RSTS/E
System Manager's Guide

Order No. DEC-11-ORSMD-A-D

Order additional copies as directed on the Software Information page at the back of this document.

digital equipment corporation · maynard. massachusetts

First Printing, July 1975

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

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

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

The HOW TO OBTAIN SOFTWARE INFORMATION pages, located at the back of this document, explain the various services available to Digital software users.

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

The following are trademarks of Digital Equipment Corporation:

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

LIMITED RIGHTS LEGEND

Contract No. _____

Contractor or Subcontractor: Digital Equipment Corporation

All the material contained herein is considered limited rights data under such contract.

CONTENTS

FIGURES

TABLES

PREFACE

This guide describes the procedures necessary to operate and manage a RSTS/E Version V06A-02 system. The reader should be familiar with the structure and programming of RSTS/E and should have a firm knowledge of time sharing software and hardware. The basis for the material in the manual is a set of programs having privileged status and programs having privileged features. The contents of this guide are of concern only to the RSTS/E system manager and users whom he designates as privileged. Any other individual should not have access to this guide without the consent of the system manager.

For more information on RSTS/E guides and manuals, consult the RSTS/E Documentation Directory.

For a quick reference to a subject in this guide, use the following list.

If you need to know about	See Section
Starting time sharing	2.1.1
Preparing a disk for use	7.7
Making a disk ready to use	7.1.2
Creating user accounts	4.1.1
Deriving system accounting data	4.2
Terminating time sharing	2.2
Gauging system performance	7.2, 7.3
Determining causes of hardware errors	6.1, 6.2
Reporting system troubles	back of guide
Creating the TTYSET.SPD file	7.4.1
Specifying terminal characteristics	7.4.2
Controlling time sharing start up	3.1.2
Managing spooled operations	5.1
Changing LOGIN job size default	3.3.2

CHAPTER 1

RSTS/E SYSTEM STRUCTURE AND SYSTEM MANAGEMENT

RSTS/E (Resource Sharing Time Sharing System/Extended) runs on a PDP-11/70, a PDP-11/40 or PDP-11/45 computer and allows simultaneous, time shared access to PDP-11 hardware resources and to RSTS/E system software components through either local or remote terminals. A RSTS/E user performs time sharing operations using the full computational and data processing power of the BASIC-PLUS language.

The current version of RSTS/E employs the same file structure and programming language of previous versions but provides the capability of compiling and running COBOL programs. This chapter introduces the system manager to the hardware and software structures of RSTS/E and provides references to further descriptions of the philosophy and uses of RSTS/E.

1.1 SYSTEM HARDWARE

RSTS/E runs on several PDP-11 processors and supports many types of mass storage and peripheral devices. The system manager can readily change the system software to add new hardware and new software options. The RSTS/E system therefore may initially be small but can be expanded as installation requirements increase. The following sections describe typical RSTS/E hardware.

1.1.1 Processors and Related Options

The PDP-11/70 processor provides the highest performance for a RSTS/E system with a large number of users. All PDP-11/70 processors include the following features:

1K words of bipolar cache memory that buffers data from main memory to provide fast program execution.

UNIBUS Map which translates UNIBUS addresses to physical memory addresses.

Memory Management Unit to allow expanded memory addressing and hardware relocation and protection.

Optional with the PDP-11/70 is the Floating Point Processor to perform either single or double precision floating point arithmetic operations and floating integer conversion in parallel with the CPU. RSTS/E supports up to 1024K words of main memory on PDP-11/70 systems.

The PDP-11/70 cache, main memory, and Input/Output structure is designed for fast program execution and high I/O throughput. Main memory is connected to the cache through a 32-bit memory bus. The cache transfers instructions and data to the central processor unit over a 16-bit data path. High speed RH70 peripheral controllers communicate with main memory over a separate 32-bit data bus. Slower peripheral controllers are connected through the UNIBUS. This multiple bus structure allows several high speed data transfers to occur in parallel.

The PDP-11/45 processor offers an intermediate RSTS/E system for a moderate number of users. The Memory Management Unit is a hardware option but is required for operation of the RSTS/E software. The PDP-11/45 optionally provides the Floating Point Processor (FPP) and a maximum of 32K words of MOS or bipolar semiconductor memory to enhance performance. Semiconductor memory is connected to the central processor unit through a high speed data path internal to the PDP-11/45 processor. Core memory and all peripheral devices are connected to the UNIBUS. RSTS/E supports the maximum amount of memory (124K words) possible for a PDP-11/45.

The PDP-11/40 processor provides an economical time sharing system for a smaller number of users. The Extended Instruction Set (EIS) and Memory Management Unit are hardware options but are required for operation of the RSTS/E software. Core memory can be expanded to 124K words. All memory and peripheral devices are connected through the UNIBUS.

The PDP-11/70 and PDP-11/45 FPP units support either 2-word (32 bit) or 4-word (64 bit) precision. The units substantially improve performance for scientific applications requiring double precision floating point calculations and for business applications which utilize 4-word scaled arithmetic available in BASIC-PLUS.

The PDP-11/40 FIS unit provides high speed 2-word floating point calculations.

Two- or four-word (including scaled arithmetic) software routines are available for systems which do not include a hardware floating point processor. The hardware can of course be added at any time to improve system performance.

RSTS/E systems require a hardware bootstrap loader. Either the BM873-YA or the BM873-YB is available. The BM873-YB is necessary either to bootstrap an RP04 disk and TM02/TU16 magtape or to bootstrap a non-zero disk unit.

RSTS/E requires a system clock. Either the KW11-L Line Frequency Clock or the KW11P Programmable Clock is available. The KW11P can generate interrupts based on either line frequency or crystal-controlled oscillator.

1.1.2 Disk Devices

RSTS/E supports the following types and numbers of disks.

Type	Number
RJP04 Moving(1) Head Disk System (RH70/RP04 or RH11/RP04)	A maximum of 8 RP04 40 million word disk pack drives
RP11-C/RP03/RP02(1) Moving Head Disk System	A maximum of 8 RP03 20 million word disk pack drives. (The 10 million word RP02 disk pack can be used.)
RK11/RK03 or RK05 Moving Head Disk System	A maximum of 8 RK03 or RK05 1.2 million word disk cartridge drives.
RJS04 and RJS03 Fixed Head Disk System (RH70/RS03/RS04 or RH11/RS03/RS04)	The RJS03 and RJS04 are expandable by adding either RS03 (256K words) drives or RS04 (512K words) drives. RSTS/E supports any combination of drives not to exceed 4 (for swapping and non-resident code only).
RF11/RS11 Fixed Head Disk System	A maximum of 8 disk drives to serve as system disk or storage for swapping and non-resident code.

The RSTS/E system disk can be either an RP04, RP03, or RP02 disk pack; an RK03 or RK05 disk cartridge; or an RF11 fixed head disk. The system disk can be used for swapping and non-resident monitor code as well as for system and user files. The optimal disk configuration, however, includes at least one moving-head disk as the system disk (and as the first unit of the public structure) and an auxiliary fixed-head disk as storage for swapping.

A disk configuration must provide adequate storage space for both system and user files. For example, a system with an RK11C/RK05 disk system only and no tape or other secondary storage must have at least two RK05 drives for adequate capacity and for system generation from software distributed on disk cartridges. Auxiliary disk units may be present to supplement secondary storage capacity. See the section entitled "Disk File Organization" for more information on configuration and storage requirements.

---

(1) RSTS/E supports either the RJP04 or the RP11-C/RP03/RP02 moving head disk systems but not both on the same computer.

1.1.3 Terminals

The RSTS/E terminal service supports a wide variety of local and remote terminals connected through many types of interfaces. The following terminals are available with RSTS/E.

DECwriters	LA36, LA30S (serial) and LA30P (parallel) hard copy terminals
Scopes	VT50, VT05B, and VT05 alphanumeric display terminals
Teletypes	ASR33, KSR33, ASR35 and KSR35 Teletype terminals
2741	IBM 2741 Compatible Terminals
Special	RT02 Numeric Display Terminal, GT40 Graphic Display System
Others	Definable through system function calls.

Interfaces for local ASCII terminals can be either KL11, DL11A, DL11B, DL11C, DL11D, DL11E (null modem), DC11 (null modem), and LC11 (for the LA30P DECwriter) for single lines or the DH11 16-line multiplexer. ASCII terminals on remote, dial-up lines can be connected through DL11 and DC11 single line interfaces or through the DH11 multiplexer and a DM11-BB modem control multiplexer.

Interfaces for local 2741-type terminals (which are non-ASCII) can be either DL11D, DL11E (null modem) and DC11 (null modem) for single lines or the DH11 multiplexer. Remote 2741-type terminals can be connected through either the DL11E and DC11 interfaces for single lines or through the DH11 and DM11-BB multiplexers. The RSTS-11 System User's Guide describes the terminals which have been tested under RSTS/E.

1.1.4 Additional Peripheral Devices

RSTS/E supports a variety of additional peripheral devices including two magtape systems, DECtape, several types of line printers, card readers and paper tape devices. The following are the types and numbers of magtape devices.

TJU16 Magtape System (1)	A maximum of 8 TU16 drives
TM11 Magtape System (1)	A maximum of 8 TU10 drives
TC11 DECtape System	A maximum of 8 TU56 single drives

A maximum of 8 in any combination of the following line printers are supported.

---

(1) RSTS/E supports either the TJU16 Magtape system or the TM11/TU10 magtape system but not both on the same computer.

LP11 High Speed Line Printer	230, 300, or 1200 lines per minute; either 80 or 132 columns; either upper-case only or upper- and lower-case.
LS Line Printer	60 lines per minute dot-matrix; upper-case only; 132 columns.

A single card reader from one of the following types is supported.

CD11	1000 or 1200 cards per minute punched card reader
CR11	300 cards per minute punched card reader
CM11	300 cards per minute mark sense card reader

The PC11 High Speed Paper Tape punch and PR11 High Speed Paper Tape Reader are also supported.

Each of the above peripheral devices is fully available during time sharing. Any unit can be assigned to any time sharing job as needed.

1.2 SYSTEM SOFTWARE

RSTS/E system software exists as either system code, language code, or system program code. The system code and part of the language code is tailored at system generation time according to the hardware configuration on which the system runs and the software features which are chosen by the system manager. Once the system is generated, the system code and part of the language code are frozen and alterable only by patching or by generating new code. The system program code exists in a library of programs executable by the system software or by individual users on the system. The library of programs is alterable and expandable during time sharing without requiring regeneration of the system.

1.2.1 System Code

The RSTS/E system code is stored on the system disk as a core image library (CIL). A core image library, when loaded into memory, is immediately executable by the PDP-11 computer. The system code comprises many distinct elements which are either resident in memory or on disk during time sharing. Permanently resident elements are the following:

- interrupt and trap vectors
- small and large system buffers
- system information and data tables
- disk and device drivers
- file processor modules

The following are disk resident (overlay) elements.

- file processor modules
- infrequently used utility routines
- system initialization code

RSTS/E operations start when the system disk is bootstrapped. The bootstrap routine loads the initialization code which performs many consistency checks to ensure the integrity of the software. When checking is completed, the initialization code remains resident and allows many options described in Chapter 3 of the RSTS/E System Generation Manual.

When time sharing operations are started, the initialization code is overlaid by the permanently resident system code and the BASIC-PLUS compiler and Run Time System. As time sharing operations proceed, infrequently used overlay code and system and user programs are loaded from disk as needed.

1.2.2 Language Processors

The language processors reside on the system disk in machine executable form and can be either permanently resident in memory or temporarily resident (swappable). The following are the permanently resident elements.

BASIC-PLUS text editor and analyzer
BASIC-PLUS incremental compiler
BASIC-PLUS run time system

The temporarily resident elements are the following:

RTSLIB auxiliary run time system
COBOL compiler and object time system
PDP-11 SORT11 program

The BASIC-PLUS code is loaded into memory at the start of time sharing operations and remains resident during the session. The code analyzes all BASIC-PLUS statements and generates and executes intermediate (compiled) code. Many monitor services are available to a BASIC-PLUS program through system function calls.

The auxiliary runtime system RTSLIB is loaded into memory only when a request is made to execute the COBOL compiler and object time system or the SORT11 program. RTSLIB remains resident until all COBOL or SORT11 requests are satisfied and afterwards is overlaid by system and user programs. The COBOL compiler and SORT11 program are swapped out to disk with the user job image.

1.2.3 System Program Code

A library of programs is produced and stored on disk during the system library build procedures. Both the system and users execute these programs to perform system housekeeping and common utility functions. The system manager can use the programs to monitor and regulate system usage. Some library programs can be tailored by altering the source statements supplied by DIGITAL and recompiling to replace the current copy on the system disk.

1.3 DISK ORGANIZATION

1.3.1 File Structure

A logical structure of files controls all access to system and user data on the RSTS/E system. The file structure is flexible enough so that it can control and access any type of information. The file structure design is based on the need to control and access data and code on disk.

The logical disk structure is divided into two types: public and private. The public disk structure consists of a system disk and additional public disk packs or disk cartridges. All public disks must be physically on-line and logically mounted whenever the system is running and must be accessible to all users during time sharing operations.

The system disk contains the system code, language processors, and the library of system programs. The system disk may also be used for storage of active user jobs which are temporarily swapped out of memory. Remaining space on the system disk and all space on other public disks is available for general storage of user programs and data files.

If the system disk is a moving head device, an auxiliary fixed head disk can be used as the swapping device. In such cases, the swapping device is a logical extension of the system disk and can be configured to contain, in addition to the swapping files, other frequently used system files to improve speed of access.

Any disk drives not devoted to the public structure can be devoted to private disk packs or disk cartridges. Private disks can be logically mounted and dismounted and interchanged as needed during time sharing operations. A private disk provides a means to restrict disk storage to a defined set of users. The file structure on a disk, whether it is designated public or private, is the same.

Access to files in the RSTS/E system is accomplished by two structures called a Master File Directory and a User File Directory. A Master File Directory, or MFD, exists on each disk initialized for use on the RSTS/E system. The MFD is treated as an account on the disk and catalogs other accounts on the disk. The MFD on the system disk is a special case, since it maintains a catalog of the accounts which can be used to log into the system. MFDs on other disks contain entries of accounts which can create files on that disk. Any user can access any file on any disk if the protection code of the file permits. However, only those users whose accounts are entered in the MFD of the private disk can create files on the disk.

One User File Directory, or UFD, exists for each user account on a disk. (The UFD is not actually created until a file is created for the related account.) The UFD catalogs all program and data files under an account and maintains accounting and access information for the files. The UFD contains all retrieval information for the files because each file is pure data and has no linkage nor structural information.

1.3.2 Disk Optimization

Whenever a file is opened on the public structure, the directories on every public disk are searched. The search either verifies the existence of the file or ensures its nonexistence. The overhead of searching more than one disk can be avoided by placing the file on a private disk.

It is advantageous to dedicate an entire private disk to a single large production file. This organization ensures an efficient directory structure and minimizes overhead to access file data. When more than one file is on the same private disk, it is best to dedicate a whole account to each production file. This organization minimizes directory search overhead. If more than one file is under the same account on the same disk, the large files should be created before the small ones to ensure better organization of the directory structures.

In an environment where distinct data files must be accessed by the same program, the optimal organization is to keep each file on a different private disk. If a program must access more than one file on the same disk, overhead is increased because disk head movement is required whenever a current reference to a file is not to the same file as the preceding reference. A large percentage of execution time, therefore, is spent in moving the disk head back and forth. If, however, each file referenced by the program exists on a distinct private disk, head movement is not required whenever the program references another file. Head movement is restricted to locating the data itself.

1.4 SYSTEM OPERATION CONCEPTS

Immediately after logging into the system, a user's terminal is in edit mode (BASIC-PLUS command level) and is returned to edit mode when any program execution is completed or whenever a CTRL/C is typed at the terminal. In edit mode, the system examines each ASCII text line entered by the user and determines whether that line is a system command, a BASIC-PLUS immediate mode statement, or a BASIC-PLUS numbered statement. System commands are executed immediately after being entered as described in Chapter 2 of the RSTS-11 System User's Guide.

An immediate mode statement is first translated into intermediate code, which is placed in the user's job area and executed immediately by BASIC-PLUS. (Immediate mode operations are described in Chapter 4 of the BASIC-PLUS Language Manual.) Program statements preceded by line numbers are analyzed and stored in their ASCII form in a temporary disk file named TEMPnn.TMP stored under the user's account. Each program statement is also compiled into its intermediate code representation and is placed in the user's area of memory.

Intermediate code created in the user's job area upon entry of numbered statements is not executed automatically. The related program statements can be changed. A copy of the intermediate code of the program can be transferred to disk storage (as a file with a BAC filename extension) or to an external storage medium.

The user job area is initialized at log in time and set to a size of 2K words (K = 1024). The job area can grow in increments of 1K words to a maximum size set by the system manager at the start of time sharing operations.

A user leaves edit mode when he types the RUN system command or the CHAIN immediate mode statement. In run mode, BASIC-PLUS interpretively executes the intermediate code stored in the user's job area. Following program execution, the user's terminal is returned to edit mode, signalled by printing of the READY message. The user can interrupt BASIC-PLUS by typing CTRL/C, which also returns the user's terminal to edit mode.

The RSTS/E system allows jobs to run one at a time. A job runs until it either enters an I/O wait state or exhausts the time quantum assigned by either the system or the system manager. When the currently running job ceases to run, the scheduler finds the next job that is ready to run and begins running that job. Meanwhile, the interrupt-driven I/O device handlers are processing requested data transfers. Upon completion of a transfer, the scheduler marks the job that requested the transfer as ready to run again and starts it from the point at which execution ceased.

RSTS/E attempts to keep as many jobs in memory as possible. When more memory is required to run a job than is available, the system temporarily moves some jobs out of memory and stores them in one of four files called SWAP0.SYS, SWAP1.SYS, SWAP2.SYS, and SWAP3.SYS. This operation is called swapping. When a job is again eligible to run, it is swapped back into memory. Jobs waiting for keyboard input and jobs waiting for device I/O completion are most likely stored in the swapping files, while jobs currently running or involved in disk or magtape data transfers are necessarily in memory.

As the system processes each job, it maintains accounting information in memory concerning that job. When the job is logged off the system, this information is used to update the accounting information stored on the disk for that account.

1.5 SYSTEM MANAGEMENT

Management of RSTS/E begins with providing properly tailored hardware and software configuration, proceeds through initializing the software at system generation time, and continues with the daily functioning of time sharing. To ensure that these steps of management are efficiently performed, the manager of a RSTS/E system should be familiar with time sharing concepts and practices or should have a close working relationship with a senior programmer or analyst who is experienced in time sharing.

To begin managing well, the person responsible for RSTS/E operation must have knowledge of local processing and the capabilities and structure of his system. The information supplied in the RSTS/E System Generation Manual explains critical aspects of hardware and software options and provides memory requirements to assist in configuring a RSTS/E system.

After RSTS/E software is generated, many initialization options are available to structure and control system elements. A few of the initialization options are complex and sophisticated and their efficient usage at system generation time is of paramount importance. Once time sharing begins, certain restructuring capabilities become more difficult to employ. The manager or his designate should be

careful about the first-time initialization of RSTS/E. Unfortunately, no cookbook approach can attain proper system initialization.

To manage daily time sharing efficiently, an individual must know data processing functions, implement time sharing utility operations, and be familiar with RSTS/E concepts and structure. A full set of utility programs described in this manual and in the RSTS-11 System User's Guide is available to perform necessary tasks.

To ensure that responsible individuals are kept informed, the manager should designate an individual responsible for current documentation of the system, including both locally generated and DIGITAL-supplied procedures and guides. That responsible individual should ensure that delegated members of the staff receive the latest information. In too many cases, improper utilization of resources results from responsible individuals not being aware of published information.

If special facilities are required, a set of privileged programming features is available to the developer at the local installation. The system library can be expanded with tailored utility programs. Auxiliary libraries are available for special groups of users. The descriptions of special features can be located by consulting the RSTS/E Documentation Directory.

CHAPTER 2

SYSTEM START UP, SHUTDOWN, AND AUTOMATIC RESTART

2.1 STARTING UP RSTS/E: BOOTSTRAPPING RSTS/E INTO MEMORY

Starting up RSTS/E is a two step operation. The first step requires loading the RSTS/E initialization code into memory and the second step involves using one of the RSTS/E initialization options.

The initialization code in RSTS/E exists in the system file RSTS.CIL on the system disk. The code is loaded into the lower 28K words of memory and overlaid after its execution by system and user programs. The following sections describe the ways that initialization code can be bootstrapped into memory. Chapter 3 of the RSTS/E System Generation Manual describes and explains the RSTS/E initialization options. Section 2.1.4 of this manual summarizes the options and shows how to start time sharing.

2.1.1 Bootstrapping RSTS/E via a Hardware Bootstrap Loader

The procedures to follow when bootstrapping RSTS/E depend on the type of hardware bootstrap loader and the type of disk used as the system device. The following steps explain the procedure.

1. Make sure that the system disk is physically mounted on a disk unit. (This does not apply if the system disk is an RF type disk.)

