL-C CTRL-A also aborts a *program* (like CONFIG or a compiler) that you may be running.

These CTRL sequences are handy and you'll use them often. Practice with the commands and CTRLs above until you're comfortable with them. From this, you'll notice that while CTRL-S is in effect, the system appears to have died. It hasn't — all you need do is type CTRL-Q.

A New Directory

Now you can create a new directory for the two little files. The primary reason for directories is to help organize files. Here's some background information on DG/RDOS directories.

There are two kinds of directories you can create on a disk or diskette. They are *secondary partitions*, whose size is fixed, and *subdirectories*, whose size is flexible. Each kind of directory has its own advantages.

Secondary partitions are useful when you want to restrict disk usage, forcibly. Users in a secondary partition will get FILE SPACE EXHAUSTED messages if they exceed the limit. Subdirectories can grow according to the files created in them, until they reach the limit of the parent directory (usually a disk or diskette). For diskettes, and also for hard disks, we suggest subdirectories because they are more flexible. (Subdirectories are allowed in secondary partitions, but this arrangement tends to be cumbersome.) So we will go with with subdirectories and the CDIR command. Type

```
cdir learning ↲          (Create a subdirectory.)
R
init learning ↲          (Make DG/RDOS aware of it.)
R
```

Creating directories is easy: you just use CDIR or CPART command. But, to *use* any directory other than the master directory, you must tell DG/RDOS that it's a directory. The INIT or DIR command does this.

Now, to move the files into the new directory:

```
move/v   learning   typeit.mc   myfile ↲
TYPEIT.MC
MYFILE
R
```

MOVE, another handy command, copies files into another directory. The /V switch told the CLI to verify files moved, which it did. The first argument to MOVE must be the destination directory name, here LEARNING. The following arguments are the filename(s) you want moved.

The original files are still in the master directory; the copies are in directory LEARNING. To check, make LEARNING the current directory and list the files in it:

```
dir learning ↵
R
list/e/s ↵


  MYFILE     22       12/14/84  13:54  12/14/84  [004344] 0
TYPEIT.MC 112       12/14/84  14:06  12/14/84  [004347] 0
R
```

Check the current directory:

```
gdir ↵
LEARNING
R
```

To conserve valuable disk space, delete the original MYFILE and TYPEIT.MC in the master directory. Do it by typing

```
dir DxO ↵
R
delete/v    typeit    myfile ↵
FILE DOES NOT EXIST: TYPEIT
DELETED MYFILE
R

delete/v    typeit.mc ↵
DELETED TYPEIT.MC
R
```

Whenever you create a directory (secondary partition or subdirectory), DG/RDOS adds the extension .DR to its name. So if you want to see directory LEARNING's filename, you need to check for LEARNING.DR. For example:

```
list/e    learning ⤶
R                 (Nothing: no file named LEARNING.)
list/e    learning.dr ⤶

LEARNING.DR   512  DY  12/14/84  14:23  12/14/84 [004774] 0
R
```

The fact that DG/RDOS assigns the .DR extension makes it easy to find directories using templates, described in the next section.

At this point, the file structure looks like Figure 4-1. The current directory is Dx0.

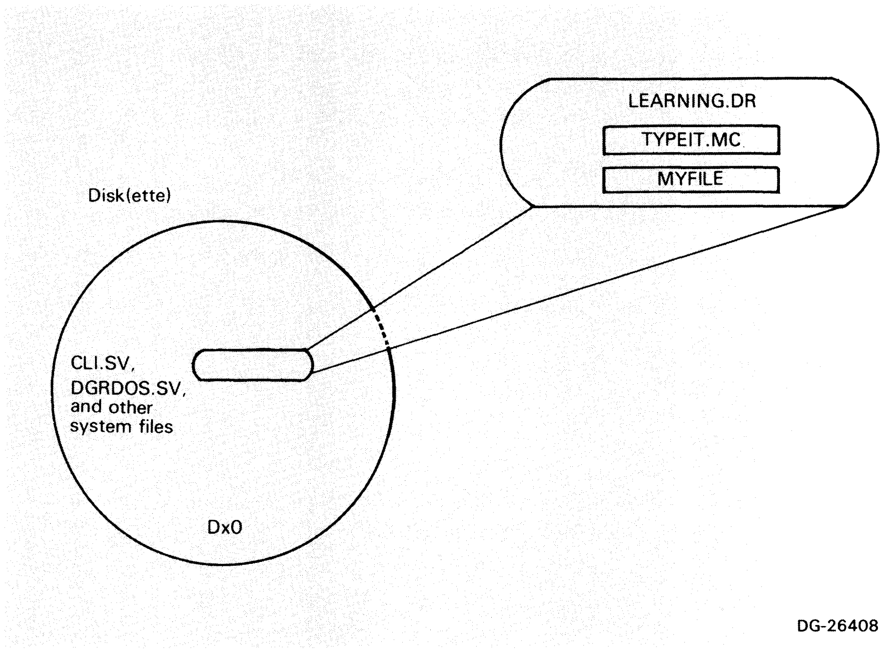


Figure 4-1 Disk(ette) with a subdirectory

Filenames and Templates

Valid DG/RDOS filenames are from one to 10 of the following characters: letters A through Z, numbers 0 through 9, and the dollar sign ($). Any filename can have a one- or two-character extension of any filename characters. The extensions help to classify files. For example,

.SV identifies an executable program (save) file,

.MC identifies a CLI macro file, and

.DR identifies a directory file (subdirectory or secondary partition).

You can use your own extensions as you wish, to help classify your own files. Extensions are especially useful with filename *templates.*

Templates to Find Filenames

People often forget the full names of files. Filename *templates* allow them access to a file without typing the full name. A template specifies a set of filenames. It's the same thing as a wild card character in MS-DOS. The most common template characters are

Character	**What it Means**
*	Match one character except a period.
	Match any series of characters not containing a period.

To see all filenames that are six characters long, with a two-character extension, type

```
list    ****** ** ↵
LOADEM.SV ...
.
R
```

To see all filenames that don't contain a period, type

```
list    - ↵
R                (None.)
```

To see all filenames whose names end in .DR (all directories):

```
list    -.dr ⤶
.
LEARNING.DR
.
R
```

Or to see all filenames whose names end in .MC (CLI macros):

```
list    -.mc ⤶
.
BYE.MC
.
R
```

To see all filenames, including periods, that begin with D, type

```
list    d-.- ⤶
.
DGRDOS.SV
.
R
```

For a hypothetical example, assume that you have a lot of CLI macro files, which conventionally end in .MC. You want to see and sort all their names. You could type

```
list/s/e    -.mc ⤶
```

The resulting display might be

```
BIORHYTHM.MC  278  01/12/85  09:01  02/01/85  [010532] 0
  BOOMER.MC    1665  12/28/84  14:48  01/28/85  [007532] 0
  FS.MC          12  10/08/84  11:33  02/01/85  [006676] 0
  UP.MC         195  10/07/84  08:55  01/31/85  [006571] 0
```

Templates can also help you to organize your files by category in different directories, back them up selectively, or delete them selectively.

There are some restrictions on template characters. For example, they don't work outside the current directory, and not all CLI commands accept them. The commands that allow template characters include BUILD, DELETE, DUMP, LIST, LOAD, and MOVE.

If you want to use a template and are not sure that your command accepts it, just try the template. But be careful with the DELETE command and templates: if you accidentally insert a space after a template character (for example, you type DELETE□-□.XX⤶), you could delete some valued files.

File Access with Pathnames and Link Entries

At this point, you're in the master directory, having just moved your files into directory LEARNING. Try to type one of the files:

```
type myfile ⤶
FILE DOES NOT EXIST: MYFILE
R
```

Try a pathname that includes directory specifier LEARNING:

```
type learning:myfile ⤶
GREETINGS FROM MYFILE.
R
```

A pathname specifies a path to the file. It includes the parent directory name, a colon, and the filename. The directory must be initialized for the pathname to work.

A pathname *can* include the "grandparent" directory name; for example, DJ0:LEARNING:MYFILE. This is extra work to no benefit, though, since — once a directory is initialized — you need not qualify its name. For example, if the pathname DJ0:LEARNING:MYFILE works, so will LEARNING:MYFILE, with fewer keystrokes.

Link Entries

Instead of a pathname, you can create and use a *link entry.* A link entry is a file whose sole function is to point to another file. But, unlike other files, a link uses practically *no disk space.*

Create a link to MYFILE in directory LEARNING:

```
link myfile learning:myfile ⤶
R
```

And try typing MYFILE:

```
type myfile ⤶
GREETINGS FROM MYFILE.
R
```

The TYPE command worked, just as if the text MYFILE were in the current directory. As you can see, the link actually *contains* the pathname, eliminating the need to type the pathname. (The link name need not be the same as the resolution filename, but if the link points to a different directory, using it will easier if the names are the same.) A link *can* exist to a file in the same directory — for example, DG supplied a link named SYS.SV to file DGRDOS.SV.

Link entries are as simple as this — either a path or different name for a file. Try listing both the link and the resolution file:

```
list/e    myfile    learning:myfile ⤶


MYFILE                  LEARNING:MYFILE
LEARNING:MYFILE   23      12/14/84  13:54  12/14/84  [004344] 0
R
```

Remove the link, check MYFILE again, and recreate it:

```
unlink/v    myfile ⤶
UNLINKED MYFILE
R
type myfile ⤶
FILE DOES NOT EXIST: MYFILE
R
link myfile learning:myfile ⤶
R
type myfile ⤶
GREETINGS FROM MYFILE.
R
```

As you create more directories and more files, you'll find greater need for link entries. They allow people in any directory to access a needed file in a central directory (like the master directory) quickly and easily, without knowing anything about the directory structure.

Here are a couple of points about links:

- If you create a link with the wrong pathname, there will be no error message — until you try to use the link. For example, type

```
link xxx yyy:zzz ↵
R                                  (No error.)
type xxx ↵
NO SUCH DIRECTORY: XXX             (Error occurs when you try
                                   to use the link.)
```

  To handle this, check the link immediately after you create it; if it doesn't work, remove it with UNLINK and recreate it.

- Never use DELETE to remove a link. This will delete the resolution file (the one linked to); the link will remain. Use UNLINK to remove links. For example,

```
unlink/v    xxx ↵
UNLINKED XXX
R
```

- If you have a lot of links, think twice before moving the resolution files to other directories. The existing links will then specify the wrong pathname and they won't work. To fix it, you'll have to unlink each link and recreate it to the new directory.

Accessing a Separate Disk(ette) as a Directory

With two diskette drives, or with a hard disk, you may want to access a separate diskette as a directory. If you have two hard disks, you certainly want to access the second hard disk.

The rules are as follows:

- The disk(ette) must be hardware formatted. This is always true for hard disks and for diskettes you get from DG. Non-DG diskettes must be hardware formatted (Chapter 6).
- The disk(ette) must be software formatted via DKINIT (Chapter 6). Diskettes need not be software formatted to be used for backup with the IMOVE program or be copied to with the FCOPY program.
- The disk(ette) must have had INIT/F typed to it from the CLI. This step actually makes it into a new DG/RDOS directory.

After these things have been done, you can insert the disk(ette) if needed and treat it just like any directory. It must be initialized (INIT without! the /F switch or DIR command) before it can be accessed. While it's initialized, you can treat it just the way you treated LEARNING: you can move files there, create links to the files, use pathnames, and so on. And, you can create and use subdirectories or secondary partitions as desired

CAUTION *After a diskette has been initialized, don't remove it from its drive until you release it (RELEASE command), or shut down DG/RDOS.*

The names of the disk(ette) directories are the same as the name of the drives: DJ1, DJO, or DE1.

For example, assume you have a fully initialized diskette (you've run all steps above on it).

Insert the diskette in a vacant drive: DJO (with a hard disk) or DJ1 (two diskette drives without a hard disk). We'll call this DJn for the rest of this section (using UPPERCASE to distinguish the "n" from the rest of the name).

```
dir DJn ↵                              (n is 0 or 1 for hard disk, 1
R                                      otherwise.)

disk ↵                                 (Check disk space on the diskette.)
LEFT: 681 USED: 23 MAX..: 681
R

type myfile ↵                          (Try to type MYFILE.)
FILE DOES NOT EXIST: MYFILE            (Error, can't find it.)
R
link myfile learning:myfile ↵          (Create link to the original file in
                                       directory LEARNING.)

R
type myfile ↵                          (Try the TYPE again...)
GREETINGS FROM MYFILE.                 (It works, via the link.)
R
disk ↵                                 (Check disk space again.)
LEFT: 681 USED: 23 MAX..: 681          (There's plenty left.)
R
gdir ↵                                 (Check current directory.)
DJn                                    (It's the separate diskette.)
R
unlink myfile ↵                        (Remove the link to clean up.)
R
```

```
release DJn ⤶                    (Release the diskette.)
R
gdir ⤶                           (Check current directory again.)
Dx0                              (It becomes the master when you
                                 release a current, nonmaster di-
                                 rectory.)
R
```

Remove the diskette from the drive.

This example shows how to maximize available disk space on any system — using an empty diskette that has links to needed files on the master directory. It's particularly apt for two diskette drive systems, since it retains almost all the space on the second diskette for file storage and execution of other programs.

Printing Files

If your system has a printer, you'll want to print files. (If it doesn't have a printer, you can have it save printer material as shown later in this chapter; skip this section.) To print one or more files, use the PRINT command. For example,

```
print typeit.mc ⤶
FILE DOES NOT EXIST: TYPEIT.MC    (Wrong directory.)
R
dir learning ⤶
R
print typeit.mc ⤶
. (pause)
```

The R prompt may return immediately or it may wait for the whole file to print — depending on where your printer is attached.

The R prompt returns immediately if the printer is connected to the CPU printer port (as secondary console/printer). For a printer on this port, DG/RDOS *spools:* it sends material to a special file, from which it is printed. You can continue typing commands while the file proceeds. If you ever want to stop the printer and empty the spool file, type the command SPKILL $LPT.

The R prompt waits for the whole file to print if the printer is attached to a multiplexor (USAM) line. DG/RDOS doesn't spool to a multiplexor line, so you must wait for printing to finish before typing CLI commands. You can stop printing at any time by typing CTRL-C CTRL-A.

Handling the Printer

1. Wait for the printer to finish printing your file (here, the file is small, so you won't have to wait long).
2. If the paper is in separate 8.5 x 11 inch sheets, simply pick up the printed file; you're done.
3. If the paper is in a pile of connected sheets, press the printer ON LINE switch. The ON LINE light should go out. (The letter- quality printer has a LOCAL switch that you use in the same way as an ON LINE switch.)
4. Press the FORM FEED switch. This tells the printer to feed a form. Tear off the paper after the blank form.
5. *Important:* press the ON LINE switch again. The ON LINE light should glow. If you forget this step, the printer can't print; all print requests that people make will wait until someone puts it back on line.

Now you can examine the printed text of TYPEIT.MC. Not exactly *War and Peace* — but printing it did show you how to work the printer. To insert paper, correct printer faults, and so on, see the printer description in the *Operations* manual that accompanies this one.

In many CLI commands — including LIST — you can send text to the printer instead of to your terminal screen. To do this with LIST, use the /L switch. This switch is very handy for things like big file sorts. For example:

```
dir Dx0 ⤶
R
list/s/e/l ⤶
(pause for filename sort)
.
.
R
```

Printing of the sorted list begins.

Taking a Break

At this point, you've learned about the terminal; started a log; created, deleted, and listed files; inserted text in files; created a directory; moved files into the directory; tried pathnames and link entries; tried using another disk(ette) as a directory; and printed a file.

You've used the commands LOG, CRAND, LIST, DELETE, XFER, TYPE, MESSAGE, LIST, GDIR, CTRL-C CTRL-A, CTRL-S, CTRL-Q (these last are really control characters, not commands), CDIR, INIT, MOVE, DIR, LINK, UNLINK, RELEASE, and PRINT. You've also used switches and the template characters * and -.

The file structure looks like Figure 4-2.

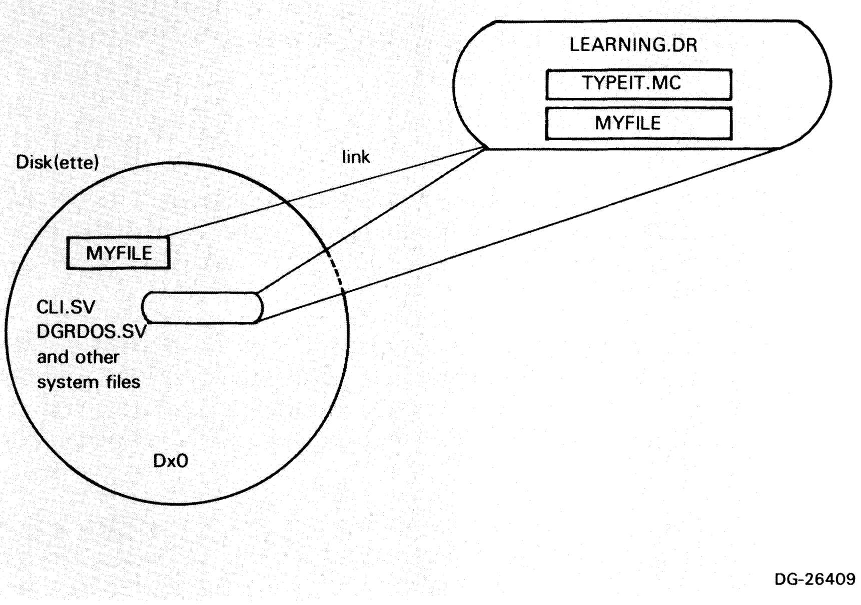


Figure 4-2 Disk(ette) with a subdirectory and link entry

You may want to stop for a while. If so, you can return to this point later.

To proceed, read on.

Running Other Programs

The CLI is a versatile program. But it's most useful as a companion to other programs — as a file maintenance aid and interface to programs like word processors, text editors, custom applications, and other operating systems like MS-DOS.

You can execute another program from the CLI by typing the program name, perhaps with one or more arguments. The new program runs *under* the CLI. When the new program terminates, or you abort it with CTRL-C CTRL-A, the CLI returns to your terminal.

For example, try program CONFIG (which is no stranger, if you configured this DG/RDOS system). CONFIG is a program supplied with all versions of DG/RDOS, and the chances are good that it's on your master directory. Type

```
 dir Dx0 ⤶
R
 config ⤶

DG/RDOS System Configuration Program ....
.
Enter name of system to configure: DGRDOS.SV
```

You've executed the CONFIG program and it's running on your terminal. Press ⤶ and it displays a screen menu:

```
 ⤶
DG/RDOS System Configuration Program ....
 .
 .
```

To exit without changing anything, type 5 ⤶; then type N⤶.

```
R
```

You terminated CONFIG and the CLI returned. This is generally the procedure you'll follow: run the new program, work with it as desired, then terminate it.

Foreground and Background Programs

DG/RDOS allows you to run a *second program* concurrently with the CLI, or with any program you run under the CLI. The second program is called the foreground program; the CLI or any other program that runs on the system console is called the background program.

The size of the foreground program you can run — in fact the usefulness of the foreground concept — depends on

- the amount of memory in your computer;
- other terminals you may have;
- whether you have a hard disk.

If you have 256 Kbytes or more of memory and a second terminal connected to the CPU printer port (Model 10 and 10/SP) or secondary console card (Model 20 or 30), you can run a CLI in the foreground. This CLI can do practically anything the background CLI can: maintain files and directories, execute programs, and so on. It's like having two independent computer systems. Usually, you need a hard disk for storage.

If you have 256 Kbytes or more of memory and one or more terminals connected to a USAM multiplexor, you can run programs like ICOBOL (Interactive COBOL) in the foreground. For storage, you need one or more hard disks.

If you have 256 Kbytes of memory with only the system console (hard disk or not), you may still find productive uses for a foreground program.

If you have 128 Kbytes of memory, a program must be very small to run in the foreground. With 128 Kbytes, skip to "Backup for Your Files."

The first step is to type

```
gmem ↲
BG: n FG: 0
R
```

GMEM describes the amount of memory in 2,048 (2 Kbyte) memory pages for background (BG) and foreground (FG) programs. Space needed for DG/RDOS, the emulator, and screen bit mapping has already been deducted. At system startup, the background has all memory pages and the foreground has 0 pages. The background keeps all pages until someone uses the SMEM command to give the foreground some pages.

The command sequence given next starts and stops a foreground CLI. It shows the GMEM, SMEM and EXFG commands and the FGND command for foreground status checking.

For a CLI to run in the foreground, the foreground needs at least 21 Kwords (42 Kbytes) of memory.

If you want to run an application like ICOBOL in the foreground, check the product Release Notice (or Table 2-5 in Chapter 2) for the number of Kbytes/Kwords required. The Release Notice gives any special instuctions needed.

```
dir Dx0 ↵                           (Get into desired directory. Here, to
                                    run the CLI, use the master directory.)
R
gmem ↵                              (Check memory.)
BG: 96 FG:0                         (Plenty of memory.)
R
smem 32 ↵                           (Give the background 32 Kwords;
                                    foreground gets the rest.)
R
gmem ↵                              (Check again.)
BG: 32 FG: 64                       (Still plenty.)
R
fgnd ↵                              (See if a foreground program is run-
                                    ning...
NO FOREGROUND PROGRAM RUNNING        (No.)
R
exfg/e cli ↵                        (Run the CLI in the foreground.)
R
fgnd ↵                              (Again, see if a foreground program is
                                    running...
FOREGROUND PROGRAM RUNNING           (Yes.)
R
.                                (time passes — a full day of applica-
                                 tions. You decide to bring
.                                down the foreground and warn the
                                 person using it, if any.)
.
CTRL-C CTRL-F                    (CTRL-C CTRL-F brings down the
                                 foreground.)
FG TERM
)
R
```

While a foreground program is running, you can run a background program other than the CLI on the system console (for example, MS-DOS), if desired. If you do this, you'll need to terminate the background program to bring down the foreground (because CTRL-A CTRL-C are meaningful only to the background CLI, not to the program you run from the background CLI).

Running a foreground program can more than double the productivity of your computer system. And you've developed a sense of how to do it from this section.

Backup for Your Files

No matter how small a computer system, it should have some procedure for file backup (copy for safekeeping). Then, if files are inadvertently deleted or otherwise lost, they can be restored from the backup medium. The DG/RDOS files you received from DG do not absolutely *need* to be copied, since they can be reloaded from the DG diskettes. But things like user directories, files, and documents might be irreplaceable, and should be backed up.

This section shows you how to back up the files you created during this session. There's more detail on all backup procedures in Chapter 7. But, since backup is a routine procedure, you should try it a as part of this session.

Backing Up Directory LEARNING

Get a diskette and remove it from the paper envelope. The diskette must be hardware formatted but need not be software formatted with DKINIT.

If there is no label on the diskette, apply one (described in Chapter 6, section "Handling and Storing Diskettes").

Now, depending on whether you have a hard disk or not, read the appropriate section next. (If you have just one diskette drive and no hard disk, go to Chapter 6, section "Copying a Diskette."

Backup from a Hard Disk

With DG/RDOS, you received several programs to back up disk(ette)-based information. With a hard disk, we recommend the program named IMOVE, because it is easier to use and more efficient than the others. (Specific tradeoffs are given in Chapter 7.)

Program IMOVE.SV should be in the master directory, copied there when DG/RDOS was installed.

If the diskette is new and you didn't get it from DG, it needs hardware formatting. Do this as described in Chapter 6, then return here. DG/RDOS can't access a diskette that hasn't been hardware formatted.

Insert the diskette in a vacant slot (DJ1 or DJ0). For this example, let's say DJ0. This diskette will be overwritten, so don't use one that has data you want.

Type the following commands:

```
dir de0 ⤶
R
imove/d/v/l    dj0    learning.dr ⤶
```
(The filename is LEARNING.DR)

Please insert disk 1, then press NEW LINE to continue. ⤶ (Press ⤶)

(pause) (If you see an error message, go to the next section.)

R

The /L switch tells the IMOVE program to list files on the printer. Get the file from the printer as described earlier. And skip the next section.

If There is No Printer The IMOVE /L switch tells IMOVE to verify on the printer, which is file $LPT. If your DG/RDOS system doesn't support a printer, it says

```
A non-existent file: $LPT
R
```

To save the verify text, create an $LPT file, repeat the IMOVE command, then rename the $LPT file as follows:

```
crand $LPT ⤶                          (Create random file
R                                      $LPT)
```

(Repeat the IMOVE command above.)

```
type $LPT ⤶                           (To check listing.)
. (file listing)
R
rename $lpt feb2784.bu ⤶              (Rename the $LPT file
                                       to a meaningful date:
                                       here, year, month,
                                       day, and .BU to show
                                       the backup date and
                                       file content.)
R
```

Having a listing file available will make restoring files much easier, if you ever need to do it.

Reading the Backup Diskette If don't know what's on a backup diskette, you can tell the system to display the filenames without copying them. To read the files, add the /F switch; to read them without trying to copy the, also add the /N switch. For example, type

```
imove/d/v/f/n    dj0 ⤶
```

Please insert disk 1, then press NEW LINE to continue. ⤶ (Press ⤶)

```
LEARNING.DR
     LEARNING:MYFILE
     LEARNING:TYPEIT.MC
```

This time, since you omitted the /L switch, the listing appears on the system console. The /N switch tells IMOVE not do copy anything — just to display names on the diskette. It's very useful to verify material copied.

You're done with the hard-disk backup. Other backups will involve more diskettes, but the concepts and steps will essentially be the same. Details are given in Chapter 7.

Later on, if you ever need to restore this directory, you'd DIR to DE0 and run IMOVE as above, without the /N switch.

Remove the diskette from drive DJ0. Then write the date, type of backup and directory (full backup, subdirectory LEARNING), and diskette number (1) on the diskette label, and replace the diskette in its envelope.

The diskette number is not needed for a single-diskette backup, but it is *critically important* for multiple-diskette backups. This is true because — to restore the files — the diskettes must be restored in the same order that they were backed up. You could easily lose track of the order in a backup that included many diskettes.

Skip to the section "Cleaning Up."

Backup from a Diskette — Two Diskette Systems

With two diskettes and no hard disk, the easiest course is to copy the whole diskette you want to back up. The easiest *program* to use for this is FCOPY. The only requirement is that the destination diskette be hardware formatted and have no bad blocks. (FCOPY will stop with an error message if it tries to write to a bad block.)

Follow these steps:

- Insert the hardware-formatted diskette in drive DJ1.
- DIR to the master directory (DJ0) and start FCOPY with the /D (duplicate) switch and pertinent diskette names:

```
fcopy/d/v    dj0    dj1 ⤶
```

- Answer FCOPY's questions by pressing ⤶ — to select the default answers — until FCOPY starts duplicating.

- Shortly, the duplicate will be done. (For any error, see Chapter 14, the error chapter). Type 3⤶ twice to exit to the CLI, as shown on the menus.
- Remove the diskette from DJ1. Write a description (like "System Diskette, practice copy with LEARNING directory" and the date on the label. You can use this diskette (just as you did the original) to start and run DG/RDOS. Return the diskette to an envelope.

You've finished the backup from diskette.There are some variations (like using the MOVE command, which requires the backup diskette to be formatted as a directory). All variations are described in Chapter 7. Chapter 7.

Cleaning Up

The session's almost done. For the next step, you should delete the new directory to maximize free disk space. (This is optional, of course; you can keep the directory. But it *is* backed up, if you ever want it back.)

To delete it, type

```
dir de0 ⤶                                       (or dj0 ⤶)
R
delete/v    learning.dr ⤶
DIRECTORY IN USE: LEARNING.DR                    (You can't delete an
                                                 initialized directory.)
R
release learning ⤶                               (Release it...)
R
delete/v    learning.dr ⤶                        (... and delete it.)
DELETED LEARNING.DR
R
```

Checking the Log File

The log file (started long ago with the LOG command) has grown pretty large by now. It remains in the master directory, where you started it. Check it with LIST/E:

```
dir de0 ↵ (or dj0↵)
R
list/e    log.cm ↵
LOG.CM 16133 D .... .... ..... ...... 1
R
type log.cm
FILE IN USE: LOG.CM            (Error message.)
R
```

Before you can read the log (TYPE or PRINT it) you must close it with ENDLOG:

```
endlog ↵
R
type log.cm ↵
.
. (record of dialog)
.
R
```

There it is — a record of all your dialog with the CLI. (Dialog with other programs, like IMOVE, isn't recorded.) You should at least examine the log file again for review of what you've done. Then you can delete it (DELETE/V LOG.CM) or not, as you prefer.

CAUTION *If you have a two-diskette system and turn on logging, be sure to turn it off (ENDLOG), or shut down, before removing the diskette logging was started. The CLI expects an open log file to be on this diskette; if the diskette is changed, you'll probably see LOG FILE ERROR on your next command. If this happens, see Chapter 14, the error chapter.*

What Next?

You have finished the entire CLI chapter. You've learned about the terminal, started a log file, created, deleted, moved, and linked files and directories; used a diskette as a directory; printed a file; learned and perhaps tried foreground programming; backed up and deleted the files you created; and reviewed the log file.

You've used the commands LOG, CRAND, LIST, DELETE, XFER, TYPE, MESSAGE, LIST, GDIR, CTRL-C CTRL-A, CTRL-S, CTRL-Q (these last are really control characters, not commands), CDIR, INIT, MOVE, DIR, LINK, UNLINK, RELEASE, PRINT, GMEM, SMEM, perhaps EXFG, IMOVE or MOVE, RENAME, and ENDLOG. You've also used switches and the template characters * and -.

Congratulations. This session included most of the concepts and commands you'll ever need with the CLI.

For a description of CLI commands, macros, and programs — alphabetically — continue to Chapter 5.

For details on formatting diskettes, skip to Chapter 6.

For details on file backup, skip to Chapter 7; or to build programs in DG computer languages, skip to Chapter 10.

Understanding DG/RDOS File Structure and Commands 5

Read this chapter when

- you want information on DG/RDOS file structure, CLI rules, or filename and device name rules;
- you want details on an often-used CLI command, macro, or program supplied with DG/RDOS.

This chapter explains DG/RDOS file structure. Then, it describes some often-used CLI commands and programs (like the ones in the Chapter 4 session). You need *not* read this chapter to start a program like FORMA-TEXT or use a BASIC programming language.

Not all CLI commands are described here; they *are* all explained in the RDOS/DOS Command Line Interpreter manual.

The major sections are

- DG/RDOS File Structure
- Commands, Macros, and Programs
- What Next?

DG/RDOS File Structure

This section explains DG/RDOS file structure — briefly — in the context of DESKTOP GENERATION systems.

If you installed DG/RDOS software (Chapter 2) or tried the CLI session (Chapter 4), you already have some sense of the directory structure.

During startup, the DG/RDOS system and CLI are read into memory from disk or diskette. This disk(ette) is the *master directory*, and it is always accessible to CLI commands that require a disk(ette). However, DG/RDOS doesn't *need* the master directory to run. You can release the master directory and open (initialize) another directory in its place (most useful with diskette-based systems).

The master directory, and any disk(ette) you want to use as a directory, must have undergone the following operations:

- a DKINIT software formatter FULL command must have run on it;
- the CLI command INIT/F must have been typed to it.

A full format and INIT/F are needed only once. They make the disk(ette) a bona fide directory, which you can later initialize into the DG/RDOS system with the INIT command (without /F!) or DIR command.

On any disk(ette) formatted as a directory, you can (but need not) create subordinate directories. These are useful in terms of file organization. Any directory you create must be initialized (just as its parent disk(ette) must be) before anyone can access its files. For example:

```
type learning:myfile ⤶
NO SUCH DIRECTORY: LEARNING:MYFILE
R
init learning ⤶
R
type learning:myfile⤶
.
.(text of file MYFILE)
.
R
```

Here's another example:

```
type dj1:filexx )
NO SUCH DIRECTORY: DJ1:FILEXX
R
init dj1 )
R
type dj1:filexx )
.
.(text of file FILEXX)
.
R
```

Initialization (with the INIT or DIR command) tells the system where a directory is. Thereafter, you can access it by directory name.

Initialization is an extra step, so you will probably want to keep the number of directories to a minimum. This depends on the medium. For a hard disk, with up to 100 times the storage of a diskette, a fairly elaborate directory structure often makes sense.

You can release an initialized disk(ette) from the system with the RELEASE command. Releasing a disk(ette) also releases all initialized directories on it.

The two types of directory you can create on a disk(ette) are

- Secondary partition. A secondary partition is limited to the size you specify when you create it. It's useful when you want to restrict the size of the directory. For example, in a multiuser system with a hard disk, each user might have a secondary partition as a "home" directory — the size could be something like 2000 blocks (on a 15-Mbyte disk).
- Subdirectory. A subdirectory grows according to the files created in it; it's limited only by the size of its parent disk(ette). Subdirectories are the more flexible of the two directory types.

A disk-based arrangement with secondary partitions and subdirectories might look like Figure 5-1. A two-drive diskette arrangement with subdirectories might look like Figure 5-2.

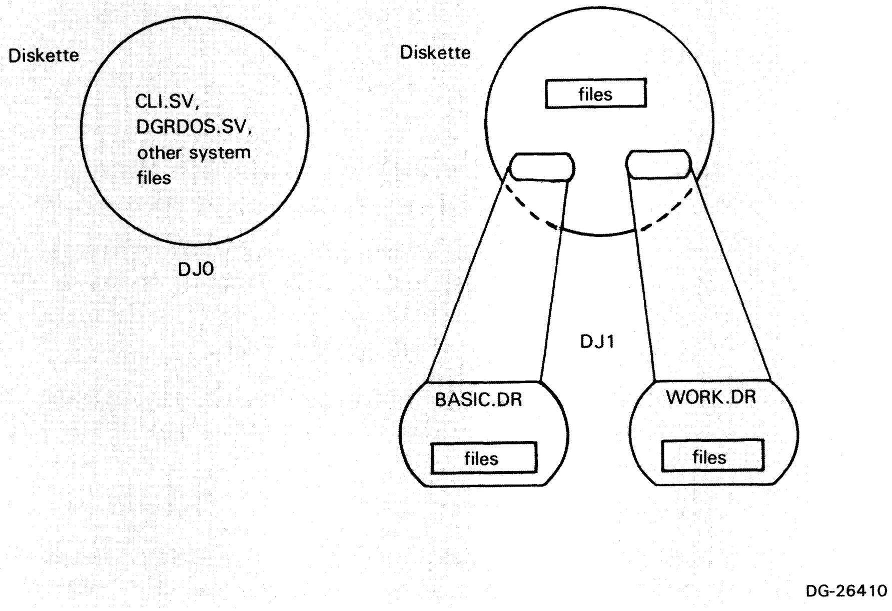


Figure 5-1 Disk with secondary partitions and subdirectory

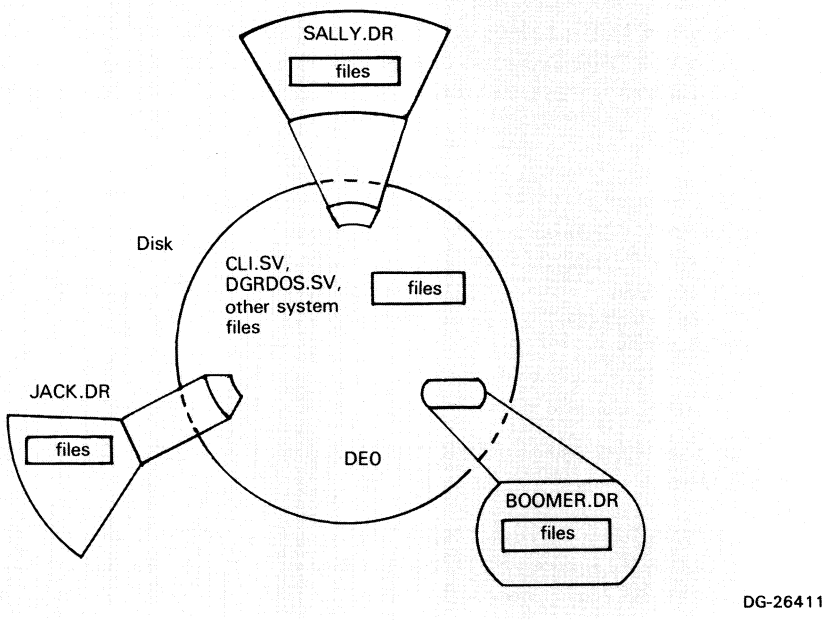


Figure 5-2 Diskettes with subdirectories

In any directory structure, people can access files outside the current directory using either a pathname (directory:filename) or a link entry (created with the LINK command). Links are easier, since a user need not know where a file is to access it. Links and pathnames are shown in action in Chapter 4.

Directory files contain other files, but in other ways they're just like any nondirectory file. You can move or dump any directory (with all it files); and you can delete any subdirectory or secondary partition that isn't initialized.

The CLI command that creates a secondary partition is CPART; the command that creates a subdirectory is CDIR (shown in action in Chapter 2). The system adds the extension .DR to the name you assign. This makes it easy to find directories with a filename template. For example,

```
list/s    -.dr ↵              (Subdirectory)
BASIC.DR 512 DY               (Subdirectory)
BIORHYTHM.DR 512 DY           (Subdirectory)
COMPUCALC.DR 512 DY           (Subdirectory)
MEMOS.DR 512 DY               (Secondary partition, 1000 blocks)
SALLY.DR 512000 CTY           (Subdirectory)
TEMP.DR 512 DY
R
```

You can then use the DIR command to make any -.DR file the current directory — and work with the files in it.

The number of directories you can create on any disk is limited only by available space. But the number of directories outside the master that can be *initialized* at once is limited by CONFIG; the default is 10.

DG/RDOS does not allow two directories of the same name to be initialized at once. This means, for example, that if users Sally and Jack, each in a secondary partition, create a subdirectory called SUBDIR, the system will recognize only the one first initialized. If Sally initializes her SUBDIR first, and Jack tries to initialize his SUBDIR, there will be no error message; but all references to SUBDIR, regardless of pathname, will go to Sally's SUBDIR. For this reason, be sure to arrange your disk structure so that every subdirectory and secondary partition has a unique name.

Filenames and Templates

Filenames can be up to 10 letters, numbers, or special characters: ? or $. Each filename can have (but need not have) a one- or two character extension. The extension serves to identify the kind of file. Some useful extensions, assigned by the system, are

name.DR	identifies a subdirectory or secondary partition;
name.MC	identifies a CLI macro file;
name.SV	identifies an executable program (save) file.

You can use a template character to match all or part of a filename and extension. A template specifies a set of filenames. The template characters are

Character	**What it Means**
*	Match any character except a period. For example, template A** be matched by AAA, A$Z, and A11 but not AAA.SV or A.AA
	Match any series of characters not containing a period, in filename or extension. For example, A- would be matched by ABCD, AB$BILLING, or any A- character without an extension. Template A-.- would be matched by any A filename *with* or without an extension.

You can also use templates with the /N switch to omit filenames. For example,

```
list/s   -.dr    -$-.dr/n ↵
```

lists all directories in the current directory, except those with $ in their names.

Not all commands accept templates; but it can't hurt to try them. Templates are shown in action in Chapter 4.

Protecting Files (Changing File Attributes)

You may want to protect certain files (like directories) from deletion, or prevent nondirectory files from being modified. The CHATR command (change attributes) allows this. If you make a file permanent (P attribute), it can't be deleted. For example,

```
chatr mydir.dr p ↵
R
delete/v mydir.dr ↵
PERMANENT FILE: MYDIR.DR
R
```

The disadvantage of giving a file the P attribute is that you must use the /A switch to list, move, dump, or load the file; otherwise, the file will be ignored.

Device Names

You can use any valid name for a disk(ette) file, but certain devices have reserved names. (You can't name other files these names, or delete or modify these reserved names.) The DG/RDOS reserved device names that pertain to DESKTOP GENERATION systems follow.

DJ0	Primary (rightmost) diskette drive.
DJ1	Secondary diskette drive.
DE0	Primary hard disk.
DE1	Secondary hard disk.
$LPT	Printer, if specified to CONFIG. If a printer was not configured, you can create and use your own $LPT file for printer material.
$TTI	System console keyboard.
$TTO	System console screen.
$TTI1	Foreground terminal keyboard.
$TTO1	Foreground terminal screen.
$PLT	Plotter.
MT0	Magnetic tape drive. The first tape file is MT0:0, the second MT0:1, third MT0:2, and so on. MT0 is a reserved name only if a tape drive was specified to CONFIG.
QTY	Multiplexor (USAM). The line 0 filename is QTY:0, the line 1 name is QTY:1, line 2 is QTY:2, and so on. These names are often used by the system but rarely used by people.

In CLI commands, you can use device names just as you use filenames — according to the command syntax shown in this chapter and the CLI manual.

Commands, Macros, and Programs

This section describes common CLI commands, and macros and programs supplied with DG/RDOS for desktop systems.

About CLI Commands and Macros

The CLI has over 50 built-in commands — and it is shipped with at least one macro (disk file that contains one or more CLI commands). You can do a lot of work using only a few CLI commands.

CLI Errors and Error Messages

When the CLI cannot execute a command, it displays an explanatory message called an *error message*. Then it stops and waits for another command. Generally, CLI errors are okay; everyone makes them, and usually all you need do next is retype the command correctly. A typical error message is

```
FILE DOES NOT EXIST: filename.SV
```

If the explanatory text allows you to understand what went wrong, you can correct it and continue with your desired operation. If you *can't* understand what went wrong, look up the error text in Chapter 14, the error chapter. Chapter 14 tells you how to cope with the most common, important errors — not only with CLI errors, but errors that occur in other programs.

Multiple Commands and Long Command Lines

To stack more than one CLI command or macro on a line, separate the items with a semicolon; for example:

```
list    myfile;    type    myfile ⤶
```

To continue a command onto another line, type a caret (press the SHIFT and number 6 keys), and press ⤶. The CLI will then let you continue typing on the next line. When you want the CLI to execute the command, press ⤶ without typing a caret first. For example:

```
ty^⤶
pe ^⤶
myfile ⤶
```

Extra spaces or tabs within CLI command lines have no effect. But neither a space nor a tab is allowed before a switch (for example, DELETE/V MYFILE).

Command, Macro, and Program Summary

This section lists the common commands, macros (macro filenames end in .MC), and programs (program filenames end in .SV). Each description explains what the item does, the format for using it, reasons why you might use it, and then shows an example. Table 5-1 summarizes the commands, macros, and programs.

Table 5-1 DG/RDOS commands, macros, and programs (continues)

Command, Macro, or Program Name	What It Does
BOOT	Starts a new DG/RDOS system or program like DKINIT
BUILD	Builds a file consisting of filenames
BYE.MC	Shuts down the DG/RDOS system
CDIR	Creates a subdirectory (variable-size directory)
CHATR	Changes file attributes, for permanence or read protection
CONFIG.SV	Checks or changes hardware and software parameters in a a DG/RDOS system
CPART	Creates a secondary partition (fixed-size directory)
DELETE	Deletes one or more files
DIR	Changes the current directory
DISK	Displays the amount of disk space left and used
ENDLOG	Closes the log file (described under LOG)
EXFG	Executes a program in foreground memory
FCOPY.SV	Copies a diskette or file to another diskette
GDIR	Gets the current directory name
GMEM	Gets the amount of memory in background and foreground
GTOD	Gets system time and date
IMOVE.SV	Copies files to or from diskette or tape
INIT	Opens a directory or tape drive, or creates a new file directory
LINK	Creates a link entry to a file in any directory
LIST	Describes file names and statistics

Table 5-1 DG/RDOS commands, macros, and programs (concluded)

Command, Macro, or Program Name	What It Does
LOADEM.SV	Loads a terminal emulator into memory or disk(ette) if needed
LOG	Logs terminal dialog in a disk file
MESSAGE	Displays text on the screen.
MOVE	Copies one or more files to any directory
PRINT	Starts printing a file
RELEASE	Releases (closes) a directory or tape drive
RENAME	Renames a file
SDAY	Sets the system date
SEDIT.SV	Edits disk file locations
SMEM	Sets memory allocation for background and fore-ground programs
SPKILL	Stops printing and deletes the spool file
STOD	Sets the system time
TYPE	Types one or more files on the terminal screen
UNLINK	Removes a link entry
XFER	Copies the contents of a file into another file

BOOT
Starts a new DG/RDOS system or program like DKINIT

```
BOOT  { disk:program }
      { program      }
```

The BOOT command tries to shut down, then start up the system or program you specify. If you omit the program name, it asks Filename? The disk can be any initialized disk or diskette; DJ1, DJ0, and so on.