2. Make sure that the system disk drive is READY and is in the WRITE ENABLE condition.

3. Ensure that the console terminal is on line.

4. Refer to Appendix A for the proper bootstrap operation.

When the initialization code is loaded, it prints the system header and installation name followed by the OPTION: message. If no messages appear after performing the bootstrap operation, ensure that the console terminal is on line and retry the bootstrap procedure. If the initialization code prints the message FATAL ERROR OCCURRED DURING CILUS PHASE, chances are the system disk is not in the WRITE ENABLE condition. To recover, write enable the disk and retry the bootstrap procedure.

If the automatic restart facility is to be enabled, set the CPU Switch Register to 777777; that is, all switches are in the up position. The automatic restart facility remains enabled as long as the CPU

Switch Register remains set to 777777. If any switch between positions 15 and 0 is set to its 0 position (that is, to its down position), the automatic restart facility is disabled for the duration of this condition.

This method of bootstrapping the initialization code is independent of any previous contents of memory and requires only that the system image be the first Core Image Library (CIL) module in the CIL file and that a special bootstrap routine reside in the first 256 words of the system disk. The RSTS.CIL file can never be moved from its place on the system disk. These conditions apply on all RSTS/E systems.

2.1.2 Bootstrapping RSTS/E After a System Halt

If the RSTS/E system halts as a result of the SHUTUP program being run or as a result of a catastrophic error or system crash, the halt address is always 54. The address lights indicate 56 in such a case, since the program counter contains the address of the next sequential instruction. (See Section 2.3.1 for a description of catastrophic errors and system crashes.) The manner in which the the system can be handled in such a situation depends upon the configuration and the results desired. The alternatives are described in the following sections.

2.1.2.1 Requesting a Memory Dump and Automatic Restart - If the RSTS/E system halts at address 54 and the crash dump facility was enabled at the start of time sharing operations, a memory dump and automatic restart can be performed. The following procedures prescribe the steps to accomplish a memory dump and automatic restart.

1. Ensure that the CPU is in a halt state as described in Section 2.2.
2. Ensure that the CPU HALT/ENABLE switch is set to its ENABLE position.
3. Set the CPU Switch Register to 000052.
4. Depress the CPU LOAD ADRS switch.
5. Set the CPU Switch Register to 777777.
6. Depress the CPU START switch.

After RSTS/E starts from address 52, it checks to determine if the CPU Switch Register is set to 777777. If the Switch Register is set to 777777, the system writes the contents of all critical memory into the CRASH.SYS file. The system then bootstraps the RSTS/E initialization code into memory in the special automatic restart mode described in Section 2.3.2.

If the crash dump facility was enabled at the start of time sharing operations but, after starting from address 52, the system finds the Switch Register set to something other than 777777, a halt immediately occurs again at address 54. If the crash dump facility was not enabled at the start of time sharing operations, the system halts immediately at address 54. It is impossible to obtain a crash dump. The system can be restarted as described in Section 2.1.2.2.

2.1.2.2 Requesting an Initialization Option - If the RSTS/E system halts at address 54, the initialization code can be bootstrapped in the normal start up mode. The procedures to request an initialization option are as shown below.

1. Ensure that the CPU is in a halt state as described in Section 2.2.

2. Ensure that the CPU HALT/ENABLE switch is set to its ENABLE position.

3. Set the CPU Switch Register to 000050.

4. Depress the CPU LOAD ADRS switch.

5. Depress the CPU START switch.

Control is passed to the initialization code, which prints the system installation name followed, on the next line, by the OPTION query. See Section 2.1.4 for the valid responses to the OPTION query. If the system halts at address 54 after a system crash and if the Switch Register was set to something other than 777777, simply pressing the CONT switch bootstraps the initialization code in the normal start up mode.

2.1.3 Bootstrapping RSTS/E from ROLLIN

If the stand-alone program ROLLIN is in memory and if the RSTS/E system disk is physically mounted on unit 0, in the READY condition and in the WRITE ENABLE condition, the RSTS/E system can be bootstrapped into memory. The /BO:dev switch described in Chapter 4 of the PDP-11 ROLLIN Utility Program document, DEC-11-OROAA-B-D, bootstraps unit 0 of the device specified. After RSTS/E is bootstrapped into memory, it prints the system installation name on the console keyboard printer followed, on a second line, by the OPTION query.

2.1.4 Starting Time Sharing - START Example

When the system disk is bootstrapped, the system heading and OPTION: message is printed. The user has available the options summarized in Table 2-1.

Table 2-1
Initialization Option Summary

Short Form of Operator Response	Meaning	Section in SYSGEN Manual
PA	Alter the RSTS/E System Code to correct problems.	3.2
DS	Initialize and optionally format a disk.	3.3
RE	Create or rebuild the system files in account [0,1] on the system disk.	3.4
SE	Set keyboard defaults for disabling lines and for enabling DH11 lines as local or with modem control.	3.5
DE	Establish or change default start up conditions.	3.6
ST or LINE FEED key	Start time sharing operations.	3.7
UN	Diagnostic aid used in conjunction with the START option to bypass the enabling of all terminal interfaces except the console interface.	3.8
BO	Bootstrap a device.	3.9
LO	Load a stand-alone program from the RSTS/E CIL.	3.10
AS VT LA	Set the fill factor of the console terminal to that of the device specified.	3.11

The initialization options are described in Chapter 3 of the RSTS/E System Generation Manual. The following sample printout shows the use of the START option.

```
RSTS V06A-02 SYSTEM #2027

OPTION: START


YOU CURRENTLY HAVE: JOB MAX = 20, SWAP MAX = 28K.

JOB MAX OR SWAP MAX CHANGES ? OLD

ANY MEMORY ALLOCATION CHANGES ? OLD

CRASH DUMP? OLD

DD-MMM-YY? 12-JUN-75
HH:MM? 15:05


SYSTEM INITIALIZATION PROGRAM            V06A-03

COMMAND FILE NAME? MONDAY.CTL
DETACHING...



^C
HELLO 1 / 2
PASSWORD:
JOB(S) 1 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?
1 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT



READY

^C
RUN $UTILTY

READY

SYSTEM UTILITY PROGRAM 'UTILTY V06A-03'
? ADD RTSLIB
? LOAD RTSLIB/ADDR:116
? UNLOAD RTSLIB
? EXIT

READY

^C
RUN $QUEMAN

READY

QUEMAN  V06A-07 - RSTS V06A-02 SYSTEM #2027
STARTED AT: 15:05    ON 12-JUN-75
*^\DE
DETACHING...
```

```
^C
HELLO 1 / 2
PASSWORD:
JOB(S) 1 4 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?
2 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT



READY

^C
RUN $SPOOL

READY

SPOOL V06A-09 - RSTS V06A-02 SYSTEM #2027
LP UNIT #? 0
DETACHING...



QUEMAN MESSAGE: LP0SPL ( 2 ) PUT ONLINE AT 15:05
^C
HELLO 1 / 2
PASSWORD:
JOB(S) 1 2 4 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?
3 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT



READY

^C
RUN $BATCH

READY

BATCH V06A-10

BATCH UNIT BA?

RSTS/E IS NOW ON THE AIR...
DETACHING...



QUEMAN MESSAGE: BATCH3 ( 3 ) PUT ONLINE AT 15:06
^C
HELLO 1 / 2
PASSWORD:
JOB(S) 1 2 3 4 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?  1
ATTACHING TO JOB 1
RUN $ERRCPY

READY

ERRCPY V06A-01
DETACHING
```

2.2 HALTING THE RSTS/E SYSTEM

A halt in the RSTS/E system is caused by an orderly occurrence of events or by randomly setting the CPU console HALT/ENABLE switch to its HALT position. The PDP-11/70 and PDP-11/45 are considered in a halt state when both the RUN and PAUSE indicators on the CPU console panel are not lit. Otherwise, the CPU is running. For the PDP-11/40, consult the following chart to determine the state of the CPU based on the condition of the Console Status Lights.

Condition		
RUN light	CONSOLE light	Meaning
OFF	OFF	CPU is powered down or bus is hung.
OFF	ON	Impossible or bus is hung.
ON	OFF	CPU is running.
ON	ON	CPU is halted.

The SHUTUP program described in Section 3.2 shuts down the RSTS/E system in an orderly fashion. SHUTUP eventually halts the CPU at address 54. The program ensures that all files are properly closed and that system accounting information is accurately updated. The halt leaves the program counter loaded in such a fashion that depressing the CONT switch on the CPU console panel causes the RSTS/E initialization code to be bootstrapped into memory from the RSTS.CIL file stored on the system disk.

Halting RSTS/E by moving the HALT/ENABLE switch on the CPU console panel to its HALT position is dangerous. Clean-up operations may not be completed; disk storage allocation tables and file directories may be left in obsolete states; file data can consequently become corrupted; and accounting information may be lost. The only way to recover from such a disorderly halt and to salvage possibly vital file information is to raise the HALT/ENABLE switch back to its ENABLE position before any other action is taken, to depress the CONT switch, and thereby to return the RSTS/E system to the state in which it was before the halt occurred.

The following is a sample printout of the SHUTUP system program.

```
RUN $SHUTUP
AUTOMATIC SYSTEM SHUTDOWN PROGRAM

HOW MANY MINUTES UNTIL SYSTEM SHUTDOWN? 5
HOW MANY MINUTES BETWEEN WARNING MESSAGES? 1
5 MINUTE WARNING MESSAGE SENT
FURTHER LOGINS ARE NOW DISABLED
4 MINUTE WARNING MESSAGE SENT
3 MINUTE WARNING MESSAGE SENT
2 MINUTE WARNING MESSAGE SENT
1 MINUTE WARNING MESSAGE SENT
FIRST QUEMAN MESSAGE SENT
QUEMAN MESSAGE: SHUTDOWN BEGUN - QUEMAN WILL PROCESS NO MORE QUE COMMANDS
QUEMAN MESSAGE: LP0SPL ( 2 ) REQUESTED OFF-LINE -- TAKEN OFF-LINE AT 15:15
FINAL WARNING MESSAGE SENT
QUEMAN MESSAGE: BATCH3 ( 3 ) REQUESTED OFF-LINE -- TAKEN OFF-LINE AT 15:15
PASS 1 OF LOOKING FOR STILL ACTIVE JOBS
QUEMAN MESSAGE: FINAL SHUTDOWN PHASE -- KILLING SPOOLERS
SECOND QUEMAN MESSAGE SENT
QUEMAN MESSAGE: QUEMAN SHUTTING DOWN
FROM NOW ON JOBS WILL ONLY BE 'KILLED'
PASS 2 OF KILLING STILL ACTIVE JOBS
UNLOADING RTSLIB RTS
REMOVING RTSLIB RTS
NON-SYSTEM DISKS WILL NOW BE DISMOUNTED
ALL (DISMOUNTABLE) NON-SYSTEM DISKS ARE NOW DISMOUNTED

ALL SET TO PROCEED WITH SYSTEM SHUTDOWN

PLEASE WAIT FOR THE COMPUTER TO ACTUALLY 'HALT'
WHEN IT DOES, PRESSING 'CONT' WILL BOOT BACK RSTS/E
```

2.3 AUTOMATIC RECOVERY AND RESTART FACILITIES

2.3.1 Nature and Causes of Catastrophic Errors and System Crashes

A catastrophic error or a system crash is an error-trap to vector 4 or vector 10. (For information on error traps, see Section 5.3 in the PDP-11/70 Processor Handbook and Section 2.7 in either the PDP-11/40 Processor Handbook or the PDP-11/45 Processor Handbook.) Such traps can be caused, for example, by referring to a nonexistent (or non-responding) UNIBUS address (bus time-out trap), by referring to an odd address with an instruction that requires a word address, or by attempting to execute a reserved or nonexistent instruction.

Catastrophic errors and system crashes, therefore, can be due to any of four types of problems: (a) configuration errors, (b) privileged account programming errors, (c) hardware malfunctions, and (d) system software malfunctions. Each of these is discussed individually below.

2.3.1.1 Configuration Errors - If the software configuration and the hardware configuration do not correspond exactly, bus time out traps occur whenever the software attempts to address a peripheral interface which, for the software, logically exists but which for the hardware does not physically exist. Thus, RSTS/E SYSGEN program configuration questions must be answered accurately to reflect the actual hardware on which the system runs.

A common configuration error results from reporting a device as present when the device is not on the system. This type of error condition is not detected until a reference is made to the device during time sharing. When a reference is made to the nonexistent device, a trap through location 4 is made and the system crashes. Refer to the discussion of T4 errors in Section 6.2.

Another type of configuration error involves not reporting a device that is in fact attached to the computer. If the unreported device is part of the floating address and vector scheme, the floating address and vector assignments of other devices are incorrect. This type of configuration error usually results in the disabling of devices that are present. Messages are printed at the start of time sharing to report devices which are being disabled. If any devices reported disabled are actually attached to the system, chances are that the floating address and vector assignments are incorrect. The user must determine what device is not reported and must regenerate RSTS/E properly. For more information on this type of problem, see the discussion entitled "Terminal Interface Considerations" in Section 2.7 of the RSTS/E System Generation Manual.

Another type of configuration error is made when the devices are reported correctly but the jumpers are improperly cut. As a result, the hardware vector and addresses are incorrect. The tables in Appendix G of the RSTS/E System Generation Manual give the correct fixed and floating address and vector assignments.

2.3.1.2 Privileged-Account Programming Errors - The RSTS/E system software is designed to protect itself against programming errors perpetrated under non-privileged accounts. The system itself, upon detecting such an error, aborts execution of the potential error and reports a corresponding error message to the guilty user.

The RSTS/E software is vulnerable, however, to certain types of errors perpetrated under privileged accounts. By intent and design, the system manager (and those to whom he assigns any [1,*] account) have been given extensive powers which are not available under non-privileged accounts. Refer to the RSTS/E System Programming Manual for a discussion of privilege. These powers allow privileged users to modify System Library programs or to create utility programs in such a fashion that they can access parts of memory. Extreme care must be used when programming with privileged SYS system functions. A mistake can cause the system to take an error trap.

2.3.1.3 Hardware Malfunctions - Hardware malfunctions can be responsible for crashing the system. If unexplainable and random-type system crashes or catastrophic errors occur (particularly on systems which hitherto have been functioning well), it is likely that a hardware problem has arisen. Hardware can be diagnosed by examining the error logging printouts. Refer to Sections 6.1 and 6.2 for information on error logging.

2.3.1.4 System Software Malfunctions - Although every attempt has been made to detect and eliminate system software errors, the paths through the RSTS/E software are incalculably numerous. It is possible that, given certain conditions and certain sequences, the RSTS/E software can trap to vector 4 or vector 10. If a problem of this type is discovered (it should be reproducible in a defined environment and under defined conditions), a DIGITAL Software Specialist should be contacted. If new problems of this type become known, DIGITAL reports them as described on the next to last page of this guide.

2.3.2 Automatic Recovery from Catastrophic Errors and Crash Dump

Whenever a trap to vector 4 or vector 10 occurs, the system distinguishes the trap as one of two categories: it is either (a) a catastrophic error which affects only one particular user or (b) a system crash for which some software or hardware problem is possibly responsible. The handling of system crashes is treated below.

The handling of catastrophic errors is as follows. The system determines which user was responsible for the error-trap. It flags that user's job with a special code which causes the system to reinitialize that user's job area completely when it is next his turn to run. The system prints on that user's terminal the message CATASTROPHIC ERROR or some other fatal system error message followed by the text PROGRAM LOST-SORRY. The reinitialized user is in the same state as he would be if he had just logged into the system. The system resumes normal time sharing operations.

When the system detects a condition from which it cannot recover, it performs an automatic restart only if both of two conditions are fulfilled:

1. The crash-dump facility must have been enabled at system start up time (possible only when the CRASH.SYS file exists), and

2. The CPU's Switch Register must currently be set to 777777.

If either condition is not fulfilled, the system does not take the automatic restart path but simply halts at address 54.

If the system halts at address 54, the operator may choose one of two procedures.

1. He depresses the CPU Console CONTinue switch, which causes the system to be bootstrapped into normal system start up mode.

2. The operator starts the CPU at address 52 with CPU Switch Register set to 777777 (see Section 2.1.2.2). This causes the system first to write the contents of memory onto the CRASH.SYS file (provided the crash-dump facility had been enabled) and then to be bootstrapped from disk in the special automatic restart mode described below.

If the system takes the automatic restart path, no halt occurs. Instead, the system first writes the critical contents of memory into the CRASH.SYS file and then bootstraps itself into memory from the system disk. After the system has been bootstrapped into memory, control jumps to the initialization routines. At this point the system recognizes the fact that it was not activated through a normal system start up but rather through an automatic restart and consequently initializes itself in automatic restart mode. If two system crashes occur within the same minute (more accurately stated, two error-traps within the same minute), the system halts at address 54. This protects the system against an infinite loop of error-traps caused by some repeating hardware malfunction.

2.3.3 Automatic Restart Mode Initialization

When the system is initialized in automatic restart mode, control by-passes all parts of the initialization code which call for operator intervention and initializes the system using information already stored in memory. The system logs Job 1 into the system on KB0: under account [1,2] and causes it to run the System Library program INIT beginning at line 100. Since, in automatic restart mode, INIT begins at line 100 (rather than at its lowest line number), it takes directions from the CRASH.CTL System Library file (rather than from the START.CTL file). The CRASH.CTL file must contain INIT commands which perform all operations the system manager considers necessary in the case of an automatic restart. See Section 3.1 concerning the control of system start up.

The printout which appears on the console terminal is similar in format to the following. Section 3.1 explains why most of these lines appear.

```
SYSTEM START UP, SHUTDOWN, AND AUTOMATIC RESTART

SYSTEM HAS BEEN RELOADED;  ATTEMPTING AUTO-RESTART.

READY

SYSTEM INITIALIZATION PROGRAM


RUN $ERRCPY
READY
DETACHING
```

CHAPTER 3

CONTROLLING TIME SHARING

3.1 CONTROLLING SYSTEM START UP - INIT

The system manager can control system start up by means of the system program INIT. System start up occurs when the START option of the initialization code is executed or when automatic restart occurs. At start up, the monitor runs INIT which, in turn, executes special commands from a control file. The actions which occur can be controlled by tailoring these commands to local requirements.

To control system start up efficiently, the system manager must understand the conditions in effect at start up time. The following conditions pertain.

1. Login attempts are prohibited (the monitor disables the login capability).

2. The monitor logically mounts only the system disk.

3. No output is made to any terminal.

4. The monitor logs the console terminal (KB0:) onto the system under the system library account [1,2].

5. At the console terminal, the monitor executes the command equivalent to CHAIN "INIT" or CHAIN "INIT" 100.

INIT runs as a consequence of condition (5).

If INIT runs as a result of the START option, it executes from the lowest line number and prints the following question at the console terminal.

```
SYSTEM INITIALIZATION PROGRAM V06A-03
COMMAND FILE NAME?
```

The name of the control file to execute must be typed. The control file must exist on the system disk (SY0:) under account [1,2]. If only the RETURN key is typed, INIT uses START.CTL on the system disk (SY0:) under account [1,2].

If INIT runs as a result of a system crash and automatic restart, it executes from line 100 and does not print a question. It reads the file CRASH.CTL on the system disk (SY0:) under account [1,2].

Any error encountered when INIT accesses the control file generates messages in the following manner.

```
<text>   - ERROR IN READING  <spec>

READY
```

INIT prints the BASIC-PLUS error message text and the specification of the file it attempted to read. INIT terminates but the console terminal remains logged into the system. After the error is fixed, initialization can be restarted by typing RUN $INIT.

A control file contains special commands for initializing the system for time sharing, recovering system crash information, and performing other locally required operations. The following sections describe the INIT commands and their usage.

3.1.1 INIT Program Commands

The RSTS/E system is not fully initialized until INIT runs and executes commands in the control file. The INIT commands are described in Table 3-1. For example, the system possibly uses other disk devices in addition to the system disk. By means of the MOUNT command, the system manager can make such devices in the public and in the private structure immediately available to users before they can log into the system. The local start up procedures must include making the specified devices physically ready on the proper drive units.

To execute other actions on the system, the system manager can cause INIT to execute BASIC-PLUS commands and programs. The LOGINS command enables further logins on the system. The LOGIN command automatically logs a specified job onto the system at a designated terminal to allow execution of commands and programs. For example, the FORCE command can run the TTYSET system program at the terminal and can subsequently execute commands to establish terminal characteristics of certain keyboards. The SEND command prints text on all on-line terminals.

Table 3-1
Control File Commands

Command Name and Format(1)	Use
LOGINS	Allows users at both local and remote terminals to enter the system.
SEND␣xxx	Transmits the text xxx to all keyboards currently on line except on the console keyboard (KB0:).
@name	Causes INIT to process the indirect control file specified by name. INIT returns to the next command in the current control file when indirect file is completed. Indirect control files can contain @ commands to allow nesting to ten levels.
LOGIN␣KBn:␣[n,m]	Logs the terminal specified by KBn: onto the system using the account indicated by [n,m]. INIT automatically looks up the password. NOTE Unless INIT has been detached, the LOGIN KBn: command cannot be used on KB0: because INIT is already running on that terminal.
FORCE␣KBn:␣xxx	Causes the text xxx to be placed in the input buffer of the terminal specified by KBn: and executed as if typed at that terminal. The text can be any BASIC-PLUS command or system command. NOTE If the ^ character is the first character of the text xxx, a CTRL/C is placed in the terminal buffer ahead of the specified text xxx. However, INIT generates an error if an attempt is made to force a ^C to KB0:.

(1)The notation ␣ indicates that a space character is required.