If the current directory contains the program, you can omit the disk specifier.

On a hard disk, the master directory (DE0), has the systems you most often boot often run (DG/RDOS systems and DKINIT).

If you specify a disk, it must be inserted (if a diskette) and initialized (INIT or DIR). The disk(ette) must also have a bootstrap root (installed by MBOOT.SV), and a copy of MBOOT.SV on it. If not, shutdown will occur but the Filename? question won't be asked.

The same constraints as for normal shutdown apply. For example, BOOT won't shut down the system if a foreground program is running.

Why Use It?

BOOT is very handy when you want to format diskettes (bring up DKINIT) or start a different DG/RDOS system. It takes the place of the BYE⤶ and Filename? name⤶ steps.

Example

```
dir de0 ⤶
R
boot dkinit ⤶
```

```
MASTER DEVICE RELEASED            (System closes the master direc-
                                  tory.)

DISK INITIALIZER REV x.xx         (DKINIT announces itself.)
DISK DRIVE MODEL NUMBER?
```

BUILD

Builds a file consisting of filenames

BUILD newfilename filename *[filename] [...]*

BUILD creates file **newfilename** and copies all qualifying **filenames** into it. If the **newfilename** already exists, BUILD deletes and recreates it. You can use templates and/or switches to select filenames.

In **newfilename**, the filenames are arranged in lines of 80 characters or less, and each line is terminated with a line continuation character (^). This allows a large number of filenames to be read from the file later.

Why Use It?

Most CLI commands and DG/RDOS programs don't have date switches, nor do they accept filename template characters. The BUILD command can overcome this — allowing you to build a file from filenames selected by date and/or template.

Later, by enclosing the BUILD filename in commercial at signs (@), you can have a command or program act on all the enclosed files.

A simple example is

```
print    a- ⤶
ILLEGAL FILE NAME: A-
R
build    afiles    a-    -.sv/n ⤶
R
print @afiles@ ⤶
.
. (printing begins)
R
```

BUILD (continued)

Switches

/A	Includes names of permanent files (P attribute, CHATR command) and nonpermanent files. If you omit this, BUILD will not include the names of permanent files.
mm-dd-yy/A	Includes names of files created on or after this day (mm and dd can be one or two digits). For example, 10-25-84/A specifies files created on or after October 25, 1984.
mm-dd-yy/B	Includes names of files created *before* this day. Syntax is the same as the date/A switch; you can use either or both date switches.
name/N	Omits names of files that match name or name template.

Example

```
list/s/e    -stat-.-    10-15-84/a    10-22-84/b    -.sv/n ↵


ACCTSSTAT   3278 D 10/15/84  09:01  10/30/84  [010532] 0
INVSTAT.VR  2365 D 10/20/84  14:48  11/28/84  [007532] 0
PAYSTAT     6788 D 10/18/84  11:33  01/02/85  [006676] 0
PURCHSTAT   4997 D 10/21/84  08:55  12/03/84  [006571] 0
R


build    stat$files    -.-    10-15-84/a    10-22-84/b    -.sv/n ↵
R
print @stat$files@ ↵
.
. (printing begins)
```

Here, a person lists all files (in the current directory) whose names have the text string STAT, that were created between October 15 and 22, 1984, and that do not end in .SV. The person then builds the names into file STAT$FILES and prints this file indirectly — which prints the *contents* of all filenames in STAT$FILES.

For another example, see IMOVE.

BYE.MC
Shuts down the DG/RDOS system

```
BYE
```

BYE.MC is a DG-supplied macro that shuts down DG/RDOS.

Why Use It?

Typing BYE is easier than typing the RELEASE Dx0 command (Dx0 is the master directory name). But, since BYE.MC is a macro file and not a command, it doesn't work from just any directory, unless you have linked to it or type a pathname (for example, `DJ0:BYE↵`).

Example

```
bye ↵
FOREGROUND ALREADY RUNNING: DE0          (Error, foreground is running.)
R
```

(Make sure foreground users are logged off.)

```
CTRL-C CTRL-F            (CTRL-C CTRL-F terminates foreground.)
FG TERM

bye ↵             (Try BYE again.)

STARTING SYSTEM SHUTDOWN        (DG/RDOS closes the disk and signs off.)
MASTER DEVICE RELEASED

Filename?              (Ready for another system or program.)
```

CDIR

Creates a subdirectory (variable-size directory)

```
CDIR directory-name
```

The CDIR command creates an empty subdirectory within the current directory. The new directory has the extension .DR, and will grow automatically according the files created in it.

Any DG/RDOS directory except the master must be initialized (INIT or DIR command) before it can be used.

You can create a subdirectory in a disk(ette) or secondary partition, but not in a subdirectory.

Why Use It?

Subdirectories are a useful feature of DG/RDOS, desirable when you want a directory of flexible size. CDIR allows you to create a directory.

Examples

```
cdir newdir ↵
R
list    -.dr ↵
ALPHA.DR CTY 61440
NEWDIR.DR DY 512
R
dir newdir ↵
R
```

These commands create NEWDIR in the current directory, list all directories (-.DR extension) in the current directory, and make NEWDIR the current directory.

CHATR
Changes file attributes, for permanence or read protection

```
               ⎧ + ⎫                       ⎧ + ⎫
CHATR filename ⎨ - ⎬ attributes  [filename ⎨ - ⎬ attributes]
               ⎩ 0 ⎭                       ⎩ 0 ⎭
```

The CHATR command adds or removes attributes from any file (**filename**) except one whose attributes are protected.

File attributes and characteristics are one-letter symbols that describe a file and the restrictions on it. The letters appear after the after the filename in a LIST display. Some examples:

CONFIG.SV ... SD	(S is an attribute, D a characteristic.)
MYDIR.DR ... DY	(D and Y are charactertics.)
SYS.DR ... APWDY	(P and W are attributes, others characteristics.)

To add an attribute, use +**attribute**; for example, +P

To remove an attribute, use -**attribute**; for example, -P

To remove all attributes, use 0 (removes all but S attribute)

Why Use It?

If a file is deleted and it hasn't been backed up, it is lost. There may be times when you want to protect a file against accidental or malicious deletion or change, or times when you want to prevent a file from being read. Also, after you copy an .SV file using the XFER command, you must give the new file the S attribute. The CHATR command allows you to do all of these things.

File permanence does have a disadvantage. If you make a file permanent, you must remember to include the /A switch to list, move, dump, or load the file. And, anyone who knows about CHATR can change the attributes you assign.

CHATR (continued)

The file attributes are

N	No link access; a file can't be accessed by link while it has this attribute.
P	Permanent. A file cannot be deleted or renamed while it has the permanent attribute. (But the parent directory, if not a disk(ette), can be deleted unless *it* is permanent.) To list, load, move, or dump a permanent file, you must add the /A switch to the command — an extra step that you might forget.
R	Read protected. No one can read *this* file (TYPE, PRINT), edit it with a text editor, or copy it via DUMP, IMOVE, MOVE, or XFER while it has the R attribute.
S	Save file. An .SV file needs this attribute to execute. The linker program RLDR assigns it when the save file is created.
W	Write protected. No one can modify *this* file. DG text editors work with a copy of the file, then rename it; thus W doesn't prevent someone from text editing a file. To protect from text editing, use R. Also, W does not protect a file from deletion.

The file characteristics are

A	Attribute protected. You can't change the attributes or the characteristics of this file.
C	Contiguously-organized file; its disk blocks are contiguous.
D	Randomly-organized file.
L	Link entry.
T	Secondary partition.
Y	Directory.

Example

```
chatr    mydir.dr    +p ⤶              (Make MYDIR.DR permanent.)
R
delete/v    mydir.dr ⤶                 (Try and delete it.)
PERMANENT FILE: MYDIR.DR               (Can't delete it.)
R
list mydir.dr ⤶                        (List it.)
R                                      (Doesn't show up.)
list/a     mydir.dr ⤶                  (Add /A switch to LIST.)
MYDIR.DR 512 PDY                       (With /A, MYDIR.DR shows up.)
R
chatr    mydir.dr    -p ⤶              (Remove permanent attribute.)
R
```

CONFIG.SV

Checks or changes hardware and software parameters in a DG/RDOS System

`CONFIG` *[system-name] [dialog-file/V]*

CONFIG is a program supplied with DG/RDOS to help you specify memory allotments, a foreground terminal, printer, and multiplexed communications lines, with terminals and/or a printer.

If you include *system-name* , CONFIG doesn't ask for a system name. If you don't specify a name, CONFIG will ask for a name or default (default is DGRDOS.SV).

If you include *dialog-file/V* and, later, choose to change the system file, CONFIG will store the CONFIG values you select in *dialog-file* — useful when you want to review the values.

CONFIG then offers a series of menu items for which you can change values as desired. When you're done, you can have the changes made to the system file, or leave the file unchanged.

CONFIG details appear in Chapter 2.

Why Use It?

DESKTOP GENERATION computers can run a printer, up to 5 user terminals, a communications line, or a modem. They can also run other software, like FORMA-TEXT, COMPUCALC, Interactive COBOL, and Business BASIC. CONFIG allows you to tailor the DG/RDOS system to support these.

Example

```
CONFIG OCT20.DG/V ↵

Enter name of system to configure: DGRDOS.SV ↵
.
. (edit parameter values)
.
5 ↵ (exit)

Do you want to install changes in the system (Y/N): Y ↵ .
R
BOOT DGRDOS ↵              (Start up the newly-configured system.)
.
.
```

CPART
Creates a secondary partition (fixed-size directory)

```
CPART directory-name max-disk-blocks
```

The CPART command creates an empty secondary partition within the current directory. The new directory has the extension .DR. It will grow as files are created in it, up to the limit of `max-disk-blocks`. A disk block holds 512 bytes (characters).

Any DG/RDOS directory except the master must be initialized (INIT or DIR command) before it can be used.

You can't create a secondary partition with fewer than 48 disk blocks. If the `max-disk-blocks` number is not an integer multiple of 16, DG/RDOS reduces it to the largest integer multiple.

You can create a secondary partition in a disk(ette), but not in a secondary partition or subdirectory.

Why Use It?

Secondary partitions are a useful feature of DG/RDOS, desirable when you want a new directory of limited size (perhaps to control a user's disk space consumption). Also — unlike subdirectories — the DISK command returns the amount of space left in the secondary partition, not the parent disk. CPART allows you to create a secondary partition.

Examples

```
cpart sally 500 ⤸
R
list/s    -.dr ⤸
ALPHA.DR 512 DY
NEWDIR.DR 512 DY
SALLY.DR 256000 CTY
R
dir sally ⤸
R
```

These commands create secondary partition SALLY, with 500 blocks, in the current directory, list all directories (-.DR extension) in the current directory, and make SALLY the current directory. Since each block holds 512 bytes (characters), directory SALLY can hold files that total more than 250,000 bytes.

DELETE

Deletes one or more files

```
DELETE filename [filename] [...]
```

The DELETE command can delete any file (directory or nondirectory) that's not permanent. You can use pathnames to specify the filenames. Filename templates are allowed.

To delete a directory, release it (RELEASE name), then delete directory-name.DR.

To protect files from deletion, use the CHATR command to make them permanent.

CAUTION *Use UNLINK,* not *DELETE, to remove a link entry. If you use DELETE, it will delete the resolution file, not the link entry.*

Why Use It?

As people use a computer system, they create files, which accumulate and consume valuable disk space. Some of these files become obsolete or redundant. The DELETE command allows people to remove files from active disk(ette)s.

Periodically, you (or someone) should use DELETE to remove files that are no longer needed. This is critically important in a diskette-only system. In a multiuser system, individual users, who know their files, should be the ones to clean up.

Switches

/C	Confirms deletion. DG/RDOS will display each matching filename and wait. If you press ↵, it will delete the file. If you type any other character, it won't delete the file.
name/N	No deletion. Don't delete files that match the name or template.
/V	Verifies deletion. DG/RDOS displays the name of each deleted file.

Examples

```
delete/v    xfile ↵
DELETED XFILE
R
delete/v    foo.dr ↵
FILE IN USE: FOO.DR
R
release foo ↵
R
delete/v    foo.dr ↵
DELETED FOO.DR
R
```

These commands deleted file XFILE and directory FOO.DR in the current directory.

```
delete/v/c    my-.- ↵
MYFILE.1 ↵                    (Confirm with ↵.)
DELETED MYFILE.1              (It deletes and confirms.)

MYFILE n ↵                    (Say no....)
                              (System says nothing.)
MYFILE.BU ↵                   (Confirm with ↵.)
DELETED MYFILE.BU             (It confirms deletion.)
R
```

Here, the /C switch told the CLI to confirm before deleting (and /V verified as usual). The person chose to delete file MYFILE.1, chose not to delete MYFILE, and chose to delete MYFILE.BU.

DIR

Changes the current directory

DIR *[directory-pathname]*

The DIR command changes the current directory. If any directory in the directory-pathname has not been initialized, DIR initializes it.

To learn the current directory name, type GDIR↲.

CAUTION *Be sure that any diskette that you've initialized (via DIR or INIT) is released before you remove it from its drive. The command RELEASE DJx' does this. If you do accidentally remove an initialized diskette without releasing it, try inserting it in the original drive and typing the RELEASE command.*

Why Use It?

Often, it's easier to work within the directory that holds the files you want than to type pathnames or use links to these files. The DIR command allows this — and initializes any uninitialized directories at the same time.

Examples

```
dir dj0:learning ↲
R
gdir ↲
DJ0:LEARNING
R
dir    %mdir% ↲
R
gdir ↲
DE0
R
```

These commands make directory LEARNING on DJ0 the current directory, get the directory name, then make the master directory the current directory and check the name again. (The %mdir% is a CLI variable that contains the name of the master directory.)

DISK
Displays the amount of disk space left and used

```
DISK
```

The DISK command displays how much disk space is left (LEFT), how much is occupied by files (USED) and the largest number of contiguous free blocks (contiguous free storage space, unbroken by files.)

The disk figures pertain to the current disk(ette), unless the current directory is a secondary partition. (In terms of size, secondary partitions are treated like disks, so DISK returns figures based on the maximum size of the partition.)

A model 6231 hard disk has about 31,000 disk blocks for file storage; a model 6301 hard disk has about 80,500 blocks. A model 6268 diskette has about 720 blocks. Each disk block can store 512 bytes (characters).

Why Use It?

If any disk(ette) becomes filled with files, system performance will suffer. Eventually, if the disk(ette) fills up, all work will stop until a person frees space by deleting files.

The DISK command tells you how much space is left on the disk. You can then use this information to decide when to clean up (by copying files to diskette, if needed, and deleting them). The LIST command tells the size of each file.

DISK (continued)

Examples

```
dir de0 ↵                                        (Hard disk.)
R
disk ↵                                           (Check disk usage.)
LEFT: 24009 USED: 7095 MAX. CONTIGUOUS: 22388    (Plenty left.)
R
dir sally ↵                                      (Secondary partition.)
R
disk ↵                                           (Check.)
LEFT: 624 USED: 400 MAX CONTIGUOUS: 587          (Not too bad.)
R
dir dj0 ↵                                        (Diskette.)
R
disk ↵                                           (Check.)
LEFT: 220 USED: 500 MAX CONTIGUOUS: 96           (Not much left.)
R
```

EXFG
Executes a program in foreground memory

```
EXFG program
```

The EXFG command tries to execute the program you specify in foreground memory. For this to work, the following must be true:

- enough channels for the program must have been specified with CONFIG. The default is 0 channels, which doesn't allow any program to execute.
- for a large program, your computer needs at least 256 Kbytes of memory.
- you must have allotted enough memory to the foreground with the SMEM command.

Some programs, like the CLI, and multiuser programs like ICOBOL and Business BASIC, need one or more user terminals if they are to run in the foreground. The foreground program can't use the background terminal (system console). Even without a second terminal, though, you may be able to use the foreground for things like compilations and assemblies.

If the foreground program terminates within a few seconds with a FG TERM message, this means it hit a fatal error condition. There's no easy way to find the cause. For multiuser products, the problem may be the product configuration; consult the product manual and/or Release Notice for the latest configuration information. (There's some information in Chapter 2, Table 2-5.)

You can tell if a foreground program is running by typing FGND↲.

To terminate the foreground program, type CTRL-C CTRL-F to the CLI on the system console.

Why Use It?

The ability to run a second program — simultaneously with the background program — can more than double the productivity of your computer system. Several other programs, like COMPUCALC and ICOBOL, must run in the foreground if they are to serve multiple users. The EXFG command is needed to start a program in the foreground.

EXFG (continued)

Switches

/E Equal priority for both programs. If you omit this, the foreground program has priority over the background program, and gets a greater share of system resources.

Example

```
exfg compucalc ⤶
INSUFFICIENT MEMORY TO EXECUTE PROGRAM: COMPUCALC.SV
R
gmem ⤶
BG: 65 FG: 0
R
smem 22 ⤶
R
exfg compucalc ⤶
R
. (foreground program prompt appears on foreground terminal(s);
. program runs for awhile)
.
R
CTRL-C CTRL-F         (Bring down foreground.)
FG TERM
```

Here, a person tries to execute a program in the foreground, gets an informative error message, sets the foreground to have more memory, and then successfully runs the program. Later, she shuts down the foreground with CTRL-C CTRL-F, preparing for *system* shutdown.

FCOPY.SV

Copies a diskette or file to another diskette

```
FCOPY { [source-diskette destination-diskette] }
      { [source-file destination-file]         }
```

The FCOPY program can duplicate a diskette or copy a file to another diskette using one diskette drive or two drives. FCOPY requires only that both diskettes be hardware formatted. If you want to duplicate a a diskette, the destination diskette cannot have any bad blocks.

Using screen menus, FCOPY prompts you to the desired operation. If you omit needed information in the command line, the program will prompt for it in a menu.

Details on running FCOPY appear in Chapter 6.

NOTE *Be sure and tell FCOPY to verify the duplicate or copy. You can do this with the /V switch by answering an FCOPY question.*

Why Use It?

FCOPY is useful whenever you want to copy a diskette, in any format.

On a single-diskette system without a hard disk, FCOPY offers the *only* way to duplicate diskettes for backup, or transfer a file from another diskette to your system diskette.

Also, FCOPY can duplicate any hardware-formatted diskette, like the Customer Diagnostics diskette. Even if you have two diskette drives, FCOPY is preferable to the Customer Diagnostics copy function because DG/RDOS can stay up while you run it.

FCOPY.SV (continued)

Switches

/C	Copies a file.
/D	Duplicates a diskette.
/V	Verifies the copy or duplicate. *Always include this switch.*

Examples

```
fcopy ⤶
DG/RDOS Diskette Transfer Utiltity
.
. (It displays menu; FCOPY operations proceed)
R

fcopy/d/v    dj0    dj0 ⤶

DG/RDOS Diskette Transfer Utility
.
. (It displays menu; FCOPY operations proceed)
R
```

GDIR

Gets the current directory name

GDIR

The GDIR command displays the current directory name.

Why Use It?

For many DG/RDOS operations, you need to know the current directory name.

Example

```
gdir ⤶
LEARNING
R
message the current directory is %gdir%. ⤶
THE CURRENT DIRECTORY IS LEARNING.
R
dir %mdir% ⤶
R
gdir ⤶
DEO
R
```

In this series of commands, GDIR gets the current directory name; then, the MESSAGE command displays the name using the CLI variable %gdir%; and finally, the DIR command gets to — and GDIR confirms — the master directory.

GMEM
Gets the amount of memory in background and foreground

GMEM

The GMEM command displays the memory allotment — in 2 Kbyte pages — for the background and foreground.

Why Use It?

If you want to run two programs simultaneously, you need to give enough memory to each program ground. GMEM tells you how much each ground has. You can then change the allotment —if desired — the SMEM command.

Example

```
gmem ⤶
BG: 67 FG: 0
R
smem 26 ⤶
R
gmem ⤶
BG: 26 FG: 41
R
exfg myprogram ⤶
.
.
```

Here, GMEM and SMEM display, set, and verify memory allotment for execution of program MYPROGRAM.

GTOD
Gets system time and date

```
GTOD
```

The GTOD command displays the system time and date.

Why Use It?

You may simply be curious about the date and/or time.

Or, you may want to check system calendar and clock. It's very important that these be correct, since the creation dates of all files are based on them. The GTOD command tells the system date and time.

If needed, you can change the date with the SDAY command; or you can change the time with the STOD command.

Example

```
gtod ↲
11/17/84 13:24:40           (November 17, 1984, 1:24:40 p.m.)
R
stod 14 25 ↲                (Set clock to 2:25 p.m.)
R
gtod ↲
11/17/84 14:25:13           (Okay...)
R

message The date is %date% and the time is %time% ↲
THE DATE IS 11-17-84 AND THE TIME IS 14:25:26
R
```

This sequence shows the system clock being checked, then set; finally it shows two CLI variables that you can use in macros.

IMOVE.SV
Copies files to or from diskette or tape

```
IMOVE/D  ⎧ DJ0 ⎫  [filename] [filename][...]
         ⎨ DJ1 ⎬
         ⎩ MT0 ⎭
```

The IMOVE utility was supplied with DG/RDOS to help you back up and restore disk-based files. It can copy files from either hard disk or diskette, but is most useful for backup and restoration of a hard disk. (With diskettes only, it's more useful to copy an entire diskette — using FCOPY or other method — for backup.)

The /D switch is mandatory, unless you ask for help by typing IMOVE/H. By default, files are copied from the current directory to the backup medium. To restore, from backup medium to the current directory, use the /F switch.

If you omit a *filename*, IMOVE copies *all* files in and beneath the current directory. If you include a filename, then, for each name you include, IMOVE copies the file and (if it's a directory) all files in it. Template characters are not allowed.

One way to do an incremental backup is to build an indirect file with BUILD, using date switches, then use the indirect filename (in @ signs) as the pathname.

Preferably, start backups from a specific directory; for example, the master directory (DE0). If you back up from the master, all user and system files will be backed up, and can be restored easily and simply.

If the material to be copied requires another diskette, IMOVE will prompt for it. Later, if your site needs to restore material, the backup diskettes must be inserted in the *same* order that they were originally filled. If the order is wrong, the restoration won't work. Therefore, we suggest that you apply a label (if the diskette doesn't have one), and note the sequence number and date on the label. To write on a diskette label, use a felt-tipped pen to avoid scoring the diskette surface.

The diskettes you use for backup must be hardware formatted, but need not be software formatted with DKINIT.

Why Use It?

It's prudent to back up your disk-based material — in case someone accidentally deletes valuable files, or in case material is lost for any reason.

Depending on how many files change each day, you might choose a full backup each month or so, and an incremental backup each week, or on alternate days.

The whole backup issue gets more detail in Chapter 7.

Switches

/C	Converts files for AOS/VS, AOS, or MP/AOS operating systems. Use /C only to convert, not back up, files.
/D	Dump format. This switch is always required (except with /H).
/F	From backup medium. This tells IMOVE to copy from backup media to the current directory. If you omit it, IMOVE copies *to* the backup medium.
/H	Help. Displays help text about IMOVE command line and switches.
/L	Lists pathnames copied to the printer ($LPT). If your system doesn't support a printer, IMOVE will list pathnames to file $LPT — if you create file $LPT (CRAND□$LPT↵) before typing the IMOVE/L command. After the backup is done, perhaps you can move file $LPT to another diskette, and print it on a different system. A paper record of files backed up is important. This switch overrides /V.
/N	No transfer. IMOVE displays the names that it *would* copy, but doesn't write anything to the destination medium. This switch is very helpful when you want to see either what would be backed up, or what's on an IMOVE backup diskette.
/O	Overwrites files that already exist (relevant on a file restore only). If a file in the current directory has the same pathname as a file on backup medium, IMOVE deletes the file in the current directory and replaces it with the file from diskette. (To do this conditionally, based on date, use /R.)

IMOVE.SV (continued)

/R Recent. Overwrites files that exist based on date (relevant on a file restore only.) If a file on the backup medium has a more recent creation date than a file with the same pathname in the current directory, IMOVE deletes the file in the current directory and replaces it with the newer version. The /R switch is required for incremental restores.

/T Tape. The backup medium is tape (device MT0) not diskette (DJx).

/V Verifies names copied to the terminal.

Examples

Prepare for backup by inserting a diskette in drive DJ0.

```
dir acctsdue ↓
R
imove/d/l    dj0 ↓

Please insert disk 1, then press NEW LINE to continue. ↓
.
. (IMOVE prompts for additional diskettes as needed)
.
R
```

This IMOVE sequence starts a backup of all nonpermanent files in directory ACCTSDUE to the diskette in DJ0. It lists all files copied to the printer. (Printing doesn't begin until the backup is done.)

For the next example, a person inserts the first diskette in the backup set in drive DJ0.

```
dir acctsdue ⤶
R
imove/d/f/r/l    dj0 ⤶
```

Please insert disk 1, then press NEW LINE to continue. ⤶

.
. (IMOVE prompts for additional diskettes as needed)
.

R

This is a restore variation of the previous command. The IMOVE/D/F command starts restoring to the hard disk the file structure that was originally copied. If any files in the backup medium are newer than those on the hard disk, IMOVE will delete and replace them. All filenames restored are listed on the printer, where they can be compared to the original backup listing. (Printing doesn't begin until the restore is complete.)

For more examples, see Chapter 7.

INIT

Opens a directory or tape drive, or creates a new file directory

```
INIT     { directory-pathname }
         { MTO                }

INIT/F   { disk(ette) }
         { MTO        }
```

The INIT command, without /F, opens a directory or tape drive (MTO) for access to its files. It does not change the current directory (as does the DIR command).

CAUTION *Be sure that any diskette that you've initialized (via INIT or DIR) is released before you remove it from its drive. You can use the command RELEASE DJx↵ to do this. System shutdown also does it. If you do accidentally remove an initialized diskette without releasing it, try inserting it in the original drive and typing the RELEASE command.*

The number of directories that can be initialized at one time is specified to CONFIG; the default is 10 directories.

INIT/F Command

INIT/F is really a different command from INIT. It writes a new file directory to the disk(ette), preparing it for file storage. This effectively destroys existing files on the disk(ette) by eliminating pointers to them. For tape, the net effect is the same; INIT/F eliminates access to any tape files on the tape, effectively destroying them.

After you type INIT/F, the system asks for confirmation. If you type Y, the full initialization occurs. If you type any other character, the command is cancelled.

Why Use Them?

The INIT command is needed for a tape drive; and it's useful when you want to initialize a directory without changing the current directory.

The INIT/F is needed to set up newly-formatted disk(ettes) for file storage. (If you try to INIT or DIR to a disk(ette) that has never been fully initialized, DG/RDOS will say FILE DOES NOT EXIST. It needs the file directory written by INIT/F.)

Example

```
init dj1:mydir ↵
R
type mydir:myfile ↵
.
. (text of myfile)
.
```

These commands initialize directory DJ1 (if not already initialized) and directory MYDIR, then type a file in MYDIR.

```
init/f dj0 ↵
CONFIRM? Y ES
R
dir dj0 ↵
R
list ↵
R
list/a ↵
SYS.DR 3345 APWDY
MAP.DR 288 APWY
R
```

This sequence shows an INIT/F to the diskette in DJ1. Note that — after the INIT/F — no nonpermanent files show up in DJ1. The only ones are SYS.DR (system file directory) and MAP.DR (system space allocation directory).

LINK
Creates a link entry to a file in any directory

`LINK link-entry-name [directory:] resolution-filename`

A link entry is a file that contains only a pathname. The LINK command creates a link entry (`link-entry-file`) to another file (`resolution-filename`) that resides in the specified directory. The `link-entry-name` is created in the current directory unless you specify a different directory.

For the resolution file's directory, use only the immediate parent directory (for example, for resolution file DJ0:LEARNING:MYFILE, use LEARNING:MYFILE). A link can have only one directory in its resolution file pathname.

The link entry will work only if

- the resolution file exists in the directory indicated by the link; and
- the directory that holds the resolution file is initialized.

Also, generally, the link entry should have the same extension as the resolution file. For example, the link entry SPEED.SV to resolution file SPEED.SV will work, but a link entry named SPEED to the same file won't work. The system doesn't check for the conditions above until you try to *use* the link, so we suggest you try the link immediately after creating it to see if it works.

CAUTION *When you want to remove a link entry,* don't *use the DELETE command. DELETE will delete the* resolution file *and leave the link intact. To remove a link entry, use the UNLINK command.*

Why Use It?

Links conserve disk space. A link gives easy access to a file from a different directory, yet consumes only a few bytes of space. Links are most useful from user-created directories to often-used files (like utility program files) in the master directory.

Example

```
dir mydir ⤶
R
imove/d/v    dj1 ⤶                            (Try to execute IMOVE pro-
                                              gram.)
FILE DOES NOT EXIST: IMOVE.SV                 (Error.)
R
link    imove.sv    %mdir%:imove.sv ⤶         (Create link, using %mdir%
                                              variable which contains
                                              name of master directory.)
R
imove/d/v    dj1 ⤶
Please insert disk 1...                       (IMOVE executes.)
```

This example shows creation and use of a link entry to the backup program IMOVE.SV in the master directory.

The next example occurs in a two-diskette system with no hard disk.

```
gdir ⤶
DJ0
R
link basic.sv dj1:basic.sv ⤶          (Create link to BASIC interpreter
                                      on DJ1.)
R
basic ⤶
NO SUCH DIRECTORY: BASIC.SV           (Misleading error message.)
R
init dj1 ⤶                            (Initialize the directory.)
R
basic ⤶                               (Try again.)
..BASIC REV x.xx                      (BASIC executes.)
*
```

In this example, the resolution file (BASIC interpreter) is on a separate diskette, which isn't initialized. The system reports a NO SUCH DIRECTORY message. Then, initializing the directory lets the link find the resolution file, and BASIC executes.

LIST
Describes file names and statistics

`LIST` *[pathname] [pathname] [...]*

The LIST command displays information on files in any directory. If you omit a *pathname* , it describes all files in the current directory.

Templates are allowed (and extremely useful) in arguments, but work only in the current directory; for example,

`LIST A-↲` will work but

`LIST SUBDIR:A-↲` won't work.

To list a file outside the current directory, specify the whole pathname (for example, `LIST SUBDIR:ABEL↲`).

LIST *ignores* permanent files unless you include the /A switch.

Why Use It?

LIST is your primary source of information on files in a DG/RDOS system. You'll probably use it more often than any other command.

Switches

/A	Includes names of permanent files (P attribute, CHATR command) and nonpermanent files. If you omit this, LIST will omit names of permanent files.
/E	Everything. List every type of information. To the display, this adds: file creation date and time, time last read, starting disk block address, and file use count.
/K	No links; omits the names of link entries. See also /N.
/L	Lists pathnames to the printer (file $LPT) instead of the terminal screen. If your system doesn't support a printer, LIST/L will write the pathnames to file $LPT, if you create $LPT (CRAND $LPT↲) before typing the LIST/L command.

mm-dd-yy/A	After date. Includes names of files created on or after this day (mm and dd can be one or two digits). For example, 10-25-84/A specifies files created on or after October 25, 1984.
mm-dd-yy/B	Before date. Includes names of files created *before* this day. The syntax is the same as the date/A switch; you can use either or both /A or /B date switches.
/N	Links only; list link entries only.
name/N	Omits names of files that match name or name template.
/S	Sorts the filenames alphabetically.

Examples

```
list/s ⤶
```

```
ALLEN.TT       54  D
JANUARY.DB  11340  PD
LEARNING.DR   512  DY
MACRO.MC       34
R
```

Here, someone lists and sorts all nonpermanent files in the current directory. (For an explanation of the letters that follow the file size, see the CHATR command.)

```
list/s/e   -.dr   10-15-84/a   10-22-84/b ⤶
```

```
BIORHYTHM.DR  512 DY  10/15/84  09:01  10/30/84  [010532] 0
BOOMER.DR     512 DY  10/20/84  14:48  11/28/84  [007532] 0
PAYSTAT.DR  32000 CTY 10/18/84  11:33  01/02/85  [006676] 0
SAVANTS$.DR   512 DY  10/21/84  08:55  12/03/84  [006571] 0
R
```

Here, someone lists all directories (in the current directory) that were created between October 15 and 22, 1984.

LOADEM.SV

Loads a terminal emulator into memory or disk(ette) if needed

```
LOADEM
```

The LOADEM program was supplied with DG/RDOS to load terminal emulator firmware on Model 10 and 10/SP systems. LOADEM can also install the correct terminal emulator file on the system disk(ette).

For systems with US or UK keyboards, the emulator filename is

- MD211AMUK.TX, accessed by link MTERM$.EM, for monochrome monitor; or
- CD211AMUK.TX, accessed by link CTERM$.EM, for color monitor.

Why Use It?

Model 10 and 10/SP systems need a terminal emulator to allow lowercase letters and bit-mapped graphics on the system console. Program LOADEM checks to see if this emulator has been loaded into the computer's memory — and loads the emulator if needed. LOADEM can also select the correct emulator file for your terminal and copy it onto the system disk(ette).

You need to use LOADEM very rarely — only when building a DG/RDOS system disk(ette). However, DG/RDOS always runs LOADEM at start up. LOADEM is the program that displays

```
WELCOME TO THE DG DESKTOP GENERATION
```

For modem 10 or 10/SP systems, LOADEM also displays the message

```
EMULATOR IS BEING LOADED

... Emulator load completed.
```

or

```
Emulator is already loaded.
```

Example

```
loadem ⤶

WELCOME TO THE DG DESKTOP GENERATION

Emulator is already loaded.
R
```

LOG
Starts logging terminal dialog in a disk file

`LOG` *[password]*

The LOG command starts recording terminal dialog in file LOG.CM, in the current directory. If LOG.CM doesn't exist, the system creates it; if it does exist, the system appends to it.

You can't read (TYPE or PRINT) the log file while logging is occurring. Nor can you rename or delete it. To stop logging, use the ENDLOG command. You must type ENDLOG with the original password if the person who started logging specified a password.

To start a new log file, stop logging if needed (ENDLOG); then rename or delete the old log file and type `LOG⏎`.

The log file records only CLI dialog — not dialog with other programs like BASIC or IMOVE.

CAUTION *If you start the log file on a diskette, be sure to type ENDLOG before you release and remove the diskette. If you substitute a different diskette for the diskette on which the CLI expects to find LOG.CM, you'll probably lose the log file. The CLI will report a LOG FILE ERROR, explained in Chapter 14, the error chapter.*

Why Use It?

Occasionally you may want a record of terminal dialog — for example, as the basis for a procedures guide, or as a record of how long a program took to run. The LOG and ENDLOG commands provide a record.

You can append the time to the CLI prompt by typing .↵ (period and ↵). Later, to remove the time, type .↵ again. Having the time appended to the prompt is sometimes useful in log files.

The log file grows as you type commands, and will come to consume a lot of disk space. To save space, you may want to back up the log file(s), then delete them. On diskette-only systems, space issues may prevent you from using log files very much.

Switches

/H	Writes a header, with directory and date information, at the beginning of the log file.
directory/O	Writes the log file itself to directory instead of the current directory.

LOG (continued)

Examples

```
log ↲
R
message "This dialog shows where to find, and how to execute, MYPROG."↲
THIS DIALOG SHOWS WHERE TO FIND, AND HOW TO EXECUTE, MYPROG.
R
dir routine:daily ↲
R
message "We're here, let's start - and time - program MYPROG."
WE'RE HERE, LET'S START - AND TIME - THE DAILY PROGRAM MYPROG.
R

gtod; myprog ; gtod ↲          (Technique to time a program.)
08/14/84 13:17:13
.                   (MYPROG executes.)
.
08/14/84 15:23:55
R
message "MYPROG is done - exit to master directory." ↲
MYPROG IS DONE - EXIT TO MASTER DIRECTORY.
R

type log.cm ↲                       (Check log file.)
FILE IN USE: LOG.CM                 (Error)
R

endlog ↲                            (Close the log file...)
R
type log.cm ↲                       (Try again.)
.
. (text of log file)
R
rename log.cm myprog$use ↲          (Rename log to meaningful
                                    name.)
R
```

This example shows how you might use the MESSAGE command, in conjunction with other CLI commands, to describe a procedure. Some editing — with a text editor — would clarify things by eliminating duplication.

MESSAGE

Displays text on the screen

MESSAGE *[" [text] "]*

The MESSAGE command allows you to write messages to the terminal, and, if logging is on, to the log file. If you use quotation marks (pairs required), the CLI displays text as you wrote it; the only change is lowercase to UPPERCASE. If you omit quotes, the CLI treats punctuation as usual — parentheses, angle brackets, and percent signs are treated as special characters (as usual in the CLI).

Why Use It?

It's often useful to be able to document what's happening with the CLI — for CLI macros or while logging terminal dialog to disk (LOG). And, you can use the /P switch to interact with the person at the terminal.

Switches

/P	Pause after displaying the text, display the message STRIKE ANY KEY TO CONTINUE, and wait for a keystroke.

Example

```
xfer/a    $tti    md.mc ⤶
message All directories in the master directory are ⤶
dir    %mdir% ⤶
list/s/e    -.dr ⤶
dir    %ldir% ⤶
↓              (Press downarrow key on terminal.)
```

This macro lists all directories (-.DR extension) in the master directory. CLI variable %mdir% contains the name of the master airectory; variable %ldir% contains the name of the last directory (thus the macro returns the person who uses it to his original directory).

For another example, see LOG.

MOVE

Copies one or more files to any directory

`MOVE directory-name` *`[filename] [...] [old-filename/S new-filename]`*

The MOVE command moves a copy of one or more files to the directory named in `directory-name`. The *`filename(s)`* must be in the current directory. Templates are allowed with *`filenames`*.

If you omit *`filename`* arguments, DG/RDOS copies all files in the current directory to the destination directory. This may waste a lot of disk space through file duplication; do it only when you really *want* to duplicate all files.

To create a copy of a file in the current directory, use the *`old-filename/S new-filename`* approach. For example,

```
move/v   %gdir%   myfile/s   myfile1 ⤶
```

This is useful to create a backup version of a file.

MOVE ignores permanent files unless you include the /A switch.

Why Use It?

From time to time, you may want to move one or more files from one directory to another; for example, to reorganize your files. You could use XFER for this, but MOVE retains the original name, creation date, and so on, and thus serves better to identify the file. Also, MOVE allows template characters and has date switches.

Generally, to conserve disk space, you might want to have the original files deleted (/D switch) after the move.

To back up files to diskette for safekeeping, we recommend that you use a backup program like IMOVE. If you use MOVE, the diskette must have been formatted with the DKINIT formatter. Also, MOVE can't back up to more than one diskette, thus you need multiple MOVE commands, and keeping track of files copied becomes tedious.

Switches

/A	All files. Includes permanent files (P attribute, CHATR command) and nonpermanent files. If you omit this, MOVE will skip permanent files.
/D	Delete originals; deletes source file after copying them. This switch prevents file duplication.
/K	No links; omits the names of link entries. See also /N.
/L	Lists filenames moved to the printer (file $LPT) instead of the terminal screen; overrides the /V switch.
mm-dd-yy/A	After date. Includes names of files created on or after this day (mm and dd can be one or two digits). For example, 10-25-84/A specifies files created on or after October 25, 1984.
mm-dd-yy/B	Before date. Includes names of files created before this day. The syntax is the same as the date/A switch; you can use either or both /A or /B date switches.
name/N	Omits names of files that match name or name template.
/S	Sorts the filenames alphabetically.
/R	Recent overwrite. If a file is newer than a file with the same name in `directory-name`, the system deletes the older file and replaces it with the newer one.
/V	Verifies files moved by displaying their names.

MOVE (continued)

Examples

```
move/v    letters    -.lt ↲
$STONE.LT
T$A$WOLFE.LT
.
.
R
```

Here, MOVE copies all files that have the extension -.LT into directory LETTERS.

```
move/v/r    acctsdue    -ax.-    -pd-.-/n    3-1-84/b ↲
ACEDEPT$AX.
ZRETAIL$AX
BHABER$AX.03
.
```

Here, MOVE copies all files whose names end in AX (but don't contain characters PD) that were created before March 1, 1984, into directory ACCTSDUE.

PRINT

Starts printing a file

```
PRINT pathname [...]
```

The PRINT command sends the text of the file named in **pathname** to the printer, $LPT.

If the printer is connected to the CPU printer port, DG/RDOS copies the text to a spool file, from which it is printed. The R prompt returns to your terminal almost immediately. If, with spooling, you want to stop printing and empty the spool file, type

```
SPKILL $LPT ⤶
```

DG/RDOS uses the master directory for the spool file. Thus, spooling won't work if you release the master directory, then print a file.

There is no spool file for multiplexor (USAM) lines. If your printer is connected to a USAM line, the CLI prompt won't return to your terminal until the entire file has been printed (or you interrupt printing). You can interrupt printing on a USAM line by typing CTRL-C CTRL-A.

If the printer produces garbled copy, perhaps both foreground and background programs are trying to use it simultaneously. DG/RDOS can't distinguish print requests by ground, so try to arrange to have only only one ground issue print requests at once.

Switches

pathname/n	Prints n copies of **pathname**. The n must be a single digit, but you can add other switches; for example, PRINT MYFILE/9/3 prints 12 copies.

Why Use It?

PRINT is the best way to print a file. It is the *only* way to print from the CLI without tying up your terminal until the file is printed.

Examples

```
print    myfile    dj1:yourfile ⤶
R
```

This command prints MYFILE and DP1:YOURFILE on the printer.

RELEASE

Releases (closes) a directory or tape drive

$$\texttt{RELEASE} \quad \left\{ \begin{array}{l} \texttt{directory} \\ \texttt{MT0} \end{array} \right\}$$

The RELEASE command closes a directory or tape drive. For a directory, RELEASE updates and closes all files (including all subordinate directories), and removes the directory from DG/RDOS. For a tape drive (MT0), RELEASE simply rewinds the tape.

Release of the master directory may cause problems later if logging is on (LOG command) or if you want to print files using the PRINT command.

System shutdown releases all directories and devices. To shut down, use the BYE macro.

If two programs are running, each program has initialized a directory, and one program releases the directory, DG/RDOS displays

DIRECTORY SHARED: directory

This is an informational message only — it doesn't indicate an error condition.

If an initialized diskette is removed from its slot without being released, there may be problems when you later try to access the diskette. (If you *do* accidentally remove an initialized diskette from its slot without releasing it, try returning it to the slot and typing RELEASE DJn⤶ — n is 0 or 1.)

Why Use It?

After a directory has been initialized, it must be released to ensure file integrity. After a tape has been accessed, it must be rewound so you can remove the cartridge. The RELEASE command does these things.

Example

```
release dj0 ⤶
DIRECTORY NOT INITIALIZED: DJ0
release dj1 ⤶
R
```

RENAME
Renames a file

`RENAME oldname newname` *`[oldname newname] [...]`*

The RENAME command renames the file you specify in **oldname**, giving it the name **newname**. The file must be in the current directory.

A program (save) file can't execute without the .SV extension, so don't give any such file a name without the .SV extension.

CAUTION *Some DG products must have their original names to work properly. If want your system to function normally, do not rename any of the DG-supplied directories or files, unless the DG documentation or Release Notice tells you to do so.*

Why Use It?

RENAME is handy when you want to give a file a new, more descriptive name. (Often, as you work with your own files, you'll think of better names for them.)

If you rename a directory to which there are link entries, the links won't work. You'll need to remove each one (UNLINK) and recreate it (LINK) with the new directory name.

Example

```
rename    xxdir    archives ⤶
FILE DOES NOT EXIST: XXDIR
R
rename    xxdir.dr    archives.dr ⤶         (Include the .DR extension.)
R
```

This changes the name of directory XXDIR.DR in the current directory to ARCHIVES.DR.

SDAY
Sets the system date

```
SDAY mm-dd-yy
```

The SDAY command sets the system date (calendar). Numbers `mm` and `dd` can be one or two digits.

To *display* the system date, type GTOD↵.

Why Use It?

It's very important that the system date be correct: the creation dates of all user files depend on it. SDAY allows you to set the date — for example, if you type the wrong date at startup.

Example

```
gtod ↵
11/04/83 13:22:03
R
sday 11 4 84 ↵
R
gtod ↵
11/04/84 13:22:25
R
```

SEDIT.SV
Edits disk file locations

`SEDIT filename`

The SEDIT program allows you to change the contents of disk file locations: the codes that indicate numbers, letters, or instructions to the computer. The file can be a program file (.SV) or a text file.

CAUTION *Do not use SEDIT on any file supplied by DG unless a DG person or product Release Notice tells you to do so.*

Why Use It?

Under some circumstances, a person may need to apply patches (corrections) to a disk file. SEDIT allows this. Usually, you won't need SEDIT; we describe it in this book because it ships with DG/RDOS.

Switches

/N	Do not search for a symbol table. Use this if there is no symbol table or to edit a text file.
/Z	File starts at location 0. (Operating system files start at location 0. Program files to execute *under* DG/RDOS start at location 16 octal, the SEDIT default.)

Example

```
sedit/n    myprog.sv ↲
SEDIT REV x.xx
.404/66544 ↲
```

(SEDIT prompt is a period (.) Display location 404, which has program NMAX — highest memory location, in two-byte words, in octal.)

```
.ESC Z
DONE
R
```

(To exit, type BREAK/ESC Z.)

SMEM

Sets memory allocation for background and foreground programs

```
SMEM memory-pages-for-background
```

SMEM sets the amount of memory (in 2-Kbyte memory pages) for the background program. All the rest of the available memory goes to the foreground.

The GMEM command tells how much memory is available to both grounds.

The CLI requires a minimum of 21 pages; to execute LIST commands, it may need more pages.

You can use SMEM only from the background CLI, when no foreground program is running.

Why Use It?

At startup, all memory is given to the background program.

If you want to run a foreground program (run two programs simultaneously), you must allot enough memory to the foreground for the desired program to run. SMEM lets you do this. You can then execute the program in the foreground with EXFG.

Example

```
exfg compucalc ↵
INSUFFICIENT MEMORY TO EXECUTE PROGRAM: COMPUCALC.SV
R
gmem ↵
BG: 67 FG: 0
R
smem 26 ↵
R
gmem ↵
BG: 26 FG: 41
R
exfg compucalc ↵
.
. (foreground program prompt appears on foreground terminal(s)).
.
```

Here, GMEM and SMEM display, set, and verify memory allotment for execution of the COMPUCALC program.

SPKILL

Stops printing and deletes the spool file

```
SPKILL $LPT
```

On systems with a printer connected to the CPU printer port, the SPKILL command deletes the spool file. This stops printing immediately on such systems. (If the printer is connected to a multiplexor line, there is no spool file; you must type CTRL-C CTRL-A to stop printing.)

Files to be printed aren't affected by SPKILL — only the spool file of all queued print requests is affected.

To restart the printer and spooling, type another PRINT command.

Why Use It?

Sometimes you will want to stop the printer — for example, to

- stop printing files that you don't want printed (for example, if the printer is printing a binary file, or intermixing parts of print requests from foreground and background).
- empty the spool file so you can shut down the system.

On systems with a printer on the CPU printer port, the SPKILL command is the only good way to kill an active printing job.

Example

```
print longfile/3 ⤶
R
. (printing begins)
.
spkill $lpt ⤶
R
(printing stops)
```

STOD

Sets the system time

STOD *[hh [mm [ss]]]*

STOD sets the system time. With no arguments, it specifies midnight (the last second of the current day). With one argument, it sets the hour (24-hour clock) with minutes and seconds 00:00. With two arguments, it sets the minute, with seconds 00. With three arguments, it sets hours, minutes, and seconds. As an example, 13 22 40⤶ is 1:22:40 p.m.

To see what time the system thinks it is, type GTOD⤶.

Why Use It?

STOD is similar to SDAY, and it should be as accurate as possible. Use it if the system clock is wrong (the system has been brought up with the wrong time), or when a daylight savings time shift changes the real world time.

Examples

```
gtod ⤶
04/10/84 14:17:46
R
stod 15 18 ⤶
R
gtod ⤶
04/10/84 15:18:08
R
```

This command sequence advances the system clock one hour.

TYPE

Types one or more files on the terminal screen

TYPE pathname *[pathname] [...]*

The TYPE command copies the file named in pathname to your terminal screen.

TYPE works properly only for text (ASCII) files; for example, those whose names end in .MC. (If you try to TYPE a file whose name ends in .SV, you'll get strange results — and might lock the terminal in graphics mode. If the display is garbled, and CTRL-C CTRL-A has no effect, press CMD and ERASE PAGE; then try CTRL-C CTRL-A again.)

When there is too much text to fit on the screen, use CTRL-S to suspend display and and CTRL-Q to resume it. To interrupt a command (as always), type CTRL-C CTRL-A.

Why Use It?

TYPE is the fastest and easiest way to see what's in a file. You can use it on CLI macros and other text files.

Examples

```
type bye.mc ⤶
.
. (text of BYE.MC)
R

type learning:myfile ⤶
.
. (text of LEARNING:MYFILE)
R
```

These commands type the contents of two files on the terminal screen.

UNLINK

Removes a link entry

UNLINK link-entry-pathname

The UNLINK command removes a link entry. It has no effect on the resolution file.

Template characters are allowed; for example, UNLINK -.SV is valid.

Why Use It?

UNLINK is the *only command* that removes a link entry. If you use DELETE, the system will leave the link intact but delete the *resolution file*. You would use UNLINK whenever you wanted to remove (perhaps to recreate) a link.

/C	Confirm each link removal. The system displays each matching link name on the screen. If you press ⤶, it removes the link. If you press any other character, the link remains.
/V	Verify each link removed by displaying its name.

Example

```
unlink/c/v    -.sv ⤶
IMOVE.SV $
MYPROG.SV ⤶
DELETED MYPROG.SV
NONENTITY.SV ⤶
DELETED NONENTITY.SV
BOOMER.SV X
R
```

This example shows a sequence in which the person at the terminal chose to keep two link entries: IMOVE.SV and BOOMER.SV.

XFER

Copies the contents of a file into another file

`XFER source-file destination-file`

The XFER command can the contents of a file on any device to another file on any device.

If the destination file is a disk file and you omit switches, XFER tries to create the file. To add a file to an existing disk file, use the /B switch. You can use pathnames for either source or destination file.

Why Use It?

XFER allows you to create a file containing text — useful for macros and other shortcuts. If you did not acquire a text editor (SPEED) with your system, this is the *only* easy way to create a file of text. You can use the MESSAGE command to insert text for later display.

XFER also allows you to add text to a file or build one large file from smaller ones. And, XFER can duplicate a file in the same directory easily. Unlike the MOVE command, it copies the *contents* of a file, without the name, creation date, and so on.

Switches

/A	ASCII transfer, needed to type text from the terminal into a file.
/B	Appends the source-file to the end of existing file. You must use this switch to add material to an existing disk file.
/R	Random organization for the destination file. Random organization is required for program (.SV) files. (Also, if you use XFER to copy an .SV file on disk, you must give it the S attribute — CHATR command — afterward.)

Example

```
xfer/a    $tti    today.mc ↲
message Today's files are ↲
list/e/s/k    %date%/a ↲
↓             (Press the downarrow key to end text insert.)
R
```

This example shows creation of a macro, TODAY.MC, which can be executed via TODAY↲. TODAY.MC displays TODAY'S FILES ARE followed by an alphabetical list of the filenames created or modified today in the current directory. The %date% is a CLI variable, designed for CLI macros. Used within a command, variables expand to a specific value; %date% expands to the system date.

The CLI session in Chapter 4 gives other examples of text transfer via XFER.

What Next?

This chapter is really a reference chapter. By command, it describes most of the information you need to make good use of the CLI.

You may want to proceed to format diskettes (next chapter), backup (Chapter 7), or a text editor (Chapter 9).

Formatting and Copying Diskettes 6

Read this chapter when

- you want some pointers on handling diskettes;
- you want to hardware format a diskette;
- you want to software format a diskette, to make it into a DG/RDOS directory or system diskette;
- you want to copy a diskette.

This chapter gives details on handling and formatting diskettes, making them into DG/RDOS directories, and copying them. The major sections proceed

Needs for Various Applications

Table 6-1 describes the diskette formatting operations you need for different operations.

Table 6-1 What a diskette needs for various operations

Operation	Hardware Format*	Make into DG/RDOS Directory	Make into DG/RDOS System Diskette
DIR to it	Y	Y	
INIT it	Y	Y	
Use for backup (with DDUMP utility or MOVE command)	Y	Y	
Use for backup (with IMOVE, FCOPY, Customer Diagnostics, or DUMP command)	Y		
Use for MS-DOS or CP-M/86 storage and access	Y		
Use to store and access individual files and directories	Y	Y	
Start and run DG/RDOS from it	Y	Y	Y

**Diskettes you acquire from DG are already hardware formatted.*

How To Tell What a Diskette Needs

Generally, you should use the diskette label to record the status of a diskette. Thus, if it has no label and is a DG diskette, it's hardware formatted but nothing else. If it has no label and isn't a DG diskette, it needs hardware formatting. If there is a label, the label tells all.

But, you can tell roughly what a diskette needs by inserting it in a drive and trying to INIT it from DG/RDOS (for example, INIT DJ1 ↵). When you do this, the system will display a message as follows:

Error message FILE DATA ERROR generally means the diskette needs hardware formatting.

Error message DISK FORMAT ERROR generally means the diskette is hardware formatted but not software formatted.

Error message SYS.DR ERROR generally means the diskette is hardware formatted but not software formatted, and it has been written to by a program like IMOVE. It may be a backup diskette.

Error message FILE DOES NOT EXIST generally means that the diskette is software formatted but the INIT/F command hasn't been typed to it.

No error message and the CLI R prompt means that the diskette was software formatted, had INIT/F typed to it, and is a DG/RDOS directory.

Handling and Storing Diskettes

Diskettes are relatively fragile — thin plastic, protected only by paper inner and outer envelopes. Some handling cautions and hints follow.

- Store diskettes in their outer envelopes. Remove a diskette from its outer envelope just before you use it.
- Hold a diskette by the edges of the inner envelope only. Avoid touching the diskette surface (exposed in oval sections on each side of the inner envelope). The oil on your finger could make that part of the diskette unreadable.
- If a diskette has no paper label, apply one. Properly labeled diskettes make life easier. Labels go on the smooth, seamless side of the inner envelope, at the top (with the write-enable notch to the right). *Avoid the oval cutout where the diskette surface is exposed.* Remove the sticky-backed label from its backing and apply it to inner envelope.
- To write on a diskette label, use only a felt-tipped pen. A pencil or ball-point pen can score the diskette surface — destroying some or all data on it.
- A diskette must be properly inserted — the system can't read one that's improperly inserted. When you insert a diskette, hold it by the edge, with label (seamless) side facing right and the write-enable notch up. The diskette should slide in smoothly and come to a firm stop.

- Diskettes wear. If, on a diskette you've used before, you see the message FILE DATA ERROR: name, this means file *name* is unreadable. Try releasing the diskette, reinserting it, and re-initializing it. If the error recurs, check for new bad blocks as described in Chapter 14. If you can recover from the error, move or back up all the files you care about to another diskette or the hard disk. The error probably means that the original diskette is wearing out.
- Don't bend or twist a diskette. A crease on the surface means data loss.
- Cold and heat can harm diskettes. Keep them temperate (between 10 and 50 degrees C, between 50 and 125 degrees F).
- A magnetic field can erase part or all data on a diskette. Keep them away from electric motors, magnets, and transformers.
- Don't turn computer power off while a diskette is inserted; remove it first. Turning off power while a diskette is inserted can lead to data loss.
- To write-protect a diskette, put tape over the write-protect notch (a quarter-inch slot on the edge of the inner envelope). A write-protected diskette cannot be written to: you will see an error message on attempts to write to such a diskette. The system can't initialize a write-protected diskette (INIT or DIR command) as a directory. Also, the Customer Diagnostics diskette won't work properly if you write-protect it; and the FCOPY program can't duplicate a write-protected diskette.

Hardware Formatting Diskettes (Customer Diagnostics)

The Customer Diagnostics diskette can hardware format diskettes and do many other things, like test system hardware and copy diskettes.

You can copy the Customer Diagnostic diskette using its own copy function *or* the FCOPY program, described later in this chapter.

To hardware format a diskette, follow these steps.

1. If DG/RDOS is running, shut it down (BYE ↵). If power is off, turn it on.

2. Type the break sequence (CMD and BREAK/ESC keys) to get the ! prompt.
3. Get the DG Customer Diagnostics diskette and insert it in drive DJ0.
4. Type `20H` to start from diskette.

   After about 20 seconds, the diagnostics program displays

   ```
   AMOUNT OF MEMORY FOUND: n KB

   CURRENT INVENTORY LIST
   .
   . (devices)

   IS THIS ... CORRECT (YES OR NO)?
   ```

5. If the list is not correct, you should fix it as described in the Customer Diagnostics chapter of your hardware *Testing* manual. (For the purpose of formatting diskettes, you can always answer YES ⤶). For example,

   ```
   YES ⤶
   ```

   Some 20 seconds pass; then it displays another menu:

   ```
   DG SERIES DIAGNOSTIC SYSTEM
   .
   .
   3 - DISPLAY DISKETTE MENU
   .
   ENTER NUMBER OF ACTION DESIRED:
   ```

6. Choose the number of the diskette menu. Here, it's

   ```
   3 ⤶
   ```

   A few seconds pass; then it displays the diskette menu:

   ```
   DISKETTE UTILITY MENU
   ========

   1. FORMAT DISKETTE
   2. FORMAT AND VERIFY DISKETTE
   .
   SELECT OPTION (1..n)
   ```

7. A plain format takes 30 seconds for a 360-Kbyte diskette. A format with verification (VERIFY) takes an additional 5 minutes. Generally, you should choose verification (VERIFY) — especially if you will use the diskette for data backup. For example,

   ```
   2 ↵
   ```

   And it displays a menu like this:

   ```
   DISKETTE FORMATTER
   ==================

   *** WARNING ***
   THIS PROGRAM DESTROYS ALL DISKETTE DATA

   Remove all diskettes from drives to prevent data loss.

   SELECT DRIVE (0,1)
   ```

8. Just to be prudent, remove the Customer Diagnostics diskette from drive DJ0. It needn't be inserted for you to *format* diskettes (although it is needed if you want other operations).
9. Insert the diskette you want to format in a drive, say DJ0.
10. Specify the diskette drive by typing 0 (for DJ0) or 1 (for DJ1). For example,

    ```
    0 ↵

    Select DG 9-sector or IBM 8-sector format (DG,IBM):
    ```

11. DG 9-sector format is required with any operating system except CP/M-86 (and DG format works even with CP/M-86). So you'll nearly always answer

    ```
    DG ↵
    ```

    The diskette formatter now tries to hardware format the diskette:

    ```
    FORMATTING DISKETTE IN DRIVE n
    ```

    (If it hits an error — like a misinserted diskette, or no diskette in the drive — it displays `UNIT n FATAL ERROR... and Hit any key to continue.)` Remove the diskette, reinsert it, and return to step 7 — skip step 9.)

    After 30 seconds or so, it says

```
FORMATTING FINISHED
```

And, if you selected the verify option, it displays a VERIFYING message and spends 5 minutes verifying the diskette. Then it displays

```
VERIFY FINISHED
n SOFT ERRORS
n PERMANENT ERRORS
```

and it returns to the diskette utility menu. Soft errors are read/write errors that disappeared on retry. Permanent errors indicate bad blocks. You can't use FCOPY with a diskette that has a permanent error, but you *can* use it as a DG/RDOS directory if you run DKINIT on it to remap the bad block(s).

12. Remove the newly-formatted diskette from the drive, apply a diskette label with something like "Hardware formatted" and the date written on it.

    To format another diskette, return to step 7.

The diskette(s) formatted are okay for backup via IMOVE or FCOPY. If you want any of them to be a DG/RDOS directory, continue to the next section. To copy, skip to the copy section.

Making a Diskette into a DG/RDOS Directory

There are two things involved: a software format (DKINIT FULL command), and an INIT/F command. The diskette(s) must be hardware formatted as described above; if not, DKINIT will fail, reporting address errors.

To make a diskette into a directory via DKINIT, follow these steps (bulleted to avoid confusion with the numbered steps above).

- Press the ALPHA LOCK key so you won't have to worry about upper- and lowercase.

- Start DKINIT. If DKINIT is on your system disk(ette) and DG/RDOS is running, type BOOT %MDIR%:DKINIT ⤶. Skip to DISK DRIVE MODEL NUMBER?

  If DKINIT is not on your system diskette, shut down DG/RDOS (if running). Type the break sequence (CMD and BREAK/ESC) to get the ! prompt. Get the Stand-Alone Utilities diskette and insert it in drive DJ0.
- Start up as usual (20H or 26).
- When it asks Filename?, type **DKINIT** ⤶

```
Filename?, DKINIT ⤶

DISK INITIALIZER REV n.nn
DISK DRIVE MODEL NUMBER?
```

- Put the diskette you want to format in a drive (remove the Stand-Alone Utilities diskette if needed).
- Type the model number of the diskette drive and press the CR key:

```
6268 CR          (You must use the CR key with DKINIT.)

DISK UNIT?
```

- Type the name of the drive that holds the diskette you want to format, and press CR. For example,

```
DJ1 CR

.
. (it confirms the disk type)

COMMAND?
```

- Type FULL CR

```
COMMAND DESTROYS ANY PREVIOUS DG/RDOS DISK STRUCTURE.
DG/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) ?
```

- Each pattern takes about 2 minutes on a 360-Kbyte diskette. You should run all patterns, since an undetected bad block can cause data loss. In any case, run at least two patterns. For example,

```
5 CR            (CR key)

*** PATTERN #1 (155555) ***
.
. (Delay while patterns run — be patient. DKINIT identifies
. bad blocks in the form DISK ERROR - BAD BLOCK=n.)
.
*** ALL PATTERNS RUN ***

DO YOU WISH TO DECLARE ANY BLOCKS BAD
```

- `THAT ARE NOT ALREADY IN THE BAD BLOCK TABLE? NO CR` (CR key)

```
DEFAULT REMAP AREA IS n BLOCKS LONG
.
```

- `REMAP AREA SIZE (TYPE RETURN FOR DEFAULT)? CR` (CR key)
- `REMAP AREA START BLOCK NUMBER (TYPE RETURN FOR DEFAULT)? CR`

```
DEFAULT FRAME SIZE IS n
.
```

- `DISK FRAME SIZE (TYPE RETURN FOR DEFAULT)? CR`

```
FULL DISK INIT COMPLETE
COMMAND?
```

- If you want to software format another diskette, type `FULL ↵` and return to the NUMBER OF PATTERNS question. Repeat this whole cycle until you've software formatted all the diskettes you want.
- When DKINIT asks COMMAND, type `STOP CR`.
- For a diskette-based system, insert your system diskette in DJ0 (remove the DKINIT diskette from DJ0 first, if needed).
- Bring up DG/RDOS: `26H` or `20H`, then date and time.
- The next phase is to type INIT/F to each diskette you software formatted. Assuming there is a newly-formatted diskette in a drive from DKINIT, type INIT/F to that drive, and confirm. For example,

```
INIT/F    DJ1 ↵
```

- *CONFIRM?* Y *ES* (Press Y to confirm)

  R

  The diskette is now a bona fide DG/RDOS directory. Release it by typing **RELEASE DJn ⤶**.

  Remove the diskette from the drive and write something like "DG/RDOS directory (DKINIT)", and the date, on the label.

- For every diskette on which you ran DKINIT FULL, insert the diskette and repeat the previous two steps. (INIT/F, confirm, release, remove from drive, and update the label).

Congratulations. You now have one or more diskettes to use as DG/RDOS directories: just insert, INIT (or DIR to), and move files to and from as needed. When you're done, release the diskette.

Making a Diskette into a DG/RDOS System Diskette

Using system diskettes for backup extends the versatility of your backups; you can start DG/RDOS and run it from this diskette just as if it were your original. The system files will consume over 300 disk blocks — space lost to normal data storage.

A system diskette is a diskette formatted as a DG/RDOS directory (above), with a bootstrap root (installed by the MBOOT program) *and* the files shown in Table 6-2.

Table 6-2 DG/RDOS system files needed on a system diskette

Mandatory	**Optional (highly recommended)**
BYE.MC (shutdown macro)	CONFIG.SV (system configuration program)
CLI.SV (CLI)	
DGRDOS.SV (system)	FCOPY.SV (diskette copy program)
LOADEM.SV (emulator loader)	IMOVE.SV (backup program)
MBOOT.SV (bootstrap program)	
-.TX (template for emulator file)	
-.EM (template for emulator link)	
SYS.SV (link entry to system file)	

To make a system diskette from a diskette formatted as a directory, follow these steps:

1. Get a diskette formatted as a DG/RDOS directory and insert it in a drive (for example, DJ1).
2. Start MBOOT. If DG/RDOS is running, type `%MDIR%:BYE ↵`. Otherwise, start up as usual (`20H` or `26H`).
3. When it asks Filename?, type `MBOOT ↵` :
4. *Filename?* `MBOOT ↵`

   ```
   BOOT REV n.nn
   Bootstrap Device Specifier?
   ```

5. Type the diskette drive name; for example, `DJ1 ↵`.
6. *Install Bootstrap (Y OR N)?* `Y` (Press Y)

   ```
   Done.
   Bootstrap Device Specifier?
   ```

7. Type the diskette drive again; for example `DJ1 ↵`
8. *Install Bootstrap (Y OR N)?* `N` (Press N)
9. *Filename?* `disk:SYS ↵` (Type the master disk name and :SYS; for example, `DE0:SYS ↵` or `DJ0:SYS ↵`.)

   ```
   DG/RDOS REV x.xx
   ```

10. *DATE (M/D/Y)?* `11 22 84 ↵` (Current date)
11. *TIME (H:M:S)?* `14 10 ↵` (Current time, 24-hour clock)

    ```
    WELCOME TO THE DG DESKTOP GENERATION

    .
    R
    ```

12. Initialize the diskette; for example,

    ```
    init dj1 ↵              (Or dj0 ↵)
    R
    ```

13. Now to move some system files onto the diskette. Choose from Table 6-2, earlier. For example, for a diskette in DJ1, you might type

    ```
    move/v    dj1    bye.mc    cli.sv    config.sv    dgrdos.sv    fcopy.sv ↵
    ```

    . (verifies)
    R

    ```
    move/v    dj1    loadem.sv    mboot.sv    -.tx    -.em    sys.sv ↵
    ```

    . (verifies)
    R

14. Shut down and test the newly-created system diskette:

    ```
    bye ↵
    STARTING SYSTEM SHUTDOWN
    MASTER DEVICE RELEASED

    Filename?
    ```

15. If the newly-created system diskette isn't in drive DJ0, move it there.
16. Type the break sequence (CMD and BREAK/ESC).
17. `! 20H` (Type 20H)
18. *Filename?* ↵
19. *DATE (D/M/Y)?* `11 22 84 ↵` (Type current date.)
20. *TIME (H:M:S)?* `14 15 ↵` (Type current time.)
    WELCOME TO THE DG DESKTOP GENERATION
    R
21. For disk space check and listing, type

    ```
    disk ↵
    .
    list/s/e ↵
    ```

    . (listing)
    R

22. Congratulations. Your newly-created system diskette is up and running. You can shut it down (BYE ↵) and store it safely.

Copying a Diskette

There are two ways to do this. They are the FCOPY diskette copy program (which also allows you to copy *files*), and the Customer Diagnostics Copy function.

Both programs require hardware-formatted (but not software formatted) diskettes.

The FCOPY program runs under DG/RDOS (just like any other program), and it can copy using only one diskette drive. The Customer Diagnostics copy function is faster, but it requires two diskette drives; and DG/RDOS must be shut down. You can start the Customer Diagnostics copy the same way you hardware format a diskette (above). Select the COPY DISKETTE entry from the Diskette Utility Menu and the program will then lead you through the copy.

FCOPY can copy 9-sector (DG format) diskettes but cannot copy 8-sector (IBM format) diskettes. The Customer Diagnostics copy can copy a diskette that has either format.

Using FCOPY to Copy a Diskette

Program FCOPY.SV, supplied with your DG/RDOS system, can copy a hardware-formatted diskette, or file on diskette, using only one diskette drive. This makes FCOPY ideal for single-diskette systems without a hard disk, since FCOPY can produce backup diskettes *and* to copy a file from one diskette to another. FCOPY is also useful for any system, since it can run while DG/RDOS runs.

FCOPY is easy to use; just get to the master directory and type FCOPY ⏎. FCOPY then leads you through menus to the option you want. You can specify source and destination files with switches, or omit them and have FCOPY ask for them.

When it needs an answer, FCOPY displays a blinking prompt at the bottom of the screen. While it's working, it displays ***Reading...*** and ***Writing...*** status lines.

To duplicate a diskette using just one drive, FCOPY prompts you to insert the source diskette. Press ↲; remove the source and insert the destination diskette, and press ↲ again, and so on. The number of cycles depends on how much memory your system has and the amount of material on the diskette.

FCOPY keeps track of the source diskette and will prevent you from accidentally copying the destination diskette to the source.

The FCOPY command format is

```
FCOPY     [source-diskette destination-diskette]
FCOPY     [source-file destination-file]
```

FCOPY has the following switches:

/C Tells FCOPY that this is a file copy.

/D Tells FCOPY that this is a diskette duplication.

/V Tells FCOPY to verify the copy. Be sure to use this switch *or* choose the verify option in a menu.

If you omit needed arguments or switches, FCOPY will ask for them in screen menus.

FCOPY Sessions

The following sessions show FCOPY duplicating a diskette using one drive (DJ0), then using two drives (DJ0 and DJ1). If you have only one drive, try the first session. With two diskette drives, try the second session.

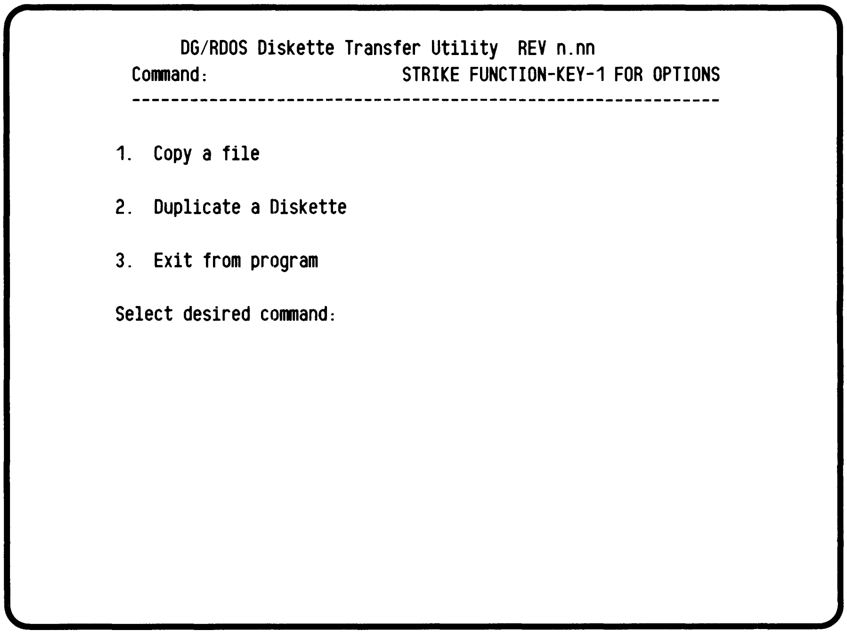
For either session, you need a source diskette (try a DG/RDOS system diskette) and a blank, hardware formatted diskette. If FCOPY hits an error, check the message (as always) in Chapter 14, the error chapter.

Duplicating a Diskette Using One Drive

To duplicate a diskette using one drive, you'll need a source and destination diskette (described above). Get them and follow these steps:

1. DIR to the master directory and start FCOPY:

```
dir dj0 ↲          (or dir de0 ↲)
R
fcopy ↲            (Start FCOPY)
```



```
         DG/RDOS Diskette Transfer Utility  REV n.nn
     Command:                      STRIKE FUNCTION-KEY-1 FOR OPTIONS
     ------------------------------------------------------------

   1.  Copy a file

   2.  Duplicate a Diskette

   3.  Exit from program

   Select desired command:
```

2. This is the FCOPY Main Menu. To check options, press function key 1 (the left key in the top row).

   f1

   (1) Continue (2) Restart (3) Return to Main Menu
   Select desired option: 1

   Function key 1 tells the program to display some general choices. Do this whenever you're confused or you want to restart.

3. Notice that *Select desired option:* is blinking. It does this whenever FCOPY wants some human interaction (like a choice or a confirmation). The default answer is 1, Continue, which is what you want. So press ↵ to select the default option:

   ↵

   and the cursor returns to *Select desired option.*

4. You want to duplicate a diskette, so choose 2. Duplicate a diskette by typing

   ```
   2 ⮐
   ```

   It now displays another menu, with specific questions about diskette names or file names, and verification:

5. Specify the source and destination diskette names (DJ0 and DJ0) and choose verification:

   ```
   Enter source diskette name: dj0 ⮐
   Enter destination diskette name: dj0 ⮐

   Do you want verification of the duplicate? (Y/N) N
   ```

6. Always choose verification. If you don't choose it, FCOPY might not detect bad blocks on the destination diskette — which could result in an unusable copy. Type

   ```
   Y ⮐

   Writing time will be longer due to the verification option.

   Select CONTINUE if the pathnames above are correct.
   (1) Continue (2) Restart (3) Return to Main Menu
             Select desired operation: 1
   ```

   Here, FCOPY asks for confirmation, with blinking prompt.

7. Since the pathnames (diskette names) are correct, press ⮐ to say Continue. (You can restart FCOPY — when it's waiting for an answer — by pressing function key 1, then typing 2 ⮐.)

   ```
   Insert SOURCE diskette: dj0
           Select .....        (Prompt blinks)
   ```

8. Insert the source diskette in drive DJ0.

9. Press ⮐

   ```
   Elapsed time min : sec
   Reading from source file...
   ```

   The prompt stops blinking; and FCOPY displays the elapsed time for this operation. Some time passes, then

   ```
   Insert DESTINATION diskette: dj0
           Select .....        (Prompt blinks)
   ```

10. Remove the source diskette from drive DJ0 and insert the destination diskette in DJ0. Be sure to insert it correctly.

11. Press ↵.

    ```
    Elapsed time min : sec
    Writing to destination file...
    ```

    And soon it says

    ```
    Insert SOURCE diskette: DJ0
    ```

    FCOPY has read, and written, as much material as will fit in memory.

12. FCOPY now takes you back to step 8 and through step 11 until it has read all material on the source diskette and written it to the destination diskette. This may take 7 or more read/write cycles.

    If you mix up the diskettes (which can easily happen), FCOPY displays *Incorrect diskette*; you can insert the correct diskette and press ↵ to recover. For other messages, see Chapter 14, the error chapter.

    Then the program says:

    ```
     Transfer and verification completed

    (1) Continue (2) Restart (3) Return to Main Menu
              Select...
    ```

13. To make another copy, remove the newly-copied diskette from drive DJ0. Press ↵; then return to step 7.

    To return to the Main Menu, type 3 ↵.

14. To exit from the program, type 3 ↵ into the Main Menu. FCOPY prompts you to insert the system diskette in its drive:

    ```
    INSERT SYSTEM DISK. STRIKE ANY KEY TO CONTINUE.
    ```

15. Remove the newly-duplicated diskette from DJ0 and insert the original source diskette in DJ0.

16. Press ↵

    After 20 or so seconds, the CLI prompt returns:

    ```
    R
    ```

    The diskette duplication is done.

17. To check it, type BYE ⤶. Then insert the newly-duplicated diskette in DJ0; and in response to the Filename? prompt, type DJ0:SYS ⤶, and see if DG/RDOS comes up on the copy. If so, type LIST/E/S. If this works, the copy is okay. Shut down (BYE⤶) and remove the copy from the drive.

Actually, you can shorten an FCOPY session somewhat by using switches in the command line (described in FCOPY, Chapter 5). For example, FCOPY/D/V DJ0 DJ0 would save a few steps. But the example shows the *whole* dialog.

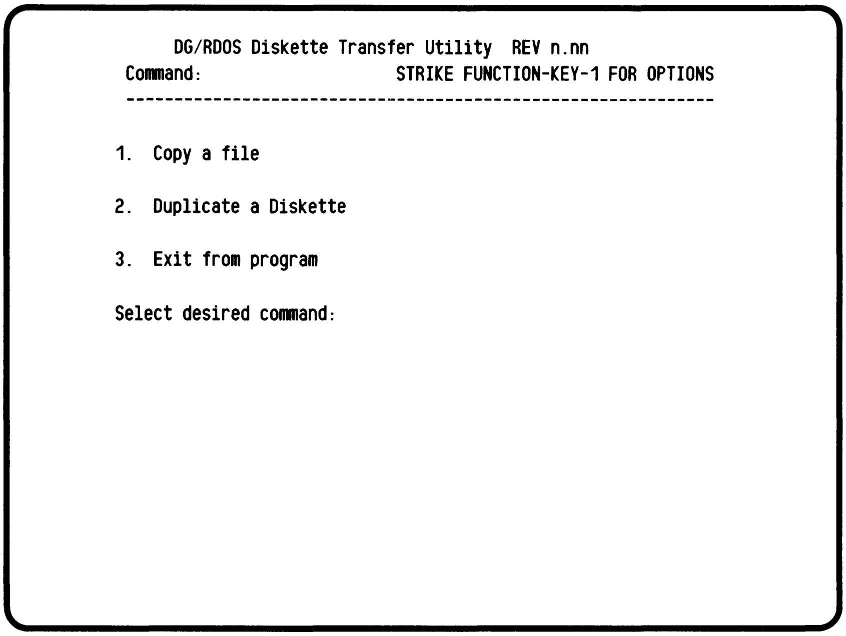
Skip the next section, unless you have two diskette drives.

Duplicating a Diskette Using Two Drives

To do this, you need two diskette drives. You also need a source and destination diskette. If none are handy, use a DG/RDOS system diskette and a blank, hardware formatted diskette. Then follow these steps:

1. DIR to the master directory and start FCOPY:

```
dir dj0 ⤶          (or dir de0 ⤶)
R
fcopy ⤶            (Start FCOPY)
```



```
     DG/RDOS Diskette Transfer Utility  REV n.nn
 Command:                       STRIKE FUNCTION-KEY-1 FOR OPTIONS
 -------------------------------------------------------------

1.  Copy a file

2.  Duplicate a Diskette

3.  Exit from program

Select desired command:
```

2. This is the FCOPY Main Menu. To check options, press function key 1 (the leftmost key in the topmost row of keys).

   ```
   f1

   (1) Continue (2) Restart (3) Return to Main Menu
           Select desired operation: 1
   ```

   Function key 1, pressed while FCOPY is waiting for an answer, tells FCOPY to display general choices. Press this key if you want to restart while FCOPY is asking a question.

3. Notice that *Select desired option:* is blinking. It does this whenever FCOPY wants some human interaction (like a choice or a confirmation). The default answer is 1, Continue, which is what you want. So press ⤶ to select the default option:

   ```
   ⤶
   ```

4. The cursor returns to *Select desired command.* You want to duplicate a diskette. So choose 2. Duplicate a diskette by typing

   ```
   2 ⤶
   ```

5. Specify the source and destination diskette names (DJ0 and DJ1) and choose verification:

   ```
   Enter source diskette name: dj0 ⤶
   Enter destination diskette name: dj1 ⤶

   Do you want verification of the duplicate? (Y/N) N
   ```

6. Always choose verification. If you don't choose it, FCOPY might not detect bad blocks on the destination diskette — which could result in an unusable copy. Type

   ```
   Y ⤶

   Writing time will be longer due to the verification option.

   Select CONTINUE if the pathnames above are correct.
   (1) Continue (2) Restart (3) Return to Main Menu
           Select desired operation: 1
   ```

   Once again, FCOPY asks for confirmation, with blinking prompt.

7. Check the source and destination diskette pathnames. If the names are wrong and/or you want to restart, type 2 ⤶. If the pathnames are correct, press ⤶ to say Continue.

   ```
     Insert SOURCE diskette: dj0
             Select .....          (Prompt blinks)
   ```

8. Insert the source diskette correctly into drive DJ0.
9. Insert the destination diskette correctly into drive DJ1.
10. Press ⮠.

    ```
    Elapsed time: min : sec
    Reading from source file...
    ```

    The prompt stops blinking; and FCOPY displays the elapsed time for this operation. Some time passes, then

    ```
    Insert DESTINATION diskette: dj0
              Select .....          (Prompt blinks)
    ```

11. Press ⮠.

    ```
    Elapsed time min : sec
    Writing to destination file...
    .
    Reading from source file...
    ```

    FCOPY now reads and writes in steps until it has read all material on the source diskette and written it to the destination diskette. (If it displays any messages about bad blocks, check the message in Chapter 14, the error chapter.)

    When done, FCOPY says

    ```
      Transfer and verification completed

    (1) Continue (2) Restart (3) Return to Main Menu
           Select...
    ```

12. To make another copy, insert the source diskette in drive DJ0 and press ⮠; then return to step 7.

    To return to the Main Menu, type 3⮠. After a delay, the CLI prompt returns:

    ```
    R
    ```

    The diskette duplication is done.

13. To check it, type BYE ⤶. Then insert the newly-duplicated diskette in DJ0; and in response the the Filename? prompt, type DJ0:SYS ⤶, and see if DG/RDOS comes up on the copy. If so, type LIST/E/S. If this works, the copy is okay. Shut down (BYE ⤶) and remove the copy from the drive.

Actually, you can shorten an FCOPY session somewhat by using switches in the command line (described in FCOPY, Chapter 5). For example, FCOPY/D/V DJ0 DJ1 would save a few steps. But the example shows the *whole* dialog.

This ends the FCOPY diskette duplicating sessions. To learn about copying a single file, try a session of your own. For a *file* copy, both source and destination diskettes must have a DG/RDOS directory structure.

What Next?

This chapter gave some pointers on handling diskettes, and explained how to hardware format diskettes, make them into DG/RDOS directories, make them into system diskettes, and copy them. Doing these things involved the Customer Diagnostics diskette, the DKINIT software formatter, the INIT/F and other CLI commands, and the FCOPY diskette copy program.

You might want to read more on backup procedures (next chapter) or programs in the DG/RDOS Development Kit (Chapter 9).

Backing Up and Restoring Files 7

Read this chapter when

- you want to plan backup procedures;
- you want to back up your files for safekeeping;
- you want to restore material from the backup diskettes.

This chapter explains the whys and wherefores of disk backup — sometimes called archiving. The major sections are

- Why Do It?
- Scheduling Your Backups
- Backing up and Restoring Files with a Hard Disk
- Backing up Files on a Diskette-based System
- What Next?

Why Do It?

A lot of information — critically important, irreplaceable information — can be stored in a computer system. The disk(ette) that stores it is not invulnerable. Accidental (or malicious) deletions or mechanical failure could destroy information on it.

Backup procedures ensure that files — like documents, reports, letters, business data, and programs — can be restored if lost.

If information is created and stored on your computer system, you need some procedure for file backup.

Planning and Scheduling Backups

The time your site spends on backups will vary with the rate that new information develops, and how important that new information is.

Basically, there are two variations of backup: full backup, which copies all files; and incremental backup, which copies all files that have changed since a certain date. When you want to restore, you restore the last incremental backup and proceed, latest to earliest, through the incremental backups until the file(s) you want have been restored. If this does not restore the desired file(s), you then restore from the full backup.

If your system has a hard disk, a full backup of the entire disk may fill scores of diskettes and take hours. You'll probably use incremental backups more often than full backups. You may want to organize the disk so that only one or two directories contain files that change often. At minimum, with a hard disk, we recommend a full backup each month and an incremental backup each week. This way, in the worst case, you can't lose more than a week's work.

On a system without a hard disk, incremental backups are usually not needed. Since all material fits on one diskette, it's easier to copy it all, whether or not it has changed, than to keep track of incremental backup sets. At minimum, with a diskette-only system, we recommend that you copy diskettes with important new material each week. This way, in the worst case, you can't lose more than a week's work.

You may want to do backups more often, based on the amount of work you can afford to lose. For example, if losing more than a day's work would be disastrous, you should back up every day — perhaps doing *only* one crucial directory.

You *can* base your backups on the development of new information; for example, do backups often during periods of high activity. But, like any other relatively boring task, backups are more likely to happen if they are done at regular intervals.

Backing up and Restoring Files with a Hard Disk

A system with a hard disk has a lot of storage space — it may even have multiple users — and new information may develop on it rapidly. Backup procedures with a hard disk require some planning — depending on the medium you'll use.

If You Have a Tape Drive

On the optional cartridge tape drive, each cartridge holds 15 Mbytes. One cartridge can hold all files on a full 15-Mbyte disk or a substantial percentage of all files on a 39-Mbyte disk.

On a system with a tape drive, you may want to use the YBURST utility (which is significantly faster than other backup programs) instead of IMOVE. Or, you can use the DUMP command, which has date switches that make incremental backups easy. To use YBURST, see the *RDOS/DOS Backup Utilities* manual. If you use DUMP, you'll restore material with the LOAD command. Details on DUMP and LOAD appear in the *RDOS/DOS CLI Manual.*

With tape, the backup and restore procedures covered here apply in principle, but you dump to and restore from a tape file (MT0:0, MT0:1, MT0:2, and so on) instead of to and from diskette.

If you don't have a tape drive, you'll need to use diskettes for backup — via one of the programs supplied with DG/RDOS for that purpose.

Which Backup Program Should I Use?

With DG/RDOS, you received several programs to back up disk(ette)-based information. With a hard disk, we recommend the program named IMOVE, for the following reasons:

- You don't need to shut down DG/RDOS to run IMOVE;
- IMOVE does not require diskettes to be formatted into directories before use (allows you to skip the DKINIT Full command and INIT/F steps before using a diskette for backup);
- IMOVE uses diskette space efficiently;
- IMOVE has a Help feature, which you can tap by typing IMOVE/H;
- If desired, IMOVE can convert files for use on other DG operating systems (AOS, AOS/VS, MP/AOS, MP/AOS-SU); thus it may benefit you to be familiar with it.

Other backup programs are

DDUMP.SV, DLOAD.SV	The directory dump and directory load programs provide fast backup to — and restoration from — multiple diskettes. Each diskette must have been full-formatted with DKINIT.
FDUMP.SV, FLOAD.SV	These programs dump and restore using mag tape only.
YDBURST.SV, YBURST.SV	These programs dump and restore from multiple diskettes (YDBURST) or tape (YBURST). They are fast, but require that DG/RDOS be shut down. If you have a tape unit, you might want to consider YBURST, since it's *much* faster than any other tape method.

Details on these programs appear in the manual, *RDOS, DOS, DG/RDOS Backup and Move Utilities.*

Full Versus Incremental Backups

A full backup of all nonsystem files — for one fairly full hard disk — can take hours and use many diskettes. An incremental backup copies new and changed files — and may take only one or two diskettes and a few minutes. Restoring material from both isn't difficult: you restore each incremental set (latest to earliest), then the full backup set.