Table 3-1 (Cont.)
Control File Commands

Command Name and Format(1)	Use
MOUNT␣dev:id	Causes the disk unit specified by the device designator dev: and by the pack identification (id) to be logically recognized by the RSTS/E system. Additionally, the MOUNT command as used in the control file causes a clean operation (if necessary) and an unlock operation. After a clean operation, INIT prints the message CLEAN COMPLETED. Refer to Section 7.1.2 for information concerning mount, clean and unlock operations.
DETACH	Causes INIT to detach from the console terminal and print the message DETACHING. This action allows the console terminal to be used for other programs while INIT runs. INIT is reattached by the ATTACH command.
ATTACH	Attaches INIT to the console terminal which can not be in use by another job. This action allows INIT to be terminated normally.
BYE or END	Causes execution of the INIT system program to be terminated. BYE causes the job running under account [1,2] to be logged out, thus freeing the console keyboard for other use and preventing unauthorized use of account [1,2]. END must be used in place of BYE when running ERRCPY since END does not logout the job running ERRCPY under account [1,2]. See Section 6.1 for a description of ERRCPY.

(1)The notation ␣ indicates that a space character is required.

The following sample START.CTL file and accompanying explanation show the usage of INIT commands.

```
MOUNT DK1:PACK1                          (line a)
MOUNT DK2:PRIV1                          (line b)
LOGINS                                   (line c)
LOGIN KB1: [1,5]                         (line d)
FORCE KB1: RUN $TTYSET                   (line e)
FORCE KB1: KB1:                          (line f)
FORCE KB1: VT05B                         (line g)
FORCE KB1: EXIT                          (line h)
FORCE KB1: BYE F                         (line i)
@COPY                                    (line j)
FORCE KB0: RUN $ERRCPY                   (line k)
SEND RSTS/E IS NOW ON THE AIR...         (line l)
END                                      (line m)
```

Other disks are made available on the system by the commands at lines a and b. The sample commands assume unit 0 is being used as the system disk. Line a causes the system to recognize the additional disk cartridge PACK1 of the public structure on RK11 drive unit 1. All public disks (except the system disk) must be mounted in this manner if all user files are to be available immediately. The system also cleans (if necessary) and unlocks PACK1 so that users can create files on it.

Line b causes the same results as line a but for the disk cartridge PRIV1 in the private structure.

Logins must be enabled before attempting to log a terminal onto the system. The LOGINS command at line c is required to enable logins and to allow users access to the system.

The command at line d logs in a job at keyboard number 1 under account [1,5]. INIT automatically looks up the password of the account. In this manner, password secrecy is maintained.

The commands at lines e through h run the TTYSET system program to set terminal characteristics. For more information on TTYSET, see Section 6.7.

At line i, keyboard number 1 is logged off the system to prevent unauthorized use of the account. At line j, the indirect control file COPY.CMD is run.

The command at line k causes the console terminal (KB0:) to run the ERRCPY system program under account [1,2]. This action enables the RSTS/E system to take advantage of hardware error logging on the system.

The SEND command notifies users that time sharing operations have begun. The message is printed on all terminals on line to RSTS/E.

The END command at line m terminates the INIT program running at the console terminal. However, the command leaves KB0: logged into the system so that the command forced into the keyboard buffer at line j can be executed. Since ERRCPY detaches from the terminal, KB0: does not remain logged into the system after initialization.

The commercial at character (@) with a file specification in the control file causes INIT to close the current control file, open the

indirect control file specified, and process its commands. The indirect control file can contain INIT commands like those in the START.CTL and CRASH.CTL files. When INIT completes processing the commands in the indirect control file, it closes the file, reopens the calling control file, and continues processing the remaining commands.

If an extension and account are not specified in the control file specification of the @ command, INIT employs an extension of CMD and searches for the file in the system library account in the public structure. The following sequence of commands shows the procedure.

```
MOUNT DK1:PRIV
@DK1:[1,35]CONTL
END
```

INIT mounts the private disk PRIV on RK unit 1 and processes commands in the indirect control file CONTL.CMD on account [1,35] of that disk.

INIT discards an END or a BYE command in an indirect control file. The END or BYE command in the primary control file terminates the processing of INIT commands.

3.1.2 Creation and Usage of Control Files

The INIT system program control files must contain commands to properly initialize the RSTS/E system. The control files must be stored on the system disk which can be mounted on any drive. The files included in the RSTS/E system generation kit are only samples and, without modification, may not execute properly on a given system. The system manager must ensure that the files include the necessary commands to initialize the local installation. To replace the sample START.CTL file supplied with the RSTS/E kit, run PIP and proceed as follows:

```
RUN $PIP
PIP RSTS V06A-02 SYSTEM #880
#SY0:START.CTL<KB:/FA

     (Enter new INIT commands.)

^Z
#
```

The procedure shown ensures that the new file is created on the system disk (SY0:). The SY0: designator is critically important on systems with multiple disks in the public structure. Its use explicitly indicates the system disk. The file must reside on the system disk since, when INIT runs, only the system disk is mounted.

It is important that both control files perform the following functions:

1. Mount all non-system public disks,
2. enable logins,
3. set keyboard characteristics for non-ASR33 terminals,
4. run error logging and spooling programs, and
5. establish the auxiliary run time system RTSLIB.

Since, at start up time, the system sets the characteristics of all terminals to those of an ASR33-type terminal, the TTYSET system program should be run to set the correct characteristics. For example, if keyboards 1 and 3 are LA30S-type terminals to run at 300 baud, if keyboard 2 is an ASR33-type terminal, and if keyboard 4 is a VT05B to run at 2400 baud; the system manager can specify the following sequence of commands to properly set the characteristics.

```
LOGIN KB4: [1,5]
FORCE KB4: RUN $TTYSET
FORCE KB4: KB4:
FORCE KB4: VT05B
FORCE KB4: KB1:
FORCE KB4: LA30S
FORCE KB4: KB3:
FORCE KB4: LA30S
FORCE KB4: EXIT
```

This sequence of INIT commands shows the optimum use of a fast terminal on which to execute commands. Before forcing the next command to a terminal, INIT waits until the terminal output buffers are empty and the job is in the keyboard input wait state.

Therefore, it is advantageous to force commands to a terminal which generates output at the highest speed since that terminal's output buffers empty most quickly. Note, in the example, that keyboard 4 is established as a VT05B whose default output speed is 2400 baud. (Refer to the table of TTYSET macro commands in Section 4.5 of the RSTS-11 System User's Guide for the default characteristics of different terminal devices.) The interface speed is the important factor since it is not necessary that a terminal actually be connected to the interface unless a visual record of the start up procedure is desired.

In the case of INIT running as a result of a system crash, it is important that commands be executed to obtain crash-dump information. To discover the cause of a system crash, the system manager can specify that INIT run the crash analysis program ANALYS. The following commands inserted in the CRASH.CTL file ensure that the analysis occurs.

```
LOGIN KB4: [1,5]
FORCE KB4: RUN $ANALYS
FORCE KB4: [0,1]CRASH.DMP
FORCE KB4:
```

For more information on ANALYS, see Section 6.2.

The system manager can specify commands to be executed on KB0: even though INIT runs at the console terminal. The system assigns job number 1 to INIT since it is the first job to run at start up time. Because INIT runs at the console terminal, commands forced to KB0: are not executed until INIT terminates. To prevent premature termination, INIT does not allow a CTRL/C combination to be forced to the console terminal. Therefore, the ^ character must not be used alone in a FORCE command on the console terminal. Other character combinations such as ^Z are allowed.

If the system manager wishes to run a certain job as job number 1, he can specify the proper commands to the console terminal. For example, it is often desired to run the ERRCPY system program as job 1. The

following command in the control file ensures this action.

```
FORCE KB0: RUN $ERRCPY
```

ERRCPY runs and detaches immediately after INIT terminates as the result of the END command.

The system manager can initialize the system to handle the auxiliary run time system by commands to run the UTILTY system program. The commands in the control file show the procedure.

```
FORCE KB1: RUN $UTILTY
FORCE KB1: ADD RTSLIB
FORCE KB1: LOAD RTSLIB/ADDR:56
FORCE KB1: UNLOAD RTSLIB
FORCE KB1: EXIT
```

Without these commands in the control file, users on the system can not execute programs using the auxiliary run time system. For more information on these UTILTY commands, see Section 7.1.4.

INIT allows a choice of control file on normal start-up to initialize the system in alternate ways. For example, if the system disk is mounted on unit 1 instead of unit 0, an alternate control file can mount other public and private disks on proper units. Moreover, separate control files can be used to perform other than daily tasks. For example, a special control file can be used to run the REORDR program once a week before normal time sharing commences.

Any alternate control file must reside on the system disk under account [1,2] for the same reasons START.CTL and CRASH.CTL must reside there. If the control file specified is not on account [1,2] on the system disk, INIT begins initialization and encounters the error. To recover, INIT must be rerun from the console terminal and a standard control file must be specified. Alternate control files can be created and properly transferred to the system disk under normal time sharing.

3.1.2.1 START.CTL File Example - The following is an example of a START.CTL file which sets keyboards 1 and 2, starts the spooling programs SPOOL and BATCH, initializes the auxiliary run time system, and runs the ERRCPY program.

```
MOUNT DK1:PRIV1
MOUNT DK2:PRIV2
LOGINS
LOGIN KB1: [1,2]
FORCE KB1: RUN $TTYSET
FORCE KB1: KB1:
FORCE KB1: VT05B
FORCE KB1: KB2:
FORCE KB1: LA30S
FORCE KB1: EXIT
FORCE KB1: RUN $QUEMAN
FORCE KB1: \DE
LOGIN KB1: [1,2]
FORCE KB1: RUN $SPOOL
FORCE KB1: 0
LOGIN KB1: [1,2]
```

```
FORCE KB1: RUN $BATCH
FORCE KB1:
LOGIN KB1: [1,2]
FORCE KB1: RUN $UTILTY
FORCE KB1: ADD RTSLIB
FORCE KB1: LOAD RTSLIB/ADDR:56
FORCE KB1: UNLOAD RTSLIB
FORCE KB1: EXIT
FORCE KB1: BYE/F
FORCE KB0: RUN $ERRCPY
SEND RSTS/E IS NOW ON THE AIR ...
END
```

3.1.2.2 CRASH.CTL File Example - The following is an example of a CRASH.CTL file which runs the appropriate programs to recover crash information and initializes the system for time sharing.

```
MOUNT DK1:PRIV1
MOUNT DK2:PRIV2
SEND RSTS/E RECOVERING FROM A CRASH...
LOGINS
LOGIN KB1: [1,2]
FORCE KB1: RUN $TTYSET
FORCE KB1: KB1:
FORCE KB1: VT05B
FORCE KB1: KB2:
FORCE KB1: LA30S
FORCE KB1: EXIT
FORCE KB1: RUN $ANALYS
FORCE KB1:
FORCE KB1: ANALYS.TMP
FORCE KB1: RUN $ERRCRS
FORCE KB1: ERRCRS.TMP
FORCE KB1:
FORCE KB1: RUN $ERRDIS
FORCE KB1: ERRCRS.TMP
FORCE KB1: ERRDIS.TMP
FORCE KB1: ALL/S
FORCE KB1: ALL
FORCE KB1: /K
FORCE KB1: RUN $QUEMAN
FORCE KB1: \DE
LOGIN KB1: [1,2]
FORCE KB1: RUN $SPOOL
FORCE KB1: 0
LOGIN KB1: [1,2]
FORCE KB1: RUN $BATCH
FORCE KB1:
LOGIN KB1: [1,2]
FORCE KB1: RUN $UTILTY
FORCE KB1: ADD RTSLIB
FORCE KB1: LOAD RTSLIB/ADDR:56
FORCE KB1: UNLOAD RTSLIB
FORCE KB1: EXIT
FORCE KB1: BYE/F
FORCE KB0: RUN $ERRCPY
SEND RSTS/E IS NOW ON THE AIR...
END
```

3.1.2.3 Simplified CRASH.CTL File Example - The following is an example of a CRASH.CTL file which runs the appropriate programs to recover crash information and executes the indirect command to initialize the system for time sharing.

```
SEND RSTS/E RECOVERING FROM A CRASH...
LOGINS
FORCE KB1: ^SET VT05B
LOGIN KB1: [1,2]
FORCE KB1: RUN $ANALYS
FORCE KB1:
FORCE KB1: ANALYS.TMP
FORCE KB1: RUN $ERRCRS
FORCE KB1: ERRCRS.TMP
FORCE KB1:
FORCE KB1: RUN $ERRDIS
FORCE KB1: ERRCRS.TMP
FORCE KB1:
FORCE KB1: ALL/S
FORCE KB1: ALL
FORCE KB1: /K
@$START.CTL
END
```

3.1.2.4 Indirect Control File Example - The following is a sample CRASH.CTL file which runs appropriate programs to recover crash information, processes an indirect control file to initialize the system for time sharing, and queues the crash information files for printing.

```
SEND RSTS/E RECOVERING FROM A CRASH...
LOGINS
FORCE KB1: ^SET VT05B
LOGIN KB1: [1,2]
FORCE KB1: RUN $ANALYS
FORCE KB1:
FORCE KB1: ANALYS.TMP
FORCE KB1: RUN $ERRCRS
FORCE KB1: ERRCRS.TMP
FORCE KB1: [0,1]CRASH.SYS
FORCE KB1: RUN $ERRDIS
FORCE KB1: ERRCRS.TMP
FORCE KB1: ERRDIS.TMP
FORCE KB1: ALL/S
FORCE KB1: ALL
FORCE KB1: /K
@$START.CTL
LOGIN KB1: [1,2]
FORCE KB1: QUE ANALYS.TMP/D,ERRDIS.TMP/D
FORCE KB1: BYE/F
BYE
```

3.2 PERFORMING SYSTEM SHUT DOWN - SHUTUP

The shut down procedures for the RSTS/E system are critically important. If system shut down is not conducted in an orderly and careful fashion, valuable user data can be irretrievably lost. To fully understand shut down procedures, knowledge of other RSTS/E system procedures is necessary. The system manager should familiarize himself with the concepts presented elsewhere in this manual under the titles "On-Line System Control", "Disk Management", "Monitoring System Status" and "Spooling Operations".

To perform orderly shut down procedures, the system manager must run the SYSTAT and SHUTUP system programs. SYSTAT is described in Section 7.2. By running SYSTAT, the system manager can determine the active jobs on the system and the disk and assignable devices in use. Any active, detached jobs can be attached to a terminal prior to shut down. SHUTUP is run to perform the orderly shut down procedures.

The SHUTUP system program can be run only from the console terminal. The program should be stored in its compiled form in the system library with protection code <124>. Figure 3-1 shows the use of SHUTUP.

```
RUN $SHUTUP
AUTOMATIC SYSTEM SHUTDOWN PROGRAM

HOW MANY MINUTES UNTIL SYSTEM SHUTDOWN? 5
HOW MANY MINUTES BETWEEN WARNING MESSAGES? 1
5 MINUTE WARNING MESSAGE SENT
FURTHER LOGINS ARE NOW DISABLED
4 MINUTE WARNING MESSAGE SENT
3 MINUTE WARNING MESSAGE SENT
2 MINUTE WARNING MESSAGE SENT
1 MINUTE WARNING MESSAGE SENT
FIRST QUEMAN MESSAGE SENT
QUEMAN MESSAGE: SHUTDOWN BEGUN - QUEMAN WILL PROCESS NO MORE QUE COMMANDS
QUEMAN MESSAGE: LP0SPL ( 2 ) REQUESTED OFF-LINE -- TAKEN OFF-LINE AT 15:15
FINAL WARNING MESSAGE SENT
QUEMAN MESSAGE: BATCH3 ( 3 ) REQUESTED OFF-LINE -- TAKEN OFF-LINE AT 15:15
PASS 1 OF LOOKING FOR STILL ACTIVE JOBS
QUEMAN MESSAGE: FINAL SHUTDOWN PHASE -- KILLING SPOOLERS
SECOND QUEMAN MESSAGE SENT
QUEMAN MESSAGE: QUEMAN SHUTTING DOWN
FROM NOW ON JOBS WILL ONLY BE 'KILLED'
PASS 2 OF KILLING STILL ACTIVE JOBS
UNLOADING RTSLIB RTS
REMOVING RTSLIB RTS
NON-SYSTEM DISKS WILL NOW BE DISMOUNTED
ALL (DISMOUNTABLE) NON-SYSTEM DISKS ARE NOW DISMOUNTED

ALL SET TO PROCEED WITH SYSTEM SHUTDOWN

PLEASE WAIT FOR THE COMPUTER TO ACTUALLY 'HALT'
WHEN IT DOES, PRESSING 'CONT' WILL BOOT BACK RSTS/E
```

Figure 3-1
Sample SHUTUP Printout

When SHUTUP runs, it prints its header line, followed by the first of two queries. The first query asks for the number of minutes before the system can be shut down, and the second query asks the intervals between warning messages. After the two queries are answered, the

SHUTUP program proceeds with its actions. Further logins are disabled to prevent more users from entering the system. Messages are sent to all on-line terminals at the interval specified by the system manager. Each message tells how many minutes are left until system shutdown. When no time is left, any terminal still logged into the system is automatically logged out. Jobs still active are terminated. All non-system disks are then dismounted. When SHUTUP terminates, it actually halts the machine at address 54.

On systems running QUEMAN and spooling programs, the SHUTUP program includes procedures for communicating with the QUEMAN program. SHUTUP notifies QUEMAN that time sharing operations are stopping. QUEMAN, in turn, notifies the spooling programs which kill themselves when they are finished. When all spooling programs terminate, QUEMAN sends a message to SHUTUP and immediately terminates itself. Unless otherwise specified, SHUTUP does not proceed with the system shut down until it receives this message from QUEMAN.

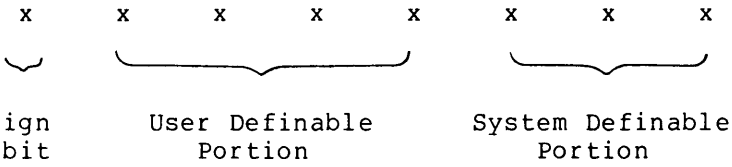
On systems with the auxiliary run time system, SHUTUP unloads the module and removes its entry from the table of run time systems. This procedure enables SHUTUP to dismount the disk on which the module resides.

It is necessary to establish administrative procedures governing RSTS/E system operational hours. All users can be kept informed by means of the NOTICE.TXT file which is printed when a user successfully logs into the system. If system shut down times are fixed, users can plan a work load and properly complete their processing within the allotted hours of scheduled time sharing.

3.3 SETTING JOB PRIORITY, RUN BURST AND MAXIMUM SIZE - PRIOR

The PRIOR system program reports the priority, run burst and maximum size assigned to an existing job. The system manager can change any of the current values to increase or decrease the chance of gaining run time in relation to other running jobs, to determine how much CPU time the job can have when it is compute bound and to increase the area a job can occupy.

The system runs jobs on the basis of priority. The higher a job's priority, the better are its chances of obtaining run time in relation to other running jobs. Priority is determined by an 8-bit priority byte, as shown below.



By running PRIOR, a privileged user can set the user definable portion of the priority byte for his own job or for another job on the system. Since the three system definable bits are normally zero, standard priorities are multiples of 8 between -120 (lowest priority) and +120 (highest priority). Zero is a legal priority. When PRIOR changes a priority, it truncates any value not a multiple of eight to the next lowest legal priority. For example, if the system manager specifies a priority of +10, PRIOR sets the value to +8.

All of the system definable bits are normally off (zero). The least significant bit is set when a keyboard delimiter is typed and the job was waiting for keyboard input. Keyboard delimiters are the CR, LF, FF, CTRL/Z, and ESC (or ALT) characters. The next significant bit is set whenever the CTRL/C combination is typed and can be set at any time. Finally, the most significant bit of the system definable portion is set by a SYS system function call. All system definable bits are cleared when another program is chained or when the system prints READY.

The system definable portion of the priority byte is always less significant than the user definable portion. Therefore, the system definable bits affect priority only within the user definable priority range. If two jobs are running under priority -8, for example, the user who types the CTRL/C combination has a higher priority (that is, priority -6 in this case) than the user who does not type that combination. On the other hand, a third user with priority 0 supersedes two other users whose priorities are -8 and -6.

When a user first logs in on a system, LOGIN is run with priority 0. LOGIN automatically sets the user's job to priority -8. This is the default priority with which most or all of the jobs are run. Only in unusual circumstances should priorities other than -8 be assigned.

On occasion, the user may want to run a non-urgent program that requires a great deal of computation. If time is not a factor in obtaining results, the privileged user can decrease the job priority to improve efficiency for the other users on the system. Conversely, infrequently used detached programs often have higher priorities (typically priority 0) since they must run quickly when needed but do not run compute bound for an extended period and do not run often.

Run burst is the maximum time a job can run compute bound before another job obtains access to the CPU. Each unit of run burst time is equal to 1/60th or 1/50th of a second, depending on the system's power line frequency. (Systems running with the KW11P clock at crystal speeds, rather than at line frequency, have a run burst unit of 1/50th of a second.) If the system is operating off a 60 Hz power line, one run burst unit equals 1/60th of a second. In that case, six units equal 1/10th of a second, which is the run burst default value.