If you do few full backups and many incremental backups, you'll spend less time doing backups and more time restoring files (for example, restoring 12 incremental sets and 1 full set). Also, you'll need to keep track of more backup sets. Another disadvantage is that *all* files that were ever backed up are restored from the backup diskettes. This means that, if you have to restore a whole disk, someone will need to go through the directories and delete all the old files that were deleted since the last full backup. (The computer can't tell the difference between files that were deleted intentionally and files deleted accidentally.)

On the other hand, if you do nothing but full backups, restoration is easy: you don't need to maintain or restore any incremental backups, and little cleanup is needed if you have to restore a whole disk. But, the amount of diskettes and time involved may be unacceptable. Generally, a good compromise is one full backup followed by from four to seven incremental backups.

If important new files are written to only one or two directories, you might consider doing regular backups (incremental or full) only of these directories. After all, you can always restore DG-supplied software from DG-supplied diskettes — with some extra effort.

Backup Sets of Diskettes

A 39-Mbyte hard disk can hold over 80,000 disk blocks; a 15-Mbyte hard disk can hold about 30,000 disk blocks. A 360-Kbyte diskette can hold about 720 disk blocks. Over a hundred diskettes might be needed for a full hard disk. You'll need few diskettes to start because your disk will be nearly empty. To get the approximate number of diskettes needed, type DIR DE0;DISK ↵ and divide the USED number by 720 (with 360-Kbyte diskettes).

Assume two to five diskettes for each incremental backup, and multiply this number by the number of incremental backups you plan between full backups. Add the total incremental backup number to the full backup number. This gives the approximate number of diskettes needed for one backup set.

You *can* get along with one backup set — but ideally, you should have alternate sets. This allows you to keep the last backup set(s) intact, and use the *oldest* backup set for the new backup.

Backup Pointers

Generally, to have backups work and be restorable, and to keep them as short as possible, you should observe the following guidelines:

- Avoid backing up material you don't *need* to back up. For example, you already have the DG/RDOS system files on the system backup diskette you made in Chapter 2 — you don't need to copy them even in a full backup. This is true of most of the DG-supplied software you'll put on the hard disk. You can build full and incremental files that tell the IMOVE program to skip system filenames (shown below in "Full Backup Example" and "Incremental Backup Example").

- Start each backup set from the same directory. Ideally, all backups would start from the master directory — copying all files on the disk, and allowing them to be restored very simply. But this may be impractical. If you do back up from different directories — based on activity — keep the diskette sets in one place, and label the diskettes well. It can be complicated to restore a directory using backups started from different directories.

  Users can back up at will from any directories they create, but this is no substitute for a general backup. (For example, some products create files in their own directories, and these files may not be backed up if the backup starts from a user's personal directory.)

- Do backups when the system is idle. If any file is open when the backup program tries to copy it, the file *won't be copied.* According to Murphy's Law, such files will be ones you want to restore, and they won't have been backed up. You can avoid this by having the foreground program (if any) shut down, and doing the backup from the background CLI.

- Be sure to have enough diskettes on hand. If you run out of them and can't complete a full backup normally, you must abort it; this means that the backup will be incomplete. To make it complete, you'll need to start again from the beginning with enough diskettes.

- Don't write-protect diskettes. If a diskette is write-protected, it cannot be written to, and the backup will stop until you insert a write-enabled diskette.

- Don't use questionable diskettes for a full backup. If a diskette has many bad blocks, the IMOVE program will abort with a "Too many bad sectors" error message. You will need to restart the whole backup operation. This can be very frustrating near the end of a full backup. If it happens, be sure to replace this bad diskette with a good one — and label the new diskette.

- You can abort a backup run with CTRL-C CTRL-A, but don't do it unless you must. If stopped for any reason, even a power failure or fatal error, a backup is incomplete — and it must be started all over again. Don't let anyone cut computer power (tantamount to a power failure) during a backup. This is *always* a no-no when DG/RDOS is running.

- Allow enough time. Each 360-Kbyte diskette takes roughly a minute to fill; then you must change it.

- It's critically important to label each diskette in a backup set, and to store diskette sets in an organized way. You'll be reusing diskettes for later backups, and if sets get mixed up, you could lose data.
- Make sure the sequence of each diskette is maintained. The first time you do a full backup, write "FULL BACKUP" or equivalent, the original current directory, and the sequence number (1, 2, 3, ...) on the diskette label after each diskette is filled. You might also write the date. Use only a felt-tipped pen to write on a diskette label. If a diskette has no paper label, apply one to the seamless side of the inner envelope, then write on it as above.

  For incremental backup sets, write "INC BACKUP" or equivalent, the current directory name, the sequence number, *and the date* on each label. The date is especially important if there will be multiple incremental backups between full backups.
- Follow any backup instructions supplied with your applications software. Some products require specific backup and restoration procedures. If so, it's essential to follow these procedures.

Streamlining Full Backups

Full backups can consume many diskettes. One way to minimize the number, and shorten backup time, is to omit system files from the backup. You can always restore them, if needed, from the system diskette(s). Filename templates allow you to skip nearly all system files. For example, take the following commands:

```
dir de0 ↵
build/a    fbackup    -.-    -.sv/n    -.lm/n    -.ol/n    -.lb/n    n-d.sr/n ↵
imove/d/l    dj0    @fbackup@ ↵
```

BUILD builds a filename file (FBACKUP), that includes all filenames *except* program (.sv) files and DG/RDOS system files, that are in directory DE0.

The IMOVE program doesn't allow templates. But, by building a file of desired filenames (as shown above), then enclosing the name filename with commercial at signs (@), you can employ templates indirectly. The backup shown above will include all files on the entire disk, except save and system files, which you can restore from DG-supplied diskettes.

This technique can eliminate several diskettes for each full backup (more if you have the Development Kit).

Incremental Backups

For an incremental backup, you must specify files by date. IMOVE doesn't have date switches, so you'll use the BUILD command — which *does* have date switches, to build a file of filenames. Then enclose the BUILD filename in commercial at signs (@name@) to back up all files named in the name file.

BUILD doesn't work with pathnames (directory:filename) into a subdirectory or secondary partition. To do an incremental backup of each of these, you can BUILD a separate file of names in the directory; then use IMOVE to produce a backup set of diskettes for *that directory.*

To minimize time and work, you may want to organize the disk so that only one or two directories hold the changed files. This will keep the number of backup sets small and manageable. Many DG products (like CS accounting) are organized to allow this approach.

Every other day, you might back up two volatile directories — say BOOMER and USERS — by date and/or other characters using the BUILD and IMOVE commands. Some examples follow.

```
dir boomer ⤶
R
build    b$backup    -.-    3-20-84/a ⤶
```

(The backup name file will hold names of all files that were create or modified on or after March 20, 1984. An easier way to do the date is shown later on.)

```
R
type b$backup ⤶
.
.
```

(Check the backup name file.)

```
R
list/e    @b$backup@ ⏎
```

(Check the names and creation dates of files to be backed up.)

```
.
.
R
```

(If the filenames and dates in the name file are okay, continue with the backup. If not, delete the name file and recreate it using a different BUILD command.)

```
imove/d/l    dj0    @b$backup@ ⏎
.
.
```

(Start the backup.)

You can eliminate a lot of typing in your incremental backup procedures by creating and using a macro, since so many of the commands are the same for each backup. The only really unique part of each backup is the date, which you can specify by easier method, shown later on.

Backup Example

For this example, assume that a DG/RDOS system is built, with other support software, in April 1984. Information starts accumulating on it immediately. The person acting as system manager decides on a monthly full backup and alternate daily backups of two directories: ACCTSDUE and USERS.

The full backup will occur on the last Friday of each month, from the master directory (DE0).

The April full backup takes only 10 diskettes. Incremental backups of ACCTSDUE average 3 diskettes and incremental backups of USERS average 4 diskettes.

In May, the full backup takes 15 diskettes. Incremental backups of ACCTSDUE and USERS directories stay the same — at 3 and 4 diskettes.

Now, we'll show the main actions and dialog at the system console for the June full backup (on Friday, June 29), and for the first following incremental backups (on Wednesday, July 4, allowing for Monday as the holiday).

Full Backup Example

1. Joan, the person who does backups, prepares to bring down the foreground by making sure that no users will lose work when this happens.

   Bringing down the foreground isn't mandatory, but it ensures that all files are closed, and it helps speed up the backup (slightly). (The foreground comments don't apply if you're not running a foreground program; check with FGND ↵.

2. She returns to the master directory, and terminates the foreground:

   ```
   dir de0 ↵
   R
   CTRL-C CTRL-F
   FG TERM
   ```

3. She sets up for the full backup by creating an indirect file — excluding all DG/RDOS system files. Generally, for a full backup, this need be done only once. The indirect file can be reused for subsequent full backups.

```
BUILD    fbackup    -.-    dgrdos.-/n    ^ ↵
cli.-/n    loadem.sv/n    mboot.sv/n    ^ ↵
config.sv/n    imove.sv/n    fcopy.sv/n    ^ ↵
sedit.sv/n    dkinit.sv/n    -.lm/n    ^ ↵
-.lb/n    n-d.sr/n ↵
```

(Builds indirect file FBACKUP. The BUILD command deletes any existing file with the same name first.)

```
R
type fbackup ↵
.
.
R
```

(Checks contents of indirect file.)

```
list @fbackup@ ↵
.
.
```

(Checks filenames and dates.)

```
R
```

(If the names look okay, she proceeds. Otherwise, she delete and recreates the indirect file.)

4. She starts the full backup, including the /L switch to specify a listing file of filenames copied. This is your decision, but we recommend it — the listing tells which files were backed up.

   If your system lacks a printer, type **CRAND $LPT⤶** to create a file to receive the filenames copied.

   ```
   imove/d/l    dj0    @fbackup@ ⤶
   ```

   Please insert disk 1, then press NEW LINE to continue.

5. She inserts the "FULL BACKUP" diskette labeled number 1 in the primary (right) unit.

   She presses ⤶.

   .
   . (it verifies files copied to diskette or printer... diskette fills up.)
   .

   This disk has been filled: DJ0
   Please insert disk 2, then press NEW LINE to continue.

6. She replaces the diskette with the next one and presses ⤶.

   She repeats this step as long as the program prompts for another diskette. For this example, let's say it goes to diskette number 13. Some files are copied to the diskette. Then:

   R

   The full backup is done. The listing file starts printing on the printer.

   Or, on a system without a printer, the listing file remains as file $LPT in the master directory. If the latter, it's important to find a way to print it, perhaps by moving it to another DG system (via diskette) and printing it. After it's been printed, someone should delete it, since it consumes a lot of disk space. However, before file $LPT has been moved and printed, it should be renamed to a name that indicates the date. For example, JUN2984.BU indicates June 29, 1984 — with .BU to indicate a back up listing.

   Each backup listing should be stored in the same place as the backup set of diskettes.

6. With the full backup done, Joan stores all diskettes safely, in order, in their outer covers, away from strong magnetic fields.

After a backup, DG/RDOS can be shut down or the foreground can be brought back via the EXFG command. In this example, she would probably shut down, turn everything off, and go home.

Incremental Backup Example At the sample site, the weekend passes — and the date for the next incremental backups arrives.

1. As last week, Joan prepares to bring the foreground down by making sure that no users will lose work.
2. She returns to the master directory and brings down the foreground:

```
dir de0 ↵
R
CTRL-C CTRL-F
FG TERM
```

3. If the system lacks a printer, she types

```
CRAND DE0:$LPT ↵
```

4. Now she sets up the *date* for the incremental backups. (You will probably want to automate this operation — and the backup name file handling that follows — by putting all that you can in a CLI macro.) Note that if you do backups every day, you can skip this step and just use %DATE%/A in the BUILD command that creates the file of filenames.)

`gtod ↵`	(Gets date.)
`16:57:34 07/04/84`	
`R`	
`delete date$old ↵`	(Deletes the obsolete backup date file.)
`R`	
`rename date date$old ↵`	(Renames the last backup date file. This may be important, since it gives the date of the previous backup.)
`R`	
`xfer/a    $tti    date ↵` `06-30-84^ ↵`	(Creates a new date file, with the cutoff date for the backup. Here, the last backup was done at day's end on June 29, so June 30 is the proper start date for the backups. The caret line-continuation character after the date is essential.)
`↓`	(Presses the downarrow key.)
`R`	

5. After putting the date in a file, she goes to a specific directory (ACCTSDUE) and does one incremental backup:

```
dir acctsdue )
R
delete ibackup$old )                          (Deletes the obsolete incre-
                                              mental backup name file.)

R
rename ibackup ibackup$old )                  (Saves the last incremental
                                              backup name file.)
R                                             (Builds a backup file using
build    ibackup    -.-    @de0:date@/a )     date stored in the master
                                              directory DATE file. As
                                              mentioned above, if you
                                              do backups every day, you
                                              can just use %DATE%/A
                                              instead of
                                              @DE0:DATE@/A.)

R
type ibackup )                                (Checks the contents of the
.                                             backup name file.)
.
R                                             (Checks names and cre-
list/e    @ibackup@ )                         ation dates of files to be
.                                             backed up.)
.
.                                             (The backup file is okay.)
R                                             (Starts the incremental
imove/d/l    dj0    @ibackup@ )               backup.)
.
.                                             (Feeds diskettes into DJ0
.                                             on prompt, labeling each
                                              one with the date.)

R
```

This incremental backup is done. She dismounts the diskette and stores all these diskettes as part of the ACCTSDUE diskettes.)

6. With the first backup (ACCTSDUE) done, she proceeds to do directory USERS.

   She inserts the first diskette in the USERS set in DJ0.

`dir de0:users ↵`	(Directory to back up.)
R `delete ibackup$old ↵`	(Deletes the obsolete backup name file.)
R `rename ibackup ibackup$old ↵`	(Renames the last backup name file.)
R `build    ibackup    -.-    @de0:date@/a ↵`	(Builds the indirect name file, based on date. As mentioned above, for daily back-ups, you would use %DATE%/A instead of @DE0:DATE@/A.
R `type ibackup ↵` . .	(Checks filenames...)
R `list/e    @ibackup@ ↵` .	(Checks filenames and dates.)
R	(They look okay...)
`imove/d/1    dj0    @ibackup@ ↵`	(Starts backup of this directory.)
. .	(Feeds diskettes into DJ0, labeling each.)
R	

With the second directory — USERS — incremental backup done, she removes the last diskette and stores the diskettes in this set.)

7. With a printer, the listing file prints. Without a printer, she renames the $LPT file to the date (04JUL84.BU or something) and plans to take it to a system with a printer.

After two more days, Joan does another incremental backup. The steps and diskette sets are the same.

As you can see, the procedure is methodical — and repetitive. But it can be extremely important. Some of the tedium could be eliminated by putting commands and prompts into CLI macros.

Chapter 5 gives syntactical and practical details on the BUILD command and IMOVE program.

Restoring Material from Diskette

Restoring falls into two categories: restoring one or more files, and restoring an entire hard disk. The first category is the most common, easiest, and fastest.

Restoring One or More Files

Usually, you need do this when someone has accidentally deleted a file (directory), or group of files. Perhaps someone was careless with DELETE and a template character — or, for whatever reason, you want to restore files from backup diskette set.

Restoring a System File

If the file(s) deleted or damaged is a DG/RDOS system file, and the system is still running, get out the DG/RDOS system backup diskette (made in Chapter 2), insert it in a drive, DIR to it, delete the bad file, and MOVE a good copy of the file to DE0. Then release, remove, and store your system backup diskette.

If DG/RDOS is not running and cannot run without the file, get out your DG/RDOS backup diskette as above and insert it in drive 0. Start from the diskette (20H); then, after DG/RDOS is up, delete the bad file, and MOVE the file to DE0. Next, DIR to DE0, and boot the DG/RDOS system on the disk. Then remove the diskette and store it safely.

Essential DG/RDOS system files include DGRDOS.SV, CLI.SV, and other files, described in Chapter 6, Table 6-2.

Restoring User Files

There are two things to consider when you want to restor a user file: the diskette set(s) needed, and the filename (directoryname) to restore.

The diskette set(s) you use to restore depend on the directory and date that the lost file(s) were created. (If the files were created since the last backup, then they cannot be restored.) Otherwise, use the backup that occurred soonest after the files were last changed or created (incremental or full).

If you can't determine the date the files were modified, check the backup listing or listing file (if any). If the name of a lost file appears in any listing, then you know the file is in that backup.

In the worst case, without a listing or dates, you will probably need to restore the entire last full backup — unless you know that the file(s) were backed up as part of a specific directory backup. If there are incremental backups that may hold the file, restore the earliest one first, then the next, then the next — and so on. This is a good reason to keep a backup listing — especially for incremental backups.

After you decide on the diskette set(s), choose your filename or directory name arguments to IMOVE. Filename templates are not allowed; neither are pathnames (directory:file).

If the backup contains one or more directories that you want to restore, you can save time and effort by specifying the directory name *with the .DR extension*. This will restore the directory (if it doesn't already exist) and any files in it that don't already exist.

Specifying one or more nondirectory filenames — as with directories — can also save time and effort.

Be sure about any directory and/or nondirectory names you specify — if you make a mistake, IMOVE will run through the entire backup without restoring the desired file.

You must restore in the same directory that the backup was started from. If a directory does not exist (perhaps it was accidentally deleted), take action as follows:

- If individual directory backups contain all the files you want to restore, then recreate the directory with the CDIR or CPART command, DIR to it, and restore using the individual backups.

- If individual directory backups don't contain all the files you want to restore, use the last full system backup and specify the name if the directory. Then use the incremental backups of the directory, latest to earliest, until you've restored them all. The steps are as follows:
  - Make DE0 the current directory.
  - Mount the first diskette of last full system backup.
  - Use IMOVE to restore the directory by typing

    ```
    IMOVE/F/D/L    DJ0    directory.DR⤶
    ```
  - Make the restored directory the current directory.
  - Mount the first diskette of last/next incremental backup of the restored directory.
  - Use IMOVE to restore the directory files:

    ```
    IMOVE/F/D/L/R    DJ0 ⤶
    ```
  - Repeat the last two steps for all incremental backups of the restored directory.

In nearly all cases, when you restore, use the IMOVE /R switch to ensure that later versions of files will replace their earlier versions.

File Restoration Examples

As an example, assume Jack accidentally deleted two files, named REPORT.JN and SUMMARY. He last changed REPORT.JN "about a week ago" and SUMMARY yesterday. The files were in directory USERS. Fortunately, USERS is backed up every other day.

First, you'd get out the USERS backup listing for about a week ago, and try to find the filename REPORT.JN there. And you'd look at the most recent backup listing for SUMMARY. (Possibly, you'd have Jack do the actual searching for the names.) If either name couldn't be found, you'd work backward through the listings until found it.

If you couldn't find either filename, you'd have to decide whether it was worthwhile to restore the last full system backup (IMOVE with a USERS.DR filename argument).

Assuming you found the filenames, each in its own individual USERS set, you'd follow these steps:

1. Get the diskette sets you need.
2. Warn other users that you are going to use the diskette drive (so no one will try to write to diskettes).
3. Mount the first diskette in the first set (say the earlier USERS backup) and type

```
dir de0:users ⤶
R                                    (Restore REPORT.JN) (Feed disk-
imove/f/d/v   dj0   report.jn ⤶     ettes as prompted.) (Success!)

R
.
.
.
REPORT.JN
R
```

4. Put the older USERS set away and insert the first diskette in the newer set (with SUMMARY).
5. Type

```
imove/f/d/v    dj0    summary ⤶   (Restore SUMMARY)
.
.                                  (Feed diskettes as prompted.)
.                                  (Restored.)
SUMMARY
R
```

6. Tell Jack to check the files and make sure they're okay.

If Jack had deleted a file in the master directory, assuming the backup structure above, you'd have had a choice: load from the last full system backup, with all its diskettes; or rebuild the file from memory; or — if the file(s) were of marginal value — forget them.

Restoring an Entire Hard Disk - When and How

The time may come when you need to restore all backed-up material to a hard disk. This time may have occurred when you type 26H and nothing happens, or if a DKINIT PARTIAL command aborts with the message TOO MANY DISK ERRORS TO COMPLETE.

If you know that you need to restore all material to a hard disk, follow these steps. The steps apply even if you have two hard disk units, and are rebuilding only one unit.

1. Make sure the disk(s) is connected properly — described in the hardware manual shipped with this one.
2. Get the most recent *DG/RDOS system* diskettes you received from DG. There should be four or five of these.
3. Return to Chapter 2 and execute all the numbered steps there.
4. Get your DG/RDOS system backup diskette (made in Chapter 2), with the configured system file. Insert it in DJ0, DIR to DJ0, and type

   ```
   MOVE/V/R    DE0    DGRDOS.SV ⤶
   ```
   (Or other system name, if you didn't configure the default system, DGRDOS.SV.)
5. Install other DG/RDOS software products from their diskettes (unless all of their files were included in the backups, which might be true).

   Then, if you have done backups, proceed as follows to restore user files. (If you haven't done backups, there's no point in reading further in this section.)
6. Get your last full backup set and all your incremental backup sets of diskettes. If no incremental backups were done since the last full backup, use only the full backup.
7. Insert the first diskette of the last full backup set in drive DJ0; and use IMOVE to restore it to the hard disk (IMOVE/F/D/L DJ0 ⤶). You'll need the listing to compare to the last backup listing.
8. Insert the first diskette of the *last* incremental or individual directory backups (if any were done) in DJ0. If the directory from which the backup was started differs from the master directory, DIR to it. Then restore using IMOVE (IMOVE/D/F/L/R DJ0 ⤶).

9. Repeat step 8 until you've done all the incremental (or individual directory) backups.

10. If you see a DISK SPACE EXHAUSTED: DEO message, abort the IMOVE run with CTRL-C CTRL-B. The message means that the hard disk is full. You (or other users) must delete some previously deleted and backed up files on the disk, to free some space, before this backup will fit.

    When you have some disk space free, say 500 to 1000 blocks or more (use DISK ⤶), retry the IMOVE run that produced the error.

11. Make sure you return all diskettes to their outer envelopes and store them safely.

12. You're done! You've recreated the entire hard disk — hopefully having lost a only a little work (the files created or changes made since the last backup occurred).

After you restore a whole disk, your next backup should be a full backup.

Backing up Files on a Diskette-based System

The easiest way to do this is to use the FCOPY utility. FCOPY works with either one or two diskette drives.

Each backup diskette will serve as a system diskette if need be. You can retain backup diskettes for a certain, given period — say 3 weeks — then reuse them for backup.

FCOPY also allows you to copy a *file* from a diskette. This is essential if — on a single diskette system — you want to copy any file from another diskette to your system diskette. Details on running FCOPY appear in Chapter 6.

What Next?

This chapter gave details on backup procedures: handling diskettes; planning; backing up and restoring with a hard disk; and backing up with a diskette-based system.

You might want to learn about hardware upgrades and software updates in the next chapter or computer languages in Chapter 10.

Adding New Software and Hardware 8

Read this chapter when

- you have just acquired a new device, like a USAM multiplexor, and installed it in your computer;
- you receive an update (new revision) of a software product you already have on your system;
- you have acquired a new software product, like COMPUCALC, ICOBOL, BASIC, or other computer language.

This chapter tells you how to handle new hardware and software acquired *after* DG/RDOS has been running for a while. The major sections are

- After Adding New Hardware
- New Revisions of DG Software
- Adding DG Software You Don't Have
- Software Created by Other Vendors
- Getting Help
- What Next?

After Adding New Hardware

Hardware upgrades are available for many desktop configurations. For example, you can add a communications multiplexor, terminals, or other hardware to many systems. Your DG sales representative can tell you more about this.

After you install your new hardware, and test it as described in the hardware documentation, you may need to reconfigure your DG/RDOS system to support the hardware. This is true if you add a USAM, an additional USAM terminal or foreground terminal, more memory, a cartridge tape, and other things. Review the CONFIG dialog in Chapter 2 to see if you need to rerun CONFIG. If you need to CONFIG, do it. (If you bought new software, you may want to reconfigure to best support it, as well.)

After installing, testing, and configuring (if needed) new hardware, you should install any new software received with the hardware as described in the next section.

If you add a new hard disk, you must format it with the DKINIT software formatter before you can use it. Follow the DKINIT and INIT/F steps in Chapter 2 (steps 9 through 20, and 33 and 34). The DG/RDOS device name of the first hard disk is DE0; the name of the second hard disk is DE1.

New Revisions of DG Software

DG continually improves its software products and updates its technical manuals. It sends the improved products and manuals to all customers on the optional Software Revision Service.

If you subscribe to the Software Revision Service, you will periodically get diskette sets and new manuals. Generally, you should install the new software.

DG/RDOS Revisions

Each DG/RDOS revision includes new versions of *all program and support files*, on a DG/RDOS diskette set. It also includes all pertinent DG/RDOS manuals, including a printed *Release Notice* with changes we haven't had time to include in the manual(s). DG/RDOS revisions proceed by tenths; for example, 1.10, 1.20, 1.30, and so on.

After receiving a new DG/RDOS revision, read the Release Notice to check for any special warnings. The Release Notice also names all files, pertinent manuals, and new features in the release. Each file in the release will replace the current version on your system disk(ette) (if there is a current version). This will bring *all* your DG/RDOS system files up one revision level.

The files shipped in revisions are designed to go in the master directory, DE0 or DJ0.

How you install a revision depends on whether you have a hard disk or a diskette-based system.

Installing a New Revision of DG/RDOS — Hard Disk Systems

To install a new revision of DG/RDOS on your hard disk, proceed as follows:

1. Rename the DG/RDOS system and CONFIG program to include their (old) revision names. You may need these programs in future, and they would be deleted and replaced by newer ones if you didn't rename them. For example, if your old revision was 1.09, type

```
dir de0 ⏎
R
rename dgrdos.sv dgrdos109.sv ⏎
R
rename config.sv config109.sv ⏎
R
```

2. If there's a diskette in drive DJ0, release it and remove it from the drive. Insert your new revision (for example, 1.10) System diskette in drive DJ0.

3. Make DJ0 the current directory, and move its files to the hard disk:

```
dir dj0 ⏎
R
move/v/r de0 ⏎
.
. (it verifies files moved)
.
R
release dj0 ⏎
R
```

4. If you received another diskette (Stand-Alone Utilities diskette), remove the diskette from unit 0 (right unit), and insert the Stand-alone Utilities diskette. Repeat the previous step.
5. Remove the diskette from DJ0. If you have a Model 20 or 30, skip to step 12.
6. Rename the old link entry file (the name ends in .EM) and the old emulator file (name ends in .TX). For example,

```
list    -.em ⤶
MTERM$.EM
R
rename    mterm$.em    mterm$109.em ⤶
R
list    -.tx ⤶
MD200.TX 10591 D
R
rename    md200.tx    md200$109.tx ⤶
R
```

7. Shut down DG/RDOS, turn power off, and power up again to have the new emulator loaded.

```
de0:bye ⤶

STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED
Filename?
```

Type the break sequence (CMD and BREAK/ESC keys).

```
!
```

Turn computer power off.

8. Turn power on and wait through the test sequence. Then bring up DG/RDOS from the hard disk:

```
! 26H

FILENAME? ↵

DG/RDOS REV x.xx                  (Notice the new revision
                                  number.)
DATE (M/D/Y)? 10 20 84 ↵          (Type the current date.)
TIME (H:M:S)? 13 ↵                (Type the current time.)

WELCOME TO THE DG DESKTOP GENERATION

EMULATOR FILE NOT FOUND: xTERM$.EM
DO YOU WISH TO LOAD THE EMULATOR (Y/N)?
```

9. You want to load the emulator, so type Y

```
INSERT EMULATOR DISKETTE IN DRIVE 0
TYPE C TO CONTINUE ...
```

10. Insert the D200 Emulator diskette (shipped with the DG/RDOS diskettes) in DJ0.

11. Type C

```
EMULATOR IS BEING LOADED ...

.
Emulator load completed.          (Note lowercase letters.)

xD211yyyy.TX is being copied to the master directory
R
```

12. To test the new DG/RDOS system, shut it down and turn power off:

```
de0:bye ↵

STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED
Filename?
```

Type the break sequence (CMD and BREAK/ESC keys).

```
!
```

Turn computer power off.

13. Turn power on and wait through the test sequence (if any). Then bring up DG/RDOS:

```
! 26H

Filename? ⤶

DG/RDOS REV x.xx                          (Notice the new revision
                                          number.)

DATE (M/D/Y)? 10 20 84 ⤶                  (Type the current date.)

TIME (H:M:S)? 13 ⤶                        (Type the current time.)

WELCOME TO THE DG DESKTOP GENERATION
```

On Model 10 or 10/SP, it says LOADING THE EMULATOR, then Emulator load completed.

```
R
```

At the R prompt, the new system is running. Continue to the next step.

(If R doesn't appear, or the system doesn't work, perhaps not all files were moved correctly. Type the break sequence, return to step 2.)

14. Try typing a few commands in lowercase letters:

```
disk ⤶
LEFT:n USED: n MAX. CONTIGUOUS: m
R
list/s/e ⤶
.
R
```

If you can type lowercase letters, and the LIST command works, the installation is okay. Remove any diskette from drive DJ0 and continue to the next step.

(If you can't type lowercase letters, this probably means you moved the wrong emulator, or no emulator, to the hard disk. Check the emulator files with LIST (LIST/E -.EM -.TX⤶). Delete the -.EM and -.TX files and return to step 9. If this doesn't help, then delete the new emulator and rename the old emulator to its original name (for example, MD200.TX). Shut down, turn power off, and restart. You'll run with the old emulator.

15. The next step is to run CONFIG on the new system. If you remember what your parameters were for the old system, fine. If you can't remember, start the old CONFIGxxx.SV (renamed above), specify the old system (DGRDOSxxx.SV), and check the values.

    Start the new CONFIG and specify a dialog file using the form

    ```
    CONFIG dialog-filename/V
    ```

    The dialog-filename/V switch tells CONFIG to record the parameters you choose (Y) in a dialog file, allowing you to to TYPE them easily for later revisions (instead of having to check by running the old CONFIG on the old DG/RDOS system). Including the date in the dialog filename can be quite helpful when you configure the system again later.

    For example,

    ```
    config dgrdos1123.cf/v ↵          (1123 represents November 23.)
    ```

    To CONFIG, specify the values you want for the system. These will be the same values as for the old system — unless you installed new hardware or software that requires a change. If you think a change might be needed, check the CONFIG dialog and tables in Chapter 2.

16. When you've configured your system, and had the changes made to the file, start it:

    ```
    boot sys ↵

    MASTER DEVICE RELEASED

    DG/RDOS REV x.xx
    DATE (M:D:Y)?
    ```

    And bring up DG/RDOS as always.

17. If you received the Development Kit, insert the diskette in DJ0 as above, and move (MOVE/V/R) the files into the directory that holds the previous revision's Development Kit programs — generally the master directory, DE0. It's important to install the new revision of Development Kit software, since an old development program might not work perfectly with a new revision of DG/RDOS.

    If you received the Sysgen files, install them in your Sysgen directory (created as in Chapter 2), as above.

You're done. You've installed the new revision of DG/RDOS. You need not reformat disks or diskettes for the new revision. (DKINIT produces revision-independent formats).

If, for any reason, you want to install an *old DG/RDOS revision*, get out old revision's diskettes and follow the procedure above. Use the MOVE/V/O command instead of the MOVE/V/R command to delete the same filename regardless of date.

As you can see, installing a revision isn't difficult. However, it does assume that you haven't moved DG/RDOS files into a user-created directory. If the DG/RDOS files aren't in the master directory, the new DG/RDOS files won't replace them.

Needless to say, keep all the system diskettes you receive from DG, and those you make yourself, in a safe place.

Installing a New Revision of DG/RDOS — Two Diskette Drive Systems

To create a system diskette with a new revision of DG/RDOS on it, you have a choice. You can build a system diskette as described in Chapter 2, or you can do it quickly using FCOPY as follows:

1. Get a blank or scratch diskette. If this has not been hardware formatted, format it as described in Chapter 6. Insert this diskette in drive DJ1.
2. Insert the DG-supplied system diskette (with the new revision) in drive DJ0.
3. Bring up the new revision of DG/RDOS (20H, ↵, date, time.)

4. Run the FCOPY program to duplicate the DG/RDOS diskette. For example:

   ```
   fcopy/d/v dj0 dj1 ⤶
   ```

   Run through the FCOPY duplicate sequence (details are in Chapter 6, "Copying a Diskette"). If FCOPY shows any bad blocks on the destination diskette, set the diskette aside and use another. (The diskette with bad blocks will serve for storage if you run DKINIT on it, later).

5. After FCOPY says Transfer completed, return to the CLI. Shut down DG/RDOS (BYE⤶).

6. Remove the diskette from DJ0 and store it safely. Move the diskette in DJ1 to DJ0 and start DG/RDOS from it (20H, ⤶, and so on).

7. When the R prompt appears, type a few commands, like LIST/E/S and DISK to check out the duplicate. If it doesn't work, put the DG-supplied DG/RDOS diskette in DJ0, the copy in DJ1, and return to step 1.

8. Using your memory and Tables 2-2 and 2-3 (Chapter 2), selectively delete all files you don't need from your new system diskette.

   If you want any non-DG/RDOS files from your old system diskette, get the old system diskette. Insert it in drive DJ1, DIR to it, and move the desired files to DJ0. Then release the DJ1 diskette and store it.

   If you have a model 20 or 30 system, skip to step 16.

9. Delete the old link entry file (the name ends in .EM) and the old emulator file (name ends in .TX). For example,

   ```
   list/s    -.em    -.tx
   MD200.TX 10591 D
   MTERM$.EM
   R
   delete/v    mterm$.em    md200.tx ⤶
   DELETED MTERM$.EM
   DELETED MD200.TX
   R
   ```

10. Shut down DG/RDOS, turn power off, and power up again to have the new emulator loaded.

```
dj0:bye ↵

STARTING SYSTEM SHUTDOWN
MASTER DEVICE RELEASED
Filename?
```

Type the break sequence (CMD and BREAK/ESC keys).

```
!
```

Turn computer power off.

11. Turn power on and wait through the test sequence. Then bring up DG/RDOS:

```
! 20H

FILENAME? ↵

DG/RDOS REV x.xx                    (Notice the new revision
                                    number.)
DATE (M/D/Y)? 10 20 84 ↵            (Type the current date.)
TIME (H:M:S)? 13 ↵                  (Type the current time.)

WELCOME TO THE DG DESKTOP GENERATION

EMULATOR FILE NOT FOUND: xTERM$.EM
DO YOU WISH TO LOAD THE EMULATOR (Y/N)?
```

12. You want to load the emulator, so type Y

```
INSERT EMULATOR DISKETTE IN DRIVE 0
TYPE C TO CONTINUE ...
```

13. Insert the D200 Emulator diskette (shipped with the DG/RDOS diskettes) in DJ0.

14. Type C

```
EMULATOR IS BEING LOADED ...

.
Emulator load completed.          (Note lowercase letters.)

Insert system diskette in drive 0
TYPE C TO CONTINUE...
```

15. Remove the diskette from drive DJ0. Insert your system diskette in drive DJ0. Type C.

    The program displays

    ```
    xD211yyyy.TX is being copied to the master directory
    R
    ```

16. To test the new DG/RDOS system, shut it down and turn power off:

    ```
    dj0:bye ⤶

    STARTING SYSTEM SHUTDOWN
    MASTER DEVICE RELEASED
    Filename?
    ```

    Type the break sequence (CMD and BREAK/ESC keys).

    ```
    !
    ```

    Turn computer power off.

17. Turn power on and wait through the test sequence (if any). Then bring up DG/RDOS:

    ```
    ! 20H

    Filename? ⤶

    DG/RDOS REV x.xx               (Notice the new revision
                                   number.)
    DATE (M/D/Y)? 10 20 84 ⤶       (Type the current date.)
    TIME (H:M:S)? 13 05 ⤶          (Type the current time.)

    WELCOME TO THE DG DESKTOP GENERATION
    ```

    On Model 10 or 10/SP, it says LOADING THE EMULATOR, then Emulator load completed.

    ```
    R
    ```

18. Try typing a few commands in lowercase letters:

```
disk ⤶
LEFT:n USED: n MAX. CONTIGUOUS: m
R
list/s/e ⤶
.
R
```

If you can type lowercase letters, and the LIST command works, the installation is okay. Remove any diskette from drive DJ0 and continue to the next step.

(If you can't type lowercase letters, this probably means you moved the wrong emulator, or no emulator, to the hard disk. Check the emulator files with LIST (LIST/E -.EM -.TX⤶). Delete the -.EM and -.TX files and return to step 9.

19. The next step is to run CONFIG on the new system. If you remember what your parameters were for the old system, fine. If you can't remember, get out the old system diskette, put it in DJ1, DIR to it, start the DJ1 CONFIG, and check the old system values.

Start the new CONFIG:

```
config ⤶
```

To CONFIG, specify the values you want for the system. These will be the same values as for the old system — unless you installed new hardware or software that requires a change. If you think a change might be needed, check the CONFIG dialog, and tables, in Chapter 2.

20. When you've configured your system, and had the changes made to the file, start it up:

```
boot sys ⤶

MASTER DEVICE RELEASED

DG/RDOS REV x.xx
DATE (M/D/Y)?
```

And bring up DG/RDOS as always.

You're done. You've created a new system diskette with the new revision of DG/RDOS.

If, for any reason, you want to run an *old DG/RDOS revision*, get out the old revision's system diskette and start DG/RDOS from it.

Needless to say, keep all the system diskettes you receive from DG, and those you make yourself, in a safe place.

New Revisions of Other DG Products

The software products you can acquire with DG/RDOS — like COMPUCALC, or ICOBOL — have a revision procedure similar to that of DG/RDOS. If you receive a new revision of any such product, install it, using the instructions on the product Release Notice. General guidelines are given in Chapter 2, "Installing other DG Products." Use the MOVE/R command, according to the product Release Notice.

Adding DG Software You Don't Have

If you acquire a DG product you don't have, install it as described on the product Release Notice. For some products, you may want (or need to) rerun CONFIG to change parameters for the product. Some guidelines for popular products are given in the CONFIG section of Chapter 2.

Software Created by Other Vendors

There may be periodic revisions to non-DG products, like MS-DOS, CP/M-86, and their dependent software. Such revisions often originate with the owner of the software, and DG simply configures the revised software for use on DESKTOP GENERATION systems. Such revisions are described on the product Release Notices.

Getting Help

If you acquired your DESKTOP GENERATION system from a authorized dealer, consult this dealer for help. If you *didn't* acquire your system from an authorized dealer, dial

1-800-DATAGEN

for the latest information on service and maintenance contracts.

What Next?

This chapter explained how to configure your system to hardware upgrades, new software, and new revisions to existing software.

Next, you might want to read about DG computer languages in Chapter 10.

Using the SPEED Text Editor and Other Development Kit Programs

9

Read this chapter when

- you want to create or edit text, using the SPEED Text Editor;
- you want an introduction to other programs in the DG/RDOS Development Kit, like the Macroassembler (MAC) or Relocatable Loader (RLDR).

The SPEED text editor program lets you write and edit text, including source program files. SPEED and the other development products are intended to help you write and build your own programs.

This chapter shows you how to use the SPEED text editor, in a working session; then it describes other development programs. The major sections proceed

- A Session with the SPEED Editor
- Other Development Programs

Before you can use SPEED or any other development program, the program should have been installed on the hard disk. Installation is easy: it requires only inserting the Development Kit diskette in DJ0, making it the current directory (DIR DJ0 ↓), moving all files to the hard disk (MOVE/V/R DE0 ↓), releasing it (RELEASE DJ0 ↓), and storing it. (The Development Kit programs can *run* from diskette — but at a high cost in performance.)

A Session with the SPEED Editor

This section shows you how to use the SPEED text editor in a working session. It explains enough SPEED features to let you use it. You can find a complete description in the *RDOS/DOS Superedit Text Editor User's Manual.* (Superedit is SPEED's formal name.)

SPEED Features

SPEED is *character string oriented.* This means that you insert and edit character sequences at the current character position.

To show the current character position, SPEED uses a *character pointer* (CP), displayed as a parenthesized caret (^). For example:

This is a line of (^)text.

Here, the CP appears between a space and the beginning of *text.*

To insert or edit text, place the CP where you want, then type the appropriate command(s). One SPEED command — I — serves to insert new text. Your I commands may include only one character or many lines of text.

SPEED Prompt and Delimiters

When SPEED is ready for a command, it displays an exclamation point (!), prompt. You then type a command and terminate it by pressing the BREAK/ESC key — the key alone, not with CMD.

The BREAK/ESC key echoes (displays) as $ when you press it. For example, if you type the command 20L and press BREAK/ESC twice, it displays as *20L$$.* When this chapter shows $, it means "press the BREAK/ESC key." To terminate a command string and tell SPEED to take action, type $ twice ($$).

In a change command (C), one $ (BREAK/ESC) character serves as a text delimiter. For example, the command

```
CHello$Bye$$
```

tells SPEED to change the next `Hello` to `Bye`.

A $ is also needed after a search or change command if you want to put other commands on the same line. For example, the command

```
CHello$Bye$LT$$
```

tells SPEED to change `Hello` to `Bye` and type the whole line.

As usual, with all commands, $$ terminates the command string. If you put multiple commands on a line, SPEED executes them sequentially, from left to right. If SPEED finds an error, it describes the error, then ignores the rest of the command line.

Edit Buffer

The *edit buffer* is an area of the computer's memory where SPEED works on your file during the editing process. The commands you type change the contents of the buffer; but the buffer isn't saved on disk until you type `UE` or `US`.

The edit buffer holds the entire current page (between form feed characters); if there are no form feeds in the file, the buffer holds the whole file.

SPEED Commands

The editing you will do in the SPEED session, and the commands you'll use, are

Act	**Command**
Edit a file.	SPEED filename
Insert text in the file.	Itext
Type line(s) of text.	nT
Move to the beginning of a line.	L
Jump to beginning of buffer.	J
Jump to end of buffer.	ZJ
Search for a character string.	Sstring
Change a character string to another string.	Cstring$string
Kill (delete) line(s) of text.	nK
Repeat (iterate) one or more commands.	<commands$;>
Update edit and halt. Saves the edit buffer (with all your changes) on disk and returns to the CLI. The old file is deleted.	UE$H
Update edit, save backup, and halt. Same as UE, but saves the old file as a backup file.	US$H

Special Keys and Control Sequences

Some special keys and control sequences that SPEED recognizes are

Key or Character	**What it Does**
BREAK/ESC	Delimits SPEED commands. Pressing this is shown here as $, which is the way it echoes on the screen. One $ separates commands; two $$ terminates a command string and tells SPEED to execute the command(s) in the string.
TAB key or CTRL-I	Produces a tab. Tabs are frequently used in the syntax of the FORTRAN and COBOL languages.
CTRL-S	Suspends display; useful for reading long files.
CTRL-Q	Continues display suspended by CTRL-S.
DEL	Deletes the preceding character. SPEED does not erase the character; instead, SPEED echoes the character erased. For example, if you typed `This` and pressed DEL four times to erase it, you'd see *`ThissihT`* on the screen. Then, type the text you really wanted.
⤶ (NEW LINE)	Ends a line of text and moves the cursor to the start of the next line.
CTRL-C CTRL-A	While you are typing, erases the line and returns the ! prompt. This is useful when you want to start typing from scratch. While SPEED is executing a command, CTRL-C CTRL-A interrupts the command and returns the ! prompt. This is useful when you want to stop a long operation, like a search.
CTRL-C CTRL-B	Aborts the current SPEED session without saving your edits — if any — on disk, and produces a break file (BREAK.SV) in the current directory. Generally, avoid this sequence.