If a specific job is assigned a run burst of 6 units but does not require that much compute bound time, the system automatically transfers control to the next user before the six units have been used. One tenth of a second is generally considered the best run burst time period to insure efficient overall system operation. If a job is guaranteed to become I/O bound (that is, I/O stalled) after a certain amount of computations, PRIOR can be used to specify a run burst period larger than 6. In many cases, a run burst of greater than 6 units has a significant effect on long computational programs.

For scheduling efficiency and accounting data accuracy, run bursts can only be set to multiples of 1/10th of a second. For example, the permissible values for run burst on a 60 Hz system are 6,12,18, and upward. PRIOR automatically rounds any specified run burst value to the next lowest 1/10th second multiple.

The maximum size assigned to a job refers to the available memory space. By specifying the maximum size of a job up to 16K words of memory, a system manager can allow certain jobs to run programs larger than 8K words. This assigned limit does not affect privileged, compiled programs. Thus, a user with a small amount of space can still run system programs that would normally exceed the memory limit.

3.3.1 Running PRIOR

PRIOR is called as follows.

```
RUN $PRIOR
'PRIOR' PRIORITY, BURST, MAXIMUM CHANGER
```

The first query line printed is:

```
ENTER ANOTHER JOB NUMBER?
```

If the current job is to be checked, type the CR key alone. If, however, the job to be checked is not the one under which PRIOR is running, type the job number to be considered. A job number less than 1 or greater than the maximum number of assignable jobs returns the error message ILLEGAL JOB NUMBER ENTERED. Only active, running jobs can be referenced; unassigned job numbers return the above error message.

PRIOR now prints the current priority, run burst and maximum size assigned to the specified job. For example:

```
CURRENT STATISTICS ARE
     -8 PRIORITY
      6 RUN BURST
    16K SIZE MAXIMUM
```

The next query line is:

```
ANY CHANGES?
```

If any or all of this information is to be changed, type Y in response to this query. Typing N or the CR key alone automatically ends the program.

If the user indicates that changes are to be made, PRIOR prints a query line for each parameter in turn. If the value assigned to any parameter is not to be changed, type N or the CR key alone to skip to the next query line. When the typed response to the query line is Y, the message CHANGE IT TO? is printed. Type the new specification as shown below in the sample dialog.

```
RUN $PRIOR
'PRIOR' PRIORITY, BURST, MAXIMUM CHANGER
ENTER ANOTHER JOB NUMBER?

CURRENT STATISTICS ARE:

     -8 PRIORITY
      6 RUN BURST
    16K SIZE MAXIMUM


ANY CHANGES? Y
CHANGE PRIORITY? Y
        CHANGE IT TO? -16
CHANGE RUN BURST? N
CHANGE SIZE MAXIMUM? Y
        CHANGE IT TO? 8


CURRENT STATISTICS ARE:
     -16 PRIORITY
      6 RUN BURST
     8K SIZE MAXIMUM


ANY CHANGES? N


READY
```

Once the last query line has been answered, PRIOR prints the new statistics for verification and prints the message:

```
ANY CHANGES?
```

Typing Y allows the user to make additional changes to this information. Typing N or the CR key alone ends the program.

3.3.2 Changing LOGIN to Set Maximum Job Size

The LOGIN system program sets the swap maximum to 8K words for all users except those whose project numbers are one. This action means that privileged users run with a swap maximum of 16K words. Since, on many systems, programs must run under nonprivileged accounts in job

areas larger than 8K words, it is necessary to modify LOGIN to set a swap maximum larger than 8K words.

To modify the LOGIN.BAS program, the system manager must alter the J% = 8% statement in the first physical line of the multiple statement line at line number 15010 and compile the program on the system library account. The following statement sets the priority, run burst, and swap maximum factors.

```
15010  J% = 8%
       : J% = 16% IF (A% AND -256%) - 256%
       : I$ = SYS(CHR$(6%)+CHR$(-13%)+CHR$(-1%)+
                CHR$(-1%)+CHR$(-2%)+
                CHR$( 0%)+CHR$( 6%)+
                CHR$(-1%)+CHR$( J%))
       : RETURN
```

Change the value 8% in the statement J% = 8% to any value less than or equal to the current default swap maximum used at system start up time. Compile the program on the system library account and set the protection code to <232> as follows:

```
COMPILE SY0:$LOGIN

READY

NAME "$LOGIN.BAC" AS "$LOGIN.BAC<232>"

READY
```

It is recommended that the system manager not replace the original source file LOGIN.BAS with the modified version.

CHAPTER 4

ACCOUNT CREATION AND ACCOUNT STATISTICS

4.1 CREATING AND DELETING USER ACCOUNTS - REACT

The system manager or a privileged user creates and deletes accounts by use of the REACT system program. The REACT system program enters user accounts on and deletes user accounts from either the system device in the public structure or individual private disk devices.

REACT is called by using the RUN command as is shown below.

```
RUN $REACT
```

REACT responds by printing the following message which requests that the user specify a function.

```
'REACT' SYSTEM ACCOUNT MANAGER
FUNCTION?
```

The three valid functions are described in Table 4-1, and explained in the following sections.

4.1.1 Creating Individual Accounts - ENTER Function

The ENTER function creates individual user accounts. When the system manager runs REACT, he invokes the ENTER function by typing E in response to the request for a function. Upon recognition of the E response, REACT prints a series of questions. A response to each question must be typed by the user before the appearance of the next question. The questions are explained in Table 4-2.

Table 4-1
REACT System Program Functions

Function	Abbreviation	Purpose
ENTER	E	To enter individual accounts on system disk or a private disk.
DELETE	D	To delete individual accounts from the system disk or a private disk.
STANDARD	S	To create standard user accounts on the system disk from the ACCT.SYS file at system generation time.

The following is a sample dialogue for the ENTER function.

```
RUN $REACT
'REACT' SYSTEM ACCOUNT MANAGER
FUNCTION?  E
PROJ,PROG?  100,100
DISK:PASSWORD?  DEMO
QUOTA  500
CLUSTER SIZE?  0
PROJ,PROG?  ^Z

READY
```

If the system manager enters an account on a private disk, he permits the owner of the account to create files on that disk. Prior to using REACT, the pack must be logically mounted and placed in the unlock state by means of the UTILTY system program. Refer to the discussion under the title "Disk Management" in Section 7.1.2.

If the system manager enters an account on the system disk, he permits the owner of that account access to the RSTS/E system and use of storage space within the public structure. When a new account is created, REACT also places an entry for the new account in the ACCT.SYS file.

Table 4-2
Responses to ENTER Function Queries

Question	Response Format	Meaning
PROJ,PROG?	n,m	The user account number to be entered in the MFD, where 1<=n<=254 and 0<=m<=254.
	CTRL/Z	The user terminates the dialogue and REACT by typing the CONTROL key and Z combination simultaneously.
DISK:PASSWORD?	passwd	To enter a password for an account on the system disk MFD, where password is from 1 to 6 alphanumerics. No value for DISK need be specified since the system disk is assumed.
	dev: passwd	To enter a password to an account on a private disk, where dev is the device designator and passwd is from 1 to 6 alphanumerics. For example: DK1:PASS The disk specified must be logically mounted and in the unlock state by means of the UTILTY program prior to invoking REACT.
QUOTA?	n	The number of 256-word blocks of disk storage the user account is allowed to retain at LOGOUT time where 0<=n<=65,535 and 0 means no quota is imposed upon the user's account. Therefore, a value of 0 limits disk storage retention only by the amount imposed by the UFD clustersize. LOGOUT is the only DIGITAL-supplied program that checks the quota and, moreover, monitors storage retention only on the public structure.
CLUSTER SIZE?	n	The account UFD cluster size where n is 0, 1, 2, 4, 8, or 16. If 0 is specified, the pack cluster size is used. If non-zero, the value for n must be at least the pack cluster size. Cluster sizes of 1,2, or 4 are recommended for most accounts. The maximum number of files a user is allowed to create is approximated by multiplying the UFD cluster size by 72.
ACCOUNT NAME?	x	Optional. Used by GRIPE system program to identify account owner.

4.1.2 Deleting Accounts - DELETE Function

The DELETE function removes individual user accounts from the system disk or from a private disk. As in the case of the ENTER function, a private disk must be logically mounted and in the unlock state prior to deletion of an account. In addition, before an account is deleted, the UFD of that account must contain no files. To remove all files from the UFD, the ZERO command of the UTILTY system program or the /ZE switch of the PIP system program must be used. See the description of the UTILTY system program in Chapter 7 of this guide and of PIP in Chapter 4 of the RSTS-11 System User's Guide.

The DELETE function is invoked by typing D in response to the request for a function. The REACT program prints a series of questions explained in Table 4-3.

Table 4-3
Responses to DELETE Function Queries

Question	Response Format	Meaning
PROJ,PROG?	n,m	The user account number to be deleted from the MFD.
	CTRL/Z	The user terminates the dialog and REACT by typing the CONTROL key and Z key combination simultaneously.
DISK?	null	By typing the RETURN key, the system device is specified.
	dev:	The device designator of a private disk. For example: DK1: The disk must be logically mounted and in the unlock state by means of the UTILTY system program prior to running REACT.

The following is a sample dialog for the DELETE function.

```
RUN $REACT
'REACT' SYSTEM ACCOUNT MANAGER
PROJ,PROG?  100,100
DISK?  DK1:
PROJ,PROG?  ^Z

READY
```

4.1.3 Automatic Creation of User Accounts - STANDARD Function

The STANDARD function in the REACT system program is provided to facilitate automatic creation of a large number of user accounts when the system disk is built. Explanation of the STANDARD function is presented in detail in Section 4.4 of the RSTS/E System Generation Manual. A few ancillary remarks are made here.

The user creates the ACCT.SYS file as shown in Section 4.3.5 of the RSTS/E System Generation Manual. The file ACCT.SYS is stored in the system library account [1,2]. ACCT.SYS is an ASCII text file, each line of which is formatted with the following: the items which would be specified by the user in response to the questions of the ENTER function and a name item. Each line of the file represents a single account to be created. The general format is as follows.

```
proj,prog,passwd,quota,cluster,name
```

The items proj, prog, passwd, quota, and cluster are described under the ENTER function. The name can contain any additional information about the account such as the owner's name. The name item is not used by REACT but is used by GRIPE. The item, name, must contain no commas, single quotes, or double quotes. The accounts [1,1] and [1,2] can appear in the ACCT.SYS file although they have been created previously during the REFRESH action of the system generation procedure. These account entries in ACCT.SYS are only used by the GRIPE system program.

4.2 PERFORMING SYSTEM ACCOUNTING OPERATIONS - MONEY

The MONEY system program enables the system manager to extract system accounting information for all accounts in the system or for selected accounts. MONEY can be run by a non-privileged user to obtain his own account information (excluding password) as described in Chapter 4 of the RSTS-11 System User's Guide.

MONEY is called by typing the following command while logged into the RSTS/E system.

```
RUN $MONEY
'MONEY' SYSTEM ACCOUNTING PROGRAM
```

If the caller is a privileged user, a sequence of option queries is printed at the keyboard. Typing an answer to one query causes the next one to be printed. The queries and the explanation for each are given in Table 4-4. The account data given as output for each account is described in Table 4-5.

The MONEY system program can be run during normal time sharing. No conflicts arise if the system attempts to update a user's accounting information while the MONEY program is accessing it. When the RESET option is used, MONEY reads and resets to zero the user's accounting information before any system action can update the values being read and reset. Thus, no user accounting information is lost.

Some of the items output can be used to weigh billing or evaluate usage. The item KCT, in effect, reflects system usage more accurately than CPU-TIME. For example, two users may each exhaust one minute of CPU-TIME in an accounting period. However, one user may tie up 2K words of memory while the other may occupy 6K words each time he runs. The first user's KCT value is incremented by 2 for each tenth of a second his 2K job is running. With the 6K user, each tenth of a second he runs, his KCT value is incremented by 6. The 6K user is tying up more system resources and this is reflected in his higher KCT value. Thus, a user's average job size can be gained by dividing the number of KCT's reflected for the accounting period by the number of tenths of seconds derived from the value of CPU-TIME. Referring to the example of values in Table 4-5, user [100,100] average job size of 3.6K is computed by dividing the number of KCT's, 3000, by the number of tenths of seconds derived from CPU-time, 832.

Table 4-4
MONEY Program Options

Option Query	Reply	Explanation
OUTPUT DEVICE?	dev:filename.ext	A file structured or non-file structured device can be specified. Indicating a disk file reduces processing time. For example: MONEY.DMP MONEY writes the data to the file which can later be queued for printing.
	RETURN key	Output is printed at the terminal.
PRINT PASSWORDS?	YES	Typing YES (or any string beginning with Y) causes passwords to be printed. Typing anything else omits passwords.
RESET?	NO	Typing NO (or any string not beginning with Y) causes the accumulated accounting data to be preserved.
	YES	Typing YES (or any string beginning with Y) causes the following items to be reset to zero. CPU-TIME KCT's CONNECT TIME DEVICE TIME The data is reset after the information is dumped.
DISK?	dev:	Type the specific disk device designator with unit number n to select the accounting data from a private pack.(1)
	RETURN key	The accounting data selected is for all public disks.

(1)Meaningful accounting data on a private disk is account number, number of blocks occupied, disk quota, and UFD cluster size. The system updates only the system disk MFD with CPU time, KCT's, connect time, and device time.

Table 4-4 (Cont.)
MONEY Program Options

Option Query	Reply	Explanation
SELECTIVE?	NO	Typing NO (or any string not beginning with Y) causes accounting data for all accounts (on the private pack or on the system, whichever the reply to the DISK? query indicates) to be dumped.
	YES	Typing YES (or any string beginning with Y) causes an additional query ACCOUNT?
ACCOUNT?	n,m	The account query appears if the reply to the SELECTIVE? query is YES. Accounting data is dumped for the account specified by the project-programmer number [n,m], following which, the query is repeated.
	CTRL/C CTRL/Z	Typing CTRL/C or CTRL/Z terminates the program run.

Table 4-5
MONEY Program Output

Header	Example	Description
ACCT	100,100	Project-programmer number (account)
PASSWORD	DEMO	Account password given at login time
CPU-TIME	1:23.2 (one minute, 23.2 seconds	Number of hours:minutes: seconds.tenths of a second of processor time the account has used since the last reset.
KCT's	3000	Core usage factor (kilo-core-ticks). One KCT is the usage of 1K of core for one tenth of a second.
CONNECT	2:34 (2 hours, 34 minutes	Number of hours and minutes (hh:mm) of terminal connect time.
DEVICE	20 (20 minutes)	Number of hours and minutes (hh:mm) of device usage time, excluding disks.
DISK	100	Number of 256-word blocks of disk storage allocated.
QUOTA	500	Number of 256-word blocks the account is allowed to retain at logout time.
UFD	2	UFD cluster size

The value under the header description DISK reflects the actual number of blocks tied up in file allocation on disk. It is not necessarily the same value reported by the CATALOG system command. A file may occupy 1 block on a disk as reflected by the CATALOG command, but ties up 3 additional blocks of disk storage if the file cluster size is 4 blocks. In essence, the user is depriving the system from claiming four contiguous blocks in the cluster although the file is currently occupying only one block.

The information given under the header UFD is the user file directory cluster size. No other system program returns this value which the system manager specifies when the account is created by REACT. The cluster size is provided in the MONEY output for information purposes only and has no accounting value.

It is advised that the system manager periodically execute the RESET option to prevent overflowing the accounting values stored on disk. The following list shows the maximum times that can be stored for each statistic without an overflow.

	On Disk	In Memory
Device Time	1092 hours	1092 hours
Connect Time	1092 hours	68 hours
CPU Time	116 hours	29 hours
KCT's	116 hours at 16K words	29 hours at 16K words

At logout time, the system updates the values on the disk with accumulated values from memory. Thus to prevent loss of accounting data, the user must log a job off the system before any of the values in memory overflow. In like manner, the system manager must execute the RESET option before the values on disk overflow. The sizes of the accounting data fields on disk allow approximately one week of continuous system operation without overflow. Therefore, MONEY must be run with the RESET option at least once per week. The MONEY program indicates reset is in effect by adding the text WITH DATA BEING RESET to the accounting printout header line.

4.3 DISK SYSTEM CATALOG - SYSCAT

The SYSCAT (system catalog) system program prints a current directory listing of any disk. The system manager or a privileged user must specify the file or device on which the information is to be printed. A standard CAT command prints a listing of the user's files on a device. SYSCAT prints a file listing of all files on a given disk. SYSCAT is called as follows:

```
RUN $SYSCAT
```

The first query line printed is:

```
OUTPUT CATALOG TO?
```

Type the device or file on which the catalog is to be printed. If the CR key is typed alone, SYSCAT prints the information on the user keyboard.

The next query line printed is:

```
CATALOG OF?
```

Type the specification of the disk to be examined. If the CR key is typed alone, SYSCAT prints a catalog of the system disk(s). Shown below is a sample dialog.

```
RUN $SYSCAT
OUTPUT CATALOG TO?
CATALOG OF? DK0:
SYSTEM CATALOG OF DK0: ON 18-JUL-74 02:14 PM
```

SYSCAT prints the name of the disk, the current date and time and continues with the catalog.

SYSCAT prints each account in the order of its creation date and lists the quota for each account. The number following the word DISK is the effective number of blocks used, as a function of cluster size. For example, if one block is used for a specific file, but the file cluster size is 8, the number of effective blocks used is 8.

The number following the word UFD indicates the user file directory cluster size. The first six columns of information in the catalog listing correspond to the standard CAT printout. The last column indicates the cluster size (in blocks) of each file. After each account listing, SYSCAT prints the number of files and the number of blocks used in the account. For example:

```
2 FILES CLAIMING 7 BLOCKS IN ACCOUNT
```

The following sample of a partial listing of a system disk demonstrates the format SYSCAT employs.

```
RUN $SYSCAT
OUTPUT CATALOG TO?
CATALOG OF?
SYSTEM CATALOG OF SYSTEM DISK(S) ON 25-SEPT-74 10:58 AM
```

ACCOUNT CREATION AND ACCOUNT STATISTICS

```
ACCOUNT [1,1]              QUOTA 0            DISK 0          UFD 16

ACCOUNT [0,1]              QUOTA 0            DISK 4292       UFD 4

BADB   .SYS        0          63       29-JUL-74 04-MAR-74 10:00 AM 2
RSTS   .CIL        300        63       19-AUG-74 30-JUL-74 12:10 AM 2
SATT   .SYS        10         63       06-AUG-74 30-JUL-74 12:10 AM 2
SWAP0  .SYS        2048       63       28-AUG-74 30-JUL-74 12:10 AM 2
SWAP1  .SYS        672        63       30-JUL-74 30-JUL-74 12:10 AM 2
SWAP2  .SYS        976        63       20-AUG-74 30-JUL-74 12:10 AM 2
OVR    .SYS        28         63       30-JUL-74 30-JUL-74 12:10 AM 2
ERR    .SYS        8          63       22-AUG-74 30-JUL-74 12:10 AM 2
BUFF   .SYS        12         63       10-SEP-74 30-JUL-74 12:10 AM 2
CRASH  .SYS        52         63       23-SEP-74 30-JUL-74 12:10 AM 2
 10 FILES CLAIMING 4106 BLOCKS IN ACCOUNT

ACCOUNT [1,2]              QUOTA 0            DISK 1858       UFD 16

LOGIN  .BAC        23         124      25-SEP-74 01-AUG-74 09:19 AM 4
LOGOUT.SYS         15         124      25-^C

READY
```

CHAPTER 5

SPOOLING OPERATIONS

Spooling operations on RSTS/E depend upon interjob communication between a spooling program and a queue management program QUEMAN. Possible spooling programs are SPOOL, BATCH and RJ2780. SPOOL executes requests for a line printer unit and facilitates more efficient use of the device. BATCH executes requests on a pseudo keyboard device and provides non-attended job operations. RJ2780 is optional on all systems and executes requests for communications between two computer systems. QUEMAN manages all requests for spooling programs and ensures that requests are handled properly.

Interjob communication is attained by QUEMAN sending and receiving messages with the send/receive system function call. A user job must be privileged to run QUEMAN since it declares itself a receiving job on the system. Spooling programs on the system also employ the send/receive message facility to communicate with the common receiving job QUEMAN.

QUEMAN creates a common file QUEUE.SYS on account [1,2] on the system disk. The file accommodates up to 254 queued requests for spooling programs. Each request handles a maximum of 14 files. Each job request queued with the /AFTER option occupies two request slots in QUEUE.SYS.

The QUEMAN program must have write access to the QUEUE.SYS file since it creates requests for spooling programs and updates control information in the file. A spooling program receives requests by notifying QUEMAN that it is ready to process. QUEMAN, in turn, accesses the queue file to determine if any requests exist for that spooling job. If no queued requests exist, the spooling program performs an indefinite sleep operation until awakened by QUEMAN with a request to process. In this manner, one program alone is responsible for maintaining the queue file.

A user creates a request for a spooling program by running the QUE program. When QUE executes a command to queue a request, it sends messages to the QUEMAN program rather than directly writing the request to QUEUE.SYS. However, when a user requests to list current requests, QUE reads the information directly from QUEUE.SYS. This action possibly confuses a user if he tries to list a queue immediately after he types a request. Often QUEMAN is in the process of parsing the new request when QUE accesses QUEUE.SYS to list pending requests. For more information on the QUE program, see Section 4.11 of the RSTS-11 System User's Guide.

QUEMAN enters requests in the QUEUE.SYS file based on a priority number between 1 and 255 specified in the QUE command. QUEMAN inserts

a request with a given priority number ahead of those with lower priority numbers. Routine requests are assigned a priority of 128. A privileged user can assign a priority higher than 128 and thus can cause QUEMAN to process the request before others of lower priority. All users can assign a priority of less than 128 and can thus cause QUEMAN to process the request only after all others of higher priority are processed. In this manner, requests of a non-critical nature do not slow up the processing of routine requests.

Whenever a spooling program encounters an error, it interrupts processing and informs QUEMAN of the error. QUEMAN generates a request for user action and broadcasts the appropriate identifying information on the system console terminal. A user on the system handles the request by removing the error condition and typing a reply to QUEMAN. QUEMAN, in turn, clears the request from its internal tables and sends the reply to the spooling program. The spooling program, upon receipt of the reply, continues processing based on the user's request.

For a spooling program servicing a line printer, certain errors are possibly self-correcting. This situation arises because the device handler for the line printer tests the ready status of an off line unit every 10 seconds. If the user removes the cause of the error (for example, fixes a paper jam) and puts the line printer on line again, the line printer software detects the ready status and automatically continues output. As a result, the spooling program detects that the error condition is cleared and informs QUEMAN. Finally, QUEMAN removes the request for user action since the spooling program has continued processing the queued request.

Because of the highly interdependent nature of the queue management and spooling mechanism, it is recommended that the user not terminate any of the programs involved. Such termination possibly results in loss of data or destruction of the queue file. The user can leave terminating of spooling operations to the system shutdown procedures.

The SHUTUP system program executes shutdown procedures which preserve file integrity. SHUTUP sends two messages to QUEMAN. The first message initiates QUEMAN's off-line procedure which entails informing each spooling program to complete processing the current file. Thereafter, a spooling program closes all files, tells QUEMAN it is off-line, and kills itself.

QUEMAN waits for all spooling programs to go offline. After SHUTUP has killed all attached jobs, it sends a second message to QUEMAN telling QUEMAN to kill all remaining spooling programs and to kill itself. The text SECOND QUEMAN MESSAGE SENT appears on the terminal. SHUTUP waits 15 seconds for a reply from QUEMAN. QUEMAN proceeds to kill any remaining spooling jobs, to send the reply to SHUTUP that all spooling jobs are killed, and to kill itself. After receiving the reply from QUEMAN, SHUTUP waits 10 seconds before it begins killing detached jobs.

If SHUTUP does not recieve a reply from QUEMAN within 15 seconds, it prints the text QUEMAN NOT RESPONDING - RETRY? Typing NO causes SHUTUP to proceed shutting down the system. Typing any string not beginning with N causes SHUTUP to retry the second message to QUEMAN.

5.1 OPERATING THE QUEUE MANAGER - QUEMAN

To run QUEMAN, the job must be privileged and the user must type the following command while at BASIC-PLUS command level.

```
READY

RUN $QUEMAN
QUEMAN V06A-07 - RSTS V06A-02 SYSTEM TEST
STARTED AT 11:31 ON 29-APRIL-74
*
```

QUEMAN opens the system file QUEUE.SYS. If the file does not exist, QUEMAN creates and initializes it. QUEMAN notifies the user of this action by printing the following message.

```
NO QUEUE FILE FOUND -- WILL INITIALIZE
```

If the file exists and QUEMAN has write access to the file, it prints the * character indicating its readiness to accept commands. Another program having write access to the file causes QUEMAN to print the message QUEUE FILE OPENED BY ANOTHER PROGRAM ALREADY...CAN'T RUN STOP AT LINE 30. At this point, the user must determine which job has the file $QUEUE.SYS open and must terminate that job. By typing the CONT system command in response to the STOP AT LINE 30 message, the user causes QUEMAN to retry the open operation.

To cause QUEMAN to DETACH, type \DE in response to the * character. For example,

```
*\DE
DETACHING...
```

As a result, QUEMAN prints the DETACHING message and detaches itself from the keyboard.

NOTE

> On ASR33 type terminals, the backslash character is generated by typing the SHIFT key and the L key simultaneously. The system echoes the SHIFT/L combination by printing the \ character.

In general, QUEMAN should be run in the DETACHed state, except when some actual interaction with it is required.

To execute QUEMAN commands, the user must attach the QUEMAN program to the terminal and type the particular command. For example, to terminate the QUEMAN program, perform the following actions as shown in the sample dialog.

```
ATTACH 5

PASSWORD:
ATTACHING TO JOB 5
'QUEMAN' ATTACHED
TYPE  \DE TO DETACH
*\EX

READY
```

The ATTACH command described in Section 4.1 of the RSTS-11 System User's Guide attaches the job to the terminal. The QUEMAN program prints the 'QUEMAN' ATTACHED message to indicate that it is attached to the terminal.

NOTE

At no time should the user type the CTRL/C combination. Such action possibly destroys the QUEUE.SYS file and causes the program to print the QUEUE FILE ENDANGERED message and to terminate execution.

The asterisk character (*) indicates that QUEMAN is ready to execute a command. To terminate QUEMAN, type \EX. Subsequently, the program closes its files and exits to the system monitor as indicated by the READY message.

Other commands recognized by QUEMAN are listed and described in Table 5-1. A command is preceded by a backslash character to distinguish it from other possible responses.

Whenever QUEMAN requires help from the user, it performs a standard procedure. For example, if a spooler encounters a hung line printer, it sends a message to the QUEMAN program. QUEMAN broadcasts the standard text on the system console keyboard in the following format:

```
****QUEMAN (j) m:n REQUEST:  text ?
```

where:

j is the job number under which QUEMAN runs.

m is the message identification number between 0 and 255 which QUEMAN relates to a request for user action.

n is the logical name of the requesting program. Examples of names are LPnSPL, BATCH or RJ2780. The character n in LPnSPL denotes the line printer unit to which the SPOOL program directs output.

text indicates QUEMAN must process a request from a spooling program. The text between the colon and the ? characters is the action message to which the user must respond.

QUEMAN performs no further processing for the designated spooling job unless the user attaches QUEMAN to the terminal and responds to the request or unless QUEMAN deletes the message for the spooling job.

Table 5-1
QUEMAN Commands

Command	Meaning
\IN	Initialize the QUEUE.SYS file. All pending jobs for all devices are removed and the protection is set to <40>.
\EX	Immediately close all files, remove QUEMAN from system receiver table and reset the priority. It is recommended that the user terminate all spooling programs before typing the \EX command. See Section 5.4 for terminating instructions.
\RE:n	Remove the spooling job indicated by the job number n from QUEMAN internal tables. The user can determine the proper n for a spooling program by using the \ST command.
\ME	Print at the terminal any pending messages from spooling jobs.(1)
\ST	Print at the terminal the status of spooled jobs currently running on the system.
\DE	Detach QUEMAN from the terminal.

(1)The QUEMAN program does not ordinarily print messages when it is attached to a terminal. When attached to a terminal, QUEMAN prints messages under the following circumstances.

1. If ten messages have been received since the job was attached,
2. If the user attempts to detach, or
3. If the user types the \ME command.

To respond to a request from QUEMAN, the user must type a line containing the identification number of the request being serviced and the command to the indicated spooling job. For example:

```
****QUEMAN   (05)   1:LP0SPL REQUEST:LP0 HUNG?
ATTACH 5
PASSWORD:
ATTACHING TO JOB 5
'QUEMAN' ATTACHED
TYPE \DE TO DETACH
*1 CO
*
```

The request LP0 HUNG is an error message generated by the spooling program SPOOL running for line printer unit 0 (LP0:). The request identification 1 tells the user that this is the first active request for user action. The user can determine the cause of the error condition and remedy it. He then attaches the QUEMAN job to the terminal and types a response to the asterisk character printed after QUEMAN attaches. The user types the identification number of the request to which he is responding (1) and the proper command (CO) to the spooling program. QUEMAN prints the asterisk again. The user can wait to determine if the response to the request is sufficient to correct the error.

If a line printer error condition is removed before the user types a response to the request, the system automatically resumes printing according to the option the requester specified when he queued the file. The spooling program SPOOL notifies QUEMAN, which deletes the request. QUEMAN, in turn, notifies the user by printing a message. The following sample printout shows the action.

```
****QUEMAN (05) 2:LP0SPL REQUEST: LP0:HUNG?
REQUEST #2 DELETED BY LP0SPL
```

(For more information regarding SPOOL, see Section 5.2.)

QUEMAN handles a response to an action request by passing the text typed to the spooling program. The spooling program runs and processes the text. In the example shown, SPOOL executes the CO command of the SPOOL program. If the user's action properly remedied the hung condition of line printer unit 0, the CO command causes SPOOL to perform the defined operation. When the command or the user's action is not sufficient to continue processing, the spooling program generates another request message for the QUEMAN program.

If the user types an invalid message number, QUEMAN prints text similar to the following sample.

```
MESSAGE #n NOT OUTSTANDING
*
```

The user is allowed to type a command or another response to the action request.

QUEMAN prints error messages when the user types an invalid command or response to the * character. These messages are listed and described in Table 5-2.

The QUEMAN program monitors the activity of spooling programs and broadcasts information messages to the system console terminal. For

example, if the user runs SPOOL, QUEMAN broadcasts a message similar to the following.

```
QUEMAN MESSAGE:  LP1SPL (4) PUT ONLINE AT 04:23 PM
```

This message informs the user that a spooling program started execution on line printer unit 1 with job number 4.

When the user types the CTRL/C combination to a SPOOL program, QUEMAN typically prints the following text.

```
LP1SPL MESSAGE:  ^C TO SPOOLER
QUEMAN MESSAGE:  LP1SPL (4) REQUESTED OFF-LINE -- TAKEN
                 OFF-LINE AT 04:22 PM
```

Table 5-2
QUEMAN Error Messages

Message	Meaning
INVALID RESPONSE -- x	The response x typed by user was neither a response to the action request nor a valid command.
JOB n NOT ON-LINE	An attempt was made to release a job by the \RE:n command and that job is not running under QUEMAN control.
MESSAGE #n NOT OUTSTANDING	The user typed an identification number indicated by n. QUEMAN determines the number requires no user action response, and continues operation.
QUEUE FILE ENDANGERED	QUEMAN detects a CTRL/C and attempts to exit without destroying the QUEUE.SYS file.
SPOOLERS ON-LINE -- CAN'T INIT	User types the \IN command, not executable by QUEMAN since spooling programs are currently running.
BAD JOB NUMBER	User typed the \RE:n command and n is not a valid number. For example, \RE:A.

The first line records that the spooling program for LP1: detected a CTRL/C combination. The second line informs the user that QUEMAN removed the job from its control tables at a specific time.

The BATCH program sends messages to QUEMAN which, in turn, broadcasts them to the system console terminal. In such a manner, BATCH maintains a log of the jobs it executes. For example, QUEMAN prints such a sequence of messages from BATCH.

```
BATCH MESSAGE:  COMPAR: STARTED AT 03:02:22 AM

      .

      .

      .

BATCH MESSAGE:  COMPAR: COMPLETED AT 03:08:09 PM
```

The text BATCH MESSAGE identifies the information as coming from the BATCH program. The text COMPAR identifies the BATCH job name currently being executed. The remaining portion of each message tells the time in hours, minutes, and seconds for the start and completion of the job. For more information on BATCH operation, see Section 5.3.

If any messages occur while QUEMAN is attached to a terminal, the program saves the QUEMAN job messages until the user types either the \ME command or the \DE command. In this manner, the user can type responses to requests without interruption from QUEMAN. Additionally, if new messages are pending and the user types the \DE command, QUEMAN prints the messages and gives the user a chance to respond to an action request before it detaches itself. For example,

```
*\DE
ADDITIONAL MESSAGES:
QUEMAN MESSAGE: LP1SPL (4) RELEASED -- TAKEN OFF-LINE AT 05:17 PM

TYPE \DE TO DETACH

*
```

The user types the \DE command to QUEMAN when new messages are pending. QUEMAN prints any additional messages (but not all pending messages) and prints TYPE \DE TO DETACH followed by the * character. The user can then type a response to the action request or can type a QUEMAN command.

5.2 LINE PRINTER SPOOLING PROGRAM - SPOOL

The line printer spooling program runs without user intervention and transfers files from disk, DECtape or magtape to a line printer. To run SPOOL, the user job is logged into the system under a privileged account. To start SPOOL, type the following command.

```
RUN $SPOOL
```

SPOOL runs and checks that the project number of the account is 1. If the account is not privileged, the program prints the PROTECTION VIOLATION message and terminates. Otherwise, the program prints two lines. The first line tells the program and system names and version numbers and the second line requests the unit number of the printer. For example,

```
SPOOL V06A-09 - RSTS V06A-02 SYSTEM #880
LP UNIT#?
```

SPOOL checks that the unit number entered is between 0 and 7. If the number is invalid, SPOOL prints the query again.

With the line printer unit number, the user can specify the following options.

Option	Meaning
/AS	Reserve the line printer unit to the job as if the user had typed the ASSIGN LPn: command.
/LN:nnn	Set the default form length to nnn which can be between 1 and 127. This value is used in the line printer mode option.
/WI:nn	Set the line printer width to nn to properly adjust the width of the job header pages. This is useful when 80 column paper is used in a 132-column unit. The option does not prevent the program from sending lines longer than nn to the printer. A line longer than 80 characters prints off the page if the printer is configured for 132 columns but 80 column paper is used.

For example, to set the line printer width to 80 columns and assign line printer unit 1 when SPOOL starts, type the following.

```
LP UNIT #?  1/WI:80/AS
```

After entering a valid response to the UNIT # query, SPOOL opens the character generation file CHARS.QUE on the system library account. If the file does not exist, the program prints the message CHARS.BAS HAS NOT BEEN RUN -- CAN'T RUN and terminates. The user must run the CHARS.BAS program described in Appendix C.

The program next determines the width of the printer by inspecting a parameter in the device data block.

SPOOL next communicates with the QUEMAN program which initializes entries in its control tables for the related device. If the queue

manager is not running, SPOOL prints the following message:

```
QUEMAN NOT RUNNING -- CAN'T RUN
```

SPOOL terminates. The user must run QUEMAN as described in Section 5.1 and run SPOOL again. If a spooling program is currently running for that device, the system generates an error (ERR=18) and the program prints the following message.

```
ILLEGAL SYS ( ) USAGE AT LINE 21010 -- SPOOLER WILL HALT
```

The program attempted to declare the job as an eligible receiver and determined that SPOOL is currently active for that device. As a result, it terminates. Otherwise, SPOOL subsequently prints the following message:

```
DETACHING...
```

The program then enables CTRL/C trapping and detaches itself from the terminal.

To properly terminate SPOOL, the user attaches QUEMAN to the terminal and types the \RE command with the job number of SPOOL. If the user does not know the proper job number, he can type the \ST command to gain a status report. A typical procedure is shown in the following dialog.

```
ATTACH 2

PASSWORD:
ATTACHING TO JOB 2
'QUEMAN' ATTACHED
TYPE \DE TO DETACH
* \ST
 2 SPOOLERS ON-LINE
 3      LP0SPL  LP0  2  0000
 4      LP1SPL  LP1  2  0000
*\RE:4
*\DE
ADDITIONAL MESSAGES:
QUEMAN MESSAGE:  LP1SPL (4) RELEASED  AT 05:17 PM
TYPE \DE TO DETACH
* \DE
DETACHING ...
```

Before detaching, QUEMAN prints an information message concerning the LP1SPL job. For more information, see the description of terminating spooling programs in Section 5.4.

The user can terminate SPOOL by attaching it to a terminal and typing the CTRL/C combination. This method is not recommended since the SPOOL program possibly is printing a job or QUEMAN is currently sending a print request to the SPOOL program. If the user does terminate SPOOL by typing the CTRL/C combination, SPOOL kills the job under which it is running. This action leaves the terminal logged off the system. QUEMAN subsequently prints the following messages at the system console terminal.

```
LP1SPL MESSAGE:  ^C TO SPOOLER
QUEMAN MESSAGE;  LP1SPL (4) REQUESTED OFF-LINE--TAKEN
                 OFF-LINE AT 04:22 PM
```

The first message informs the user that the LP1SPL job detected the CTRL/C combination. The second message reports that QUEMAN has removed the LP1SPL job from its internal tables.

Normally, the SPOOL program starts when the INIT system program executes commands in the START.CTL and CRASH.CTL files. For example, a portion of a typical START.CTL file appears as follows.

```
FORCE KB9:  RUN $QUEMAN
FORCE KB9:   \DE

      .

      .

      .
FORCE KB12: RUN $SPOOL
FORCE KB12: 0
```

These commands assume both keyboards are logged into the system under privileged accounts. For more information on the INIT system program, see Section 3.1 of this document.

5.2.1 Recovery from Line Printer Errors

Control of error handling in the spooling process is by interaction between the spooling program and the QUEMAN program. For example, if the supply of paper is exhausted or if the paper jams, SPOOL generates the DEVICE HUNG OR WRITE LOCKED error (ERR=14), sends a message to QUEMAN, and discontinues processing. QUEMAN prints a message at the system console terminal (KB0:) similar to the following:

```
****QUEMAN (02)   3:LP0SPL REQUEST:  LP0:  HUNG?
```

The message tells the user that the spooling job for line printer unit 0 (LP0SPL) has generated request number 3. Before SPOOL can resume, the user must correct the error. If SPOOL determines that the error is gone, it continues printing based on the options specified when the requester queued the file for printing. In such a case, SPOOL notifies QUEMAN which, in turn, deletes request number 3 and prints a message to that effect.

If the user cannot correct the error, or wishes to perform some other operation, he must leave the line printer off-line, attach the QUEMAN job to a terminal and type the request number followed by a SPOOL command. Table 5-3 describes the valid commands. If the user types an invalid command, SPOOL causes QUEMAN to print the ILLEGAL RESPONSE message.

Table 5-3
SPOOL Commands

Format	Operation	Meaning
CO	Continue	Continues by printing the current job according to options the user specified to QUE.
DE	Defer	Places current job at the end of the queue to be rerun at a more convenient time.
RE	Restart	Continues by printing the entire job again from the job header page onward.
KI	Kill	Terminates the current job and removes it from the queue file.

To respond to a pending action request, leave the line printer off-line, note the job number reported in the QUEMAN message, and type the ATTACH command with the number as follows.

```
ATTACH 2
PASSWORD:
ATTACHING TO JOB 2
'QUEMAN' ATTACHED
TYPE \DE TO DETACH
*
```

To restart the job from the beginning, type the request number with the RE command.

```
*3 RE
*\DE
```

QUEMAN passes the string RE to the spooling program related to request number 3 and prints the asterisk character. The user can then place the line printer on line. SPOOL runs and begins printing the job from the job header. The user then types \DE to detach QUEMAN. For more detailed information on QUEMAN, see Section 5.1.

5.2.2 Line Printer Output

SPOOL generates job header and file header pages to identify print jobs and files within a print job. Both types of header page contain identification and general accounting information. The identification information consists of large, easily readable, block letters created from the character generation file CHARS.QUE. The accounting information is in standard letters and placed on the page according to the type of header.

The job header identification consists of the account number of the user requesting the job and the job name the user gave in the QUE command. If no job appeared in the QUE command, SPOOL prints the filename of the first file in the request as the job name.

General accounting information for the job header is centered on the page and is offset from the identification information by two rows of special characters. The accounting information contains three lines of data. The first line consists of the job name, current date, current time, and requester's account in the following general format.

JOB name PRINTED ON date AT time FOR USER [n,m]

The second line comprises the date and time when the user created the request and the device for which he created the request. The third line gives the QUE options the user specified in the job identification part of the QUE command. If the user gave no options, SPOOL prints only /MODE=0.

The file header identification shows, on separate lines, the filename and extension of the file SPOOL printed. If SPOOL does not print the file because of an error, the identification and accounting information are replaced by an error message framed above and below by five rows of special characters. The error message has the following format:

FILE filename specification -- RSTS error message

The file specification includes the device, filename, extension and project-programmer field. The error message is the text generated by the system upon encountering such an error. These errors are summarized in Appendix C of both the BASIC-PLUS Language Manual and the RSTS-11 System User's Guide.

The accounting information for the file header appears below the identification and is framed, above and below, by two rows of special characters. The first of two lines of data gives the job name used when SPOOL printed the file and shows the current date, time, and account as the job header accounting information does. The second line gives the complete file specification and the QUE options the user specified in the file identification part of the QUE command. The QUE options are preceded by the text SWITCHES=. If the user gave no options in the QUE command, SPOOL prints the assigned default values. For more information on the QUE options, see Section 4.11 of the RSTS-11 System User's Guide.

5.2.3 Job Error Messages

Errors SPOOL encounters during printing are reported in the line printer output. SPOOL differentiates the error message from requested output by framing the text with five rows of special characters.

The text of the error message is split in two parts separated by - characters. The first part of the text varies according to the type of error. If the error is other that DEVICE HUNG OR WRITE LOCKED, the message begins with the RSTS error message text. Otherwise, the text begins with the line printer device designator followed by the words HUNG ERROR. The remainder of the first part qualifies the nature of the error. If the error is file related, SPOOL includes the text FILE followed by the filename specification.

The second part of the text varies according to the operator or system action taken to recover from the error. The following list shows the texts and related meanings.