Cursor and Case of Characters

When you are inserting text or typing a command, the cursor shows your position. The cursor is a box to the right of the ! prompt or the current character.

You can type SPEED commands in either UPPERCASE or lowercase letters, or both. The FORTRAN and COBOL compilers require statement keywords (like READ) in UPPERCASE; they accept either case in text strings. This session shows UPPERCASE for SPEED commands; it shows UPPERCASE and lowercase for the text itself.

If You Make Mistakes

As you proceed through the session, it's okay to make mistakes — everyone does, and they help teach. The main point is to learn the way SPEED works.

Although SPEED's error messages are terse, SPEED will usually protect your work from editing errors. Don't type Y or H as the first letter of a command (next to the ! prompt), though, or SPEED may terminate without saving your edits.

As with any program, CTRL-C CTRL-B will abort SPEED and return to the CLI. But CTRL-C CTRL-B nullifies all your edits as well, so use it only when you must. To stop a SPEED command without aborting SPEED, use CTRL-C CTRL-A.

Starting SPEED

To begin an editing session, get into the directory that holds SPEED. This is usually the master directory, but you can check for SPEED in any directory. DIR to the directory and type

```
list/a    speed.sv    speed.er ↵
```

After you've found the SPEED files, execute SPEED.SV. The command to do this has the form

```
SPEED filename
```

The filename includes the name of the file you want to edit. If the filename already exists, SPEED will copy it into the edit buffer. If the file does not exist, SPEED will create a new, empty file and you'll begin with an empty buffer. In the latter case, SPEED will display

```
CREATING NEW FILE
```

Note: If SPEED displays the message

```
EXPANDED ERROR TEXT NOT AVAILABLE
```

this means that SPEED can't find files CLI.ER and SPEED.ER, which it needs for error message text. Files CLI.ER and SPEED.ER were shipped with the Development Kit. CLI.ER *must* be in the master directory (DE0), and SPEED.ER must be *accessible* from the current directory (via the file itself or a link). Check with the LIST command. If you can't find CLI.ER in the master directory, or SPEED.ER isn't accessible in the current directory, get the Development Kit diskette, insert it, DIR to it, and move CLI.ER and SPEED.ER to the master directory. In any case — whenever you want to use SPEED from a nonmaster directory — DIR to this directory and create links as follows:

```
LINK    SPEED.SV    dir-name:SPEED.SV )
LINK    SPEED.ER    dir-name:SPEED.ER )
```

The dir-name is the name of the directory that holds the SPEED files — usually the master directory, DE0.

A Note of Caution

SPEED is a powerful editor — practically a text-processing language — but its power and speed can sometimes make life difficult for a new user. Until you learn it well, you should update your file often, saving a backup version with the US and H commands (US$H$$) to minimize lost effort.

If, at any time, text seems to have vanished from the edit buffer after a command, type US$H$$ to update the file and save the original (saved in filename.BU). Then, examine both the changed and original files with either SPEED or the CLI TYPE command and use the one you want.

Although SPEED's error messages are terse, SPEED will usually protect your work from editing errors. Be careful about starting a command with H or Y. This can happen if you forget the I to start an insert; for example, if you type `Hello..` instead of `!IHello..` When SPEED sees a command that begins with H or Y, it takes action that wipes out the edit buffer — which means that any changes you've made in this session are lost.

SPEED Session

To illustrate each SPEED command, we'll create and edit a prototype source program named MYPROG.XX. Make sure the ALPHA LOCK light is off. If it's lit, press the ALPHA LOCK key (to the right of the space bar) so that you can use lowercase letters.

Get to the desired directory (DIR) and type

```
speed myprog.xx ↵

CREATING NEW FILE           (If the file doesn't exist.)
!
```

You received the CREATING message because MYPROG.XX didn't exist before you ran SPEED.

As you type during this session, you will probably make typing mistakes. Use DEL to erase the wrong characters (SPEED echoes each one erased, instead of physically erasing it from the screen). Or, to erase a whole line you've typed, type CTRL-C CTRL-A.

Inserting New Text (I) and Typing Lines (T)

SPEED is displaying its ! prompt. Type the following, *precisely as shown,* with the boxes representing spaces, but don't include the ! prompt.

```
! IThis□is□source□line□1. ↵
Thisis□source□line□2.$$          (To delimit, press BREAK/ESC twice.)
!
```

You've inserted two lines of text. To see them, use the T command. A handy variation of T (#T) types all the lines in the buffer. Try it:

```
! #T$$
This is source line 1.
Thisis source line 2. !
```

If you use T alone, SPEED types the current line, showing the position of the character pointer (^). Try it:

```
! T$$
Thisis source line 2.(^)
!
```

The CP shows your position after the last inserted character. Since you didn't type ⤶ at the end of the second line, the CP is after the period in ..line 2. This lets you repeat insert commands in the same order that you would type words or lines of text on a typewriter. Just remember that the I command inserts text after the CP.

Now to add some text. Type

```
! I ⤶
This□is□source□line□3. ⤶
$$
```

Here, from the end of line 2, you inserted ⤶ and another line of text. Check them all again by typing #T and T. These commands — I and T — are commands that you'll use often.

T (Type) Command Variations

With SPEED, you'll use the T command extensively to see what's going on. There are several forms of T, including

#T	Types the entire buffer.
T	Types the current line and show the CP.
nT	Types n lines, including the current line.
0,nT	Types the buffer from its beginning to character n.

A reminder on the I command — Don't forget to type I before the text you want to insert. If you forget I, the text you type may be lost. Worse still, if the text begins with H or Y, you could lose *all* your edits. Forgetting I is a common mistake. As you gain experience with SPEED, including the I will become automatic.

Generally, because I is easy to forget, don't try to insert much text at once. To start, it's better to insert only a few lines, type $$, and review what you've typed. Then continue inserting with a new I command.

Moving the CP to the Start of a Line (L)

Type

```
! -2L$$
! T$$

(^)Thisis source line 2.
! #T$$

This is source line 1.
Thisis source line 2.
This is source line 3.
!
```

As you can see, the L command moves the CP around. With a negative number n, it moves the CP n NEW LINEs backward. With a positive number n, L moves the CP n NEW LINEs forward. Without a number, L moves to the beginning of the current line. If number n is out of range (like -500L), L simply moves to the beginning of the first line or beginning of the last line.

Type

`! 1LT$$`	(Use L and T command. The $ separator isn't needed between L and T.)
...	(SPEED displays text and CP.)
`! -400LT$$`	(Use L and T again..)
...	(SPEED displays text and CP.)
`! 500LT$$`	(L and T again...)
`(^) !`	(SPEED displays CP.)

Sequential 20LT commands can help you read forward (in longer files); -20LT commands help you read backward.

Jumping CP to Beginning or End of Buffer (J, ZJ)

The J command gets you to the start of the buffer, ZJ to the end of the buffer. Type

`! JT$$`	(Go to start of buffer and type the first line. As with L, you need not separate J and T with $.)
(^)This is source line 1.	(SPEED does it.)
`! ZJT$$`	(Go to end of buffer and type line.)
(^) !	(SPEED does it.)
`! IThis□is□line□4. ⤶ $$`	(Insert text at the end of the buffer.)
`! #T$$`	(Type entire buffer.)

```
This is source line 1.
.
This is line 4.
!
```

As shown, ZJ is a convenient command when you want to append text to to the file; you can simply start the insert after ZJ executes. (And J is easier than -nL to get you to the start of the buffer.)

Searching for a Character String (S)

S allows you to find a character string, and it places the CP after the found string. Type

`! Sline□1.$T$$`	(Search for the character string `line□1.` and type the line.)
Error: Unsuccessful search Sline 1.$!	(No luck.)

SPEED searches from the CP onward, so the search failed — because the CP was beyond the string you specified. When a search (or change) command fails, SPEED puts the CP at the beginning of the buffer. Therefore, another identical search should succeed. Type

```
! Sline□1.$T$$              (Search for the string line□1. and type
                            the line.)
This is source line 1.(^)   (Found.)
!
```

When a search (or change) command succeeds, SPEED puts the CP after the last character sought. To search for ↲, type

```
! S ↲
$T$$                        (Search for ↲.)
(^)Thisis source line 2.    (Found.)
!
```

As you saw, SPEED can find nonprinting characters like ↲. You'll see a use for this later in this session. As with any successful search, the CP is after the last character sought: after the ↲ on line 1, which is the beginning of line 2.

For a search (or change) to succeed, you must type in precisely the character string you want. For example:

```
! Sline2.$$                    (Search for string line2.)
Error: Unsuccessful search     (Not found — because we omitted the
Sline2.$$                      space between line and 2).

!
```

SPEED does not allow a space between the S (or C) command letter and the string. If you insert one, SPEED won't be able to find the string. For example, type

```
! S□This$$                     (Search for string □This)
Error: Unsuccessful search     (No luck.)
S This$$
!
```

The case of characters is important in search or change commands. For example, type

```
! SLINE$T$$                      (Search for string LINE and type it.)
Error: Unsuccessful search       (No LINE found.)
SLINE$T$$
! Sline$T$$                      (Try string line and type it.)
This is source line(^) 1.        (String line is found.)
!
```

You'll be using S a lot, so try a few more:

```
! Shis$T$$     (Search for his)
...
! Ss□s$T$$     (Search for s□s)
...
!
```

Changing a Character String (C)

The C command, like S, is a search command; but instead of simply searching, it searches and changes. C has the form

```
Cstring1$string2
```

where `string1` is the string you want to change to `string2` and `$` (as usual) means pressing the BREAK/ESC key.

You can use the C command to correct the typing error in line 2:

```
! J$$                           (To start of buffer.)
! S1.↵
$T$$                            (Position at start of line 2 and type.)
(^)Thisis source line 2.        (Done.)
! Cis$□is$T$$                   (Change is to □is and type it.)

(This(^)is source line 2.       (Not what we wanted.)
!
```

With C — as with S — you need to be accurate and specific. Fix it by typing

```
! J$CTh□isis$This□is$T$$        (Combine J, C, and T commands.)

This is(^) source line 2.       (Success.)
!
```

C can also extend changes over more than one line. For example, type

```
! Cline□4.⤶
$line□4.⤶                    (Change line 4.⤶ to line 4.⤶ plus
This is source line 5.⤶      This is source line 5.⤶)
$$
! #T$$

This is source line 1.
This is source line 2.
This is source line 3.
This is line 4.
This is source line 5.       (Done.)
!
```

You can also use the C command to delete text. Use the form Cxxx$$ where xxx is the text you want to delete. This command changes the characters of the text to nothing, effectively deleting the characters.

C — like S — is an extremely useful SPEED command. It allows you to edit without pinpointing the CP with other commands.

Deleting Lines (K)

The C command allows you to delete characters and lines; but K (for Kill) is much more convenient. The command form nK deletes n lines forward for a positive n or n lines backward for a negative n. (A *line* is the characters after the CP and through the next NEW LINE character.) To display the text, type

```
! #T$$
...
...
!
```

To delete (and restore) line 3, type the following:

```
! J$S3$LT$$                   (Position at the desired line.)
(^)This is source line 3.     (Done.)
! 1K$#T$$                     (Kill it and type all lines.)
...
...                           (Old line 3 is gone.)
! IThis is source line 3.↲
$#T$$                         (Reinsert it and type all lines.)
...
...                           (Done.)
!
```

To delete *the remainder* of a line, simply put the CP before the doomed string, and type 1K. Type

```
! S5$LSThis$T$$               (Position before string.)
This(^) is source line 5.     (Done.)
! 1K$T$$                      (Kill the rest of the line.)
This(^)                       (Done.)
! I□is□source□line□5.↲
$#T$$                         (Now restore string and type all lines.)
...                           (Done.)
!
```

Often it's easier to delete part or all of a bad line and insert a new one than to try to correct the original. As you saw, K can help with this.

With the K command, we suggest that you use only positive values of n to keep your editing simple. A negative n when the CP is in the middle of a line will delete not only the previous line but also the left portion of the current line.

At this point, you've executed SPEED and created a file, inserted text, moved to different lines, jumped to the beginning and end of the buffer, searched for and changed text, and killed and restored lines. You've used the commands SPEED, I, T, L, J, S, C, and K. These commands, along with UE$H and US$H, will suffice for most editing needs.

The following commands simply make editing easier.

Getting User File Status (U?)

Type

```
! U?$$


Global:
    Input File - None
    Output File - MYPROG.XX
Local:
    Input File - None
    Output File - None
!
```

The U? command describes some things we don't explain here, but it can help you remember the name of the file you're editing.

Repeating (Iterating) Commands <commands$;>

Sometimes, you may want to execute one or more SPEED commands many times. To do this, use the form

`<command [command] [...] $;>`

where `command` is any of those we've described — generally involving Search or Change. Be sure to follow the last search or change command with a semicolon; this prevents the command from looping. The semicolon *must go* after the last search or change command. If you forget the semicolon, you'll need to break the loop with CTRL-C CTRL-A.

For example, type

```
! JT$$                          (Start of buffer and type.)
(^)This is source line 1.
! <C⤶                           (Repeat change ⤶ to ⤶ ⤶ commands...
$⤶
⤶
$;>$$                           ...exit loop and do it.)
! #T$$                          Type all lines.

This is source line 1.

This is source line 2.

This is source line 3.
...
!
```

Here, you changed every NEW LINE (⤶) in the buffer to two NEW LINES (⤶ ⤶) — producing double spacing. You could do the opposite to produce single spacing in the file. Double spacing can be handy when you want to edit a printout with a pencil or pen.

Update Exit or Update Save and Halt (UE$H or US$H)

Type

```
! UE$H$$
R
```

The UE command writes the entire current edit buffer — with your editing changes — to disk, then clears the buffer. Next, the H command halts SPEED and returns to the CLI. US$H does the same things but saves the original file as backup, adding the characters .BU to the filename.

In our sample session, saving the original file would be pointless because we created it during this session.

Try re-editing MYPROG.XX:

```
speed myprog.xx↵                 (Start SPEED on the file.)
Lower case input encountered.    (Information.)

! #T$$                           (Type all lines.)

This is source line 1.

This is source line 2.
...

! CThis□is$This□is□new$T$$       (Change line to add □new.)

This is new(^) source line 1.

! US$H$$
R
```

Because you specified a backup file, there will be both a MYPROG.XX and a MYPROG.BU in the current directory — the latter being the backup file. Specifying a backup file can be useful if you need to check the original contents of the file. For a source program backup file, you should RENAME the backup file from the CLI to have the appropriate ending; for example, rename a FORTRAN source file to have the extension .FR.

SPEED allows only one backup file for any file; so, if backup file name.BU exists, and you type US$H, SPEED will delete the older name.BU and replace it with the newer name.BU.

Actually, you can type either the UE or US commands without H, but since the commands clear the buffer, you'd have to type H anyway to get back to the CLI and re-execute SPEED. So you might as well say UE$H or US$H. There are commands to read the file back into the buffer from SPEED, but we don't describe them here.

Finally, we'll tell you that you can *omit* the $ (BREAK/ESC) command delimiter after all but Insert, Search, and Change commands. For example, JS1.$$ and UEH$$ are legal. And, you may indent from the ! prompt before you give a SPEED command. For example, in response to the ! prompt, the keystrokes

```
□□□□□□CNO$YES$$
```

and the keystrokes

```
CNO$YES$$
```

both tell SPEED to do the same thing.

Summary

In this SPEED session, you've created and edited file MYPROG.XX, using commands SPEED, I, T, L, J, S, C, K, U?, <...>, UE$H, and US$H. These commands will allow you to do nearly all the editing you want.

Two new files result from the session: MYPROG.XX and MYPROG.BU (created by the SPEED US$ command).

SPEED Command Review

Table 9-1 reviews each SPEED command you used, with a simple example.

Table 9-1 SPEED command examples

Command	Example(s)	Meaning and Result
SPEED	`SPEED FILE1↵`	Execute SPEED. If file FILE1 exists, read it into the edit buffer; otherwise create it.

I and T	`IBAL=0.↵`
	`$-1T$$`
		Insert characters BAL=0.↵ at the CP position. Type the line just inserted.

J	`JIC□MYPROG↵` `$T$$`	Jump to start of buffer, insert characters C□MYPROG↵ there; type the line.
S	`S(J.NE.0)$T$$`	Search for the characters (J.NE.0); type the line.
C	`CZZ9$ZZZ9$T$$`	Search for characters ZZ9 and, if found, change to ZZZ9; type the line.
L	`LI↵` `$T$$`	Insert a NEW LINE (↵) character before the current line; type the line.
K	`3K$$`	Kill (delete) three lines.
T	`#T$$`	Type entire buffer.
U?	`U?$$`	Get file status.
<...$;>	`J$<CYes$No$;>$$`	From start of buffer, change every Yes to No.
UEH and USH	`UEH$$`	Update; write edited file to disk.

Other Development Programs

The Development Kit includes programs other than SPEED. Table 9-2 lists and introduces the Development Kit programs (alphabetically).

Table 9-2 Development kit programs

Program Name	Filenames and Comments
Library file editor	LFE.SV and LFE.OL. The library file editor allows you to build libraries of your most commonly used program routines. It is useful if you will do a lot of program development. If you have FORTRAN IV, you also need LFE to create the FORTRAN runtime library file.
Macroassembler	MAC.SV and MACXR.SV. MAC is needed only for assembly language programming. Before using it, you need to create a symbol file (MAC.PS).
Relocatable Loader	RLDR.SV and RLDR.OL. This loader (linker) is required to build programs in FORTRAN or assembly language, but is not needed for BASIC or ICOBOL programs.
SPEED text editor	SPEED.SV and SPEED.ER. There is a SPEED session earlier in this chapter.
System Library	SYS.LB. This file contains needed routines for any FORTRAN or assembly language program you build under DG/RDOS. The RLDR linker searches the library automatically when you run the linker.

These files are usually installed — along with SPEED — in the master directory, DE0. If you want to use any of them from a nonmaster directory, create a link of the file name to DE0:name and remember to include .SV or .LB extension in the link name.

Programming with DG Computer Languages 10

Read this chapter when

- you want to write a computer program using a DG BASIC language — either Extended BASIC or Business BASIC;
- you want to write a program using a compiled language: FORTRAN or COBOL;
- you want to know about building programs in FORTRAN or COBOL.

This chapter shows how to build programs in several computer languages. It assumes that you have some experience with the language you want to use. The major sections are

- Interpreters and Compilers
- Using BASIC
- Using FORTRAN
- Using Interactive COBOL
- What Next?

These languages are available with DG/RDOS on DESKTOP GENERATION systems. Any of them must be installed before you can use it. Installation is sketched in Chapter 2 and detailed on the product Release Notice.

Interpreters and Compilers

A computer program is a series of instructions that a computer can execute. It originates with *statements,* usually written as familiar words like PRINT or WRITE. After a person writes the statements, they are translated from the familiar words to a binary form that the computer can understand.

The software that does the translation is called an *interpreter* (BASIC), or a *compiler* (FORTRAN or COBOL).

With an interpreter-based language, you execute an interpreter to create and run the program. The interpreter checks each line as you type it; then, it helps you run the program. When you are done, you leave the interpreter. To run the program again, you must execute the interpreter again.

In a compiled language, you type the statements using a text editor. All the program statements form a *source program.* After you finish typing the source program, you run the compiler on it. The compiler checks for errors; and, if it finds none, it builds a file called a *relocatable binary* file with the statements translated into binary. If the compiler does find errors, you fix them by correcting program statements with the text editor, and recompile. Next — for FORTRAN — you need to run the RLDR program on the relocatable binary. RLDR builds an executable save (.SV) file.

Finally, you execute the program. If it doesn't run properly, you go back to change the source file and recompile. With COBOL, there's another option: examining the executing program with the debugger. A debugger can set *breakpoints* wherever you want to stop the program. At each breakpoint, you can check variable values to see what went wrong. With a compiled language, you usually have to repeat the text editing, compiling, RLDR (for FORTRAN), and executing steps several times until the program does what you want.

Interpreted languages are easy to use and allow programs to be written quickly — benefits that have made BASIC the most popular language for personal computers.

Compiled languages take more work than interpreted languages, but the programs they produce run much faster. Also, compiled programs can be larger than interpreted programs, since they don't need to share memory with an interpreter.

Using BASIC

There are two BASIC languages available with DG/RDOS. They are Extended BASIC, a simple, traditional BASIC; and Business BASIC, a business-oriented BASIC that offers screen management and indexed sequential (ISAM) file management.

This section tells you how to start, write a program in, and stop Extended and Business BASIC. On the simplest level, shown here, the syntax of the BASICs is identical.

Here are the steps you follow to create a BASIC program:

1. Get into BASIC by typing the BASIC interpreter name; for example,

   `BASIC ↵` (for Extended BASIC)
   or
   `BBASIC ↵` (for Business BASIC)

2. Write or edit a series of BASIC program statements.
3. Run the program with the BASIC command:

   ```
   * RUN ↵
   ```

4. If the program runs as you want it to, go to step 6.
5. Identify problems using BASIC runtime error messages or dynamic debugging. This means inserting STOP statements at strategic points — which lets you to print variable values, then continue the program — until you've found the bugs. Then return to step 2 and fix the offending statement(s).
6. Save the program on disk by typing the BASIC command LIST "filename" ↵. Later you can bring it back into memory with the command ENTER "filename ↵".
7. You're done! Type

   ```
   * BYE ↵
   ```

   to sign off BASIC.

About DG/RDOS BASICs

A BASIC program is a series of BASIC statements. Each statement begins with a number between 1 and 9999. BASIC checks the syntax of each line as you type it. When you type the command RUN ⤶, BASIC executes the statements sequentially by number; thus, the program can do useful work.

Business BASIC variable and array names can include up to six letters and numbers, and must begin with a letter; for example, TOTAL2. String variable names can be up to five letters and numbers, followed by $; for example, TELNO$.

Extended BASIC variable and array names can include 1 or 2 characters. The first character must be a letter; the second character must be a number, for example, I1. String variable names end with an extra $ character; for example, I1$. You can type variable and array names, keywords, and strings in uppercase, lowercase, or any combination.

In either BASIC, you can declare array names with the DIM statement; for example, DIM A (100). For comments, use the REM statement. The interpreter translates all text into uppercase, regardless of the case you type.

```
10 REM - Dimension array A for the needed values.
20 DIM A (100)
```

While you are typing a program, or at any point, you can examine its statements with the LIST command. To change an existing statement, type its line number, then the new text. When you're satisfied with a program, write it to disk with the BASIC command LIST "filename ⤶". Later, you can read it back into memory with the command ENTER "filename" ⤶. From BASIC, you can print the program on the printer by typing LIST "@LPT" ⤶.

To start work on another program, type NEW ⤶ and proceed. To sign off BASIC, type BYE ⤶.

Extended BASIC Systems and Directories

For Extended BASIC, a directory named BASIC.DR is created when BASIC is installed. Generally, all the programs you create go to this directory — and Extended BASIC accesses them from this directory — automatically.

For multiuser or specific applications, someone must generate an Extended BASIC system (just as with DG/RDOS). For standard single-user situations, there's a default BASIC system that's ready to use. The default system is named XBSFP.SV (for model 10) or XBHFP.SV (other systems); for hard disk systems, it's in the master directory (DE0) and for two-diskette systems it's on a specific diskette. From the proper directory, you can start this simple Extended BASIC system by typing XBSFP ↵ (Model 10) or XBHFP ↵ (other systems).

Business BASIC Systems and Directories

For Business BASIC, several directories are created when BASIC is installed. You can run Business BASIC directly from the master directory (DE0) — although you might want to create your own user directory (say BASIC.DR) to help keep your files organized. In a multiuser system, you *should* do this — perhaps using your own name to distinguish your directory from other users'.

Business BASIC has a system generation procedure that resembles the DG/RDOS configuration procedure. System generation is needed for multiple users or specific applications. However, Business BASIC files do include a default single-user Business BASIC system that's ready to use. The system name is BBASIC.SV; with a hard disk, it's in the master directory (DE0), and with two diskette systems, it's on a specific Business BASIC diskette. From the proper directory, you can start this simple Business BASIC system by typing BBASIC ↵.

Starting BASIC

As suggested above, the name of the BASIC system can vary. Unless you know the system name, from the master directory, try

BASIC ↵ (For Extended BASIC. If you get a DOES NOT EXIST message, try XBSFP ↵ or XBHFP ↵.)

or

BBASIC ↵ (For Business BASIC)

BASIC will announce itself by displaying a program banner; for example

```
Extended BASIC Revision x.xx
*
```

When the asterisk prompt appears, you are in BASIC and can start typing commands and statements. (If you get a FILE DOES NOT EXIST message, BASIC may not have been installed, or you may be in a nonmaster directory without a link to BASIC. If the former, install it as described on the BASIC Release Notice. If the latter, create a link named xxxx.SV to DE0:xxxx.SV, where xxxx is the name of the BASIC system file.)

A Sample BASIC Program

To get into BASIC, try the program shown in Figure 10-1. Press ↵ after typing each line. With Business BASIC, you can make things a little clearer by using longer variable names: use FACT instead of F, and use FTOTAL instead of F1. The names shown work with either BASIC.

```
10 PRINT "THIS PROGRAM COMPUTES FACTORIALS."
20 PRINT "TYPE AN INTEGER FROM WHICH YOU WANT THE FACTORIAL."
30 INPUT F
40 REM - SAVE ORIGINAL VALUE. THEN LOOP, REDUCING VALUE BY 1
50 REM - AND MULTIPLYING UNTIL THE MULTIPLIER BECOMES 1.
60 LET F1 = F
70 FOR I = F-1  TO  1  STEP  -1
80 F1 = F1 * I
90 NEXT I
100 PRINT  "THE FACTORIAL OF " ;F; " IS "; F1
110 END
```

Figure 10-1 A sample BASIC program

After typing the program, list all its statements with

```
* LIST ↲
```

```
10 PRINT "THIS PROGRAM COMPUTES FACTORIALS."
20 PRINT "TYPE AN INTEGER FROM WHICH YOU WANT THE FACTORIAL."
30 INPUT F
40 REM - SAVE ORIGINAL VALUE. THEN LOOP, REDUCING VALUE BY 1
50 REM - AND MULTIPLYING UNTIL THE MULTIPLIER BECOMES 1.
60 LET F1=F
70 FOR I=F-1 TO 1 STEP -1
80   LET F1=F1*I
90 NEXT I
100 PRINT "THE FACTORIAL OF ";F;" IS ";F1
110 END
```

Note how BASIC compressed your spacing and indented the statement within the FOR/NEXT loop. Now, run the program with

```
* RUN ↲

THIS PROGRAM COMPUTES FACTORIALS.
TYPE A NUMBER WHOSE FACTORIAL YOU WANT.
? 7 ↲
THE FACTORIAL OF 7 IS 5040

END AT 0110
*
```

Next, list the program to disk, leave BASIC, re-enter BASIC, ENTER the program, and leave BASIC again:

`* LIST "FACTORIAL.BA" ↵`	Used this way, the LIST command writes the current program to disk under the specified file (path) name. The .BA extension identifies the kind of file (BASIC program).
`* BYE ↵` *R*	Leave BASIC.
`BASIC ↵` or `BBASIC ↵`	Start Business BASIC or Extended BASIC.
... *BASIC* ...	BASIC displays its program banner.
`* ENTER "FACTORIAL.BA" ↵`	The ENTER command brings a program that was LISTed to disk back into memory.
`* BYE ↵`	Leave BASIC.

Unless you save a program on disk (for example, with the LIST command), all work done on it during this BASIC session vanishes when you leave BASIC. You can use any valid filename for the program.

If a file with the name you specify already exists, when you type LIST "filename", Extended BASIC will delete the old version and replace it with the new one. Business BASIC will report FILE ALREADY EXISTS; and you must type DELETE "filename" ↵, then type LIST "filename" ↵ again.

Now, change the program so that it will rerun until you type CTRL-C CTRL-A to interrupt. First, get back into BASIC, enter the program, and list it. Then add the needed statements:

```
* 106 PRINT "TO EXIT FROM THIS PROGRAM, TYPE CTRL-C CTRL-A." ↵
* 108 GOTO 20 ↵
```

And run it again:

```
* RUN ↵
.
TYPE A NUMBER WHOSE FACTORIAL YOU WANT.
? 8 ↵
THE FACTORIAL OF 8 IS 40320
TO EXIT FROM THIS PROGRAM, TYPE CTRL-C CTRL-A.

TYPE A NUMBER WHOSE FACTORIAL YOU WANT.
? 9 ↵
THE FACTORIAL OF 9 IS 362880
TO EXIT FROM THIS PROGRAM, TYPE CTRL-C CTRL-A.

TYPE A NUMBER WHOSE FACTORIAL YOU WANT. CTRL-C CTRL-A      (Type CTRL-C
                                                           CTRL-A)

...STOP AT 0020          (IKEY AT 0020 in Business BASIC)
*
```

Now list the changed program to disk using its original name:

```
* LIST "FACTORIAL.BA" ↵          (If you see an error message, type DELETE
                                 "FACTORIAL.BA ↵, then repeat the LIST com-
                                 mand.)

*
```

You're done. You've written, run, and edited a BASIC program.

To create and run more complex programs, follow the steps shown above, using the BASIC or BBASIC command to start BASIC; use the ENTER command to read a saved program into memory; and use the LIST command to see the statements. If a program is too long to fit on the screen, use the form

```
LIST s TO e ↵
```

where s is the starting statement number and e is the ending statement number you want to see. Or, you can type LIST ↵ and use CTRL-S and CTRL-Q to suspend and continue display. Change and run the program as desired. When you have finished, save it on disk by typing LIST "filename" ↵.

If you want to have the program write to a disk file or printer, it will need to open the file or printer and write to it, using BASIC file I/O statements.

BASIC Documentation

For more information on either Extended or Business BASIC, see the preface (before Chapter 1) for the names and numbers of manuals.

Using FORTRAN

FORTRAN is one of the oldest and most popular programming languages. This section covers two different DG FORTRAN compilers: FORTRAN IV and FORTRAN 5.

These are the steps you follow to create a program in FORTRAN:

1. Create or edit a source file using the SPEED text editor, adding FORTRAN statements and comments. Running SPEED is described in Chapter 9.
2. Compile the source file by typing

   `FORT filename ⤶` (For FORTRAN IV)
   or
   `FORTRAN filename ⤶` For FORTRAN 5)
3. If there are compilation errors, return to step 1 and fix the offending statement(s). If there are no errors, continue.
4. Link the object file to produce an executable program:

   `RLDR filename` *[subprograms]* `FORT.LB ⤶` (For FORTRAN IV)
   or
   `RLDR filename` *[subprograms]* `@FLIB@ ⤶` (For FORTRAN 5)
5. Execute the program with the CLI command

   `filename ⤶`
6. If the program runs the way you want it to, go to step 9.
7. Identify logic errors using runtime error messages or incorrect output.
8. Go to step 1 and fix the erroneous statement(s).
9. You're done!

FORTRAN IV Runtime Library

If you've already built and run FORTRAN IV programs, skip this section.

For you to create a FORTRAN IV program, the FORTRAN IV runtime library, from which RLDR extracts needed routines, must exist on the disk. This library is not supplied. Someone at your site must build it (needed only once). To build it, DIR to the FORTRAN directory (which someone should have created when FORTRAN IV was installed). Check to see if the library exists by typing

```
DIR DEO:FORT ⤶          (or whatever the FORTRAN IV directory is named)
R
LIST/E FORT.LB ⤶          (See if the file exists.)
```

If CLI displays name FORT.LB, fine; the library already exists and you can skip the rest of this section. If FORT.LB doesn't show up, create it. Type

```
LIST *MPYD.LB ⤶                  (Check for hardware or software multi-
                                  ply divide option. Model 10s have the
                                  software multiply divide option; all
                                  others have the hardware multiply
                                  divide option.)

xMPYD.LB                         (It displays SMPYD.LB or HMPYD.LB;
R                                 this is the name you'll use to build the
                                  FORTRAN IV library.)

LINK LFE.SV DEO:LFE.SV ⤶         (Create a link from the FORTRAN
R                                 directory to the Library File Editor
                                  (LFE). Ignore any FILE ALREADY EX-
                                  ISTS message.)

LFE   M   FORT.LB/O   FSYS.LB   FORT<0,1,2,3>.LB   xMPYD.LB ⤶
. (20 second delay)
R
```

If there are any error messages, retype the LFE command. After you create FORT.LB, it need not be recreated (unless you want to include any user-defined libraries, in which case, insert their names at the end of the LFE command line).

Writing a FORTRAN Program

This section shows an example program, which will compile and run with both FORTRAN IV and FORTRAN 5. The program asks for a number, computes and displays the factorial of the number, and gives you the option to run it again.

There should be a FORTRAN directory to hold your FORTRAN files. Find this via LIST/E -.DR if needed and DIR to it. Using the FORTRAN directory will keep all your FORTRAN programs in one place and prevent conflicts with other programs that have the same names.

For example,

```
dir fort ⤶
R
```

Now, use the SPEED text editor to create the source file. Use the name FACTORIAL.FR (actually, you can use any valid filename, but the compilers recognize the extension .FR shown above, and the sample command lines show the name FACTORIAL).

For example,

```
speed factorial.fr ⤶

CREATING NEW FILE
I C This FORTRAN IV/5 program ....
.
.
```

(If you get a FILE DOES NOT EXIST or EXPANDED MESSAGE TEXT... message, create links to SPEED and CLI files as described near the beginning of Chapter 9.)

Type the program shown in Figure 10-2. Don't forget to insert a tab (TAB key) before each FORTRAN statement that doesn't have a C or a statement label in column 1. The compiler requires all keywords to be in uppercase.

```
C  This FORTRAN IV/5 program computes factorials. It uses a double
C  precision real number to allow large values without integer overflow.

        DOUBLE PRECISION    FACTORIAL

        TYPE "This program computes factorials."
10      ACCEPT "Type a number whose factorial you want.  ", NUMBER

C    Assign the number to the factorial variable. Set up a control variable
C    for number-1 loops. Then loop, multiplying the factorial by a
C    decreasing value, number-1 times.

        FACTORIAL = NUMBER
        NLOOP = NUMBER - 1
        DO 100   I = 1, NLOOP
                FACTORIAL = FACTORIAL * ( NUMBER - I)
100     CONTINUE

        WRITE (10, 110)  NUMBER, FACTORIAL
110     FORMAT  (1X, "The factorial of  ", I4, " is  ", F20.0, / )
        ACCEPT "To repeat, type 1<NEW LINE>. To stop, type 0<NEW LINE>. ", IANSWER
C    If the person does not type 0, repeat.
        IF  (IANSWER .NE. 0)  GOTO  10
        END
```

Figure 10-2 A sample FORTRAN 5/FORTRAN IV program

Compiling a FORTRAN Program

Now to compile the source program FACTORIAL. Depending on your FORTRAN, type

```
fort factorial ↲           (for FORTRAN IV)
```
or
```
fortran factorial ↲        (for FORTRAN 5)
```

Nearly always, the compiler will find errors the first time you compile a source program. (In this case, the example is error-free, but — for most programs — you can expect compiler error messages.)

A FORTRAN IV compiler error message includes the text of a statement line, then two numeric codes, in the form

```
;       XXXXXXX source statement line  XXXXXXXX
;       *** n  ***  CHR  m
```

Number *n* is the error code, which you can look up in the *FORTRAN User's Manual,* Appendix B. *CHR m* shows the character position (m) where the compiler thinks the error occurred. A CHR 1 message means the error probably occurred on the *preceding* line.

FORTRAN 5 compiler error messages are quite explicit, telling you clearly what lines are wrong, and what's wrong with them.

If the compiler reports errors with this example, you may have made a typing mistake. Use the SPEED editor to fix it; and try the compile line again.

Creating the Save File with RLDR

The RLDR command line for FACTORIAL is

```
rldr factorial fort.lb ↲     (for FORTRAN IV)
```
or
```
rldr factorial @flib@ ↲      (for FORTRAN 5)
```

While RLDR builds the program file, it displays

```
FACTORIAL.SV LOADED BY RLDR REVISION n date time
.MAIN
```

```
.
. (names of all routines linked into the program - be patient.)
.
R
```

In FORTRAN, or any other high-level language, RLDR errors are rare. They usually mean you mistyped a symbol name in the source file or omitted a subroutine name from the RLDR line — resulting an undefined symbol error message.

(If you get FILE DOES NOT EXIST or SYSTEM LIBRARY NOT FOUND message, check for the following links (which must exist for you to use RLDR from this directory):

RLDR.SV	DE0:RLDR.SV
RLDR.OL	DE0:RLDR.OL
SYS.LB	DE0:SYS.LB

Also, the relocatable binary(ies) you're trying to link must exist. Here, the file, created by the compiler, is FACTORIAL.RB.

Executing the FORTRAN Program

Now, you can execute FACTORIAL.SV. To execute this or any other program, type the program name and ↲. You can omit the .SV extension. For example,

```
factorial ↲

This program computes factorials.
Type a number whose factorial you want. 8 ↲        (Try 8)
The factorial of 8 is 40320.

To run again, .....             (Program types instructions.)
```

To run the program again, type 1 ↲.

```
Type a number whose factorial you want. 10 ↲       (Try 10)
The factorial of 10 is 3628800.

To run again, .....             (It types instructions.)
```

To stop the program, type 0 ↲. The program stops and the CLI returns:

```
R
```

FORTRAN (and other compiled languages) have two types of error condition: *compiler errors,* which involve things like syntax; and *runtime errors,* which occur when you run the program. Runtime errors reveal problems that the compiler can't detect, like a filename that the program can't find.

FORTRAN IV gives no error text but reports the error as ERROR n. Code n is described in the FORTRAN IV manual, Appendix B. Again, FORTRAN 5 has good runtime error messages, which are usually specific enough to allow you to fix the problem.

If you can correct a runtime error without changing the program (for example, by moving a needed file into the same directory as the program), fine. If not, you need to repeat the cycle of editing the source to fix the error, compiling, linking with RLDR, and executing.

If you cannot identify the problem, insert numerous PRINT/TYPE statements to monitor the values in variables, and pinpoint the location of the problem; then fix it and recompile, link, and execute.

FORTRAN Summary and Documentation

You're done. You've written, compiled, linked, and executed a FORTRAN program. To create and run more complex programs, follow the steps shown above, using the text editor to write or fix the source program, then compile it, link it using RLDR, and execute it.

If you want to have the program write to a disk file or printer, it will need to open the file or printer and write to it, using FORTRAN OPEN and WRITE statements.

For a names and numbers of FORTRAN IV and FORTRAN 5 manuals, see the preface (before Chapter 1).

Using COBOL

COBOL (shorthand for Common Business-Oriented Language) is the the most popular language for business data management. COBOL programs resemble English, with paragraphs, sentences, clauses, and words. Well-written COBOL programs can be — to a large extent — self-documenting.

COBOL also offers ISAM (Indexed Sequential Access Method). With ISAM, a program can find a record by index key (for example, a customer name), or it can find records sequentially (perhaps to list all customers). ISAM works well in any situation where people need to find a record by one of several keys, or to find records alphabetically. It suits both large and small business situations.

The COBOL available with DG/RDOS is Interactive COBOL (ICOBOL). ICOBOL has its own, internal ISAM; and it allows primary and alternate keys to access one record. ICOBOL can run on any DESKTOP GENERATION computer.

The details on ISAM access are outside the scope of this book. However, this section *does* show a sample ICOBOL program and how to develop it.

These are the steps you follow to create a program in COBOL:

1. Create or edit a COBOL source file. You can use the ICOBOL text editor, IC/EDIT, described in the *IC/EDIT Interactive COBOL Editor* manual; or you can use the SPEED editor (Chapter 9).
2. Compile the source file by typing

   ```
   ICOBOL filename ⤶
   ```

   If you will want to use the interactive debugger, include the /D switch; for example, type ICOBOL/D filename ⤶.
3. If there are compilation errors, return to step 1 and fix the offending line(s). If there are no errors, continue.
4. Execute the program by typing

   ```
   ICX filename ⤶
   ```

5. If the program runs the way you want it to, go to step 8.
6. Identify logic errors using runtime error messages or erroneous output. If you included the debugger switches, you can debug the program:

   ```
   IDEBUG filename ⤶
   ```

7. Go to step 1 and fix the erroneous line(s).
8. You're done!

Writing a COBOL Program

The COBOL program in this section asks for a number, computes the factorial of the number, prints the factorial, and gives the person the option to run it again.

There should be an ICOBOL directory to hold your COBOL files. Find this via LIST/E -.DR if needed and DIR to it. Using a COBOL directory will keep all your COBOL programs in one place and prevent conflicts with other programs that have the same names. For example,

```
dir icobol ↲
R
```

Now, use the ICEDIT or SPEED text editor to create the source file. Use the name FACTORIAL.CO (actually, you can use any valid filename, but .CO is the conventional COBOL suffix, and the sample command lines we show will have the name FACTORIAL). For example,

```
speed factorial.co ↲
```

Type the COBOL program shown in Figure 10-3. Don't forget to insert one or more tabs (use the TAB key) as shown. The COBOL compiler requires all keywords (but not text strings in quotes) to be in uppercase.

```
*  This COBOL program computes factorials.

IDENTIFICATION DIVISION.
PROGRAM-ID.  FACTORIAL.
ENVIRONMENT DIVISION.

DATA DIVISION.
WORKING-STORAGE SECTION.
* Declare variables for input, output, and edited output from factorial
* routine; and declare the control variable (I) and the answer variable.

77      FACTORIAL       USAGE IS COMPUTATIONAL PICTURE S99.
77      FTOTAL          PICTURE 9(18).
77      EDITED-FTOTAL   PICTURE Z,ZZZ,ZZZ,ZZZ,ZZZ,ZZ9.
77      I               PICTURE 9(18).
77      ANSWER          PICTURE X(1).

PROCEDURE DIVISION.
        DISPLAY "This program computes factorials.".
CALCULATION-LOOP.
        DISPLAY "Type a number whose factorial you want. "
                WITH NO ADVANCING.
        ACCEPT FACTORIAL.
        MOVE 1 TO FTOTAL.
        MOVE 2 TO I.

        PERFORM FTOTAL-COMPUTATION UNTIL I IS GREATER THAN FACTORIAL.

        IF FTOTAL IS LESS THAN 10000000000000000 THEN MOVE FTOTAL TO EDITED-FTOTAL
                DISPLAY "The factorial of ", FACTORIAL, " is ",
                        EDITED-FTOTAL
        ELSE
                DISPLAY "Factorial of ", FACTORIAL, " is too large for format.".

        DISPLAY " ".
        DISPLAY "To run again, press NEW LINE.  To stop, type 'S NEW LINE'. "
                WITH NO ADVANCING.
        ACCEPT ANSWER.
        IF ANSWER IS NOT EQUAL TO "S" AND ANSWER IS NOT EQUAL TO "s"
                THEN GO TO CALCULATION LOOP.
        STOP RUN.

FTOTAL-COMPUTATION.
        COMPUTE FTOTAL = FTOTAL * I.
        ADD 1 TO I.
```

Figure 10-3 A sample COBOL program

Compiling the COBOL Program

Now to compile the source program FACTORIAL. Type

```
icobol factorial.co ↵
```

If you don't use the ICEDIT editor, the compiler usually will find errors the first time you compile a source program.

ICOBOL compiler error messages are quite explicit, telling you what lines are wrong, and what's wrong with them. If the compiler reports an error with one of these examples, you may have made a typing error. Use the text editor to fix it; and try the compile line again.

The ICOBOL compiler produces an intermediate file that you can execute via the ICOBOL interpreter.