JOB RESUMED	Operator used CO command.
JOB DEFERRED	Operator used the DE command to defer the job.
JOB RESTARTED	Operator used the RE command to restart the job or the system automatically restarted the job.
JOB ABORTED	Operator used the KI command to terminate the print job.

If operator action was involved in the error, the program prints the text BY OPERATOR following the error text. If, for any reason, SPOOL cannot continue or restart, it includes the text FAILURE TO RESTART.

5.3 BATCH PROCESSOR PROGRAM - BATCH

The batch system program runs without user intervention and executes files of standardized commands queued on the related batch device. The number of batch devices possible depends upon the number of pseudo-keyboards available for use by BATCH. To run BATCH, the user job is logged into the system under a privileged account. To start BATCH, type the following command.

```
RUN $BATCH
```

BATCH runs and checks that the project number of the account is 1. If the account is not privileged, the program prints the PROTECTION VIOLATION message and terminates. Otherwise, the program prints two lines. The first line gives the program name and version number and the second line requests the unit number of the batch device. For example:

```
BATCH V06A-10
BATCH UNIT BA?
```

The user can type the RETURN key to indicate the general batch processor (BA:) or can type a decimal number to indicate a distinct batch device (BA1:, BA2:, and onward).

The unit designations for batch processors help to selectively process jobs. The general batch processor executes only those requests queued for BA:. Requests queued for batch processors BA0: through BA7: are excuted only by that respective batch unit. Requests queued for the general batch processor, however, are executed by any batch unit running.

To implement selective batch processing, run a certain batch processor during peak time sharing hours and run a different batch processor during periods of slower activity. For example, the general batch processor (BA:) can process batch requests as needed during the day. In the evening, a disignated batch unit (BA0: through BA7:) can be started to process non-urgent requests queued for it during the day.

BATCH runs and processes messages through the QUEMAN program, which must be running on the system. For more information on sending messages through QUEMAN, see Section 4.12.3.8 of the RSTS-11 System User's Guide.

5.4 TERMINATING AN INDIVIDUAL SPOOLING PROGRAM

It is possible to terminate an individual spooling program. However, it is not recommended since QUEMAN automatically terminates spooling programs when SHUTUP runs. To terminate a particular spooling job, the user must attach QUEMAN to a terminal, release the desired spooling job, and run the UTILTY system program to kill the job. It is important that the user release the spooling job by QUEMAN because requests in the QUEUE.SYS file may otherwise be lost.

The following sample dialog shows the proper way to terminate the BATCH controller program.

```
ATTACH 2
PASSWORD:
ATTACHING TO JOB 2
'QUEMAN' ATTACHED
TYPE \DE TO DETACH
* \ST
 3 SPOOLERS ON-LINE
 4            LP1SPL LP1  2  0  0  0
 3            LP0SPL LP0  2  0  0  0
 5            BATCH5 BA*  3  0  0  0
*\RE:5
*\DE
ADDITIONAL MESSAGES:
QUEMAN MESSAGE: BATCH5 (5) RELEASED--TAKEN OFF-LINE AT 12:04 PM
TYPE \DE TO DETACH
*\DE
DETACHING...
HELLO 1/100
PASSWORD:
JOB(S) 2 3 4 5 ARE DETACHED UNDER THIS ACCOUNT
JOB NUMBER TO ATTACH TO?
4 OTHER USER(S) ARE LOGGED IN UNDER THIS ACCOUNT
READY

RUN UTILTY$
'UTILTY' SYSTEM UTILITY PROGRAM
? KILL 5
? EXIT

READY

BYEF
```

If the \RE:n command is typed and the spooling program is currently processing a request, QUEMAN places that request at the beginning of the queue. When the spooling program starts again, that request is immediately restarted from the beginning. However, when SHUTUP terminates time sharing, the status of jobs terminated depends on the individual spooling program: SPOOL and RJ2780 jobs are placed at the beginning of the queue and BATCH jobs are deleted from the queue. For jobs terminated individually using the \RE:n command, QUEMAN always places a currently processing request at the beginning of the queue.

CHAPTER 6

SYSTEM ERROR DETECTION

6.1 MANAGING ERROR LOGGING - ERRCPY, ERRCRS, AND ERRDIS

Logging of hardware errors is an automatic function of the RSTS/E monitor. To gain the advantages of error logging, the system manager must properly employ the ERRCPY, ERRCRS, and ERRDIS system programs.

The ERRCPY program retrieves error-related data logged by the RSTS/E monitor. Upon occurrence of a hardware error, special routines save the contents of the device registers in small buffers and effectively send a message to the ERRCPY program. The system awakens ERRCPY which transfers the saved data to disk. Since the number of messages which can be queued at any given time is limited, ERRCPY must be running to prevent loss of valuable diagnostic information.

The ERRCRS program retrieves error-related data saved following a system crash. When a system crash occurs and certain conditions are in effect, the monitor preserves the contents of certain critical parts of the system. The system file CRASH.SYS holds this information along with other error-related device data. The ERRCRS program transfers the information from the CRASH.SYS file to another disk file which has the same format as the one created by the ERRCPY program.

The ERRDIS system program produces summaries of error-related data and formats it for output to a hard copy device. This program provides the record of errors logged on the RSTS/E system.

6.1.1 Operating and Using the Error Copy Program - ERRCPY

The error copy system program ERRCPY reads error-related information stored in the monitor part of memory and writes it to a special disk file. The system manager must ensure that the proper commands are created in the START.CTL and CRASH.CTL files as described in Section 3.1 so that ERRCPY is started and active during time sharing operations. The following discussion outlines the entire process of activating the job which runs ERRCPY.

When the RSTS/E system starts up, commands in either the START.CTL or CRASH.CTL control file are executed by the INIT system program. If the command FORCE KB0: RUN $ERRCPY appears in the control file, the command RUN $ERRCPY is placed in the input buffer of the console terminal (KB0:) as if it had been typed at the terminal. Meanwhile, the accompanying END command in the control file causes termination of the INIT system program and causes the console terminal to be placed

at BASIC-PLUS command level (edit mode), as signalled by the READY message being printed. The console terminal remains logged into the system under account [1,2].

When the system executes the command RUN $ERRCPY from the input buffer of KB0:, the ERRCPY program runs and detaches itself from the console terminal as indicated by the message DETACHING printed at the console terminal. The console terminal is no longer logged into the system, but ERRCPY continues running as a detached job under account [1,2].

When ERRCPY is activated, it exists in the SL (sleep) state and neither occupies memory storage nor uses CPU time until awakened by the RSTS/E Monitor error logging routines. When error logging detects a hardware error, it causes ERRCPY to run and write the error-related information to a special file ERRLOG.FIL. The file is stored under the system library account [1,2] on the system disk. If ERRCPY is not running, the diagnostic area can overflow and the history of subsequent errors can be lost. Therefore, the system manager must properly start the ERRCPY job.

The ERRCPY program automatically kills itself whenever logins are disabled. Therefore, if the system manager disables logins, he should restart ERRCPY.

The system manager gains information concerning the hardware errors detected and placed in the ERRLOG.FIL by running the ERRDIS system program as described in Section 6.1.3. If a system crash occurs, the system manager can retain error data by following the instructions in Section 6.2.

6.1.2 Use of the Error Crash Program - ERRCRS

The ERRCRS system program retrieves error information saved at the time of a system crash. When a system crash occurs, critical contents of memory are written to the system file CRASH.SYS if the user enabled the CRASH DUMP facility at start up time. The ERRCRS system program transfers certain error information from the file CRASH.SYS to a user designated file. The following sample dialog shows the use of ERRCRS.

```
RUN $ERRCRS
ERRCRS  V06A-01
OUTPUT FILE NAME? FILE.CRS
CRASH DUMP FILE NAME?

READY
```

ERRCRS is executed by typing the RUN $ERRCRS command from a terminal logged into the system under a privileged account. Two queries are printed. The response to the first query designates the name of a file to which error information will be written. The response to the second query is simply the RETURN key, designating the file CRASH.SYS stored under the system account [0,1]. The ERRCRS program writes the error information to the file named FILE.CRS (in this sample) and terminates automatically, as signalled by the READY message being printed.

The system manager can later print a report on the error information saved if he uses the ERRDIS system program as described in Section 6.1.3 and designates the filename specified as output of the ERRCRS

program run as the input filename for ERRDIS. It is highly recommended that users place the proper commands in the CRASH.CTL file so that ERRCRS runs automatically upon initialization of the system after a system crash.

6.1.3 Operation and Use of the Error Display Program - ERRDIS

The error display program ERRDIS allows the system manager to print a full or partial history or a full or partial summary of the error-related information preserved by the ERRCPY or ERRCRS system programs. ERRDIS prints, in an organized and formatted fashion, the error-related information read from a disk file according to options and switches specified by the system manager. The file is created by either the ERRCPY or ERRCRS system program and exists under the system library account [1,2] with protection code <60>. The disk file can maintain a history of a maximum of 880 errors and can record a maximum of 100 of any one type of error. If either of these limits is reached, ERRDIS prints in the output history a message telling how many errors were missed due to no room or to the limit of 100. The following two sections describe how to run and terminate ERRDIS and how to optimally use ERRDIS features.

6.1.3.1 Running and Terminating ERRDIS -

The system manager or privileged user runs the ERRDIS program by typing the following command while logged into the RSTS/E system.

```
RUN $ERRDIS
```

The program responds by printing a program header line, followed by three queries as shown below.

```
ERRDIS V06A-03
INPUT FILE NAME (<CR> FOR DEFAULT)?
OUTPUT TO?
OPTIONS?
```

The user types the RETURN key in response to the query concerning the input file name. The default input file name is $ERRLOG.FIL. The user can specify as input the name of the file created by the ERRCRS program. If the user types the RETURN key in response to the second query, the error-related information subsequently requested is printed at his terminal keyboard printer. To indicate a different output device, or file, type the proper specification followed by the RETURN key.

The OUTPUT TO query can be answered with the ? character. This response creates a file in the public structure under the current account. The name of the file is derived from the current date and time of day and the extension is LOG.

If the ? character is typed in response to the OUTPUT TO query, ERRDIS prints the name and extension of the resultant output file. For example,

```
OUTPUT TO?    ?
(OUTPUT FILE IS F04N59.LOG)
OPTIONS?
```

The file name is divided into 4 parts. The first part is a letter from A to L which denotes the month based on the letter's position in the alphabet. In the example, F is the sixth letter of the alphabet to denote the sixth month of the year, June. The second part of the file name is two digits from 01 to 31 to denote the day of the month. In the example, 04 is the fourth day fo the month. The third part of the name is a letter from A to X which denotes the hour from 00 to 23 (military time). Thus, the letter N is the example represents the 14th hour for 2:00 p.m. (civilian time). The last part is two numbers based on minutes.

Table 6-1
ERRDIS Options

Option Type	Option Format	Meaning
General	ALL	Error-related information for all errors is printed in the order in which they were detected and recorded, from the earliest to the most recent.
	EX	Terminate ERRDIS and exit to the Monitor.
	HE	Print the help file ERRDIS.HLP.
	MS	Missed errors.
Peripheral Errors	DH	Prints error-related information concerning the DH11 multiplexer.
	DT	TC11/TU56 DECtape.
	RF	RF11/RS11 fixed head disk.
	RC	RC11/RS64 fixed head disk.
	RK	RK11/RK05 or RK03 disk cartridge drive.
	RP	RP11-C/RP03 disk pack drive.
	RS	RH11/RS03 and RH11/RS04 fixed head disk.
	MA	RH11/TM02/TU16 magtape.
	RB	RH11/RP04 disk pack drive.
	RX	RX11/RS01 floppy disk drive.
	RJ	DP11 or DU11 interface for RSTS/2780 software.
	CD	Card reader.
	MT	TM11/TU10 Magtape.
	KB	Hung Teletype errors by job number and keyboard line number.

Table 6-1 (Cont.)
ERRDIS Options

Option Type	Option Format	Meaning
Processor Errors	T4	Traps through vector location 4.
	T0	Traps through location 000000.
	J0	JMP instructions executed to location 000000.
	RI	Reserved instruction traps.
	PF	Occurrences of power failure.
	CK	Checksum errors.
	MP	11/45 or 11/40 memory parity.
	MM	Memory management.
	??	Illegal code.

Table 6-2
ERRDIS Option Switches

Switch Format	Meaning
/T	Used with ALL option to print error data by type rather than by historical order of occurrence.
/S	Print only a summary of information of the error type indicated in the option. (Used alone, /S is meaningless.)
/K	Delete (kill) information in the error logging file. If the file is $ERRLOG.FIL, the /K option simply zeroes the file but keeps it in the directory. If the file is anything other than $ERRLOG.FIL, the /K option deletes the directory entry. In either case, ERRDIS terminates and returns control to READY.
/H	Used alone; causes a help file to be printed.
/dd-mmm-yy	Prints information concerning the error type indicated in the option if it was detected on or after the date designated by /dd-mmm-yy. For example, 19-MAY-73.
/hh:mm	Prints information concerning the error type indicated in the option if it was detected at or after the time of day designated by /hh:mm. For example, 8:50 or 20:50. If a date switch appears with a time switch, ERRDIS prints errors detected at or after the date and time of day. If a time switch appears without a date switch, ERRDIS uses the current system date.

After the user designates the output, ERRDIS prints the OPTIONS query. An option from those given in Table 6-1 can be typed. An option can be modified by any of several switches as described in Table 6-2. After output of the option or options specified is completed, the OPTIONS query is printed again. To terminate the ERRDIS program, type the EX command in response to the OPTIONS query.

```
OPTIONS?  EX
READY
```

Control is returned to BASIC-PLUS command level, as indicated by the READY message being printed.

6.1.3.2 Recommended Usage of ERRDIS - The recommended procedure for using the ERRDIS program is to daily request at least two specific options: ALL/S and ALL. The procedure entails running ERRDIS and answering the OPTIONS query in the following manner.

```
OPTIONS?  ALL/S
OPTIONS?  ALL
```

The ERRDIS program first creates a summary (/S) of all error-related information. The output starts with 4 lines of accounting data. On the first line, ERRDIS indicates the option requested, followed, on a second line, by the file name from which the information is taken (usually $ERRLOG.FIL). On the third line appears the output specification used and, on the fourth line, the time of day and current date. Following the accounting data is the summary of the total number of errors-recorded and errors missed by type and a tally of certain disk input and output information.

For the second option requested (ALL), ERRDIS prints the accounting data and the entire history of the errors logged. Information for each occurrence of a logged error is printed in chronological order, beginning with the earliest error and continuing to the most recent occurrence. The chronological order allows the user to find the occurrence of two catastrophic errors in the same minute. If this order is not important, the ALL/T options can be used to print all errors ordered by type.

For each error logged, a header line is printed which describes the type of error and the time of day and date of the occurrence. Following the header line for each error, ERRDIS prints such data as job number, keyboard number (if a hung Teletype error), processor status word (PSW) contents, and the contents (in octal) of the device registers at the time of the error. Consult the PDP-11 Peripherals Handbook for the meaning of the device register abbreviations and the types of errors encountered. (A job number of 0 indicates the null job.) A comment line is appended to some error-related information, such as that of a hung Teletype.

At the conclusion of the error history, ERRDIS prints the number of missed errors (if any) and the total number of errors listed of those logged since the beginning of the error history.

It is recommended that the user specify a disk file to contain the output of the options. The output can be gained by queuing the file on an 128 column line printer. The printouts of the complete summary and the complete history should be inspected and stored in a central location reserved for them. They provide the basis for planning preventive maintenance and the means to more readily isolate potentially dangerous hardware problems. The printouts should be available to the DIGITAL Field Service or Software Support representative. Periodically, the system manager can delete the contents of the file $ERRLOG.FIL by specifying the /K option in response to the ERRDIS program OPTION query.

The system manager should be alert for certain conditions reported by ERRDIS. Several hung Teletypes are not serious, but a steadily increasing number of hung Teletypes on a certain keyboard line indicates a possibly dangerous condition which should be remedied. Any occurrence of a T0 error is serious and indicates that an interrupting device has presented an incorrect vector location to the bus. An increasing number of disk errors (particularly on the system disk) indicates a need for immediate maintenance.

To obtain error logging printouts automatically, include the proper commands in the START.CTL file. The commands can run ERRDIS with output to a disk file and request the ALL/S and ALL/T options. The standard error logging file ERRLOG.FIL can be zeroed by the /K option. Also as part of the START.CTL file, commands can queue the ERRDIS output file for printing on line printer. The following sample commands show the process.

```
FORCE  KB1:  RUN $ERRDIS
FORCE  KB1:  $ERRLOG.FIL
FORCE  KB1:  ERRLOG.TMP
FORCE  KB1:  ALL/S
FORCE  KB1:  ALL/T
FORCE  KB1:  /K
FORCE  KB1:  QUE ERRLOG.TMP/D
FORCE  KB1:  BYE/F
```

The commands run ERRDIS, queue the output disk file so that the is deleted after printing, and log off the system. This technique ensures that error logging data is obtained without operator action and that error data is not lost due to the error logging file being filled. The printouts can be used to monitor errors and can be saved to provide a performance history.

6.2 ANALYZING SYSTEM CRASHES - ANALYS

When a system crash occurs in RSTS/E, time sharing operations are halted. If the required conditions described in Section 2.3 are met, the critical contents of memory are written into the CRASH.SYS file in the system account [0,1] and the system disk is bootstrapped in automatic restart mode.

The occurrence of a later system crash causes the CRASH.SYS file to be overwritten. Therefore, the system manager is provided with a means of retaining information in the CRASH.SYS file. This means is the ANALYS system program. The use of the ANALYS system program to document system crashes requires that the CRASH.SYS file be created at REFRESH time and that the crash dump feature be enabled at system start up time.

The ANALYS system program is invoked by the following command:

```
RUN $ANALYS
```

In response, a header line and two successive query lines are printed as follows:

```
'ANALYS' CRASH DUMP ANALYZER V06A-01

INPUT?
OUTPUT?   CRASH.DMP

READY
```

The first query line requests the name of the file to be analyzed, which by default is CRASH.SYS in account [0,1]. The user need only type the RETURN key, after which the second query line is printed. The second query line requests a disk file or a device designator for the output medium, which, for example purposes, is CRASH.DMP on the system disk. Normally, ANALYS takes about 5 minutes to run. Upon completion of the output, program execution is automatically terminated and READY is printed at the terminal.

The output of an ANALYS system program run supplies valuable hardware and software information which can be used by a software specialist to determine possible causes of system crashes.

ANALYS reports an error code in the crash dump data. Table 6-3 shows the error codes.

Table 6-3
System Crash Error Code

Error (1) Code (octal)	Meaning
-1(177777)	Unknown vector
-2(177776)	Jump to 0
41	Trap to 4
42	Trap to 10
43	Trap to 250 (Memory management violation)
44	Kernel SP Stack overflow
46	Trap to 114 (Parity memory error)
0 or other	Forced dump

The ANALYS output is complemented by the error logging printouts. The first and second items in the data labelled TOP 8. ITEMS ON KERNAL SP STACK in the ANALYS printout are the virtual Program Counter and the Processor Status Word. Use these values to compare with the data labelled VIRT PC and PSW in the ERRDIS report. If the values match, the error in the ERRDIS report is one which caused the crash.

The value of the PC minus 2 gives the location of the instruction being executed at the time of the crash. Refer to the load maps to find the module containing the instruction.

The PSW tells the current and previous modes of the processor. RSTS/E does not use supervisor mode. The only valid modes are kernel and user. If bit 11 of the PSW is 1, the processor is either a PDP-11/70 or PDP-11/45; if bit 11 is 0, the processor is a PDP-11/40.

(1)The power fail error has no code associated with it.

To obtain crash information printouts automatically, include the proper commands in the CRASH.CTL file. The commands can run ANALYS, to preserve the crash information, run ERRCRS to extract error information from the crash file, run ERRDIS to create a report and run QUE to obtain a listing of the error information. The following sample commands show the procedure.

```
FORCE  KB1:  RUN $ANALYS
FORCE  KB1:  [0,1]CRASH.SYS
FORCE  KB1:  ANALYS.TMP
FORCE  KB1:  RUN $ERRCRS
FORCE  KB1:  ERRCRS.TMP
FORCE  KB1:  [0,1]CRASH.SYS
FORCE  KB1:  RUN $ERRDIS
FORCE  KB1:  ERRCRS.TMP
FORCE  KB1:  ERRDIS.TMP
FORCE  KB1:  ALL/S
FORCE  KB1:  ALL/T
FORCE  KB1:  /K
@$START.CTL
FORCE  KB1:  QUE ANALYS.TMP/D,ERRDIS.TMP/D
FORCE  KB1:  BYE/F
BYE
```

The commands run the necessary programs to extract the crash data, start time sharing (which starts queuing and spooling operations), and queue the output file for printing. This technique ensures that crash data is obtained without operator action. The printouts can be used as a diagnasitc aid and saved to provide a performance history.

6.3 OCTAL DEBUGGING TOOL - ODT

The system program ODT opens a file, a peripheral device, or memory as an address space and allows a user to examine and change word or byte locations within the address space. As auxiliary operations, the user can list the contents of certain conventional table locations in the operating system.

The program immediately interprets and executes each character as the user types it. This action is termed ODT submode or ODT character mode and differs from the procedure used by other system programs which interpret input only after the user enters an entire line of characters. Since ODT performs processing based upon single characters typed at the terminal, its language is highly interpretive and interactive. It thus provides a quick and efficient means of finding errors in program and data files and changing data in those files for testing purposes. Because of the quickness and efficiency of ODT, it is advised that only experienced user's employ it to perform testing and error correction on a system's data base.

The program accesses and manipulates data in word and byte locations based on octal values. The word is the 16-bit PDP-11 word and can have a value between 0 (octal) and 177777 (octal), the limit imposed by 16 bits. A word has a high order (odd address) and low order (even address) byte. A byte can have a value between 0 (octal) and 377 (octal) - the limit that can be represented by 8 bits. For the purposes of clarity in this chapter, the following symbols express the octal values used by ODT.

Symbol	Meaning
n	Represents an octal integer between 0 and 17. The use of 8 or 9 generates an error.
k	Represents an octal value up to 6 digits in length. If more than 6 digits are specified or a value greater than 177777 (octal) is specified, ODT truncates the value to the low order (rightmost) 16 bits. If the octal value is preceded by a minus sign, ODT uses the 2's complement value of the number.

For example, ODT interprets the following values as shown.

```
     1          000001
    -1          177777 (2's complement)
   400          000400
-177730         000050 (2's complement)
1234567         034567 (truncated to low-order 16-bits)
```

The user can represent a location within the permissible address space by typing an octal value or an expression which reduces to an octal value. The following are the correct forms and the interpretation by ODT.

k	The 6 digit octal value of k.
n,k	The resultant address is the value of k added to the contents of the relocation register specified by n. Relocation registers are numbered from 0 to 17 (octal). See Section 6.3.4 for more information concerning relocation registers.

The special characters and symbols in Table 6-4 are recognized by RSTS/E ODT and explained in the remainder of the section.

Table 6-4
ODT Characters and Symbols

Character(s) or Symbols	Meaning
/ k/	Open the previously open location as a word or open the location designated by k as a word.
\ k\	Open the previously open location as a byte or open the location designated by k as a byte.
" k"	Give the ASCII representation of the currently open or last previously open location or of the location specified by k.
% k%	Give the ASCII representation of the Radix-50 value in the currently open or last previously open location or in the location specified by k.
RETURN key k followed by RETURN key	Close the currently open location or modify the contents of the currently open location with the value k and close it.
LINE FEED key k followed by LINE FEED key	Close the currently open location and open the next sequential location or modify the contents of the currently open location with the value of k before closing it and opening the next sequential location.
^ or ↑	Close the currently open location and open the preceding sequential location. (On some terminals, the ↑ or ^ character is typed by depressing the SHIFT and N keys simultaneously.)

(continued on next page)

Table 6-4 (Cont.)
ODT Characters and Symbols

Character(s) or Symbols	Meaning
↑ or __	Take the contents of the currently open location as a PC relative offset and calculate the next location to be opened; close the currently open location and open the location thus evaluated.
@ k@	Take the contents of the currently open location as an absolute address, close the currently open location, and open and print the contents of the location thus evaluated. If @ is preceded by k, the value k replaces the contents of the currently open location before it is closed.
> k>	Take the low order byte of the currently open location as a relative branch offset and calculate the address of the next location to be opened; close the currently open location and open and print the contents of the relative branch location thus evaluated. If > is preceded by k, the value k replaces the contents of the currently open location before it is closed.
<	Close the currently open location and open the last location explicitly open. Returns ODT to the origin of a sequence of relative locations determined by _, @, and > character operations.

(continued on next page)

Table 6-4 (Cont.)
ODT Characters and Symbols

Character(s) or Symbols	Meaning
'	Separates a relocation register number from an octal value. ODT adds the contents of the specified relocation register to the octal value following the comma and forms a relocatable address.
;	Separates multiple values in a list request using the L character and in a register operation using the R character.
.	Specifies the last explicitly open location similar to that used by the < character operation.
+ space bar	Add the preceding value and following value and use the result.
-	Subtract the following value from the preceding value and use the result.
R	Reset all relocation registers to -1 (177777).
nR	Reset relocation register n to -1 (177777).
k;nR	Set relocation register n to the value k.

(continued on next page)

Table 6-4 (Cont.)
ODT Characters and Symbols

Character(s) or Symbols	Meaning
F	Set relocation calculation for list requests using L character.
1F	Disable relocation calculation set by F character.
C	Print out Monitor table symbolic names and memory addresses.
$S	Print out the processor status word.
Q	Use the last quantity printed by ODT.
k1;k2L	Print contents of locations k1 through k2 at the terminal.
1;k1;k2L	Print contents of location k1 through k2 on line printer unit (0).
2;k1;k2L	Print contents of location k1 through k2 on another device. ODT prints DEVICE question, to which user types the device designator.

6.3.1 Running And Terminating ODT

The user runs ODT by typing the following command.

```
RUN $ODT
ODT V06A-02
FILE?
```

ODT runs and prints the question FILE. The user response to this query determines how ODT runs and what address space ODT accesses. The possible responses are listed and described in Table 6-5.

Table 6-5
ODT FILE Question Responses

Response	Meaning
Type the RETURN key only.	Allows read access to memory only if user is privileged.
Type the ALT MODE key only.	Allows read access to the file CRASH.SYS in account [0,1].
Type the LINE FEED key only.	Same as the RETURN key.
Type a file specification followed by the RETURN key or the ALTMODE key.	Allows read access to the file on the device specified. If no device is specified, the system disk is used.
Type the file specification followed by the LINE FEED key.	Allows read and write access to the file specified.

ODT determines the address space by the response to the FILE question and indicates its readiness to accept commands by printing the * character. For example,

```
FILE? ABC.DAT        (Terminate with LF)
*
```

ODT opens for read and write access the file ABC.DAT on the system disk under the current account. To terminate ODT, type the CTRL/Z combination in response to the * character. For example,

```
* ^Z
READY
```

ODT closes any file currently open and returns control to BASIC-PLUS command level.

6.3.2 Opening and Closing Locations in the Address Space (/ And \ )

ODT access the address space as either a word or a byte. The user indicates the type of access by specifying the slant (/) or reverse slant (\) character. For example,

```
*1000/
```

The user types the address 1000 (octal) followed by the / character. ODT opens the location as a word, generates a space, prints the

6-digit octal contents of the word, generates another space, and leaves the location open for change. The following demonstrates the results.

```
*1000/ 004100
```

ODT prints the contents of location 1000 as 004100. To close the location, type the RETURN key. ODT closes that location prints an * character on the next line, and does not open a new location.

The user specifies the \ character to open a location as a byte. For example,

```
*1000\ 000
*
```

ODT opens location 1000 as a byte, generates a space, prints the 3-digit octal contents of the byte, generates another space, and leaves the location open for change. To close the location, type the RETURN key. ODT closes the location, prints an asterisk character on the next line, and does not open a new location.

To change location 1000, type a new 6-digit octal value followed by the RETURN key. For example, if location 1000 is open as a word,

```
*1000/  004100  004000      (Type RETURN key)
*
```

ODT replaces the current contents 004100 with the specified contents 004000, closes the location, and prints the * character on the next line.

To determine the contents of the current word location, type the / character. For example, if the current location is 1000, the following occurs.

```
*/ 004000
```

ODT opens the current location, generates a space, prints the 6-digit contents and generates another space.

If the user types the LINE FEED key while a word location is open, ODT closes the current location and opens the next sequential location. For example, if location 1000 is open as follows,

```
*1000/ 004000
```

and the user types the LINE FEED key, the following occurs

```
*1000/ 004000
001002/ 012345
```

ODT generates a carriage return and line feed and prints the address of the next location, followed by the / and space characters and the contents of the word. The new location 1002 is open. Repetitive use of the LINE FEED causes ODT to open and display the contents of sequential locations.

If the user types the LINE FEED key while a byte location is open, ODT performs the same actions as described for a word location except that the next location is treated as a byte.

6.3.2.1 Opening the Preceding Location (↑ or ^) - Typing the up arrow (↑) character or the circumflex (^) character when a location is open causes ODT to close the currently open location and to open the immediately preceding location.

NOTE

On ASR-33 type terminals, type the ^ character by depressing the SHIFT key and the N key simultaneously.

For example, if two sequential locations are successively opened, typing the ^ character opens the immediately preceding location.

```
*1000/ 002345        (Type LF key)
001002/ 012740^
001000/ 002345
```

Typing the ^ character closes location 1002 and opens location 1000 and prints its contents. If a byte location is currently open, typing the ^ character opens the immediately preceding byte location. If the user types a value followed by the ^ character, ODT replaces the current contents with the specified value before closing the location.

If the user types the ^ character and a location is not currently open, ODT opens and prints the contents of the last currently open word or byte location. For example,

```
*1000/  002345      (Type RETURN key)
*^
001000/ 002345
```

ODT prints the address and the contents of the word location on the next line.

6.3.2.2 Opening A PC Relative Location (← or _) - Typing the backarrow (←) character or underline (_) character when a location is currently open causes ODT to add 2 to the address of the current location, to add the resultant sum to the contents of the current location, and to open the location specified by the final sum. For example,

```
*1000/ 000040_
001042/ 012345
```

ODT closes location 1000, adds 2 to the address (1000), and adds the resultant 1002 to the contents (40) of the current location. As a result, ODT opens location 1042 and prints its contents. This method of calculating the next location to open is similar to that used in relative addressing by the program counter in the PDP-11 computer. Such a method of address calculation is for position independent code.

If the contents of the current location is an odd value, ODT opens and prints the contents of the low-order byte of the PC relative location. If the user types a value followed by the _ character, ODT modifies the current contents and uses the new value to calculate the PC relative address.

6.3.2.3 Opening an Absolute Location (@) - Typing the commercial at (@) character when a location is currently open causes ODT to take the contents of the current location as the address of the next location to open. As a result, ODT closes the current location, opens the calculated location and prints its contents. For example,

```
*1006/ 001024  @
001024/ 000500
```

ODT uses the contents of the current location (1024) as the next location to open.

NOTE

On ASR-33 type terminals, type the @ character by depressing the SHIFT key and the P key simultaneously.

If the user types a value followed by the @ character, ODT changes the contents of the current location to the value and uses the new value to determine the next location to open. The method is equivalent to absolute addressing on the PDP-11 computer where the contents of the location following an instruction are taken as the address of the operand. The address is absolute since it remains constant regardless of where in memory the assembled instruction is executed.

6.3.2.4 Opening a Relative Branch Offset Location (>) - Typing the greater than (>) character when a location is currently open causes ODT to use the signed value of the low-order byte of the current location to determine an offset from the current location. ODT uses the final sum to open the next location and print its contents. For example,

```
*1032/ 000407>
001052/ 001456
```

ODT takes the contents of the low order byte (007) and multiplies by 2 to give 16 (octal). Next, ODT adds 2 to the address of the current location (1032) to give 1034. Finally, ODT adds these two quantities (1034 + 16) to give the address of the word (1052) to open. ODT closes the currently open location, opens the calculated location, and prints the address and the contents on the next line.

If the user specifies a value followed by the > character, ODT modifies the contents of the currently open location and uses the low order byte of the new value to calculate the relative branch offset location. For example,

```
*1032/   000407  301>
000636/  000010
```

ODT interprets the byte value 301 as a negative value since the high order bit is 1. The absolute value of 301 is 77 (octal) which is multiplied by 2 to give 176 (octal). ODT subtracts the relative branch offset (176) from the address plus 2 of the current location to give 636 as the address of the next location to open. ODT opens the new location and prints its contents.

6.3.2.5 Returning To An Interrupted Sequence (<) - Typing the less than (<) character causes ODT to close the currently open location and open the last explicitly open location. This command is useful, when a user has typed the _, @, and > characters, or any sequence thereof, and wishes to open the locations from which ODT calculated subsequent relative locations. For example,

```
*1032/   000301>
000636/  000010 @
000010/  123456<
001032/  000301
```

After typing the > and @ characters, ODT opens location 10. The user returns to the last explicitly open location by typing the < character. ODT opens and prints the contents of location 1032, the last location explicitly opened.

6.3.3 Printing the Contents of Locations

The user can type the L character in three ways to print the contents of locations in the address space open by ODT. For example, to print the contents of a certain range of addresses, specify the start and end addresses with the L command as follows.

```
*0;776L
```

ODT prints at the terminal, the octal contents of each word between address 000000 and 000776. Beginning each line of the printout, ODT prints the address of the first word on the line. The user can turn the printing on and off by typing the CTRL/O combination.

To print a listing on line printer unit 0, type the L command as follows.

```
*1;0;776L
```

ODT prints, on line printer unit 0, the octal contents of each word between addresses 000000 and 000776. To specify another unit, type 2 preceding the command.

```
*2;0;776L
DEVICE? LP1:
```

ODT prints the DEVICE question, to which the user types the device designator of the line printer unit to be used.

6.3.3.1 Printing ASCII Format (") - Typing the quotation marks (") character when a location is currently open causes ODT to print the ASCII representation of the word or byte. For example,

```
*1000    101 " A (Type LF key)
001001   103 " C (Type CR key)
*1000/ 41501
```

If the currently open location is open as a word, typing the " character causes ODT to print the 2-character representation of the word. For example,

```
*1032/ 034567   W9
```

The low order byte contains 167 (octal) which is W and the high order byte contains 071 (octal) which is 9.

If the user types the " character and a location is not currently open, ODT prints the ASCII representation of the previously open location.

6.3.3.2 Printing Radix-50 Format (%) - Typing the percent (%) character when a location is currently open causes ODT to print the 3-character ASCII representation of the Radix-50 word. For example,

```
*1000/ 034567 % IG1
```

If the user types a value preceding the % character, ODT interprets it as the address whose contents are to be interpreted and printed. For example,

```
*1000 % IG1
```

ODT interprets 1000 as the address to use. If the user types the % character and a location is not currently open, ODT prints the 3-character ASCII representation of the previously open location.

6.3.4 Relocation Registers

ODT has available 16 relocation registers which the user can employ to specify relative addresses. ODT initially sets the relocation registers to -1 (177777, the highest possible address) to prevent inadvertent errors in address calculation. The user sets a relocation register by typing the relative address, followed by a semicolon and the specification of one of the eight relocation registers. For example, to set relocation register 0 to 1000, type the following,

```
*1000:0R
*
```

The user can subsequently use the value in relocation register 0 as an offset or a base address in specifying a location. For example, to open location 1032, as a word, the user types the following.

```
*0,32/ 000010
```

ODT adds the offset 32 to the contents of relocation register 0 to open the location. Since relocation register 0 contains 1000, ODT opens location 1032.

To reset the contents of all relocation registers to -1, the user need type only the R character. For example,

```
*R
*
```

ODT generates the carriage return and line feed operation and prints the * character again. To reset the contents of any one register to -1, simply specify the register number followed by the R character.

For example,

```
*1R
*
```

The above command resets relocation register 1 to 177777 (octal).

ODT treats registers 0 through 7 differently from 10 (octal) through 17 (octal) when used with disk files. For registers numbered 0 through 7, the leftmost 3 digits specify a block number and the rightmost 3 digits specify a byte location within the block. For example,

```
000017
```

The value designates byte 17 (octal) in block 0 of the file. The following value:

```
3412
```

designates byte 412 (octal) in block 3 of the file. (These registers are helpful in accessing data which is partitioned in 512-byte blocks.) For registers numbered 10 (octal) through 17 (octal), the value specifies an absolute block number in the disk file. For example, the value 1000 (octal) specifies block 1000 (octal) of the disk file.

To print the contents of locations based on a fixed offset from the relocation registers, type the F character and subsequently use the L character. For example,

```
*F
*1;0;3000L
*
```

The F character conditions ODT to calculate relocated addresses for a printout. The next command tells ODT to add the offsets from 0 to 3000 to the value of relocation register 0 and print the contents of those resultant locations on line printer unit 0. The procedure is repeated for the values of relocation registers 1 through 7. The resultant printout contains a listing of the contents of addresses n,0 through n,3000 where n is between 0 and 7 (the relocation registers).

To turn off calculation of relocation addresses for a printout, type 1F. For example,

```
*1F
*
```

Subsequent listing requests generate printout for actual addresses rather than relocated addresses.

6.3.5 Interpretive Address Quantities (Q And .)

ODT uses the variable Q to store the last value which it printed at the terminal. The user can type Q to designate the value so stored.

ODT performs any valid operation requested and automatically extracts the value from Q. For example, if the user desired to increase the

value in an open location by a certain increment, he could proceed as shown in the following sample.

```
*1342/  173214 Q+10    (Type RETURN key)
*/ 173224
```

The user types 1342/ to open that location as a word. ODT prints the contents of the location and stores that value in Q. The user subsequently types Q+10 followed by the RETURN key. ODT adds 10 to the value in Q and modifies the current location with the sum. The user thus does not have to retype the number or calculate the sum. To verify that ODT has changed the location properly, the user types the / character. ODT opens the last previously open location (1342), prints the contents, and, additionally, updates the value Q with the most recently printed quantity.

The period (.) character indicates the currently open or last explicitly open location and can be typed to indicate the current address for ODT operations. This is the same address used by the < character described in Section 6.3.2.5. In most cases, the . character value is the address used by ODT when the user types the /, \, ", %, and LINE FEED characters. For example, to open as a word a location 16 bytes from the last explicitly open location, the user types the following,

```
*.+16/  012345
```

ODT adds 16 (octal) to the address given by the . character, opens the resultant address, and prints its contents.

6.3.6 Error Procedures

If the user types an invalid or unrecognized character, ODT prints a question mark (?) character, generates a carriage return and a line feed, and prints the * character. For example,

```
*1008?
*
```

ODT indicates that the character 8 is an error. The user must retype the number correctly.

If ODT encounters an error while performing output to a device, it prints the message I/O ERROR? followed by the * character. The user must correct the device error and type the command again.

CHAPTER 7

SYSTEM UTILITY OPERATIONS

7.1 GENERAL UTILITY OPERATIONS - UTILTY

While running the UTILTY system program, the system manager has on-line system control and can perform such operations as:

1. Enable and disable logins.
2. Broadcast messages and force strings to any or all keyboards on the system.
3. Terminate execution of a job (kill) or cause a remote line to hang up.
4. Set and reset system date and time.

The system manager also can perform various disk management operations such as:

1. Cause private disk packs to be used or to be prohibited from use on specified disk drive units (mount and dismount operations).
2. Prohibit or allow creation and accessing of files on specified disk drive units (lock and unlock operations).
3. Rebuild the storage allocation table on a corrupted disk (clean operation).
4. Change the number of blocks of disk storage an account can retain at LOGOUT time (quota) and change an account password.
5. Remove all files from a user's account on a specified disk drive unit (zero operation).

The commands of the UTILTY system program are presented in Table 7-1 for reference. The following sections explain the commands according to their functional uses: program control, on-line system control, disk management, and run time system control.

Table 7-1
UTILTY Commands

Category	Command and Format(1)	Use
Operational Control	LOGINS	Enables users to login to the RSTS/E system.
	SET␣LOGINS␣x	Set to x the number of user jobs which can be logged into the system at any one time.
	NO␣LOGINS	Prevents further login attempts.
	SEND␣{KBn: / ALL}␣xxx	Causes the text string xxx to be printed on the keyboard unit n or all keyboards.
	FORCE␣{KB:n / ALL}␣xxx	Causes the text string xxxx to be forced into the input buffers of keyboard unit n or all keyboards as if it had been typed in. If the ^ character is the first character of the string, it is replaced by a ^C.
	KILL␣n	Immediately terminates user job specified by n.
	HANGUP␣KBn:	Disconnects the remote line specified by KBn:.
	DISABLE␣KBn:	Disable the terminal interface of keyboard unit n until the start of the next time sharing session.
	DATE␣dd-mon-yy	Sets the RSTS/E system date to the value of day, month and year (for example, 13-NOV-72).
	TIME␣hh:mm	Sets the RSTS/E 24 hour clock to the value of hours and minutes (for example, 21:52 means 9:52 p.m.).

(continued on next page)

(1)The notation ␣ indicates that a space character is required.

Table 7-1 (Cont.)
UTILTY Commands

Category	Command and Format(1)	Use
Disk Management	MOUNT␣dev:id	Logically associates the disk residing on the specified drive unit with the pack identification label (id) so that data on the disk can be properly accessed by the system. For example, MOUNT DK1:PRIV1 associates the pack PRIV1 with the physical device designator DK1:.
	DISMOUNT␣dev:	Disassociates a disk pack from its physical drive specified by dev:. Must be used prior to removing the cartridge from the disk drive unit.
	LOCK␣dev:	Places the disk mounted on drive unit dev: in a state which prevents files from being OPENed by non-privileged users.
	UNLOCK␣dev:	Allows non-privileged users to OPEN files on the disk mounted on drive unit dev:.
	CLEAN␣dev:	Rebuilds the SAT (Storage Allocation Table) of the pack mounted and locked on disk drive unit dev:. To be used only when message DEVICE NEEDS CLEANING is printed and the device is locked.

(continued on next page)

(1)The notation ␣ indicates that a space character is required.