Executing the COBOL Program

To execute a COBOL program (compiled ICOBOL program), you start up the program in the ICOBOL interpreter, which is named ICX. So the command to execute the FACTORIAL program is

```
icx factorial ↵
```

The ICOBOL interpreter now reads FACTORIAL and starts executing it.

```
This program computes factorials.
Type a number whose factorial you want. 8 ↵        (Try 8)
The factorial of +08 is 40,320.

To run again, press NEW LINE. To stop, type 'S NEW LINE'.
```

To run the program again, press the NEW LINE key.

```
Type a number whose factorial you want. 10 ↵       (Try 10)
The factorial of +10 is 3,628,800

To run again, press NEW LINE. To stop, type 'S NEW LINE'.
```

To stop the program, type S ↵. The program stops and the CLI returns:

```
R
```

Compiled languages have two types of error condition: *compiler errors,* which involve things like syntax; and *runtime errors,* which occur when you run the program. Runtime errors involve problems that the compiler can't detect, like a filename that the program can't find.

ICOBOL has good runtime error messages, which are usually specific enough to allow you to fix the problem. If you can correct a runtime error without changing the program (for example, by moving a needed file into the same directory as the program), fine. If not, you need to repeat the cycle of editing the source to fix the error, compiling, and executing.

If you cannot identify the problem, you can use the interactive ICOBOL debugger.

Using the ICOBOL Debugger

An interactive debugger can really ease the debugging phase of program development. To use it, compile your program with the /D switch. To execute, start the program in the ICOBOL debugger instead of the interpreter:

```
IDEBUG filename ↲
```

ICOBOL debugger commands are described in the ICOBOL utilities manual, named in the preface (before Chapter 1).

COBOL Summary and Documentation

You're done! You've written, compiled, and executed a COBOL program. For more complex programs, the development steps are similar to those above: write the source file, compile, and fix syntax errors; then start in the interpreter. To use a debugger, include the /D switch to compile; and type IDEBUG filename to execute.

If you want to have a program write to a disk file or printer, the program will need to declare the file or printer, open it, and write to it, using file I/O keywords defined in the COBOL documentation.

For a names and numbers of ICOBOL manuals, see the preface (before Chapter 1).

What Next?

If you want to learn about other software, proceed to the appropriate chapter; or, if you want, review earlier material. Or, you can start writing your own programs, using the pertinent DG language manuals described above.

Using the DG/RDOS Sort/Merge Program (CSSORT) 11

Read this chapter when

- you want to learn about data records;
- you want to learn to use the DG/RDOS Sort/Merge utility.

The DG/RDOS Sort/Merge program, CSSORT, can sort and merge records stored in randomly or contiguously organized DG/RDOS files. CSSORT was originally developed for use with Interactive COBOL and Commercial Systems (hence its name). CSSORT has become the official DG/RDOS Sort/Merge. It's useful whenever you want to re-order records — created by an application program or directly by yourself, with a text editor.

The major sections in this chapter proceed

- Records and Record I/O
- The DG/RDOS Sort/Merge Program (CSSORT)
- Merging
- CSSORT Error Messages
- What Next?

Records and Record I/O

Records pertain to all computer programming — not just to CSSORT. Records are groups of characters that a program reads from and writes to devices — like the terminal and disk files.

Records can be read and written by a text editor, in which case they are lines of text like these. Each line is a record read from the terminal and written to disk. Or, records can be read and written by a computer program that you write. Or, they can be read and written by CSSORT.

Computer systems can use any of several different methods to distinguish one record from the next record. The most common methods are

- using a special character to separate the records. The system treats all characters up to the next special character as a record. Records separated this way are generally called *data-sensitive* records. (In CSSORT operations, data-sensitive records are called *line-sequential* records.) The special character that separates records is called the *delimiter*. A standard delimiter is CR (carriage return). DG/RDOS translates all ⤶ characters into CR internally. Each line of text you type using the CLI or a text editor is treated as a line-sequential record; the ⤶ press at the end, translated into CR, is the delimiter.

  The system sees three different characters as line-sequential delimiters: CR, form feed (page break), and null. If a program does a line-sequential read from a file, and reads one of these characters, DG/RDOS treats the characters preceding this character as the record.

  Line-sequential records are most useful for writing and reading text, because the delimiters ⤶ (CR) and form feed (page break) occur as natural line and page terminators in the text.
- using a constant length for the records. The system treats each group of this length as a record. Such records are called *fixed-length* or *fixed* records. In CSSORT operations, fixed-length records are called *fixed-sequential* records. With fixed-sequential records, the system does not look for a delimiter; it just considers each group of characters as a record.

  Fixed-sequential records are useful for information that adapts well to a constant length, like telephone numbers. Sort and merge operations are faster with fixed records than line-sequential records, since the system doesn't have to search for a delimiter.

- A third kind of file, indexed (ISAM) files, allows access to records by one or more index *keys*. ISAM files are the most versatile of all. You can create and access ISAM files using Interactive COBOL (ICOBOL) or Business BASIC as described in the product documentation.

A program specifies the kind of record format (line- or fixed-sequential) when it writes to the file.

To use CSSORT on a file with line-sequential records, you must tell it to treat records as line-sequential, using the /L switch.

Record Fields

Fields are character positions within records. For example, take the following record (shown in Figure 11-1).

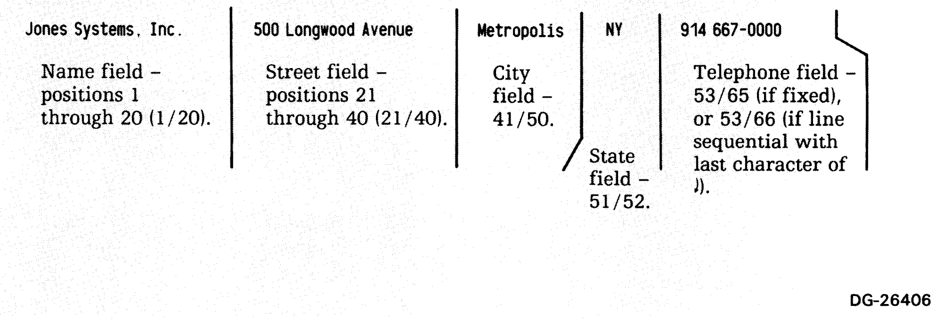

Jones Systems, Inc.
500 Longwood Avenue
Metropolis
NY
914 667-0000
Name field – positions 1 through 20 (1/20).
Street field – positions 21 through 40 (21/40).
City field – 41/50.
State field – 51/52.
Telephone field – 53/65 (if fixed), or 53/66 (if line sequential with last character of J).
DG-26406

Figure 11-1 Anatomy of a record

This record has five fields. Other records in the same file would normally also have five fields, with identical starting/ending character positions. This allows all the records to be read (and the fields changed as needed) consistently. It also allows you to sort the records by any of the five fields. For example, you can sort by name key (positions 1/20) or state key (51/52), or both. ("Sort" means reordering records by numeric or alphabetical order.)

The DG/RDOS Sort/Merge Program (CSSORT)

The CSSORT Sort/Merge program can process DG/RDOS random- and contiguously-organized files. These files show a "D" or "C" as attributes after their byte lengths when you LIST them. CSSORT can't sort a DG/RDOS sequentially-organized file (displayed by the LIST command with no letter after its byte length). "Sequentially-organized" has no relation to the term "line sequential" or "fixed sequential."

CSSORT produces as output a new DG/RDOS randomly organized file. You can use this file as input for CSSORT sort or merge operations.

The merge operation accepts as input up to six sorted files of the same type and merges them into a single output file.

CSSORT can produce different output files from a given input file. You can tell it to reformat records for the output file — using its /R switch. This allows you to create tailored reports from a master file. Records can be sorted on any COBOL data type in ascending or descending order or according to a sequence you define.

CSSORT sorts an input file and produces a new sorted output file. The input and output files are different; that is, CSSORT doesn't write within one copy of a file. Files that have been sorted on the same key can be merged in a separate operation into one file.

You can define a collating sequence by creating a file that contains the desired alternate sequence and specifying this sequence's filename in the command line.

By default, CSSORT always reports the sort or merge statistics. You can specify a disk filename for these statistics and — if desired — print the file.

CSSORT assumes 7-bit characters. It cannot sort 8-bit characters according to the collating sequence in the international character set.

You can run CSSORT from the system console or from the foreground master console.

CSSORT Example

The CSSORT program offers many options. But the explain it is to show a short sample sort.

Assume there is a randomly-organized file named CTEST.IN. Its records are ASCII characters terminated by carriage returns (CRs). The fields in each record are as shown in Figure 11-1: name in positions 1 through 20; street in positions 21 through 40; city/town in positions 41 through 50; state in positions 51 and 52; and phone number in positions 54 through 66.

Let's say that CTEST.IN holds the following records:

```
LEVIATHAN PRODUCTS  44 ALOUETTE LANE   ST PAUL   MN 612 100-0000
SYNERGY MORNING INC 95 BLEEKER STREET  NEW YORK  NY 212 899-0000
APEX PRODUCTIONS LTD 88 POWELL STREET   OAKLAND   CA 415 885-0000
JONES SYSTEMS INC   500 LONGWOOD AVENUEMETROPOLISNY 914 667-0000
ALPHA DESIGNS CORP  0 NEWBURY STREET   BOSTON    MA 617 266-0000
```

To try the example, use a text editor or the XFER command, as follows. (For UPPERCASE and lowercase text in records, see "Changing the Collating Sequence", below. In any case, XFER can't insert lowercase letters in a file, since the CLI makes them uppercase before it writes them.)

To help identify multiple spaces, we've used periods. You can type spaces instead of the periods if you want. The output examples include the periods.

```
DIR DEO ⤶           (Or go to the directory that holds the CSSORT files.)
R
DELETE/V    CTEST.IN ⤶          (Start with a clean beginning.)
.
. (Ignore a "FILE DOES NOT EXIST" message.)
R
XFER/A    $TTI    CTEST.IN/R ⤶       (Use /R switch for random organiza-
                                      tion.)
```

```
LEVIATHAN PRODUCTS...44 ALOUETTE LANE...ST PAUL...MN 612 100-0000 ⤶
SYNERGY MORNING INC..95 BLEEKER STREET..NEW YORK..NY 212 899-0000 ⤶
APEX PRODUCTIONS LTD 88 POWELL STREET...OAKLAND...CA 415 885-0000 ⤶
JONES SYSTEMS INC....500 LONGWOOD AVENUEMETROPOLISNY 914 667-0000 ⤶
ALPHA DESIGNS CORP...0 NEWBURY STREET...BOSTON....MA 617 266-0000 ⤶
```

```
↓             (Press the downarrow key.)
R
LIST CTEST.IN ⤶
CTEST.IN 330 D
R
```

The file contains 330 characters: 5 records, each with 65 letters and CR.

To sort the file by name, type

```
CSSORT    CTEST.IN/L    CTEST.OT/O/L    1:20/K ⤶
```

CSSORT then displays a screenful of statisics, and the CLI prompt returns:

```
R
```

Examine the input and output files:

```
LIST/S    CTEST.- ⤶
CTEST.IN 330 D
CTEST.OT 330
R
```

Check out the sort:

```
TYPE CTEST.OT ⤶

ALPHA DESIGNS CORP...0 NEWBURY STREET...BOSTON....MA 617 266-0000
APEX PRODUCTIONS LTD 88 POWELL STREET...OAKLAND...CA 415 885-0000
JONES SYSTEMS INC....500 LONGWOOD AVENUEMETROPOLISNY 914 667-0000
LEVIATHAN PRODUCTS...44 ALOUETTE LANE...ST PAUL...MN 612 100-0000
SYNERGY MORNING INC..95 BLEEKER STREET..NEW YORK..NY 212 899-0000
R
```

The alpha sort on the name field worked — ALPHA, APEX, JONES...

Other approaches could sort by state key and secondarily by name key, as follows:

```
51:2/K 1:20/K
```

or by telephone area code with a secondary name key, like this:

```
54:3/K 1:20/K
```

You can begin to see the organizational power of the CSSORT Sort/Merge. (Just as an aside, CTEST.IN could be sorted as either a line-sequential or fixed-sequential file, since all records are the same length, and all are terminated by CR.)

Command Line Analysis

The meaning of each piece of the CSSORT command line is as follows:

CSSORT	Starts the CSSORT Sort/Merge program.
CTEST.IN/L	Identifies the input file we created. There is no /O switch, therefore CSSORT assumes that this is the input file. Switch /L means the file's records are line-sequential (they end in CR), as described above). For fixed-sequential records, we'd say `CSSORT CTEST.IN/S 66/N PARTS.OT/S/O 1:20/K ↲`
CTEST.OT/O/L	The /O switch identifies the output file. The /L switch tells CSSORT that the file's records are line sequential.
1:20/K	The key field (to sort on) begins with the first character (byte). The field continues for 20 characters, the length of the name field (hence 1:20). There's just one key field.

The next section describes the CSSORT command format and switches.

Sort Command Format

To sort, you must specify at least an input file and output file (outfile/O switch). Also, you must use either the infile/L switch (for line sequential records) or the number/N switch (for fixed sequential records). There are many other possibilities. The entire command line for a sort operation is

```
CSSORT[/N]  { infile/S number-bytes/N outfile/O/type         }  ^
            { infile/type [number-bytes/N] outfile/O[/S]     }

key1/K [... key8/K] ^

[field-specifier1/F ... field-specifier8/F] ^

[collating-file/C] ^

[workfile1/W ...workfile6/W] [auditfile/A]
```

CSSORT Switches

All the CSSORT switches, alphabetically, are

auditfile/A	Names the file to which you want Sort/Merge statistics written. By default, the utility writes them to your terminal. The statistics include filenames and types, record and key specifiers, collating file, time, and number of records. If you omit this, and /N, the utility writes to your terminal. Do not specify the printer ($LPT) as both the audit and output file.
collating-file/C	Names the file containing the user-specified collating sequence. The default is the standard 7-bit ASCII sequence. You can define a nonstandard collating sequence only with keys of the ASCII data type (default, key/K switch); with other types, CSSORT ignores user-defined collating sequences. Creating your own collating sequence — for upper- and lowercase records, for example — is explained later, in "Changing the Collating Sequence."

field-specifier/F

Specifies the position and length of an input field to be moved to the output record. Field specifiers allow you to reformat records, copying only selected portions to the output file. Field specifiers are optional. You can specify up to 8 field specifiers, each with the following form:

```
field-position:field-length/F
```

`field-position` is the position of the first byte in the input record to be copied to the output record. The position of the first byte is 1, the second 2, and so on.

When you include one or more field specifiers, the fields appear in the output record in the order you specified with /F. You can move, duplicate, or overlap fields.

`field-length` is the field byte length.

If a file's records end in CR, you will probably want to copy the CR delimiter to the output file file (if not, the file won't be printable). You can do this only if the records are all the same length. To do it, make the last /F indicate the CR. For example, if the file has 80-character records that end in CR, use 80:1/F as the last field specifier.

Examples of field specifiers are

```
1:5/F 20:6/F
```

Tells CSSORT to create each output record from the 1st through 5th characters and the 20th through 26th characters in the input record.

```
54:3/F 1:20/F
```

Tells CSSORT to create each output record from the 54th, 55th, and 56th characters and the 1st through 20th characters in the input record (it swaps field positions in the output record).

infile/type	Defines the file type of the input file. If you omit this switch, CSSORT assumes the file is a fixed-sequential file. The values for type are

infile/I	File is an ICOBOL indexed file.
infile/L	File is a line sequential file.
infile/R	File is an ICOBOL relative file (input files only).
infile/S	File is a fixed-sequential file (default). You can skip this switch.
infile/V	File is a variable-sequential file. In these files, ICOBOL uses a 2-byte header to specify the length of each record. Thus, CSSORT can sort variable files not created by ICOBOL only if there is a 2-byte header for each record.

key/K

Specifies the position of the key in the record, and the key length, both in bytes. You can specify from one to eight keys. CSSORT will sort on the keys in the order they appear in the command line: the first key will be primary, the second secondary, and so on.

In an ICOBOL indexed file, a key specifier doesn't indicate a primary or alternate key. Instead, it specifies the portion of the *record* to be sorted.

You can specify keys to overlapping fields in the record.

Key specifiers have the following form:

`key-position:key-length/K` *[/D][/d]*

`key-position` specifies the position of the first byte in the input record. The first byte is 1, the second 2, and so on.

`key-length` is the byte length of the key in the input record. The byte length cannot exceed the record size nor run past the end of the record from the key-position.

`/K indicates a key specifier.`

For example, 3:2/K means that a key begins at byte (character) position 3 of the input record. It is a 2-byte key: characters 3 and 4 are the key.

/D	indicates descending ASCII order. The default is ascending. You can intermix ascending and descending ASCII order in different keys.
/d	indicates the ICOBOL data type (described further in the ICOBOL documentation). You can specify any of the following as d:

/A	ASCII characters (default).
/C	Computational, unsigned.
/C/S	Computational, signed.
/N	Numeric display, unsigned.
/N/L	Numeric display, leading sign.
/N/L/S	Numeric display, leading separate sign.
/N/T	Numeric display, trailing sign.
/N/T/S	Numeric display, trailing separate sign.

Examples of data types in keys are

`1:3/K` bytes 1,2,3; ascending order; ASCII.

`1:3/K/D` bytes 1,2,3; descending order; ASCII.

`13:4/K/N/L` bytes 13 through 17, ascending order; numeric display with leading sign.

/N	No statistic display. This tells CSSORT to skip the terminal display of sort statistics.
number-bytes/N	Number of bytes in the input record. This switch is mandatory *only* if the input file is fixed sequential. The record length in a line-sequential file is limited to 132 bytes, plus the delimiter (CR, form feed, or null). For a variable-sequential file, specify the maximum record size if you know it. Otherwise, the default, 4,096 bytes, will slow the sort. For example,

```
CUST3.IN 80/N
```

says that the record length in the fixed-sequential file CUST3.IN is 80 bytes.

outfile/O/type
: Defines the name and record type that you want for the output file. The name outfile cannot exist when you start CSSORT; otherwise, CSSORT will stop with an error message. The maximum filename length is 13 characters. Values for type are

outfile/O/L	Creates and writes a line-sequential file. For example, `CUST3.OT/L`
outfile/O/S	Creates and writes a fixed-sequential file (default). You can omit this.
outfile/O/V	Creates and writes a variable-sequential file.

work-filename/W
: specifies a temporary sorting file. CSSORT uses up to six work files. If you do not explicitly name them, the program creates work files named SORTW1.TP though SORTW6.TP. CSSORT creates and deletes these files for each sort. The most active work files are SORTW1.TP and SORTW4.TP. If there is insufficient disk space or a large number of records to sort, you can specify work files to be on different devices for a faster sort. For example, DE1:WORK1.TP/W indicates that WORK1.TP on DE1 is a work file. The work files' names are assigned to the user-specified names in the order that they appear in the command line. Thus, to specify work file 4 in the command line, you must include at least four work filenames.

CSSORT does not require a specific order for arguments in the command line. For example, the input, output, and audit filenames can appear anywhere in the command line. However, the order of key, field, and work file specifiers is important. CSSORT sees the first key specifier in the command line as the primary, the second as secondary, and so on.

Changing the Collating Sequence

By default, CSSORT uses the straight ASCII sequence to sort. The ASCII characters are represented by internal codes consisting of decimal integers from 0 through 127. For example, the character A has the decimal value 65. CSSORT uses these integers to compare and sort characters. Digits precede uppercase letters which precede lowercase letters.

By creating and specifying a collating-sequence-file to CSSORT, you can redefine the ASCII precedence for this sort. The leading characters get first (lowest) precedence; following characters get higher precedence. In other words, integer codes are assigned to the characters starting with 0. The remaining ASCII characters, which are not in the file, are assigned the remaining codes in their usual order. Here are the rules for defining alternate collating sequences:

- The file can contain only the ASCII characters.
- No character can be repeated.
- The file cannot be longer than 128 characters.
- Each character must fit in 7 bits (CSSORT cannot sort 8-bit characters according to the international character set collating sequence).

Upper- and Lowercase Precedence

One useful application of a different collating sequence to make UPPERCASE and lowercase characters equal. This allows you to use upper- and lowercase characters in records and have them sorted as if they were the same case.

For example, the standard sort of the records

AZTEC Trucking
Aardvark Supply
ABC Licensing

would produce the following sorted file:

ABC Licensing
AZTEC Trucking
Aardvark Supply

because lowercase follows uppercase in the standard ASCII sequence.

To correct it, you can build a collating file of the following characters (text editor or other non-CLI program required). Let's assume the file is called ULC (for UpperLowerCase).

```
01234567890AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwYyZz
```

Have the file end with Zz, not with CR or ↵. You can do this with SPEED — just don't press ↵ after the last letters. Or, if there is already a ↵, delete the last character.

Then, you sort with the name/C switch; for example

```
CSSORT    xxx/L    xxx/O/L    x:x/K    ULC/C ↵
```

to produce the following sequence of records:

Aardvark Supply
ABC Licensing
AZTEC Trucking

Generally, you should include the digits *before* the rest of the definition sequence. If you omit the digits, they will receive a lower value than any letters you specify, and thus records sorted on digit keys will appear at the end of the output file. (This may be what you want — if so, do it.)

Another Collating Sequence Example

In the following example, the user wants to generate a report that indicates by salesperson the products being marketed. The following information is needed to develop the command line:

Input Filename: SALESREC File Type: Indexed

Record Position	Contents	Byte Pos : Byte Length
1-6	Item stock number	1 : 6
7-10	Salesperson code	7 : 4
11-16	Transaction date	11 : 6
17-22	Customer number	17 : 6
23-36	Item description	23 : 14
37-40	Item class	37 : 4
41-43	Units sold	41 : 3
44-45	Discount code	44 : 2
46-46	Carriage return character	46 : 1

Output Filename: SALESITEM File Type: Fixed sequential

The output file is to be formatted to include the salesperson, the item sold, number of units sold, and the transaction date:

Record Position	Contents	Byte Pos : Byte Length
1-4	Salesperson code	1 : 4
5-10	Item stock number	5 : 6
11-13	Units sold	11 : 3
14-19	Transaction date	14 : 6
20-20	Carriage return character	20 : 1

The salesperson codes are numeric. Several years ago the format of the stock item numbers was changed. The old stock item numbers are 6 characters long; the first 2 characters are uppercase alphabetic and the last 4 are numeric. New stock item numbers are 6 digits. When sorting the file, the user wants to list the old stock items before the new ones. Thus this sort requires an alternate collating sequence in which uppercase precedes digits.

The alternate collating sequence would be created as follows:

```
xfer/a    $tti    uppercase ↵
ABCDEFGHIJKLMNOPQRSTUVWXYZ ↓       (Press downarrow at end.)
```

Since the uppercase letters are in the alternate collating sequence, they are assigned first precedence. The remaining characters (digits, lowercase letters) follow in their usual order.

The command line to generate the report is:

```
CSSORT   SALESREC/I   SALESITEM/O   7:4/K/N   1:6/K    11:6/K/N ^ ↵
7:4/F    1:6/F    41:3/F    11:6/F    46:1/F    $LPT/A    UPPERCASE/C ↵
```

An analysis of the command line follows.

CSSORT	Starts the CSSORT program.
SALESREC/I	Specifies the input filename. The /I switch indicates an indexed file.
SALESITEM/O	Specifies the output filename. Without a type switch, CSSORT will make this file fixed sequential.
7:4/K/N	Defines the primary key: the salesperson. The ICOBOL data type is unsigned numeric. This key is not affected by the alternate collating file.
1:6/K	Defines the secondary key: the item stock number. With the data type switch omitted, the default is ASCII. This key is sorted by an alternate collating sequence.
11:6/K/N	Defines the tertiary key: transaction date. The data type is unsigned numeric.This key is not affected by an alternate collating file.
^ ↵	Caret continuation sign. The sign permits the command to be continued on the next line. To separate arguments, you must insert a space before the caret, or at the beginning of the next line (or in both places).
7:4/F	Field specifier: salesperson code. Bytes 7, 8, 9, and 10 of the input file record are duplicated in bytes 1, 2, 3, and 4 of the output file.

1:6/F	Field specifier: item stock number. Bytes 1, 2, 3, 4, 5, and 6 of the input file record are duplicated in bytes 5, 6, 7, 8, 9, and 10 of the output file record. Note the necessary blank before the 1; this blank follows the ^ symbol.
41:3/F	Field specifier: units sold.
11:6/F	Field specifier: transaction date.
46:1/F	Field specifier: carriage return character.
$LPT/A	Prints audit information on the printer.
UPPERCASE/C	Alternate collating sequence filename.

The following table illustrates the reformatting done by the field specifiers. The input file format is listed in the first column; the output file format in the last.

Byte Pos : Byte Length	**Contents**	**Byte Pos : Byte Length**
7 : 4	Salesperson code	1 : 4
1 : 6	Item stock number	5 : 6
41 : 3	Units sold	11 : 3
11 : 6	Transaction date	14 : 6
46 : 1	Carriage return (CR)	20 : 1

The order of the field specifiers in the command line determines the order of fields in the output file.

The following audit information is displayed on the screen and printed on the printer:

```
CS SORT/MERGE PROGRAM REV 1.20     11/23/83    ***SORT OPERATION***
          FILENAME              FILE TYPE      MAXIMUM RECORD SIZE

INPUT:    SALESREC              INDEXED                     46
OUTPUT:   SALESITEM             FIXED-RECORD SEQUENTIAL     20
KEYS:  START BYTE * LENGTH * ASC-DESC * DATA TYPE
            7         4        A       NUMERIC, UNSIGNED
            1         6        A       ASCII CHARACTERS
           11         6        A       NUMERIC, UNSIGNED

OUTPUT FIELD SPECIFIERS :  (START BYTE , LENGTH)
        7,   4            1,   6            41,  3             11,    6
       46,   1
AUDIT FILENAME :      $LPT     SEQUENCE FILENAME:    UPPERCASE
WORK FILENAMES    :
       NONE SPECIFIED

PRESORT      11:49:28              11:50:09   NO. RECORDS IN:  673
LAST PASS    11:50:09     DONE     11:50:58   NO. RECORDS OUT: 673
FIXED SEQUENTIAL OUTPUT RECORD SIZE IS 20 BYTES.
```

Using More Than One Key

In this example, a user wants to create a mailing list from the employee record file EMPREC. Because this list is to be used for bulk mailing, the sort is by ZIP code. For internal use, the user wants the sort refined by street, then by employee name. The following information is needed to develop the command line:

Input Filename: EMPREC File Type: Fixed sequential

Record Position	Contents	Byte Pos : Byte Length
1-5	Employee number	1 : 5
6-35	Employee name	6 : 30
36-55	Street	36 : 20
56-75	City	56 : 20
76-77	State	76 : 2
78-82	ZIP Code	78 : 5
86-89	Extension	86 : 4
90-95	Date hired	90 : 6
96-96	Carriage return	96 : 1

Output Filename: MAILIST File Type: Fixed sequential

The format of the output file is determined by the field specifiers in the command line. For this example, the employee name, street, city, state, and ZIP code from the input file are to be included in the output file. These items are included in record positions 6-82 (i.e., byte position 6, byte length 77) of the input file. The output record format is:

Record Position	**Contents**	**Byte Pos : Byte Length**
1-30	Employee name	1 : 30
31-50	Street	31 : 20
51-70	City	51 : 20
71-72	State	71 : 2
73-77	ZIP Code	73 : 5
78-78	Carriage return (CR)	78 : 1

The command line to create the mailing list is:

```
CSSORT    EMPREC    96/N    MAILIST/O    78:5/K/N    36:20/K    6:30/K ^↲
6:77/F    96:1/F    $LPT/A↲
```

The following is an analysis of the command line.

CSSORT	Starts the CSSORT program.
EMPREC	Specifies the input filename. No type switch is needed because the file is fixed sequential.
96/N	This is the number of bytes (input record length). It's required because the file is fixed sequential.
MAILIST/O	Specifies the output filename. No type switch is needed because the file is to be fixed sequential.
78:5/K/N	Defines the primary key, the ZIP code field. The data type is unsigned numeric.
36:20/K	Defines the secondary key, the street address. Where there are duplicate ZIP keys, CSSORT will sort them by the street address.
6:30/K	Defines the tertiary key, the employee name. Where the there are duplicate street addresses, CSSORT will sort them by employee name.

^ ⤵	Caret continuation sign. The sign permits the command to be continued on the next line. To separate arguments, you must insert a space before the caret, or at the beginning of the next line (or in both places).
6:77/F	Field specifier: employee name, street, city, state, and ZIP code.
96:1/F	Field specifier: CR (carriage return character).
$LPT/A	Prints audit information on the printer.

The following audit information is displayed on the screen and printed on the printer:

```
FULL
CS SORT/MERGE PROGRAM REV 1.20     11/23/83     ***SORT OPERATION***

           FILENAME              FILE TYPE     MAXIMUM RECORD SIZE

INPUT:     EMPREC                FIXED-RECORD SEQUENTIAL  96
OUTPUT:    MAILIST               FIXED-RECORD SEQUENTIAL  78
KEYS: START  BYTE * LENGTH *  ASC-DESC * DATA TYPE
          78           5         A       NUMERIC, UNSIGNED
          36          20         A       ASCII CHARACTERS
           6          30         A       ASCII CHARACTERS

OUTPUT FIELD SPECIFIERS :   (START BYTE , LENGTH)
         6,  77               96,  1
AUDIT FILENAME  :  $LPT     SEQUENCE FILENAME:   NONE SPECIFIED
WORK FILENAMES    :
       NONE SPECIFIED

PRESORT      08:09:45            08:10:05  NO. RECORDS IN:      345
LAST PASS    08:10:05    DONE    08:10:35  NO. RECORDS OUT:     345
FIXED SEQUENTIAL OUTPUT RECORD SIZE IS 78 BYTES.
```

Sorting Indexed Files with Alternate Keys

In this example, the user wants to prepare a file that lists account balance, customer name, address, and account number in descending order by balance. If duplicate balances occur, the sort will be on account number in ascending order. The input file is ACCT$REC. It is indexed by account number as the primary key and customer name as the alternate key. The following information is necessary to develop the command line:

Input Filename: ACCT$REC File Type: Indexed with alternate keys

Record Position	Contents	Byte Pos : Byte Length
1-10	Account number	1 : 10
11-35	Customer name	11 : 25
36-55	Street	36 : 20
56-75	City	56 : 20
76-77	State	76 : 2
78-82	ZIP Code	78 : 5
83-91	Account balance	83 : 9
92-92	Carriage return (CR)	92 : 1

Output Filename: ACCT$BAL File Type: Line sequential

The output file is to be formatted to include the account balance, customer name, address, and account number. The output record format is:

Record Position	Contents	Byte Pos : Byte Length
1-9	Account balance	1 : 9
10-34	Customer name	9 : 25
35-54	Street	35 : 20
55-74	City	55 : 20
75-76	State	75 : 2
77-81	ZIP Code	77 : 5
82-91	Account number	82 : 10
92-92	Carriage return	92 : 1

The command line to create the file in account balance sequence is

```
CSSORT   ACCT$REC/I   ACCT$BAL/O/L   82:9/K/D/N/L   1:10/K/N ^↓
82:9/F   11:72/F   1:10/F   92:1/F ↓
```

An analysis of the command line follows.

CSSORT	Starts the CSSORT program.
ACCT$REC/I	Specifies the input filename. The /I switch indicates an indexed file.
ACCT$BAL/O/L	Specifies the output filename. The /L switch tells CSSORT to make this file line sequential.
83:9/K/D/N/L	Defines the primary key: the account balance. Switch /K indicates a key; /D specifies descending collating order; /N/L indicates a data type of numeric with leading sign.
1:10/K/N	Defines the secondary key: the account number. CSSORT wil sort duplicate balances by account number, in ascending order.
^↓	Caret continuation sign. The sign permits the command to be continued on the next line. To separate arguments, you must insert a space before the caret, or at the beginning of the next line (or in both places).
11:72/F	Field specifier: customer name, street, city, state, and ZIP code. Since the order of these fields doesn't change, they can be listed under one specifier.
1:10/F	Field specifier: account number. This is to appear as the last data field in the output file.
92:1/F	Field specifier: carriage return.

Merging

CSSORT can take up to six sorted input files and merge their records into one output file. The input files must be sequentially-organized (as produced by a CSSORT sort). If a file is already sorted on the field(s) you want to merge on (and is a sequentially-organized file), you don't need to sort it again before merging.

In essence, the CSSORT merge has the same syntax as the sort, except that you type CSSORT/M instead of CSSORT and there are two or more input files instead of one.

As with the CSSORT sort earlier, we'll show an example of the operation first, then give the syntax, switches, and a detailed example.

CSSORT Merge Example

This example builds on the one for the CSSORT sort: it takes the original sort file, created earlier, and another file, and merges them.

Just to review, the format of the records is as shown in Figure 11-1: name in positions 1 through 20; street in positions 21 through 40; city/town in positions 41 through 50; state in positions 51 and 52; and phone number in positions 54 through 66.

The file with the sorted records is named CTEST.OT. File CTEST.OT holds the following records:

```
ALPHA DESIGNS CORP...0 NEWBURY STREET...BOSTON....MA 617 266-0000
APEX PRODUCTIONS LTD 88 POWELL STREET...OAKLAND...CA 415 885-0000
JONES SYSTEMS INC....500 LONGWOOD AVENUEMETROPOLISNY 914 667-0000
LEVIATHAN PRODUCTS...44 ALOUETTE LANE...ST PAUL...MN 612 100-0000
SYNERGY MORNING INC..95 BLEEKER STREET..NEW YORK..NY 212 899-0000
```

To try this example, you need to create another file — say with three records.

To try the example, use a text editor or the XFER command, as follows. (For UPPERCASE and lowercase text in records, see "Changing the Collating Sequence", above. For lowercase, you must use a text editor or other program, not the XFER command.)

```
XFER/A   $TTI   CTEST1.IN/R ⤶
BOMBAST INDUSTRIES...1095 PARK LANE.....MAPLETON..LA 504 454-0000 ⤶
VERICOLOR VALUES INC 8 ROYALE AVENUE....WICHITA...KN 316 225-0000 ⤶
EMPIRE ARRANGEMENTS..202 ENTERPRISE WAY PORTLAND..ME 207 998-0000 ⤶
```

↓ (Press the downarrow key.)

Check CTEST1.IN:

```
LIST CTEST1.IN ⤶
CTEST1.IN 198 D
R
```

Sort the file, line-sequential, and name the output file CTEST1.OT:

```
CSSORT   CTEST1.IN/L   CTEST1.OT/O/L   1:20/K ⤶
```

.
. (it displays statistics)
.

```
R
```

Type the output file to make sure it's sorted correctly.

Check both input files:

```
LIST/S   CTEST-.OT ⤶
CTEST.OT 330 D
CTEST1.OT 198 D
R
```

The attribute identifier (D) after the file size indicates a randomly-organized file. The CSSORT merge (unlike the sort) accepts randomly- *or* sequentially-organized files.

Now to try the merge:

```
CSSORT/M/L    CTEST.OT    CTEST1.OT    CTEST.MG/O/L    1:20/K ⤶
.
. (it displays statistics)
.
R
```

Check the new, merged file:

```
LIST CTEST.MG ⤶
.
TYPE CTEST.MG ⤶
.
```

The alpha merge operation worked: ALPHA, APEX, BOMBAST.... As with a sort operation, you can sort on any field — perhaps by state and secondarily by name, or by telephone area code and secondarily by name, and so on.

Merge Command Line Analysis

The meaning of each piece of the CSSORT merge command line is

CSSORT/M/L	Starts the CSSORT program. The /M switch specifies a merge operation. The /L switch specifies the file type, line sequential. All files to be merged must be the same type.
CTEST.OT and CTEST1.OT	These are the sorted files to be merged. In a merge, CSSORT assumes that each filename without a switch is an input file.
CTEST.MG/O	This is the merge output file.
1:20/K ⤶	This is the key to sort on, characters 1 through 20. Usually, but not always, you'll want the same key for a merge sort as was used to sort the input files.

CSSORT Merge Format

The command line to merge line-sequential or variable files is

```
CSSORT/M[/type][/N] infile1 infile2 [...in-file6] ^

out-file/O/type [number-bytes/N] key1 [...key8] ^
field-specifier1/F ...[field-specifier8/F] ^
[auditfile/A] [collating-file/C] ^
```

For fixed-sequential records (default), you can omit the /`type` but must include the *`number-bytes/N`* switch. For line-sequential records, you must include the /`type` switch /S and you must omit *`number-bytes/N`*. These rules are the same as for the sort operation.

auditfile/A	Names the file to which you want statistics written. The default is the screen. A good alternative is the printer, $LPT, if you have one.
collating-file/C	Names the file containing a user-specified collating sequence. Creating your own collating sequence — for upper- and lowercase records, for example — is explained above, in "Changing the Collating Sequence." The collating sequence must be the same sequence that was used to sort the input files.
field-specifier/F	Specifies the position and length of an input field you want copied to the output record. Field specifiers allow you to reformat records, copying only selected portions to the output file. Field specifiers are optional. You can specify up to 8 field specifiers, each with the following form:

```
field-position:field-length/F
```

`field-position` is the position of the first byte in the input record to be copied to the output record. The position of the first byte is 1, the second 2, and so on. When you include one or more field specifiers, CSSORT copies fields to the output record in the order you specify with /F. For examples, see this switch in the previous major section.

key/K — Specifies the position of the key in the record, and the key length, both in bytes. You can specify from one to eight keys. CSSORT will sort on the keys in the order they appear in the command line: the first key will be primary, the second secondary, and so on.

Key specifiers have the following form:

```
key-position:key-length/K [/D][/d]
```

`key-position` specifies the position of the first byte in the input record. The first byte is 1, the second 2,

	and so on. For example, 3:2/K defines a key beginning at the third character; the key is two characters long.
	Switch /D defines a descending collating sequence for the sort on this key. Switch /d defines the ICOBOL data type. For more detail on data types and keys, see the key /K switch in the preceding major section.
/M	Defines the merge operation; without it, CSSORT will try to sort (producing an error, since there will be more than one input file).
/N	No statistic display. This tells CSSORT to skip the terminal display of sort statistics.
number-bytes/N	Number of bytes in the input record. This switch is mandatory *only* if the input file is fixed sequential. The record length in a *line-sequential* file is limited to 132 bytes, plus the delimiter (CR, form feed, or null).
outfile/O/type	Defines the name and record type that you want for the output file. The name outfile cannot exist when you start CSSORT; otherwise, CSSORT will stop with an error message. The maximum filename length is 13 characters. Values for type are

outfile/O/L	Creates and writes a line-sequential file.
outfile/O/S	Creates and writes a fixed-sequential file (default).
outfile/O/V	Creates and writes a variable- sequential file.

/type	Defines the file type of the input files. If you omit this switch for, CSSORT assumes the files are fixed sequential. The values for type are

/L	Files are line sequential.
/S	Files are fixed sequential (default).
/V	Files are variable sequential. In these files, ICOBOL uses a 2-byte header to specify the length of each record. Thus, CSSORT can sort variable files not created by ICOBOL only if there is a 2-byte header for each record.

CSSORT does not require a specific order for arguments in the command line. For example, the input, output, and audit filenames can appear anywhere in the command line. However, the order of key and field specifiers is important.

Another Merge Example

In the following example, a company does a heavy volume of business. Each quarter, the new customer list is combined with the existing customer list to produce the quarterly update list. The new customer list is an indexed file named NEWCUST. The existing customer list is also an indexed file named OLDCUST. The customer number is the index key. Both files have the same record format, which is shown next.

Record Position	**Contents**	**Byte Pos : Byte Length**
1-6	Customer number	1 : 6
7-36	Customer name	7 : 30
37-81	Address	37 : 45
82-91	Phone number	82 : 10
92-115	Buyer	92 : 24
116-119	Credit limit	116 : 4
120-120	Carriage return	120 : 1

The updated customer list will be a fixed-sequential file named UPDATE. It will be constructed alphabetically by customer name and retain the same record format.

Since input files in the merge command line must be sequential and sorted on the key to be used by the merge operation, NEWCUST and OLDCUST must be sorted on key position 7:30. Duplicates will be sorted on customer number, key position 1:6. The command lines for the sort are

```
CSSORT   NEWCUST/I   ALPHANEW/O   7:30/K   1:6/K/N   $LPT/A ↵

CSSORT   OLDCUST/I   ALPHAOLD/O   7:30/K   1:6/K/N   $LPT/A ↵
```

The record size of all 5 files — NEWCUST, OLDCUST, ALPHANEW, ALPHAOLD, and UPDATE — is 120 bytes. The two fixed sequential files, ALPHANEW and ALPHAOLD, are merged alphabetically to create the new UPDATE file. The command line is:

```
CSSORT/M   ALPHANEW   ALPHAOLD   120/N ^↵
UPDATE/O   7:30/K   1:6/K/N   $LPT/A ↵
```

An analysis of the command line follows.

CSSORT/M	Starts the CSSORT program. Without /type switch, CSSORT assumes all input files are fixed sequential.
ALPHANEW	Specifies an input filename.
ALPHAOLD	Specifies an input filename.
120/N	Specifies input record size. Because the input files are fixed sequential, they must all have the same record length.
UPDATE/O	Specifies the output filename (/O switch). There is no /type switch, therefore CSSORT creates and writes a a fixed sequential file.
7:30/K	Defines the primary key, customer name. With the data type switch omitted, the default is ASCII.
1:6/K/N	Defines the secondary key, customer number. The /N switch indicates unsigned numeric data.
$LPT/A	Prints audit information on the system printer.

Since no field specifiers are given in the command line, the output file has the same record format as the input files. The following audit information is displayed on the screen and printed on the printer:

```
CS SORT/MERGE PROGRAM REV 1.20    12/ 1/83    ***MERGE OPERATION***
INPUT FILES: TYPE--FIXED-RECORD SEQUENTIAL    MAX RECORD SIZE:  120
NAMES:      1 - ALPHANEW      2 - ALPHAOLD
OUTPUT FILE: TYPE--FIXED-RECORD SEQUENTIAL    MAX RECORD SIZE:  120
NAME: UPDATE
KEYS:  START BYTE * LENGTH * ASC-DESC * DATA TYPE
            7         30         A       ASCII CHARACTERS
            1          6         A       NUMERIC, UNSIGNED

OUTPUT FIELD SPECIFIERS :  (START BYTE , LENGTH)
      NONE SPECIFIED
AUDIT FILENAME :  $LPT     SEQUENCE FILENAME: NONE SPECIFIED
MERGE  16:35:51            DONE  16:39:05
INPUT RECORD COUNTS :
   1 -         59            2 -         948
TOTAL RECORDS INPUT  :      1,007    TOTAL RECORDS OUTPUT :    1,007
FIXED SEQUENTIAL OUTPUT RECORD SIZE :      120 BYTES
```

CSSORT Error Messages

The full text of each CSSORT error message indicates whether the error is a user error or a program error. User errors are errors in the command line. Program errors are uncovered by the operating system. They include such errors as uninitialized directories, nonexistent files, and exhausted file space. If you encounter a user error, correct the command line. If you encounter a program error, respond as with a CLI error message.

ALL INPUT FILES ARE EMPTY
You asked CSSORT to merge empty files.

COLLATING SEQUENCE FILE IS TOO LARGE
The collating sequence of a file cannot be longer than 128 characters.

DIRECTORY NOT INITIALIZED
You must initialize any nonmaster directory before you attempt to access a file in it. Do this with the INIT or DIR command.

DUPLICATE KEY FIELDS
You can't repeat a key field.

DUPLICATE CHARACTERS WERE FOUND IN SEQUENCE FILE
You can't use define any character twice in a collating sequence file.

FILE ALREADY EXISTS FILE NAMED filename
CSSORT will neither delete nor append to an output file that already exists. You can either rename the file (if you want to keep it) or you can delete it via the CLI.

FILE DOES NOT EXIST FILE NAMED filename
CSSORT can't find a file you specified. You may have made a typing error, or, if a link entry is involved, the resolution file may not exist.

FILE NOT ACCESSIBLE BY DIRECT I/O
A CSSORT sort requires a randomly- or contiguously-organized input file. Check the file attribute with LIST. (The attribute is a letter that follows the byte count; it must be R or C; no letter means that the file is sequentially organized.) If the file is sequentially organized, use XFER to copy it onto a random file (XFER name newname/R ↓).

FOUND RECORD TOO SMALL FOR KEY OR FIELD SPECIFICATION
Each record must be at least as long as the key(s), and field specifiers (if any). For a line-sensitive file, this error may indicate two CR characters in a row (which define a zero-length record). Generally, in a line-sensitive file, any double space (two CRs in a row) will produce this error. To fix it, delete all but one CR.

FOUND VARIABLE RECORD LENGTH EXCEEDING LIMIT
The header of a variable record shows more characters than the record maximum (4,096 bytes).

FOUND VARIABLE RECORD WITH ZERO OR NEGATIVE LENGTH
The header of a variable record has an invalid value.

ILLEGAL COMMAND SWITCH COMBINATION
Switches on CSSORT conflict. For example, you may have specified an indexed output file for a sort.

ILLEGAL GLOBAL SWITCH COMBINATION
You have specified more than one file type for the merge operation, for example, /L/S.

ILLEGAL RECORD SIZE
For line-sequential files, the maximum record size is 133 characters, including the delimiter. For fixed- or variable-sequential files, the maximum record size is 4,096 characters.

IMPROPER DATA FOUND IN KEY
The key was not of the data type expected; for example, it had a numeric leading sign when CSSORT expected ASCII.

IMPROPER DATA WAS FOUND IN SEQUENCE FILE
Only ASCII characters are allowed in the collating sequence file.

INPUT RECORD SIZE TOO SMALL FOR KEYS OR FIELDS
It found a record too short for the key or field specifier(s). See the message under FOUND RECORD TOO SMALL...

INSUFFICIENT MEMORY FOR PRESORT
Check the amount of memory with GMEM. Try giving this ground more pages (if possible).

INSUFFICIENT MEMORY FOR THE SPECIFIED SORT
See the message under the previous error.

INSUFFICIENT MEMORY FOR THE SPECIFIED MERGE
The size of the sort or merge exceeds the amount of memory available. Memory demand depends record size, and the size and number of the keys and fields. You must change some part of the sort or merge. Adapt the command line to include fewer or smaller keys or fields. If possible, give more memory to the ground running the CSSORT program.

INVALID OUTPUT FIELD SPECIFIER
The field specifiers must be in the form field-pos:field-length/F or field-pos.field-length/F.

INVALID KEY SPECIFIER
The key specifiers must be in the form key-pos:key-length/K or key-pos.key-length/K.

KEY RANGE ERROR
A key specifier is out of range.

MORE THAN ONE SEQUENCE FILE IS SPECIFIED
Only one alternate collating sequence is allowed per sort or merge. You may have specified more than one filename with the /C switch.

NO KEY SPECIFIERS
You must include at least one key specifier. Indicate the specifier with the /K switch.

NO INPUT FILE SPECIFIED
You must specify at least one input file when sorting and two input files when merging.

NO KEY OR OUTPUT FILE FOR SORT
One output file and one key specifier is required for sorting.

NO OUTPUT FILE SPECIFIED
An output file is required and must be specified with the /O switch.

NO RECORD SIZE SPECIFIER FOR FIXED SEQUENTIAL
If a CSSORT operation involves a fixed sequential, you must the number of bytes in each input record. Use the /N switch (number/N).

OUTPUT FILE ALREADY EXISTS
The output file, which is specified with the /O switch, cannot currently exist. To fix it, see the message under FILE ALREADY EXISTS.

OUTPUT RECORD IS TOO LONG FOR LINE SEQUENTIAL
The maximum record size for line-sequential files is 133 bytes, including the delimiter (usually CR).

*PROGRAM ERROR: **message***
Find the message in this table.

RECORD SIZE SPECIFIER PERMITTED ONLY WITH FIXED SEQUENTIAL INPUT
You cannot specify a record size unless the input file is fixed sequential.

REVISION INCOMPATIBILITY
This ISAM file is of an obsolete revision level.

THIS OUTPUT FILE MUST BE LINE SEQUENTIAL
When sorting files, the output file must be line sequential. No switch indicating the file type is allowed.

TOO FEW INPUT FILES
You must specify at least two input files for the merge operation.

TOO MANY AUDIT FILES SPECIFIED
You can specify only one one audit file.

TOO MANY INPUT FILES SPECIFIED
You can specify only one input file when sorting. In a merge operation (CSSORT/M), you can specify up to six files.

TOO MANY KEY SPECIFIERS
You can include a maximum of eight key specifiers in the command line.

TOO MANY OUTPUT FIELD SPECIFIERS
You can include a maximum of eight field specifiers in the command line.

TOO MANY OUTPUT FILES SPECIFIED
Only one output file is allowed. You may have specified the /O switch for more than one file.

TOO MANY RECORD SIZE SPECIFIERS
Each input record size must be preceded by the corresponding input filename.

TOO MANY WORK FILES SPECIFIED
You can specify up to six work files. The work files take the form workfilename/W.

USER ERROR: message
Find the message in this table.

UNIDENTIFIED LOCAL SWITCH
CSSORT doesn't recognize a switch you used with one of the arguments.

Communicating with Another System 12

Read this chapter when

- you want to send or receive information directly from another computer system (for example, send or receive a file).

This chapter outlines the DG products that allow DESKTOP GENERATION systems to communicate with other systems: larger DG systems, other DESKTOP GENERATION systems, information banks that you can dial up, and IBM systems. The major sections are

- DG/BLAST and DG/XAP™ for File Transfer
- DG/GATE™ for Information Banks
- RJE80 for File Transfer
- What Next?

For your system to communicate with another system, it needs *communications hardware:* a device called a USAM (universal synchronous/asynchronous multiplexor), with either one or four lines. There must be a link between the systems: a direct wire or a phone line (connected to the remote system via a modem).

A phone line has great advantages. The primary one is the huge international telephone network, whose lines reach practically everywhere. To use a phone line, you need a *modem*, also called a *data set*. With a modem, you dial the other system's number on a phone, wait for the appropriate tone, then make the modem connection.

In addition to the hardware described above, your system needs the appropriate software: DG/BLAST, DG/XAP, DG/GATE, or RJE80.

DG/BLAST and DG/XAP for File Transfer

These programs provide file transfer over an asynchronous line. The other system must also be running the same program.

With DG/BLAST or DG/XAP commands, you establish connection with a remote DG system, send and receive files, monitor the transmission, and close the connection.

Features of DG/BLAST and DG/XAP include

- Unattended operation with command (macro) files.
- Remote file manipulation.
- Automatic transfer restart on line reconnect after disconnect.
- Logging of communications and activity.

DG/BLAST and DG/XAP have manuals of their own, described in the preface (before Chapter 1).

DG/GATE for Information Banks

DG/GATE enables your DESKTOP GENERATION system to communicate with public information networks. DG/GATE emulates a nonintelligent ASCII terminal.

This means you can dial up and use another computer system (for example, an information pool with electronic bulletin boards that lists sales or closing Dow Jones quotations). The other system need not be a DG system.

DG/GATE features easy, menu-driven operations and programmable function keys. It can even auto-dial and log on to remote host systems. And, it can record all intersystem dialog in a disk file for later review.

Some features of DG/GATE let you

- Set automatic-dial modem control.
- Log all dialog in a disk file or on your printer.
- Give control to remote system.
- Suspend a session for temporary local work.

DG/GATE is further described in *Generating, Running and Using DG/GATE.*

RJE80

DG's RJE80 IBM emulator provides file transfer between your DESKTOP GENERATION system and any other system that runs RJE80. IBM products that can run RJE80 include the popular IBM 2780 and 3780 data communications terminals. Virtually all DG systems — including 32-bit MV/Family ECLIPSE systems can also run RJE80 — which means that RJE80 can transfer files to and from nearly any DG system.

RJE80 can receive files from remote systems without operator attention.

RJE80 is further described in its own manuals, named in the preface.

What Next?

This chapter has given a brief sketch of the communications products available with DESKTOP GENERATION systems. Next, you might want to review earlier material, or consider graphics (Chapter 13).

Looking at Graphics 13

Read this chapter when

- you want to learn something about hardware and software graphics products.

This chapter sketches the graphics hardware and DG software available with DG/RDOS systems. The major sections in this chapter are

- Graphics Hardware
- Graphics Software
- What Next?

Graphics Hardware

DESKTOP GENERATION computers — especially the Model 10/SP — include several graphics features. The model 10/SP system console monitor, for example, has a graphics command set and multiple window capability.

As options, DG offers several terminals with additional graphics features. They are

- Model 6262 color monitor CRT, with 16 colors, as system console on Model 10/SP systems only;
- DASHER D410/D460 terminal, as system console or user terminal on Model 20 and Model 30 systems.

Printers available with DESKTOP GENERATION systems include

- Model 4434 printer, an economical dot-matrix printer with graphics capability that can serve both as a printer and plotter;
- Model 4435 plotter, a color graphics plotter that can plot charts and pictures using felt-tipped pens.

Also available are

- Model 4436 mouse, an input device that you move across a a flat surface.
- Model 4437 data tablet, an input device that translates graphic information into digital information.

To work properly, the printer, plotter, and/or terminal must have been specified to the CONFIG program.

Model 10/SP System Console Monitor (Standard and Color)

Both the standard and color Model 10/SP system console monitor (screen) have some inherent graphics capability. Each monitor includes a graphics command set for character or bit-mapped graphics. For bit-mapped graphics, main memory locations are mapped to co-ordinates on the system console screen.

The graphics command set has control sequences that do things like set foreground color (color is either green or black on the standard system console, or one of 16 choices on the color monitor), start reverse video, or define a character.

The graphics commands work on both the standard and color monitors — but certain color commands have no effect on the standard monitor. All commands and control sequences are described in the *Model 10/SP Monitor and Keyboard User's Manual.*

DASHER D410/D460 Terminals

The D460 terminal features reverse video, multiple windows, slow scrolling, compressed type, and user-definable characters. The D410 terminal, although it is not a graphics terminal, does have reverse video, multiple windows, slow scrolling and compressed type.

Model 4434 Printer

The model 4434 is a versatile, economical dot-matrix printer that — via its graphics capability — can work as a plotter. It accepts 8-bit codes, adding 128 characters and symbol to the standard set. And it offers a choice of standard type (80 characters per line) or compressed type (160 characters per line).

The Model 4434 printer has a firmware program that allows you to change certain printer characteristics. Operating the model 4434 firmware and hardware is covered in the appropriate hardware *Operating* manual, described in the preface.

Model 4435 Plotter

This plotter uses two, color pens and can plot either on paper or film (for projectable view-graphs). To add additional colors to a plot, you can change pens.

Operating the model 4435 hardware is covered in the appropriate hardware *Operating* manual, described in the preface.

Model 4436-A — Mouse

The mouse is a small handheld box that connects by cable to a graphics display terminal.

Operating the mouse is explained in the *Operating* manual described in the preface.

Model 4437 — Data Tablet

The data tablet is a large tablet with an electric grid beneath its surface. It uses either a cursor puck or a stylus to generate input signals. Operating the data tablet is further explained in the *Operating* manual described in the preface.

Graphics Software

The graphics software products available with DG/RDOS include GW-BASIC (with MS-DOS).

GW-BASIC

GW-BASIC runs under MS-DOS as an extension of Microsoft BASIC-86. It can produce graphics suitable for business, technical applications, and games.

GW-BASIC supports graphics for either monochrome or color monitors. GW-BASIC's drawing statements help you create lines and circles, or paint the screen. The screen editor allows multistatement lines and recognizes special function keys that are reassignable within your program. GW-BASIC programs can call assembly language subroutines.

What Next?

This chapter has outlined DG's graphics hardware and software.

You've finished the substance of this manual. You might want to examine earlier chapters, or simply use DG/RDOS on your DESKTOP GENERATION system. Enjoy.

Responding to Errors and Error Conditions 14

Read this chapter

- if DG/RDOS stops with five-number display (message *nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn*);
- when power returns after a power failure;
- whenever you don't understand what's happening with the system;
- if nothing happens when you expect something to happen;
- when you want to know how to get help.

This chapter tries to describe every error message and condition (including no response, "nothing") that you may receive while running DG/RDOS — and tells you how to recover from the error. It can't cover *all* products: for MS-DOS, ICOBOL, BASIC, or other product error messages, see the pertinent product manual.

The major sections are

- Error Messages, Conditions, and Recovery
- About New Bad Blocks
- How to Get Help
- What Next?

Error Messages, Conditions, and Recovery

The following table, Table 14-1, describes important DG/RDOS and related program errors, alphabetically by message (if any). It includes the error messages shown above. The table gives the message, the program or situation where it may occur, the cause of the problem, and tells how to recover from the error.

Table 14-1 Error messages and recovery (continues)

Message	Source, Possible Cause(s), and Action
(nothing, or no response)	At power on. The computer may not be receiving power; or the computer unit was turned on before other devices in the system. Verify that the computer unit and separate units (like a second hard disk) are plugged in and that the outlet is live. Turn on power sequentially: first the system console, then the printer (if any), then the cartridge tape module (if any), then the second hard disk (if any), and finally, the computer unit. (On Model 10 and 10/SP systems that have a printer on the printer port, the computer tests the printer by having it print a line; if the printer cannot print, the computer won't complete its power up test. Ready the printer, then turn computer power off and on.) At power on. The system console is not turned on, or it's not on line, or it's set too dim. Check power, the ON/OFF switch behind the screen, and the ON LINE light. If the terminal has power but ON LINE is off, press the CMD key, hold it down, and press ON LINE. To check for brightness, use the control under the right front corner of the terminal.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
(nothing or no response)	At DG/RDOS startup, when you type 20 to start from diskette. Perhaps the diskette is inserted backwards. Remove and reinsert it, as shown in Chapter 2.
	At DG/RDOS startup (general). Perhaps you typed the wrong characters (not 26H or 20H). Type the break sequence (CMD and BREAK/ESC keys). This should produce the *!* prompt and you can type 26H or 20H. *Don't* type the break sequence if DG/RDOS is up (*R* prompt).
	During DG/RDOS operations, after you type an INIT or DIR command to a diskette. DG/RDOS spends 20 seconds trying to access a diskette before it gives up and displays a *DEVICE TIMEOUT* message. If a diskette is misinserted, or there is no diskette in the drive, the CLI prompt won't return for 20 seconds. Be patient. After the prompt returns, check for correct diskette insertion in the drive.
	During DG/RDOS operations. Type CTRL-Q (to undo any CTRL-S that suspended display). If this works, you've recovered.
	During normal DG/RDOS operations. If DG/RDOS is running, and power is still on to the computer, the system may be deadlocked. Type CTRL-C CTRL-A on the the system console. If there is no response or if the system really seems inert, turn computer power on and off again, and proceed as described in Chapter 3, "Abnormal Shutdown."
	While running the disk block owner program YOWNER. If YOWNER hangs for a minute or more, this probably means a bad block is the index block of a randomly organized file. See the YOWNER step (22) in the next section, "About Bad Blocks."

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
?!	At startup. Use the *numeric keypad* to the right of the main keypad to type numbers. The numeric keypad has only numbers on it, and always produces numbers without needing a terminal emulator.
!	From the system console loader program. It means that the computer is halted. If you see this when you have just turned power on, or shut down DG/RDOS, it's normal. To start DG/RDOS, type `26H`. Or, you can turn off power if you want. If you see this *!* when DG/RDOS was running, it means that DG/RDOS is frozen; someone may have typed the break sequence at the system console. To return control to DG/RDOS, type `P⤶`.
@^^**&^^^ (text stream and beeps)	From the CLI. You may have told the system to type a binary file on your terminal (this can happen if you type `TYPE CLI.SV⤶` or a template like `TYPE -.-⤶`.) When it is displaying a binary file, the terminal ignores most CTRL characters. So, press the COMMAND key (CMD) and simultaneously press ERASE PAGE. If this doesn't restore the normal display, type CTRL-C followed by CTRL-A. Then press the CMD and ERASE PAGE keys again. The CMD/ ERASE PAGE sequence tells the terminal to exit from graphics mode; and CTRL-C CTRL-A terminates the command. If this doesn't work, type CTRL-C followed by CTRL-B, and then press CMD/ERASE PAGE to terminate the current process. If the terminal remains frozen, you must turn off the computer unit power and restart (described in Chapter 3, "Power Failures.")

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
nnnnnn nnnnnn nnnnnn nnnnnn nnnnnn	From DG/RDOS. A serious error has prevented the system from continuing; and it has stopped in a *panic*. This can happen for any of several reasons. A panic can occur after you remove an initialized diskette without releasing it. There may be open files, particularly if an application program was running. DIR to each directory that was initialized at the panic, type `CLEAR/A/V/D↵`, and note the names of all files that appear (ignore SYS.DR). Any file left open by an abnormal shutdown may have contain inconsistent data. Be prepared for this — in the worst case, you might need to delete the file(s) that was open and restore from backup media. To restart, type `I` to reset the computer, then restart (26H or 20H). If panics recur, see "Persistent Problems" in Chapter 3.
ADDRESS ERROR ... perhaps followed by: *** WARNING - RDOS CANNOT BE RUN WITH THIS BLOCK BAD*	From DKINIT, on a FULL or PARTIAL run. This usually means that the diskette is not hardware formatted. Hardware format the diskette as described in Chapter 6; then try DKINIT again. If this error occurs on a hard disk, consult your authorized dealer or dial 1-800-DATAGEN for contract information.
BAD BLOCK CONTAINED IN REMAP AREA SPECIFIED. PLEASE SPECIFY ANOTHER AREA.	From DKINIT. The remap area, which provides substitutes for bad blocks, contains a bad block. For a diskette, subtract 22 from the last bad block number displayed and type this number. For a hard disk, subtract 100 from the last number displayed and type this number. Repeat this step until DKINIT accepts your answer.
Bad block discovered on destination file/device	From FCOPY diskette transfer program. The destination diskette has one or more bad blocks. To duplicate a diskette, you must use another diskette. You may be able to use the diskette as a directory (DKINIT FULL and INIT/F commands) but you can't duplicate a diskette on it. To copy a *file*, run a a DKINIT PARTIAL format on the destination diskette and try again.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
BAD BLOCK LIST IS CORRUPT – ABORTING	From DKINIT. The bad block list is invalid. Run a FULL command to create a new bad block list.
BAD BLOCK LIST IS FULL. UNABLE TO ENTER ANY MORE BAD BLOCKS	From DKINIT. The disk(ette) may need hardware formatting. Usually, for a diskette, you should discard the diskette (you can try hardware formatting with Customer Diagnostics). For a hard disk, you should consult your authorized dealer or dial 1-800-DATAGEN for contract information.
BLOCK IS PART OF BAD BLOCK REMAP AREA. IT CAN ONLY BE DECLARED BAD USING FULL INIT OR REMAP	From DKINIT. The block you declared bad is part of the remap area, where bad blocks are not allowed. You'll need to run a FULL command — adding 200 or so to the address DKINIT displays for the default *REMAP AREA START...* If the device involved is a hard disk, you might want to seek help from DG (described later in this chapter).
BOOT.SV NOT FOUND	From bootstrap root loader program. Either the disk(ette) you want to start from has no bootstrap root, or the bootstrap root is not of the current revision. Starting with DG/RDOS revision 1.10, you must use program MBOOT.SV (not BOOT.SV) to install the bootstrap. Restart (26H, and specify program **MBOOT**↲; install a bootstrap on the disk(ette) from which you couldn't start; and try again to start from this disk(ette).
BREAK	From the CLI. This can appear after you abort a program with CTRL-C CTRL-B. DG/RDOS has created a memory-image file named BREAK.SV (or FBREAK.SV for a foreground program) in the current directory. You can delete this file if you want. If you abort a program with CTRL-C CTRL-B, any file the program was writing to may not have been closed normally; it may contain inconsistent data.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
CHECKSUM ERROR	From DG/RDOS or a support program. The drive hardware couldn't read the diskette or tape. Retry. If the error recurs, try another diskette/tape or different drive (if possible). For a tape drive, sometimes cleaning the tape or cleaning the unit's read-write heads with an alcohol-soaked cotton swab will help. For a diskette drive, you might try the Customer Diagnostics diskette cleaning program.
CRITICAL DISK BLOCKS ARE BAD. additional message	From DKINIT. The disk or diskette is unusable. Possibly, hardware formatting will help.
DEVICE ALREADY IN SYSTEM	From the CLI. You tried to initialize (INIT) a directory that is already initialized. The system can't initialize it again.
DEVICE NOT IN SYSTEM	From the CLI. You tried access to a tape or disk unit without initializing it. For tape, type INIT MT0↵; for a disk(ette) based directory, you can use either the INIT or DIR command.
DEVICE TIMEOUT: device	From the CLI. The system has tried for 20 seconds to access the device, but it cannot do so. For a diskette, this probably means that the diskette is misinserted in the unit — or not inserted at all. Check the diskette in the drive.
DIRECT I/O ACCESS ONLY	From the CLI. A program tried to perform nondirect-block I/O on a file that requires direct-block I/O.
DIRECTORY DEPTH EXCEEDED	From the CLI. You (or a LOAD command) tried to create a subdirectory within a subdirectory, or a secondary partition within a secondary partition. Change directories or commands as appropriate, and retry.
Directory depth exceeded	From IMOVE, on a restore. The current directory when you started the restore was the wrong one. IMOVE is trying to recreate a subdirectory within a subdirectory or a secondary partition within a secondary partition. Abort the restore with CTRL-C CTRL-A. Get to the directory from which the backup was started (usually DE0 or DE1), and restart the restore.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action

DIRECTORY IN USE	From the CLI. It thinks that the directory you tried to delete or rename is in use. First, try releasing the directory by typing `RELEASE dirname↵` on the system console; then retry the DELETE command.
	
	If the message recurs, the foreground program may be using the directory. If so, you might not want to delete or rename the directory. If you do want to delete or rename it, have the foreground release it, then retry.
	
	If the message recurs *again*, there may be a file open in the directory. For example, you can't release the directory from which you started logging (LOG command) until you type ENDLOG. Check for open files with the LIST/U/S command. Any use count other than 0 indicates a file in use.
	
	If there are any use counts above 0 (aside from SYS.DR), try clearing use counts to 0, via the following commands:
	
	`DIR dir-name ↵`
	`CLEAR/A/V/D;    CLEAR    LOG.CM ↵`
	`CLEARED SYS.DR`
	`R`
	`RELEASE dir-name ↵`
	`R`
	
	Then retry the delete or rename. If the foreground is running, the system won't let you use the CLEAR command. So, if the foreground is running, you'll need to wait until it's terminated, or terminate it yourself, before clearing use counts.

Directory is in use, DJn	From IMOVE. Type CTRL-C CTRL-A, then type `RELEASE DJn ↵` and repeat the IMOVE command.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
DIRECTORY NOT INITIALIZED: COM.CM	You've released the master directory (DJ0 or DE0), which also releases other directories. While there is no initialized directory, you can't access files on any disk(ette). To access a file, DIR to a disk(ette); for example, type `DIR DJ0↵` and proceed. To shut down, type the break sequence (CMD and BREAK/ESC keys).
DIRECTORY SHARED	From the CLI. After you release a directory (RELEASE), this message reminds you that the other ground has the directory initialized. The message is informational only. Take no action.
DIRECTORY SIZE INSUFFICIENT	You (or a program) tried to create a secondary partition smaller than 48 blocks long. Repeat with an argument of 48 or more.
DISK FORMAT ERROR	From the CLI. The disk(ette) is not in DG/RDOS directory format. It has been hardware formatted, but not software formatted (it hasn't had a DKINIT FULL command run on it and INIT/F typed to it). The next step really depends on what you want to do. The problem may be the wrong diskette (check diskette label) or wrong command (for example, to back up to diskette using IMOVE, you don't need INIT or DIR). Or, if you want to use INIT or DIR, you must run a DKINIT FULL and INIT/F command to make the diskette into a directory.
DISK ID IS INCORRECT	From DKINIT. The disk(ette) has never been full formatted by DKINIT. Verify that the disk(ette) is not part of a backup sequence (like an IMOVE backup). Then, if you still want to make a DG/RDOS directory from the diskette, run a DKINIT FULL command on it. If this message occurs on a hard disk, you might want to seek help from DG (described later in this chapter).

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
DISK SPACE EXHAUSTED: file	From the CLI. This message means that your command cannot be completed because the current disk (or secondary partition) is full. If *file* is a secondary partition, or nonmaster disk, you will probable want to delete some files to make it usable; use the LIST date/B command to identify old files. If *file* is the master directory, you *must* free some space for system operations to continue. Check for files to delete with the LIST date/B command. With a hard disk, you must plan some procedure for backing up and deleting *many* files — there should be a minimum of 500-1000 blocks free on a hard disk. If this is a multiuser system, ask users which files they don't need, and delete them. Details on restoring the hard disk appear in Chapter 7, under "Restoring the Entire Hard Disk."
END OF FILE	From the CLI. The system hit an end of file when it tried to execute your command. Often, with tape, this means you specified a nonexistent file number. Try a lower number, for example, MT0:0.
ERROR message	Try to find the *message* in this table.

EXPANDED ERROR TEXT NOT AVAILABLE	From SPEED text editor. SPEED needs two files for its error message text. They are CLI.ER and SPEED.ER. Leave SPEED by typing H ESC ESC and fix it as follows.
	
	If you're running SPEED from the master directory, this message means that one or both files aren't available. Perhaps they weren't MOVEd over from the Development Kit diskette. Check with the LIST command. If you're running SPEED from a nonmaster directory, create needed links by typing
	
	`link    cli.er    %mdir%:cli.er ↵`
	
	`link    speed.er    %mdir%:speed.er ↵`
	
	Then, retry the SPEED command.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
FATAL SYSTEM ERROR	From Customer Diagnostics diskette hardware formatting program. Remove and reinsert the diskette, and retry the formatting operation. If it fails again, discard the diskette.
FILE ALREADY EXISTS	From the CLI. You tried to create a file whose name already exists in the pertinent directory. Or, you tried to MOVE, IMOVE, LOAD, or XFER such a file without appropriate switches (/R or /O; or, for XFER, add /B or delete the file and type XFER again).
FILE ATTRIBUTE PROTECTED	You tried to change the attributes of a file who attributes were fixed with the .CHATR system call. The attributes can't be changed.
FILE DATA ERROR: file	From the CLI. There's an inconsistency in the file you're trying to access. It displays this error on most read or write errors to disk files. If *file* is a diskette name (DJ0, DJ1), this message probably means that the diskette you tried to initialize (INIT or DIR) has not been hardware formatted. (The message is *SYS.DR ERROR* if the diskette has been hardware formatted but not software formatted.) If *file* is SYS.DR or MAP.DR, this means a system directory has inconsistent information in it. Probably, an INIT/F command is needed on the disk(ette) to fix things. You can try to back up existing files before doing the INIT/F, but the backup may itself be inconsistent.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action

FILE DOES NOT EXIST: name	From the CLI. The system can't find the file in the current or specified directory. The problem may be
	· You made a typing error.
	· The file exists, but you forgot a needed extension; for example, save files have the .SV extension, which you must type (except to execute). CLI macro files have the extension .MC; and directory names end in .DR (for IMOVE commands or DUMP commands).
	· The file exists, but in another directory, and there is no (correct) link to the file in its parent directory. This often happens after you create a new directory, DIR to it, and try to execute a program (like SPEED.SV). Use the LINK command to create a link, preferably with the same name, to the directory and file. The link to a save file must have the .SV extension — it won't work otherwise. For example,
	`link speed.sv   %mdir%:speed.sv↲`
	`link speed.er   %mdir%:speed.er↲`
	Generally, system files that many people use are installed in a specific directory (often the master directory). A pathname (directory:file), link entry, or macro is needed to access the file.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
FILE DOES NOT EXIST name (continued)	A link allows you to access the file from the directory at almost no cost in disk space. · If file *name* is DJ0, DJ1, or another disk name, this means that the disk has not been fully initialized with INIT/F. Verify that this is the right disk(ette). Then, if you still want to access it as a directory, type INIT/F to it, and confirm with Y. · Lastly, the file really doesn't exist.
FILE IN USE: file	From the CLI. You tried to read, rename, or delete a file that the system thinks is in use. If a file is open at abnormal shutdown, its use count remains nonzero when DG/RDOS comes up again; DG/RDOS thus assumes that someone is still using the file. You can check a file's use count by typing `LIST/U filename↲`. To zero the use count, type `CLEAR/A/V/D↲`. If this doesn't help, type `CLEAR/V filename↲`. Note that CLEAR does not *fix* any data inconsistency that may have developed because the file was left open.
File is not a contiguous or random file.	From FCOPY or CSSORT. The disk file you want to copy or sort is a sequential file (with no letter characteristic). FCOPY can't copy this file. To copy or sort it, transfer it to a randomly organized file as follows: `XFER    filename    newfilename/R ↲` Then retry the utility, specifying newfilename. To save disk space, you might want to delete the old file and rename the new to the old.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
FILE NOT FOUND: xxxx.SV	From the bootstrap program, at startup. It can't find the file you specified. Programs you can start from the *Filename?* question include DG/RDOS systems (DGRDOS.SV), DKINIT.SV, MBOOT.SV, YBURST, and YDBURST.
File read error	From the IMOVE program. Probably the diskette is not hardware formatted. Or maybe it was misinserted (maybe backwards). Check; if diskette was supplied by DG or you know it was hardware formatted, remove and reinsert, and press ↵.
FILE READ PROTECTED	From the CLI. You can't read the file because it has the R (read-protect) attribute. If you need to read it, type **CHATR filename -R↵**, then retry the command.
FILE SPACE EXHAUSTED	From the CLI. All tape has been used; yet more disk-based material remains to be copied. The backup to this tape file is incomplete; retype the command to a tape with more space left. For disk, see message *DISK SPACE EXHAUSTED.*
FILE WRITE PROTECTED	From text editor or CLI. You can't change the file because it has the W (write-protect) attribute. If you need to change it, type **CHATR filename -W↵**, then retry the command.
FILES MUST EXIST IN THE SAME DIRECTORY	From the CLI. Your command (probably RENAME) requires both files to be in the same directory.
FG TERM	From the background CLI. The foreground program has terminated. Either you did it from the system console with CTRL-C CTRL-F, or the program terminated normally, or the program hit a fatal error condition.
FOREGROUND ALREADY RUNNING	From the CLI. While a foreground program is running, you can't: 1) use SMEM to change memory allotments; 2) CLEAR file use counts to 0; 3) execute a program in the foreground; or 4) shut down (BYE↵). Before you can do any of these things, you must terminate the foreground program with CTRL-C CTRL-F.
ILLEGAL ARGUMENT	From the CLI. You specified an illegal argument.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
ILLEGAL ATTRIBUTE	From the CLI. You specified an illegal attribute (with CHATR). See the CHATR command in Chapter 5 for more detail.
ILLEGAL BLOCK TYPE	From the CLI. You tried to LOAD a file that was copied with XFER or to XFER a file that was copied with DUMP. Use the appropriate command.
ILLEGAL CHARACTER	From system, at startup, when you're trying to type the date and time. Turn computer power off and on, and try again. If this fails, use the numeric keypad, to the right of the main keypad, to type numbers. This keypad has only numbers on it and always produces numbers without relying on an emulator.
ILLEGAL DISK UNIT DECLARATION	From DKINIT. It doesn't recognize the disk unit name you typed. Legal names for DG/RDOS disks are DJ0, DJ1, DE0, and DE1. The problem may be lowercase, which DKINIT doesn't understand. Make sure the terminal is in alpha lock and try again.
ILLEGAL DIRECTORY NAME	From the CLI. The directory name you specified was illegal. Legal filename characters are A-Z, 0-9, and $.
ILLEGAL FILE NAME	From the CLI. The filename you specified was illegal. Legal filename characters are A-Z, 0-9, and $.
ILLEGAL INDIRECT FILENAME	From the CLI. The indirect filename you specified was illegal. Retype the command, using the form @filename@ (legal filename characters are A-Z, 0-9, and $).
ILLEGAL NUMERIC ARGUMENT	From the CLI. The problem may be: there is a nonnumeric character in a numeric argument; or the numeric argument is too large; or that the radix is wrong.
ILLEGAL PARTITION VALUE	From the CLI, after your SMEM command. You tried to: allocate more memory to the background than is available to both grounds; or allocate too little memory for the background CLI (less than 20 pages). Retry the SMEM command.
ILLEGAL TEXT ARGUMENT	From the CLI. You omitted a matching quotation mark (") in a MESSAGE command.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
ILLEGAL VARIABLE	From the CLI. The variable you typed doesn't exist, or is illegal. Retype the command, using the form %variable% (legal variables — for example, %DATE%, are described in the CLI manual).
Incorrect diskette	From FCOPY. The diskette you want FCOPY to write to (either to copy a file or to duplicate a diskette) is not the original destination diskette. Or, the diskette you want FCOPY to read from is not the original *source* diskette. Remove this diskette, insert the correct one (source or destination), and press ⤶ to continue.
INSUFFICIENT CONTIGUOUS BLOCKS	From the CLI. There are not enough contiguous free blocks on the disk to hold the contiguous file you're trying to LOAD (MOVE) onto it. The disk is probably nearly full; take action described under message *DISK SPACE EXHAUSTED.*
INSUFFICIENT MEMORY TO EXECUTE PROGRAM	From the CLI. There isn't enough memory available in the foreground to execute this program. Give the foreground more memory. Use GMEM to get memory allotment (n). If the foreground (FG) doesn't have at least 16 (pages), give it 16 pages (SMEM n-16 16⤶); and retry the EXFG command. Keep giving the foreground more memory, in increments of 2, until the EXFG command or you get an error message. Another option is to check the product Release Notice for the amount of memory required.
INSUFFICIENT NUMBER OF CONTIGUOUS FREE BLOCKS	There isn't enough contiguous space for you to create the file. Take action described under *DISK SPACE EXHAUSTED.*
INT	From the CLI. You interrupted the CLI command with CTRL-C CTRL-A. If the latter, any file the program was writing to may not have been closed normally; it may contain inconsistent data.
INVALID BAD BLOCK TABLE	From the CLI. The disk(ette) cannot be initialized in its current state. Run a DKINIT PARTIAL (if this fails, DKINIT FULL) command on the diskette; see "About Bad Blocks," in the next section.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
Invalid dump block type, DJn	From IMOVE during a restore. The diskette was not written by IMOVE — perhaps it was written by another kind of dump program. You may need to abort and restart IMOVE from the beginning, using the correct backup diskettes.
INVALID FILENAME	From IMOVE. You specified an invalid filename — perhaps an illegal template character like MYDIR:FILE-. Templates in pathname specifiers are illegal in DG/RDOS. Type the full pathname or just the directory name and .DR (for example, MYDIR.DR).
INVALID TIME OR DATE	From the CLI. You tried to specify an illegal time or date, or used an illegal character. Type **GTOD**⤶ for an example of the correct format (you can use spaces as separators).
LINE TOO LONG	From the CLI. The command line and system limit for line-oriented text is 133 characters, including the ⤶ delimiter. You typed more than this number. Retype the command.
LINK ACCESS NOT ALLOWED	From the CLI. The file's link attributes (CHLAT command) prevent access via a link entry. See CHLAT in the CLI manual.
LINK DEPTH EXCEEDED	From the CLI. This probably means the file was linked to itself. Remove the link (UNLINK linkname) and recreate it specifying the correct resolution file pathname.
LOG FILE ERROR	From the CLI. Logging was on (LOG), but the CLI could not write to the log file. This may mean you released the directory where the log file was started. If so, restore the diskette to its drive, DIR to it, restart logging, and continue (The original log file has been deleted.) Next time, before releasing the master directory, if you think you *might* be logging, type **ENDLOG**⤶. Then, after inserting and DIRing to the new diskette, type LOG to restart logging on the new diskette.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
MAP.DR ERROR	From the CLI. The space allocation directory has inconsistent information. Probably, an INIT/F is needed to fix things. You can try to back up existing files before doing the INIT/F, but the backup may itself be inconsistent.
NEW ERRORS FOUND ON DISK. UPDATING BAD BLOCK TABLE TO INCLUDE THEM	From DKINIT, on PARTIAL command. This is an informational message. One or more files may be damaged. To discover which, LIST the bad blocks when DKINIT finishes; then run YOWNER as described in the next section.
NO DEBUG ADDRESS	From the CLI. You tried to debug (DEB command) a program that does not include a debugger (RLDR/D switch).
NO DEFAULT DEVICE	From the CLI. You've released the master directory — therefore DG/RDOS has no disk to access for needed information. (This can happen if you type something like `RELEASE DE0:MYDIR↵`, which releases *all* directories involved.) DIR to the master directory (DE0 or DJ0) and continue (all directories that were initialized have been released, so you might want to reinitialize them).
NO FILES MATCH SPECIFIER	From the CLI. There are no files that match your filename template (characters - or *).
NO MORE DCBS	You (or another user or program) tried to initialize a directory via the INIT or DIR command — but the number of directories in use has reached the maximum allowed by CONFIG. For the INIT or DIR command to work, an initialized directory must be released (RELEASE command). If you get this error message often, you may want to increase the "number of directories accessible at one time..." via CONFIG. Or, you may want to reduce the number of directories in your system (by consolidating, then deleting files, and so on).

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
NO ROOM FOR UFTS	From the CLI. There are not enough channels configured for the program ground. The CLI requires 16 channels; other applications may require more. Run CONFIG and specify more channels, as described in Chapter 2. This message *can* mean that a program you built doesn't have enough channels reserved (with RLDR /C switch).
NO SOURCE FILE SPECIFIED	From a utility like the MAC assembler. You omitted a source filename from the command line.
NO STARTING ADDRESS FOR LOAD MODULE	From RLDR utility. The user program was written without a starting address and cannot be executed. Using a text editor, specify a starting address next the .END pseudo-op.
NO SUCH DIRECTORY: file	DG/RDOS can't find the directory. This may mean that the directory, or its parent directory, isn't initialized. You can check directory names within the current directory by typing `LIST -.DR↵`. Use the INIT or DIR command to initialize the directory; then try again.
NOT A LINK ENTRY	You tried to UNLINK a nonlink file. To remove nonlink files, use DELETE.
NOT A SAVE FILE	You can't execute any program that's not a save (.SV) file. It must also be a randomly-organized file and have the S attribute. Check with LIST and give the S (CHATR filename + S) if needed. If the file isn't randomly organized (R), copy it to a random file by typing `XFER filename newname/R↵`
NOT ENOUGH ARGUMENTS	From the CLI. More arguments are needed; see the command format in Chapter 5 or the CLI manual.
NOT ENOUGH ROOM FOR UPDATE. SUGGEST FULL INIT	From DKINIT. A FULL command is needed on the disk(ette). You can try to bring up DG/RDOS and back up files before running DKINIT FULL, if you want.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
OUT OF TCB'S	From the CLI. Your user program needs to have more tasks defined. Run RLDR on it again and specify more tasks (ntasks/K switch).
PARITY ERROR	From the CLI. For action, see message *CHECKSUM ERROR*.
PERMANENT FILE: name	From the CLI. You tried to delete a permanent file (P attribute). This can happen after a DELETE command or after a MOVE, IMOVE or LOAD command that specifies deletion (the /R or /O switch). If you really want to delete the file, remove its P characteristic (CHATR filename -P)). (Some system files are attribute protected, which means you can't remove a P attribute.)
PROGRAM ERROR:message	From CSSORT. See the message in Table 11-1, end of Chapter 11.
QTY ERROR	From the CLI. This means a USAM line error — probably a simultaneous read or write to the same line.
RDOS ERROR: n	From a system utility. It could not interpret the error code returned. This can happen if the program lacks disk space to create a needed file. It can also indicate a revision clash (incompatible revisions of the utility and DG/RDOS system).
Source disk block n bad, 'OWNER' can find any bad file. Select continue to skip over and continue	From FCOPY. FCOPY can't read a block on the source diskette. The block will not contain reliable information on the destination diskette. Run DKINIT PARTIAL and YOWNER (described in the next section) on the source diskette. Identify, delete, and reload the damaged file (if any). Then, if the destination diskette is a DG/RDOS directory, delete the same file on *it* and use the MOVE command (or FCOPY) to copy the reloaded file to the destination diskette.
Source disk contains marginal blocks	From FCOPY. One or more of the source disk blocks is marginal (FCOPY couldn't read it the first time but on a later retry *could* read it.) Run DKINIT PARTIAL and YOWNER on the source diskette, as described in the next section (if you care about the source diskette; the copy is okay.)

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
SPOOL FILES ACTIVE	From the CLI, when you try to shut or boot. There is material in the printer spool queue, waiting to be printed. Either wait for printing to stop, or kill the spool queue by typing `SPKILL $LPT↵` then shut down.
STACK OVERFLOW	From the CLI. There isn't enough memory available to execute your command. For some commands, the CLI needs a minimum of 21 Kwords. Check with GMEM, give the CLI at least 21 Kwords with SMEM, and try again. (If the foreground is already running, you'll need to wait until it is terminated — CTRL-C CTRL-F — to change memory with SMEM.)
SYS.DR ERROR: file	From the CLI. There's an inconsistency in the system file directory. If the *file* is a diskette name (DJ0, DJ1), this message probably means that the diskette you tried to initialize (INIT or DIR) has been hardware formatted, but *not* software formatted, and that it has been written to by a program like IMOVE or a different operating system (MS-DOS). For a diskette-based *file*, the next step depends on what you want to do. The problem may be the wrong diskette (check diskette label) or wrong command (for example, use IMOVE, without a preceding INIT or DIR, to restore from IMOVE diskettes). In any case, the directory that provokes the system error message is not reliable as a DG/RDOS directory. You will probably need to type INIT/F to the diskette. (Try to back up files before doing this if you want, although the backup may not be sound.) If the *file* is a hard disk with material you want, you should seek help (described later in this chapter) before typing INIT/F to it.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
SYSTEM LIBRARY NOT FOUND XN TMIN NO SCHEDULER......	From the RLDR linker after you try to link a program. RLDR cannot find the system library, SYS.LB. This library is included with the Development Kit; it should have been installed on DE0 when the Development Kit was installed. Check with the LIST command. Usually, this message means you didn't link SYS.LB from a nonmaster directory. From the nonmaster directory, type `link sys.LB de0:sys.lb ⮐` and try the RLDR command again.
TEXT ARGUMENT TOO LONG	From the CLI. You typed too many characters (more than 72) between quotation marks. Retype with a shorter text string.
This disk contains more than 15 bad sectors	From IMOVE, on a restore. The diskette you inserted has too many blocks. Discard it, substitute another (hardware formatted), labeled diskette, and restart the backup from the beginning.
This disk is not in sequence	From IMOVE, on a restore. The diskette is out of sequence. Insert the correct diskette and IMOVE will continue.
This disk is not part of the dump being processed	From IMOVE, on a restore. The diskette is part of a different IMOVE backup set (perhaps a different incremental set). Find and insert the correct diskette, and IMOVE will continue; or abort the restore and restart it with the correct diskette number 1.
THIS IS NOT THE SYSTEM DISKETTE TYPE C TO CONTINUE.....	From LOADEM program when you're building a system diskette). The receiving diskette must be a system diskette, with CLI.SV on it. Remove the diskette, replace it with a system diskette, and type C to continue.
TOO MANY ARGUMENTS	From the CLI. You typed too many arguments (maybe you accidentally inserted an extra space). See the command format in Chapter 5 or in the CLI manual.

Table 14-1 Error messages and recovery (continued)

Message	Source, Possible Cause(s), and Action
TOO MANY DISK ERRORS TO COMPLETE	From DKINIT. The disk(ette) has too many errors (bad blocks or address errors) to be usable. For a diskette, use the Customer Diagnostics diskette to hardware format this diskette. If this occurs while you're formatting a hard disk, consult your authorized dealer or dial 1-800-DATAGEN for contract information.
TRAP BREAK	An internal error halted the program. Retry. If this is a DG program, you may want to call your dealer or dial 1-800-DATAGEN for contact information. See also the message for BREAK.
Unable to read from destination diskette - Verification fails	From FCOPY program. On verification, FCOPY could not access the diskette. See message Unable to write to..
Unable to read source file/device	From FCOPY program. FCOPY was able to access the source diskette, but for some reason is not able to read it. Bad blocks are not the cause. Restart the FCOPY.
Unable to write to destination file/device	From FCOPY diskette transfer program. FCOPY can't write to the diskette. This may mean that the destination diskette is not hardware formatted, or that it's write protected (tape over notch). Remove and check the diskette. If you don't know that it's hardware formatted, hardware format it (Chapter 6). If it's write protected, remove the tape strip.
UNIT n FATAL ERROR... (text advice) *Hit any key to continue.*	From the hardware diskette formatter (Customer Diagnostics diskette). Remove and reinsert the diskette to be formatted, and repeat the previous steps.
UNIT IMPROPERLY SELECTED	From the CLI. You tried to access the tape drive, but the drive is not turned on, or not ON LINE.
UNKNOWN COMMAND - COMMAND IS OWNER	From YOWNER. This program recognizes only the command OWNER, which must be uppercase. Make sure the terminal is in alpha lock and type **OWNER CR**.

Table 14-1 Error messages and recovery (concluded)

Message	Source, Possible Cause(s), and Action
UNKNOWN COMMAND - COMMANDS ARE ...	From DKINIT, YBURST, or YOWNER. The program doesn't recognize the command. For DKINIT, the only commands you need in DG/RDOS are LIST, PARTIAL, FULL, DISK, and STOP. Commands must be uppercase. Make sure the terminal is in alpha lock and retype the command.
Unreadable block...	From FCOPY program. See message under Source disk block n.
USER ERROR: message	From the DG/RDOS sort/merge program, CSSORT. See the message in Table 11-1, end of Chapter 11.
VERIFYING DISKETTE (n minutes)	From the Customer Diagnostics diskette formatter. It's verifying the integrity of the format, as specified. After n minutes, it will have either verified format or displayed an error message.
YOU CAN'T DO THAT	From the CLI. Logging was started with a password, and you typed ENDLOG without the password. If you can't remember/discover the password, you can shut down DG/RDOS to close the log file.
Wrong disk. Please mount disk n and type NEWLINE to continue.	From the IMOVE program on a restore. This diskette is out of sequence. From the original group of diskettes used for the backup, find the correct diskette (n) insert it, and press ↓.

About New Bad Blocks

As time passes, the magnetic qualities of media may change — especially with diskettes — and one or more disk blocks may lose its ability to retain information. When this happens, DG/RDOS will be unable to read or write the file that owns the bad block. You may see the message

```
FILE DATA ERROR: file
```

which tells you the damaged filename. This message *usually* means that a new bad block has developed.

If the new bad block is in a critical place (like the system), DG/RDOS may panic:

```
nnnnnn nnnnnn nnnnnn nnnnnn nnnnnnn
```

or hang, doing nothing. To check for, and recover from, a new bad block, read the next section.

Finding and Fixing New Bad Blocks

The first step is to identify the new bad block(s) in such a way that they can be remapped, allowing the system to avoid them in the future.

For hard disks, there may be a special diagnostic program (described in the *Testing* manual) that does exhaustive checking to identify new bad blocks. Check the *Testing* manual to see; and consider running the diagnostic program if described and if you have a hard disk. Otherwise, continue.

Now, you should get DKINIT to identify the new bad block(s) and remap them. This may result in the loss of part or all of a file, but this loss is unavoidable. The main point is to save as many files as possible on the disk(ette).

After DKINIT, you'll want to learn which file has been damaged, so that you can delete it and load a copy from backup media. The program that tells you the "owner" file of a bad block is called YOWNER.SV.

In any case, you'll need DKINIT.SV. If you don't know which filename owns the new bad block, you'll also need YOWNER.SV. These programs are on the DG/RDOS Stand-Alone Utilities diskette. We suggest using this diskette. You can run the programs from the hard disk, if you have one; but if a new bad block is in one of the program files, this won't work and you'll need the utilities diskette anyway.)

Follow these steps:

1. Shut down DG/RDOS if it's not already down.

2. Put the terminal in alpha lock (press ALPHA LOCK until the ALPHA LOCK light glows).

2a. With a hard disk, check the *Testing* manual for description of a hard-disk diagnostic program; and run this diagnostic program if you decide it's a good idea to do so.

3. Unless you will run DKINIT from the hard disk, insert the DG-supplied diskette that has DKINIT in drive DJ0.

4. Start up from diskette (or disk):

```
! 20H          (or 26H for hard disk)
FILENAME?
```

5. Type `DKINIT ↲`

```
DISK INITIALIZER REV x.xx
   DISK DRIVE MODEL NUMBER?
```

6. Type the model number and CR (you must use the CR key, not ↲, with DKINIT and YOWNER). For diskette, the model number is 6268 for a 360 Kbyte diskette. For a hard disk, it's 6271 (15-Mbyte disk) or 6301 (39-Mbyte disk).

```
DISK UNIT?
```

7. If you want to check a diskette, make sure it is in a drive. You can remove the DG-supplied diskette from DJ0 if you need a free drive.

   Type the name of the disk drive that holds the suspected disk(ette): DJ0 or DJ1 for diskette; DE0 or DE1 for disk. For example,

```
DE0 CR         (or DJ0 CR; press the CR key)

COMMAND?
```

8. Type

```
LIST CR

6xxx DISK DRIVE ON UNIT xxx
FRAME SIZE =x REMAP AREA SIZE =x
REMAP AREA START BLOCK NUMBER =x
NUMBER OF BAD BLOCKS = n

HEAD SECTOR CYLINDER / BAD BLOCK NUMBER
x x x / number
```

9. On a piece of paper (or on the following blank lines) write the total number of bad blocks (n) and all of the bad block numbers (number), if any numbers appear.

BAD BLOCK LIST FOR UNIT D _

Total number of bad blocks ____	Date ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________
bad block number ____________	bad block number ____________

10. Type `PARTIAL CR` (the command is PARTIAL!)

```
*** CHECKING FOR BAD BLOCKS ***
```

11. DKINIT now does a read analysis on the disk(ette). For a hard disk, this takes 5 or 10 minutes (depending on capacity); for diskette, it takes less time.

DKINIT should, but may not, find the new bad block(s). For every bad block, new or old, that it finds, it displays

```
DISK ERROR - BAD BLOCK=naddr
DATA COMPARE ERROR - BAD BLOCK =number
```

When DKINIT finishes, it displays either

```
NEW ERRORS FOUND ON DISK. UPDATING BAD BLOCK TABLE TO INCLUDE THEM
```

or

```
NO NEW ERRORS FOUND ON DISK.
```

Then it displays

```
PARTIAL INIT RUN COMPLETE
COMMAND?
```

If it says NO NEW ERRORS FOUND ON DISK, this means either that the persistent problem is not a bad block or that a block is marginal — perhaps readable but not writable. You can either rerun the PARTIAL or bring up DG/RDOS and run until it fails again, with hopes that DKINIT will catch the bad block on a later run. If you decide on the former, return to step 10. If the latter, skip the rest of this section.

If DKINIT says NEW ERRORS FOUND ON DISK, the problem probably involves a bad block, which DKINIT will take out of circulation.

12. Type the LIST command again, to check the updated bad block table.

    LIST CR (CR key)

    ```
    6xxx DISK DRIVE ON UNIT xxx
    .
    NUMBER OF BAD BLOCKS = n

    HEAD SECTOR CYLINDER / BAD BLOCK NUMBER
    x x x / naddr
    .
    COMMAND?
    ```

13. Identify the new bad block(s) by comparing the list of bad blocks to the list written above. Add the new bad block numbers to the to the written list, followed by the word "New" and the date; for example, "11267 New February 29, 1984."

14. Stop DKINIT by typing

    ```
    STOP CR              (CR key)
    !
    ```

    The bad block(s) will be bypassed in future. Now, if you know the filename(s) that hold them (perhaps because you got a FILE DATA ERROR, skip to step 23.

15. Insert the DG-supplied diskette that holds DKINIT in drive DJ0, if you removed it earlier.

16. Type `20H` (or, from disk, `26H`)

    ```
    FILENAME?
    ```

17. Type `YOWNER↵`

    ```
    *** DISK BLOCK OWNER - REV x.xx
    COMMAND?
    ```

18. Type

    ```
    OWNER CR             (Press the CR key)

    INPUT DISK UNIT?
    ```

19. Type the name of the disk(ette) drive that holds the disk with the new bad blocks (DJ0, DE0, and so on) and press CR. For example,

    ```
    DJ0 CR               (CR key)

    BLOCK NUMBER (TYPE RETURN TO STOP)?
    ```

20. Type the number of the new bad block, from the list above. If there's more than one new bad block, type the number of the next bad block. For example, if DKINIT reported 00000004167, you'd type

    ```
    4167 CR              (Press CR key)
    ```

    YOWNER then repeats

    ```
    BLOCK NUMBER (TYPE RETURN TO STOP)?
    ```

21. Repeat step 20 until you've typed all the new bad block numbers. Then press CR to the bad block question:

    `CR` (CR key)

22. The YOWNER program now resolves the disk filename(s) from the block number(s) you gave it, then stops running. This takes 20 seconds or so. It displays the names as follows:

    ```
    BLOCK NUMBER n IS CLAIMED BY FILE pathname
    ```

    For example,

    ```
    BLOCK NUMBER 00004167 IS CLAIMED BY FILE DE0:MYDIR.DR:MYFILE
    ```

    Note the path name(s) displayed on a piece of paper (perhaps next to the new bad block number(s) in the list above).

    ```
    *** ALL MATCHES REPORTED ***
    !
    ```

    If you need to rerun YOWNER, return to step 16 above.

    (If YOWNER hangs — does nothing for a minute or more — this probably means that the bad block is the index block of a random file. YOWNER can't deal with this situation. Type the break sequence of CMD and BREAK/ESC keys. Bring up DG/RDOS and try to find some way other than YOWNER to identify the damaged file. For files other than program (.SV) files, the FPRINT command can help you identify a damaged file. FPRINT displays file contents; it skips over identical lines. If the display for a nonprogram file shows skipped file addresses, this may indicate a damaged file. You might also try the TYPE command to check text files.

23. Bring up DG/RDOS as usual: 2nH, ↲, date, and time.

24. The next step involves dealing with the damaged file(s), which are probably not usable in current form (although you may be able to recover parts of text files using a text editor).

    The best way to handle damaged file(s) is to delete each one and reload it from backup diskettes or from DG-supplied DG/RDOS diskettes (if a damaged file is a DG-supplied file) For a non-DG file, delete the file and restore it from backup media as sketched in Chapter 7.

    For a DG file, get the DG/RDOS system diskette, put it in drive DJ0 (with other diskette in unit 1, if you have no hard disk). Then bring up DG/RDOS from diskette (20H), and MOVE the file(s) over, described in Chapter 2, step 36 and following.

(If you're curious about the name YOWNER, it derives from program OWNER. It's the version of OWNER designed for the system that is code named YRDOS. Y (for YRDOS) plus OWNER gives YOWNER.)

How to Get Help

If you acquired your DESKTOP GENERATION system from a authorized dealer, consult this dealer for help. If you *didn't* acquire your system from an authorized dealer, dial

1-800-DATAGEN

for the latest information on service and maintenance contracts.

What Next?

This — the last chapter in the book — described error conditions, recovery, and how to get help.

Next, you may want to review earlier material, or (more likely) just *use* your DESKTOP GENERATION system.

ASCII Character Codes

A

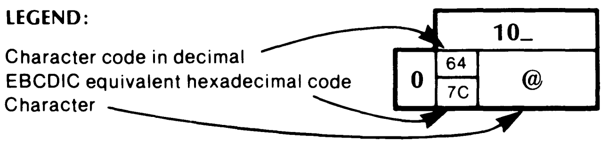
This Appendix describes the 7-bit ASCII characters and their numeric values. These characters and codes are not the same as those in the DG International character set.

To find the *octal* value of a character, locate the character, and combine the first two digits at the top of the character's column with the third digit in the far left column.

OCTAL	00_			01_			02_			03_			04_			05_		
0	0	00	NUL	8	16	BS (BACK-SPACE)	16	10	DLE ↑P	24	18	CAN ↑X	32	40	SPACE	40	4D	(
1	1	01	SOH ↑A	9	05	HT (TAB)	17	11	DC1 ↑Q	25	19	EM ↑Y	33	5A	!	41	5D	)
2	2	02	STX ↑B	10	15	NL (NEW LINE)	18	12	DC2 ↑R	26	3F	SUB ↑Z	34	7F	" (QUOTE)	42	5C	*
3	3	03	ETX ↑C	11	0B	VT (VERT TAB)	19	13	DC3 ↑S	27	27	ESC (ESCAPE)	35	7B	#	43	4E	+
4	4	37	EOT ↑D	12	06	FF (FORM FEED)	20	3C	DC4 ↑T	28	1C	FS ↑\	36	5B	$	44	6B	, (COMMA)
5	5	2D	ENQ ↑E	13	0D	RT (RETURN)	21	3D	NAK ↑U	29	1D	GS ↑]	37	6C	%	45	60	-
6	6	2E	ACK ↑F	14	0E	SO ↑N	22	32	SYN ↑V	30	1E	RS ↑↑	38	50	&	46	4B	. (PERIOD)
7	7	2F	BEL ↑G	15	0F	SI ↑O	23	26	ETB ↑W	31	1F	US ↑←	39	7D	' (APOS)	47	61	/

(continues)

LEGEND:



Character code in octal at top and left of charts.
↑ means CONTROL

OCTAL	06_			07_			10_			11_			12_		
0	48	F0	0	56	F8	8	64	7C	@	72	C8	H	80	D7	P
1	49	F1	1	57	F9	9	65	C1	A	73	C9	I	81	D8	Q
2	50	F2	2	58	7A	:	66	C2	B	74	D1	J	82	D9	R
3	51	F3	3	59	5E	;	67	C3	C	75	D2	K	83	E2	S
4	52	F4	4	60	4C	<	68	C4	D	76	D3	L	84	E3	T
5	53	F5	5	61	7E	=	69	C5	E	77	D4	M	85	E4	U
6	54	F6	6	62	6E	>	70	C6	F	78	D5	N	86	E5	V
7	55	F7	7	63	6F	?	71	C7	G	79	D6	O	87	E6	W

OCTAL	13_			14_			15_			16_			17_		
0	88	E7	X	96	79	\` (GRAVE)	104	88	h	112	97	p	120	A7	x
1	89	E8	Y	97	81	a	105	89	i	113	98	q	121	A8	y
2	90	E9	Z	98	82	b	106	91	j	114	99	r	122	A9	z
3	91	8D	[	99	83	c	107	92	k	115	A2	s	123	C0	{
4	92	E0	\	100	84	d	108	93	l	116	A3	t	124	4F	|
5	93	9D	]	101	85	e	109	94	m	117	A4	u	125	D0	}
6	94	5F	↑ or ^	102	86	f	110	95	n	118	A5	v	126	A1	~ (TILDE)
7	95	6D	← or _	103	87	g	111	96	o	119	A6	w	127	07	DEL (RUBOUT)

(concluded)

Glossary

This glossary describes computer-related terms that may be new to to you — either as words or in relation to DG's software products.

abort — the termination of a program from a serious error *or* from an interrupt sequence; for example, you can abort an executing program by typing CTRL-C CTRL-A.

ANSI — American National Standards Institute, a committee that publishes standards for a large range of things, including computer languages and tapes, machine screws, and copiers.

AOS (Advanced Operating System) — A different DG operating system: the Advanced Operating System for DESKTOP GENERATION and other 16-bit ECLIPSE computers.

AOS/VS (Advanced Operating System/Virtual Storage) — A different DG operating system: the Advanced Operating System for MV/Family 32-bit ECLIPSE computers.

argument — something that is acted upon by a command, statement, or instruction. For example, in `PRINT MYFILE ↓`, MYFILE is an argument to the PRINT command. In `PRINT "Hello"`, "Hello" is an argument to the PRINT statement.

ASCII — American Standard Code for Information Interchange. This code establishes standard numeric values for each character used in text; the numbers range from 000 for the null character to 177 (octal) for the DEL character. An international character set extends the ASCII set with numbers from 200 (octal) to 377 (octal); these numbers indicate non-US, language-specific characters (for example, the UK currency symbol).

asynchronous line — a communications line that uses an asynchronous protocol to transmit characters. In such a protocol, each character has its own "framing" information: traditionally one start bit (before the character) and one stop bit (after the character). Asynchronous lines are generally used for terminals and — sometimes — for intersystem communication.

background program — the program that's running on the system console. At startup DG/RDOS runs the CLI on the system console. From this background CLI, you can execute a different program in the background. If your system has enough memory, you can allot memory to the foreground and run a *foreground program.*

backup — files copied for safekeeping, usually onto magnetic diskettes or tape.

bad block — on the magnetic surface of a disk or diskette, a bad block is a flawed area that won't hold information. The DKINIT software formatter notes such areas so the operating system will avoid them. If DG/RDOS encounters a new bad block, it usually displays a ***FILE DATA ERROR*** message.

BASIC language (Beginner's All-purpose Symbolic Instruction Code) — an easy, interpreted language, originally developed at Dartmouth College. BASICs available with DG/RDOS include Business BASIC, with special business and ISAM capabilities; and Extended BASIC, a traditional BASIC.

baud — the rate at which a line or modem can transfer data, in bits per second. Normally, each character requires 10 bits, so characters are transferred at 1/10 the baud rate. The standard (and default) baud rate for terminals is 9600 (960 characters per second). For modems it is 1200.

bit — a Binary digIT. A bit can assume one of 2 values: 0 or 1. But 16 bits, as used in a DG computer word, can indicate 65,536 different numbers. And 32 bits, in two computer words, can indicate over 4 billion numbers.

boot — to start up an operating system or stand-alone program. Classically, a small program on the beginning of the device reads an larger bootstrap program (here, MBOOT.SV). The bootstrap program in turn asks which file you want to boot; after receiving a valid answer, it loads that program into memory and yields control to it. You can start the bootstrap procedure in different ways: by turning on power (if you have no hard disk), by typing 26H or 20H, or by typing **BOOT system-name** ↲ when DG/RDOS is running.

break character — A break or interrupt character is one that lets you stop program execution. The DG/RDOS break characters for terminals on the multiplexor are CTRL-C or CTRL-A, either of which halts execution of the current program. Break characters for terminals not on the multiplexor, for example the system console, are control character combinations, such as CTRL-C CTRL-A.

buffer — a part of the computer's memory used to receive and temporarily store disk-based information. The information may or may not be changed while in the buffer. If changed, the new information is usually written back to disk — perhaps to the file from which it came. DG/RDOS itself uses buffers (system buffers).

Business BASIC — an enhanced, business-oriented version of the BASIC language.

byte — 8 bits, capable of storing one ASCII character (for example, A) or any number from 0 to 255.

channel, I/O — a data structure used by DG/RDOS to identify and communicate with each open file.

CLI (Command Line Interpreter) — the DG/RDOS command language. CLI commands allow people to communicate with DG/RDOS. They provide control, help maintain files, execute other programs (like SPEED), and do many other things. When you start DG/RDOS it runs the CLI on the system console. If your system has a terminal connected to the printer port and enough memory, you can run a second CLI in the foreground.

CMD key — a key on the terminal keyboard that by itself does nothing, but with other characters can do things like take user terminals off and on line.

COBOL — the COmmon Business Oriented Language, a very popular programming language for business. It features English language construction, with paragraphs, sentences, and clauses. The COBOL available with DG/RDOS is Interactive COBOL (ICOBOL).

cold start — system startup that begins with computer power off, as compared to a warm start.

COM.CM — a command file, built by the CLI, for use by DG/RDOS and some utility programs. It contains the name of the last non-CLI command typed.

command — in this book, a keyword that tells the CLI, or any DG product, what to do.

compiler — a program that translates statements in a high-level programming language (like FORTRAN or COBOL) into binary instructions for the computer. The RLDR program then turns the binary instructions into an executable program file, called a save (.SV) file.

COMPUCALC™ — an electronic spreadsheet program that can help you organize and analyze data in order to create budget forecasts, sales projections, investment analyses, or similar projects.

CONFIG — a program supplied with DG/RDOS that tailors a DG/RDOS system for certain hardware (USAM, printer) and software.

console — another name for terminal. The operator's terminal is called the system console in this book. Other terminals are called user terminals.

control key — see *CTRL* key.

CPU (central processing unit) — one of the three sections of a computer system. The others are Input/Output and Main Memory. The CPU decodes and executes program instructions, performs arithmetic and logical operations, and holds critical data.

CP/M (RM)-86 operating system — stands for "Control Program Monitor", with an 8086 central processing unit. CP/M-86 is a widely used operating system for microcomputers developed by Digital Research. CP/M-86 and support software are available with DG/RDOS.

CR — the CR key, or the CR character, which acts as a line terminator for certain DG/RDOS programs. (Internally, DG/RDOS translates all ↓ NEW LINE characters into CR, so — internally — CR is the main line terminator.)

crash — see panic.

CRT (Cathode Ray Tube) — a terminal with a keyboard and television-like video screen that displays characters.

CSSORT — the Sort/Merge program for DG/RDOS systems. It helps users edit, sort, reorder, and reformat records.

CTRL key — a key, like the SHIFT or CMD key, that does nothing by itself but can do a lot with other keys. CTRL sequences are used to suspend and restore screen display and interrupt commands and programs.

current directory — the directory where you currently are; the name displayed by the GDIR⤶ command.

cursor — on a terminal screen, the cursor indicates the current position on a line. The cursor is analagous to the position of a pencil point on paper. It is a box superimposed on a character position.

data tablet — a large tablet with an electric grid beneath its surface that uses either a cursor puck or a stylus to generate input signals.

database — an information structure (usually kept in one or more files) that a program requires for proper operation.

data-sensitive record — A type of record delimited by a special, agreed-upon character. A standard delimiter is CR (translated from ⤶). Data-sensitive records are also known as *line-sequential* records.

deadlock — a condition in which a system is frozen: unable to act, or respond. Usually you must break a deadlock manually, perhaps by turning power off and on.

default, by default — a value or parameter that a program uses if you do nothing about it. For example, the CLI command LIST displays by default an unsorted list of filenames. But the LIST/S command displays the filenames sorted.

delimiter — a special character that ends each record in a data-sensitive (line-sequential) file. The system treats all characters up to the next special character as a record. A common delimiter is CR (translated by DG/RDOS from ⤶). When you type a CLI command, the ⤶ you press at the end becomes the delimiter.

device code — the number by which a computer recognizes an attached device, like a disk controller, keyboard, or USAM multiplexor. For example, the device code of the diskette controller (which runs both drives) is 20 octal.

DG/BLAST — a communications product that allows you to transfer files between systems over a communications line.

DG/GATE™ — a communications product that allows a DESKTOP GENERATION system to emulate a terminal produced by any of several major manufacturers — enabling you to to dial up and tap into another computer system (for example, an information pool with electronic bulletin boards or closing market prices). DG/GATE can record all intersystem dialog in a disk file for later review or printing.

DG/XAP™ — a communications product that allows you to transfer files between systems over a communications line.

DIP (dual in-line package) switch — a very small switch, often in groups of eight, that enables or disables a hardware function; for example, 8-bit operation on a console. An alternative to DIP switches is hardware (wire) jumpers, but DIP switches are far more convenient. You can change the setting of DIP switch with a pencil point.

directory — a file whose sole function is to contain other files. Directories can help you organize and keep track of your files; the system itself uses them for this purpose.

disk (hard) — a fast, mass storage device, with metal platters that rotate rapidly. The platters have a magnetic coating that is written to and read from. Standard DESKTOP GENERATION system hard disks hold either 39 or 15 million bytes (characters). Compare to diskette.

diskette — a flexible disk, with the magnetic coating on plastic, ranging in size from 3 to 8 inches. The standard diskette for DG/RDOS systems is 5 1/4 inches — a common size. It holds 360 Kbytes (368,000 characters). It is a double-sided, double-density diskette. CPM/M-86 and MS-DOS can also use 5-1/4 inch diskettes.

DKINIT — a DG/RDOS utility program that formats diskettes and hard disks and checks disk surfaces for flaws (bad blocks).

dump — In data processing, dump means "to copy". Often the copy is done for safekeeping, as for backup.

echo — to confirm a character by displaying it. For example, when you type a character on the keyboard, DG/RDOS reads it and echoes it on the terminal screen.

emulator — a program that enables a terminal or computer system to act like a specific (other) type of terminal.

fatal error — see *panic.*

file — a collection of information stored under a *filename.* Some device filenames are rigidly defined (for example, DJ0 for diskette); but user filenames are flexible (also see name).

filename — see *name.*

firmware — instructions that control some aspect of computer hardware. For example, in Model 10/SP computers, the terminal emulator program is a type of firmware. (The system console starts with limited ability (on power up); the emulator enables it for character and graphics handling.) A different type of firmware — which retains state when power is turned off — is used in the computer itself.

fixed record — (also fixed-length or fixed-sequential record). A fixed-length group of characters defined to be a data record. This contrasts with data-sensitive/line-sequential. With fixed records, the system does not look for a delimiter; it just considers each group of characters as a record. Fixed records can be up to 4,096 characters each.

floating-point unit (FPU) — a PC board that speeds up computations with floating-point numbers (numbers that have a decimal point). A hardware FPU is optional on some DESKTOP GENERATION systems, standard on others. Systems without an FPU use *firmware* for floating-point operations.

floppy — another name for a diskette.

foreground console — a terminal attached to the computer printer port; a second CLI and/or other software can run on this terminal.

foreground program — a second program that DG/RDOS can run simultaneously with the background program. See also *background.*

form feed — a character (CTRL-L) that tells the printer to stop printing on the current page and start at the top of the next page. Typed on the terminal screen, a form feed clears the screen.

FORTRAN — contraction of Formula Translator, FORTRAN is one of the oldest and most popular programming languages. There are two different FORTRAN compilers available with DG/RDOS systems: FORTRAN 5 and FORTRAN IV.

function key — one of the keys in the topmost row of a terminal keyboard. Each key, alone or in conjunction with the SHIFT and/or CTRL keys, can represent a command. (Pressing a key is easier than typing a command.) A product's function keys (if it has any) are identified by a shaped *template* that fits over them.

hang — see deadlock.

ICOBOL — Interactive COBOL, a DG COBOL that runs on all Data General computers.

I/O (Input/Output) — The process of reading information from a device into the computer's main memory (input) and/or writing information from memory (output). The input can come from, and the output go to: disk files, diskettes, a terminal, telephone lines, or microwave beams.

I/O channel — see channel.

ISAM — Indexed Sequential Access Method. This is a file structure offered by ICOBOL.

Kbyte — 1,024 bytes (1,024 characters). A 360-Kbyte diskette holds 368,640 characters.

line (communications) — see asynchronous line.

line (of text) — a sequence of ASCII characters that ends with either a NEW LINE, form feed, or null character.

line-sequential record — a type of record that ends with a special, agreed-upon character. A standard delimiter is CR (translated from ↵). Line-sequential is another term for data-sensitive records.

link entry — a file whose sole function is to indicate the location of another file; created with the LINK command. A link consumes almost no disk space, yet gives easy access to the file. Links are often used in subdirectories to produce access to often used files in the master directory.

loopback mode — a way of testing communications software by having a computer's transmissions come back to it (instead of going to another system). This allows local problems to be identified quickly. For normal operation, loopback mode is disabled. (These options are offered by the CONFIG program.)

macro — a sequence of instructions or commands that can be accessed by a single name. Macros are primarily timesavers, allowing people to write a series of commands only once, then execute them all by one name.

main memory — memory inside the computer, directly accessible to it. Disks provide another kind of memory, usually called "storage." Access to main memory is thousands of times faster than access to a hard disk — which in turn is much faster than access to a diskette. DESKTOP GENERATION computers are available with main memory capacity of 2,000,000 characters. DG/RDOS itself uses only 31 Kbyte of main memory.

Mbyte — Abbreviation for megabyte. In terms of computer memory, a megabyte means 1,048,576 bytes (characters); two megabytes of main memory can hold 2,097,152 characters. In terms of disk storage, a megabyte means 1,000,000 bytes; a 15-Mbyte disk can hold 15,000,000 characters.

modem — a device that connects a remote terminal to a computer over a telephone line. One modem is needed at each site. From the remote site, you dial the destination computer's number, wait for a tone, then connect the modem (either via a switch or by inserting the phone receiver into the modem).

monitor — the system console display screen.

mouse — an input device that you move across a flat surface. Movements are translated as coordinates, which a program then uses to move a cursor or draw a picture.

multiplexor — a board that allows a computer system to manage a communications line, user terminals, and a printer. It sorts the incoming signals for the computer, and ensures that the computer's response goes to the correct line. The multiplexor for DESKTOP GENERATION systems is called a *USAM.*

name — DG/RDOS filenames can be from 1 to 10 characters, including letters, numbers, and $. A filename can (but need not) have a two character extension that follows the name proper. Extended BASIC variable names are a letter and optional number. Business BASIC variable names can be up to 6 numbers and letters, and FORTRAN names can be from 1 to 32 letters and numbers. Names for these languages must begin with a letter. COBOL names can be from 1 to 32 UPPERCASE letters, numbers, and dash (-).

NEW LINE — a character (produced by the NEW LINE key, shown in this book as ⤶) that ends a line of text and starts the next line. It terminates commands to the CLI and other programs.

on line — in direct communication with the computer and under its control. For example, when a terminal is on line, the computer reads from the terminal keyboard and writes to its screen. When a terminal is off line, the computer ignores it.

operating system — a large program that manages and operates devices for users and user programs.

panic — what happens when an operating system hits a fatal error condition (an error so serious that the system cannot or dares not recover from it). The system console then prints a panic message, five groups of six digits.

parity — a method used to ensure that a character has been sent and received correctly. There are several different kinds of parity. DG/RDOS uses even parity, which means that the sending device sets the eighth bit so that the sum of all bits in the character is an even number. The receiving device then adds all the bits received and — if the sum is odd, it rejects the character with a parity error message. Parity checking is not allowed with 8-bit characters, since the 8th bit is part of the character code.

patch — a correction to a program, applied directly to the program file on disk.

pathname — a path to a file. DG/RDOS pathnames include one directory name and the filename.

pregen — a pregenerated (ready-made) version of a program. For example, DG/RDOS is a pregen RDOS that has an operating system generated and ready to configure. Pregen systems are designed to make life easier for their users.

printer — a printing device. Several printers are available with DESKTOP GENERATION systems, including the model 4434 printer (up to 160 characters per line), model 4433 printer (up to 233 characters per line), and model 4518 letter-quality printer (up to 203 characters per line). You can have files printed via the CLI command `PRINT`, and specify the printer as a listing file via the $LPT/L switch.

printer port — on a Model 10 or 10/SP computer CPU board, this is a device controller that can run a printer or foreground console.

program — a series of instructions, translated into binary codes, that the computer can execute. Text editors, the CLI, and the operating system itself are programs.

prompt — a character or phrase which indicates that a program is ready to receive a command (or more information). For example, the DG/RDOS CLI prompt is the letter R and ↲.

protocol — a set of conventions between communicating programs that defines the format and sequence of messages to be exchanged.

queue — a file designed to hold printer requests while the printer prints them. See also spool.

RDOS — a relative of DG/RDOS that allows the operating system to be generated by a user. This provides more device flexibility but is harder to use than DG/RDOS, the pregenerated (pregen) system.

record — a series of one or more characters written to or read from a file. Further described in Chapter 11.

Release Notice — a notice that names product files and recent software changes that DG hasn't yet been able to include in the product manual(s), supplied with DG/RDOS and other software as a printed listing. The Release Notice also explains how to install the product.

secondary console — a terminal attached to the printer port; also see foreground terminal.

secondary partition — a type of user-created directory whose size is fixed at creation and cannot be enlarged; created with the CPART command. The other kind of user-created directory (which *does* enlarge) is a subdirectory.

Software Revision Service — a service that provides new revisions of system and support software to customers as DG creates them. Membership is available with DESKTOP GENERATION systems.

Software Trouble Report (STR) — a formal report, made by a customer to DG through a DG service area or engineer, about a serious problem that the customer is having with the software. The cause may be a user or DG error. DG personnel try to duplicate the problem to solve it, thus need as much information about the problem as possible. Or, instead of reporting errors, an STR can simply offer suggestions.

Sort/Merge — a program, available with DG/RDOS, that allows users to edit records or reorder them in numeric or alphabetical order. The DG/RDOS Sort/Merge is named CSSORT.

source file — the file that contains the source statments of a program. If the program is written in a compiled language, the source file must be compiled before the program can be run. In BASIC, you can usually just type and run the source file. In a compiled language, the source file is the most important file (more important than compiled versions, which can easily be recreated by the compiler program).

spool, spooling — a method for storing information on disk while it's being printed. Without spooling, whenever you type a PRINT command, you can't type any more commands until the printer has finished. DG/RDOS can spool to a printer connected to any device except a USAM line. To delete the spool queue and stop spooling (as for shutdown), use the SPKIL command. (The word "spool" stands for Simultaneous Peripheral Operation On Line.)

stand-alone program — a program that runs without an operating system, relying on its own device handlers. Usually, stand-alone programs are dedicated to a specific function. The DKINIT formatter is a stand-alone program.

STR — see *Software Trouble Report.*

subdirectory — a type of user-created directory that can grow as needed to hold files within it; created by the CDIR command. Also see *secondary partition.*

support organization — the organization person committed to supply help or support. For DESKTOP GENERATION systems, this may be the authorized dealer (if the system was obtained from a dealer) or another group; for information, call 1-800-DATAGEN.

switch — aside from conventional meaning, a switch is a slash (/) followed by some characters that modify the execution of a command or macro. For example, the switch /L=$LPT selects the printer queue as a listing file (instead of the terminal).

symbol — the name that identifies some procedure, variable, array name, or location. Often created by users, but sometimes defined by the language or system. For rules on names, see *names.*

synchronous line — a communications line that uses a synchronous protocol to transmit or receive data. Synchronous lines are frequently used in long distance communication between computer systems.

SYS.DR — a file (not a directory, as its name implies) maintained by DG/RDOS to keep track of files within a directory. There's a copy of SYS.DR in each directory.

sysgen — a version of a program that a require a user to generate a system before using it. RDOS is the sysgen version of DG/RDOS. Compare with *pregen.*

system buffer — see buffer.

system console — the terminal connected directly to the computer. User terminals are not connected *directly,* but to the printer port or a USAM multiplexor that sorts incoming commands and sends responses to the correct terminal.

tape — a magnetic medium suitable for file backup and mass storage. For DG/RDOS systems, the tape unit name is `MT0`. Tape *files* are numbered sequentially from 0. After material has been written to file 0 (MT0:0), the next tape file is MT0:1.

template — the word has two meanings. First, a template is part of a filename, used with one of the template characters (- or *) to access one or more files. For example, the template `FILE-.-` matches all filenames that begin with the characters `FILE` in the current directory. Second, a template is a cardboard or plastic shape that fits over the topmost group of keys on the keyboard. These keys are called function keys and the template identifies them.

terminal — an interactive device with a keyboard for input and a screen for display. Sometimes called a console. The system console screen is called a monitor.

text editor — a computer program designed specifically to help people write and edit text. A text editor is related to a word word processor.

USAM (Universal Synchronous Asynchronous Multiplexor) — an optional board, model 4463, that can manage one or four asynchronous lines. A USAM line can be used to communicate with another system, run a printer or plotter, or run a terminal (which can be connected directly to the computer or work remotely through a *modem*).

user (as in user, system) — anyone who, in any capacity, uses a computer system.

variable — an entity that has a name and holds a value; it represents a number and is generally used in programming languages. The CLI has variables of its own, designed to make macros more useful. For example, the CLI variable %DATE% holds the current date.

utility, utility program — a program supplied by DG to help you use the system; for example, the CLI and DKINIT formatter. Some utilities are included with the operating system; others, like the CSSORT Sort/Merge, are optional extras.

warm start — a computer system startup in which computer power has stayed on since the operating system was shut down; usually faster than a cold start, since certain essential programs stay active and need not be reloaded.

YOWNER.SV — a program that identifies the filename that owns a disk block, useful to help recover from new bad blocks. (The name YOWNER comes from Y — a variation of RDOS — plus OWNER.)

Index

Within this index, "f" or "ff" after a page number means "and the following page" (or "pages"). In addition, primary page references for each topic are listed first. Commands, calls, and acronyms are in uppercase letters (e.g., CREATE); all others are lowercase.

Documentation Comment Form

Manual Title ______________________________

Manual No. ______________________________

Your Name ______________________________

Your Title ______________________________

Company ______________________________

Street ______________________________

City ____________________ State ________ Zip ____________

Please help us improve our future publications by answering the questions below. Use the space provided for your comments. Thank you.

	Yes	No
Is this manual easy to read?	☐	☐
Is it easy to understand?	☐	☐
Are the topics logically organized?	☐	☐
Is the technical information accurate?	☐	☐
Can you easily find what you want?	☐	☐
Does it tell you everything you need to know?	☐	☐
Do the illustrations help you?	☐	☐

If you wish to order manuals, contact your sales representative or dealer.

fold

Comments:

Date

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA 01772

Postage will be paid by addressee

Data General

Software Documentation
M.S. E-111
4400 Computer Drive
Westboro, MA 01581

Documentation Comment Form

Manual Title ______________________________

Manual No. ______________________________

Your Name ______________________________

Your Title ______________________________

Company ______________________________

Street ______________________________

City ____________________ State ________ Zip ____________

Please help us improve our future publications by answering the questions below. Use the space provided for your comments. Thank you.

	Yes	No
Is this manual easy to read?	☐	☐
Is it easy to understand?	☐	☐
Are the topics logically organized?	☐	☐
Is the technical information accurate?	☐	☐
Can you easily find what you want?	☐	☐
Does it tell you everything you need to know?	☐	☐
Do the illustrations help you?	☐	☐

If you wish to order manuals, contact your sales representative or dealer.

fold

Comments:

Date

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA 01772

Postage will be paid by addressee

DataGeneral

Software Documentation
M.S. E-111
4400 Computer Drive
Westboro, MA 01581

Startup

These are cold start steps. For warm start:
If system console shows *Filename?,* go to step 8.
If system console shows *!,* go to step 6.

1. Without a hard disk, insert the DG/RDOS system diskette in DJ0.

2. Make sure printer (if any) is on and on line.

3. Turn on tape drive (if any) and second hard disk (if any).

4. Turn on system console.

5. Turn on computer unit.

   Model 10 or 10/SP computer runs a test program. Without a hard disk, skip to step 7.

6. *!* **26H** (Type **26H** for hard disk, or, for diskette, **20H**.)

7. *Filename?*

8. ↵ (Press ↵ or type program name and ↵.)

   DG/RDOS REV x.xx

9. *DATE (M/D/Y)?* **12 14 84** ↵ (Type the date.)

10. *TIME (H:M:S)?* **14 30** ↵ (Type the time, using 24-hour clock.)

   .
   . (emulator messages on Model 10 or 10/SP)
   .
   R

Shutdown

1. Shut down any non-CLI program running on the system console.

   R

2. **FGND** ↵

   NO FOREGROUND PROGRAM RUNNING
   or
   FOREGROUND PROGRAM RUNNING

   If no foreground program is running, skip to step 5.

3. Warn foreground users of impending shutdown.

4. **CTRL-C CTRL-F** (To terminate foreground program.)
   FG TERM

5. **DIR DE0** ↵ (Or for diskette, type **DIR DJ0** ↵.)

6. **BYE** ↵

   STARTING SYSTEM SHUTDOWN

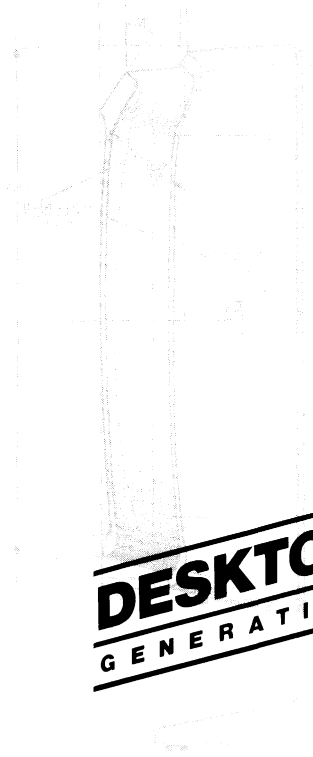
   MASTER DEVICE RELEASED

   Filename?

7. Type break sequence (CMD and BREAK/ESC keys).
   !

   Turn off devices (computer and system console last) if desired.

DataGeneral



DESKTOP GENERATION™

DG/RDOS Summary Card

DG/RDOS Commands, Macros, and Programs

BOOT $\left\{\begin{array}{l}\text{disk:program}\\ \text{program}\end{array}\right\}$
Starts a new DG/RDOS system or program like DKINIT.

BUILD newfilename filename *[filename] [...]*
Builds a file consisting of filenames.

BYE
Shuts down the DG/RDOS system.

CDIR directory-name
Creates a subdirectory (variable-size directory).

CHATR filename $\left\{\begin{array}{c}+\\ -\\ 0\end{array}\right\}$ attributes *[...]*
Changes file attributes (like P or R), for permanence or read protection

CONFIG *[system-name] [dialog-file/V]*
Checks or changes parameters in a DG/RDOS system.

CPART directory-name max-disk-blocks
Creates a secondary partition (fixed-size directory).

DELETE filename *[...]*
Deletes one or more files.

DIR *[directory-pathname]*
Changes the current directory.

DISK
Displays both the amount of disk space left and used.

EXFG program
Executes a program in foreground memory.

FCOPY $\left[\begin{array}{l}\textit{[source-diskette destination-diskette]}\\ \textit{[source-file destination-file]}\end{array}\right]$
Copies a diskette or file.

GDIR
Gets the current directory name.

GMEM
Gets the amount of memory in background and foreground.

GTOD
Gets the system time and date.

IMOVE/D $\left\{\begin{array}{l}\text{DJ0}\\ \text{DJ1}\\ \text{MT0}\end{array}\right\}$ *[filename] [...]*
Copies files to or from diskette or tape.

INIT $\left\{\begin{array}{l}\text{directory-pathname}\\ \text{disk(ette)}\\ \text{MT0}\end{array}\right\}$
Opens a directory or tape.

INIT/F disk(ette)
Creates a new file directory.

LINK link-entry-name *[directory:]* resolution-file
Creates a link entry to a file in any directory.

LIST *[pathname] [...]*
Describes file names and statistics.

LOADEM
Loads a terminal emulator into memory or disk(ette).

LOG *[password]*
Starts logging terminal dialog in a disk file.

MESSAGE *[" [text] "]*
Displays text on the screen.

MOVE dir-name *[filename] [...] [old-filename/S new-filename]*
Copies one or more files to any directory.

PRINT pathname *[...]*
Starts printing a file.

RELEASE $\left\{\begin{array}{l}\text{directory}\\ \text{MT0}\end{array}\right\}$
Releases (closes) a directory or tape drive.

RENAME oldname newname *[...]*
Renames a file.

SDAY mm-dd-yy
Sets the system date; for example,
SDAY 12-21-84 ↵.

SEDIT filename
Edits disk file locations.

SMEM memory-pages-for-background
Sets memory for background and foreground progr

SPKILL $LPT
Stops printing and deletes the spool file.

STOD *[hh [mm [ss]]]*
Sets the system time.

TYPE pathname *[...]*
Types one or more files on the terminal screen.

UNLINK link-entry-pathname
Removes a link entry.

XFER source-file destination-file
Copies the contents of a file into another file.

Data General

Data General Corporation, Westboro, Massachusetts 01580

069-000056-02