Table 7-1 (Cont.)
UTILTY Commands

Category	Command and Format(1)	Use
Disk Management (Cont.)	QUOTA␣[n,m]␣q	Sets the quantity of 256-word blocks the user account [n,m] is allowed to retain at logout time to the decimal number q. A value for q of zero means unlimited quota.
	CHANGE␣[n,m]password	Alters the password or user account [n,m] to the 6-character alphanumeric password.
	ZERO␣dev:␣[n,m]	Deletes all files from user account [n,m] on the disk drive unit specified by dev:.
Run Time System Control	ADD␣name	Add to the run time system table the run time system specified by name.
	REMOVE␣name	Remove from the run time system table the entry for the module specified by name.
	LOAD␣name/ADDR:xxx	Load into memory the run time system specified by name. The /ADDR:xxx option (where xxx is a 1K section of memory) appended to name establishes the default starting location.
	UNLOAD␣name	Unload from memory the run time system specified by name.
	NAME␣name=file	Associate name with the run time system module specified by file. The standard name is defined when the system is built. The designation file is a standard RSTS/E specification. The equals character must separate name and file.

(continued on next page)

(1)The notation ␣ indicates that a space character is required.

Table 7-1 (Cont.)
UTILTY Commands

Category	Command and Format(1)	Use
Program Control	HELP	Prints a list of valid UTILTY commands at the keyboard printer.
	CTRL/C	Terminates execution of the current operation and the UTILTY run.
	CTRL/Z	Allows completion of pending operations before termination of the UTILTY run.
	EXIT	Allows completion of pending operations before termination of the UTILTY run.

(1)The notation ␣ indicates that a space character is required.

7.1.1 Running and Terminating UTILTY

The system manager or a privileged user executes the UTILTY system program by typing the following command while logged in at any terminal.

```
RUN $UTILTY
```

The program responds by printing one header and one query line as follows:

```
SYSTEM UTILITY PROGRAM 'UTILTY V06A-03'
?
```

after which any valid command can be specified. A list of all valid commands is printed if the HELP command is typed. The query line need not be present in order to type in any subsequent commands. The program prints the ? character after each command is executed.

Termination of the UTILTY system program can be properly accomplished by typing either the CTRL/Z combination or the EXIT command. (CTRL/Z is echoed by ^Z being printed on the terminal.) If CTRL/Z or EXIT is typed at any time, the operations currently pending are properly completed, and control is returned to the BASIC-PLUS editor. The completion of the UTILTY run is signalled by READY being printed at the keyboard.

If the CTRL/C combination is typed in order to terminate the program run, any operation in progress is interrupted immediately. Control is returned to the BASIC-PLUS editor. (CTRL/C is echoed at the keyboard printer by the ^C being printed). Typing CTRL/C to terminate a UTILTY program run is not considered proper termination since the effect of uncompleted internal system operations is unpredictable. The system manager is advised to use CTRL/Z or the EXIT command to terminate UTILTY.

7.1.2 Principles of Disk Management

Certain commands of the UTILTY system program are used to manage disks on the RSTS/E system. Such commands are presented for reference in Table 7-1 under the category of disk management. To more easily convey the proper use of individual disk management commands, the following description of each command is presented with a discussion of its function and value.

7.1.2.1 Preparing a Disk for Use on a Drive - A disk pack or cartridge, to be used on the RSTS/E system, must be made known to the system. The disk is made known to the system by associating its logical name, the pack label (or identification), with the physical device unit number on which the disk resides. Such a process is called logically mounting a disk.

Logically mounting a disk asssociates the specific device designator and the exact pack label of the disk. A new disk must first be formatted and initialized for use on the RSTS/E system by the initialization option DSKINT. An already formatted disk can be initialized by the DSKINT system program. The pack label, a

6-character alphanumeric name assigned to the disk during the DSKINT dialogue, is required to logically mount the disk.

When the RSTS/E system is started, only the system disk is logically mounted. The system manager must ensure that other packs in the public structure (and perhaps the private structure) are mounted by commands from the START.CTL or CRASH.CTL file. The same principle applies to private disks added to the system after system start up. The system manager can logically mount a private disk on the system by means of the UTILTY system program MOUNT command or can use the CCL MOUNT command described in Section 4.14 of the RSTS-11 System User's Guide. The procedure to mount a disk is summarized in Table 7-2.

Table 7-2
Procedures for Using Disk Packs and Cartridges

Enter a Pack or Cartridge to the System(1)	Remove a Pack or Cartridge from the System
1. Use the SYSTAT system program to ensure that the drive unit is free.	1. Invoke UTILTY and use LOCK command on the drive unit containing the pack to be removed.
2. Place the pack in the drive. When the drive is READY, write enable it.	2. Use SYSTAT disk status report to determine the number of OPEN files. If zero, proceed. If non-zero, wait until all files are closed before proceeding.
3. Invoke the UTILTY system program and use MOUNT command to notify system of new pack.	3. With no files open on the device unit, use the DISMOUNT command to notify system that the pack is being removed.
4. Use CLEAN command if necessary.	4. Remove pack from disk drive unit.
5. Use UNLOCK command to free device for use.	

(1)The disk pack is assumed to have been initialized and formatted using the DSKINT initialization option or the DSKINT system program (Section 7.5). See Chapter 3 of the RSTS/E System Generation Manual for a description of DSKINT initialization option.

After a public system disk or a file structured private disk is mounted on the system, it must remain WRITE ENABLED. The system must be able to update access and job accounting information during time sharing. If the disk is WRITE PROTECTED, accessing a file on it is impossible. A WRITE PROTECTED system disk causes the system to crash. Any other WRITE PROTECTED disk causes the DEVICE HUNG OR WRITE LOCKED error when an attempt is made to write to it.

Under no circumstances should a public disk be removed from the system during normal system operation. Files are created and accessed in the public structure without explicit reference to a device. By removing a public disk during time sharing, the system manager denies users access to files which reside on that device.

To properly remove a private disk pack or cartridge from a drive unit, actions similar to those of preparing the disk for use must be employed. For example, if the system manager desires to replace a private disk with another private disk, he must follow a careful procedure. First, he must ensure that no files are open on the drive unit. The system manager can do this by requesting a disk status report through the SYSTAT system program. His next action is to lock the device unit by use of the LOCK command. This action ensures that non-privileged user programs cannot open any more files on the disk.

When the disk to be removed has no OPEN files and has been LOCKed, the system manager next must disassociate that disk pack from the drive unit. He does so by logically dismounting the device with the DISMOUNT command. If any files are open on the disk, UTILTY prints the ACCOUNT OR DEVICE IN USE error message and does not dismount the disk. After the dismounting action is completed, the disk pack can safely be removed from the drive unit. Any pack which is to replace the pack removed must undergo the procedures previously described for proper use of the pack.

If a disk was not dismounted properly after it was last used, UTILTY encounters the DISK PACK NEEDS 'CLEANING' error (ERR = 25) when the MOUNT command is executed. UTILTY prints the warning message DEVICE NEEDS CLEANING. The disk is mounted but locked. SYSTAT prints the text LOCKED in the disk structure report. The lock condition prevents nonprivileged jobs from accessing the disk but allows the disk to be cleaned.

An improperly dismounted disk requires cleaning because the Storage Allocation Table (SAT) in the SATT.SYS file does not reflect the actual allocation of storage. This condition results because the SAT is manipulated in memory and is not written back to the disk immediately. Following the proper dismounting procedures ensures that the SAT is updated on disk before the disk is physically removed from the drive.

Because the MOUNT command of UTILTY does not clean disks, the CLEAN command must be used to rebuild the SAT on the disk. The CLEAN routines read all the directory blocks and create a new SAT truly reflecting occupied storage. Additionally, all file access counts are reset to 0 and all files with TMP extensions are deleted. If the CLEAN command is not executed and the disk is dismounted, the error condition no longer applies but the SAT is still unreliable and requires cleaning. (The DISMOUNT command assumes the proper procedures were followed and clears the bit which indicates that cleaning is required.)

The MOUNT command of UTILTY does not unlock the disk. To allow nonprivileged jobs access to the disk, the UNLOCK command must be executed. Nonprivileged jobs can open files on the disk after it is unlocked.

7.1.2.2 Removing Files from an Account - Before an account can be deleted from the RSTS/E system or deleted from a private disk, the account must contain no files. The ZERO command of UTILTY removes all files from an account on a device. The REACT program (see Section 4.1.2) is then used to delete the account.

7.1.2.3 Changing Quota and/or Password of an Account - Each user account in the RSTS/E system has associated with it a quota of disk storage that the account can retain at logout time and a password which allows access to the system. The quota and password are specified by the system manager when the account is created.

The system manager can change the quota by use of the QUOTA command of UTILTY. The system manager specifies, in the QUOTA command, the account number and the decimal number of 256-word blocks of disk storage the account can retain at logout time. If he specifies zero for the quota, the account can retain an unrestricted number of blocks.

The system manager can change the password of an account by using the CHANGE command.

7.1.3 Operational Control of the System

Certain commands of the UTILTY system program control the operation of the system. Such commands are listed for reference in Table 7-1 under the category of operational control. These commands and examples of their possible usage are described in this section.

When the system manager is on-line in the RSTS/E system, he can monitor and control system operation. By use of the SYSTAT system program, he observes the number of free small buffers. If, for example, the number of free small buffers drops below 10, system efficiency declines. He can remedy the possible degradation of system efficiency by preventing more users from logging into the system. The system manager prevents further logins by using the NO LOGINS command. The system manager also can use the NO LOGINS command in preparing to shut down time sharing operations of the RSTS/E System.(1)

The system manager can communicate with a user at his terminal or with all users by the SEND command. The SEND command causes a specified text string to be placed in the output buffer of a terminal or all terminals and, as a result, be printed on the terminal. If a user assigns a peripheral device for an inordinately long time, for

---

(1)Use of the NO LOGINS command causes ERRCPY to terminate. If error logging is desired, ERRCPY should be run again.

example, the system manager can transmit a message requesting the user to deassign the device. By specifying ALL in place of the device designator of a single keyboard, the system manager can transmit the message to each on-line terminal in the RSTS/E system.

If it becomes necessary, during the course of system operations, to handle troublesome users, the system manager has two capabilities. He can cause a user's terminal (or all users' terminals) to execute a text string by the FORCE command. He can also terminate a user's job by the KILL command. The FORCE command places a text string in the input buffer of a specified terminal as if it had been typed by the user. If the first character of the text is the up-arrow character (↑), it is replaced by a CTRL/C (↑C). The following sequence of two FORCE commands causes a user's terminal to execute two commands which log out the job.

```
? FORCE KB4: ^BYE
? FORCE KB4: YES
```

THE KILL command terminates the user's job. The user is immediately logged off the system.

If the system manager determines that a dataset line is in use but no keyboard activity is taking place (by SYSTAT job status report), he can disconnect the dataset. The HANGUP command causes the remote line specified by KBn: to be disconnected. The hangup capability prevents a user from monopolizing the line without being charged for connect time and frees the line for other remote users.

The DATE and TIME commands allow the system manager to set the system date and the value of the 24-hour clock, respectively. The SET LOGINS command allows the system manager to control the number of users on the system. The DISABLE KBn: command can remove a terminal interface from use for the current time sharing session. To disable the interface for subsequent sessions, the SETKEY option of the initialization code can be used. (SETKEY is described in the RSTS/E System Generation Manual.)

7.1.4 Run Time System Control

The run time system control commands described in Table 7-1 allow the system manager to conduct time sharing operations with an auxiliary run time system supplied by DIGITAL. The NAME command provides the association required for a specially tailored module on the system disk to be accessed by a specified runtime system name. This association is normally made during the system library build procedures and is not required again unless the system is regenerated.

For a run time system to function properly during time sharing operations, certain actions must occur to initialize the system. The actions are normally effected by commands in the START.CTL and CRASH.CTL files and are altered only if DEFAULT conditions are altered. The following example shows which control file commands are necessary and sufficient to initialize RSTS/E to use the auxiliary run time system.

```
FORCE KB1: RUN $UTILTY
FORCE KB1: ADD RTSLIB
FORCE KB1: LOAD RTSLIB/ADDR:xx
FORCE KB1: UNLOAD RTSLIB
FORCE KB1: EXIT
```

See Section 3.1 for more information concerning control files. If these commands are not executed at the start of time sharing, an attempt to run a dependent program or object time system generates the NO RUN TIME SYSTEM error.

The ADD command in the control file adds the appropriate name to the run time system table in memory. This action is necessary since the initialization code establishes a new table each time system start up occurs. The command ensures that the appropriate name and parameters are entered in the table.

The LOAD command with the /ADDR option in the control file loads the run time system into memory at the 1K section specified. Because 1K section numbering begins at 0 and ends at n-1 (where n is the total size of memory), the 1K section number in the /ADDR:xx is one less than the physical section number. For example, to load the run time system RTSLIB in the 61st through 64th 1K sections of memory, specify /ADDR:60. The run time system is loaded from low memory to high memory at its defined initialized size. To be loaded without error, enough contiguous user space must be available starting at that location. The location specified in the /ADDR option becomes the default location at which the run time system is loaded during the current time sharing session. The location need be changed only if the system manager changes the allocation of the section of memory with either the DEFAULT or the START initialization option.

One precaution is necessary when specifying the address at which the run time system is loaded. The section of memory chosen must not fragment the user job space to prevent the run time system from executing a job. For example, assume a system has 24K words of user space available between the 36K and 60K sections of memory. Assume also that a job requires 18K words of user space to run and that the run time system requires 4K words when resident. If the loading address is 36K, the space between 40K and 60K remains available for an 18K job to run. If the loading address is 42K, the user space is fragmented into two sections - one from 36K to 42K and one from 46K to 60K. Both sections are too small to allow the 18K program to run.

The UNLOAD command in the control file frees that portion of memory occupied by the run time system. The memory becomes available as user job space and is not occupied by the run time system until a user executes the appropriate CCL command during time sharing.

To prevent use of an auxiliary run time system for the current time sharing session, the system manager can execute the REMOVE command. This purges the entry from the run time system table. The SHUTUP system program automatically performs the REMOVE action when time sharing operations are terminated.

7.2 MONITORING SYSTEM STATUS - SYSTAT

During normal time sharing operations, the system manager should monitor the status of the RSTS/E system. He gains information concerning system status either by use of the system program SYSTAT or, on systems with sufficient memory storage, by running the VT5DPY or the VT50PY program. The options and output supplied by the SYSTAT system program are described in Chapter 4 of the RSTS-11 System User's Guide. The VT5DPY and VT50PY program descriptions appear in Section 7.3. The discussion here gives the system manager guidelines on how and when to use system status information.

Several uses of SYSTAT are described elsewhere in this manual in conjunction with other system manager operations. Those instances are listed here.

1. During preparation for system shut down, to determine active jobs and disk devices and assignable devices in use.

2. In conjunction with the UTILTY system program command HANGUP, for determining misuse of a remote line.

3. In conjunction with the UTILTY commands NO LOGINS and SET LOGINS, when the number of free small buffers is less than 10.

Refer to the discussion relevant to the individual system program or system operation for more information on the above uses of SYSTAT.

Further uses of SYSTAT are listed below and discussed in the ensuing paragraphs.

4. Uncovering malfunctioning keyboards by the HUNG TTY count.

5. Guarding against a disk device filling up by watching the FREE block count.

6. Following the progress of user jobs or detached jobs by the STATE and RUN-TIME items of job status.

Item (4) refers to the HUNG TTY count reported in the SYSTAT buffer status report. A HUNG TTY count of zero is good. A HUNG TTY count of non-zero indicates the presence of a malfunctioning terminal or terminals. The ERRDIS program can identify the device or devices causing the error count. If the HUNG TTY count increases rapidly, a field service representative should be consulted.

The FREE block count mentioned in item (5) reflects the apparent number of free blocks on each disk and is given in the disk status report of SYSTAT. For practical purposes, however, such as for allocation a file on the device, all of the free blocks reported by SYSTAT may not be usable. A NO ROOM FOR USER ON DEVICE message may be generated although SYSTAT reports that enough FREE blocks exist. The file cluster size or the number of clusters required can prevent a file from fitting on the device desired. For example, a file whose cluster size is 16 and whose length is 10 blocks possibly does not fit on a device which SYSTAT reports to have 50 free blocks of file space remaining. The cluster size of 16 demands that 16 contiguous blocks of free space must exist on the device before the file can be allocated to the device. In some cases, 16 contiguous blocks simply

do not exist on a device. (It must be pointed out that RSTS/E does not allow a file to extend to another physical device.)

A further condition exists for showing NO ROOM FOR USER on a device. The UFD is perhaps full and cannot accommodate the creation of another file. The UFD cluster size was not made large enough when the account was created with REACT.

The occurrence of jobs being stalled in a resource sharing system is detectable by the means presented in item (6). If the system manager notices that a RUN-TIME value of a job is not increasing (the value is printed out in a job status report), it indicates that the job is stalled, waiting for an I/O device. One user job in the system can ASSIGN a device or keep an assignable device locked by having one file open on it. The system manager can determine the selfish user job by examining the device status report which associates the busy device with the job number of the user controlling that device. The system manager can request that the user free the device or, if that is not viable, can use UTILTY commands to force the job off the system.

The status of detached jobs is of interest also. If a detached job is reported by SYSTAT to be in the HB state (hibernate), it is never eligible for run time. The HB state indicates that the detached job generated an error or has completed execution. The problem of a detached job in the HB state is handled by logging into the system at a free terminal, by using the attach capability of LOGIN and attaching the job to a terminal. Once the detached job is attached to a terminal, messages can be printed.

7.3 DYNAMIC DISPLAY OF SYSTEM STATUS - VT5DPY AND VT50PY

The VT5DPY and VT50PY system programs display the system status on a VT05 and VT50 alphanumeric display terminal and update the status at given intervals. The programs are stored in the system library with protection code <232> and require a 14K job area to run. Since the programs run in such a large job area, it is suggested that they be used only on systems with ample memory.

Two programs are provided to display system status: VT5DPY (for the VT05 alphanumeric display terminal) and VT50PY (for the VT50 DECterminal). Both programs are the same except for the output routines. (The hardware commands and screen capacity of the VT05 differ from those of the VT50.) Each program is created at system generation time by appending the related output routines (VT05.DPY and VT50.DPY) to a common display program (DISPLY.BAS) and compiling the result.

The operation of each program is automatic or can be varied at user option. When the user starts the program, he specifies the interval at which the information on the screen is updated. The information displayed is similar to that given by the SYSTAT system program. However, the user can type commands to modify the items displayed. For more information on SYSTAT, refer to Section 4.3 in the RSTS-11 System User's Guide and Section 7.2 of this guide.

7.3.1 Running and Terminating VT5DPY and VT50PY

The command RUN $VT5DPY should be used on a VT05 terminal and the command RUN $VT50PY should be used on a VT50 terminal.

The program runs and prints the INTERVAL question to which the user can type the number of seconds between updates and any combination of the following modifiers.

Modifier	Description
/DET	Have the program run detached from this terminal or from the terminal specified.
/KBn:	Run the program at keyboard unit n if it is available. If /DET is specified, run the program detached.
/NO FILL	Set the fill factor to 0. Improves program performance when the VT05 runs at 300 baud or less.
/PRIORITY	Run the program at special priority rather than at -8 priority.

If the user types only the RETURN key or types no number with a modifier in response to the INTERVAL question, the program updates the status every 15 seconds.

Running the program detached allows the user to temporarily interrupt the program and use the terminal to run other programs. To interrupt the display, type the CTRL/C combination and the program prints a message telling the user that the terminal is available. When the user releases the terminal by logging off the system, the program automatically displays the status information on the screen again.

When the program displays the system status, it prints a header line on the top line of the screen, skips a line, and fills the leftmost half of the screen with job status information and the rightmost half of the screen with, in turn, disk structure, busy device, run time system, free buffer status, and message receiver information. Upon filling the screen the program moves the blinking cursor to the first character position on the second line of the screen. This indicates that the program is idle.

At the interval specified, system tables are interrogated and the status information on the screen is updated with any changed data. While executing routines to extract update information, the program prints the message WORKING ... and leaves the cursor positioned to the right of the message. After completing the update, the cursor is returned to its idle position.

While the cursor is at its idle position, the user can type commands to modify the contents and arrangement of items on the screen. Any command typed should be terminated by the ALTMODE key. Although any line terminator works, ALTMODE leaves the cursor positioned on the blank line. The program takes no action on invalid commands.

Some commands which add items to the screen can be preceded by a minus sign which indicates taking away the effect of the command. Table 7-3 lists and describes the commands.

7.3.2 Screen Layout

The program partitions the screen into three parts: the header line, the lefthand half and the righthand half. The components of these parts are defined in the following subsections.

Table 7-3
VT5DPY and VT50PY Commands

Command Type	Format	Description
General	C	Clear the screen and display new status.
	Sn	Display memory status in place of job status. Start with the 8K word section less than or equal to n. If n is not given, start at the beginning of memory.
	J	Display job status in standard manner.
	Jn	Display job status starting with n+1 active job. Use to overcome physical limitation of the screen.
	Xn	Change the interval to n seconds.
	X0	Update the display with an interval of 0 seconds (that is, run continuously) but lower the priority so that other jobs are not stalled.
Job Status	O	Display the account number of operator jobs as [OPR]. An operator job has a project number 1 and a programmer number less than 200.
	-O	Replaces OPR in operator account designations with the actual project and programmer numbers.
	T	Display total CPU time each job has expended. The time is displayed as number of hours, minutes, seconds, and tenths of seconds under the RUN-TIME column.
	+	Display the increment of CPU time each job has expended since VT5DPY last updated the screen. User can return to total CPU time by typing T.
	%	Display the amount of CPU time each job has expended as a percent of the total CPU time expended. User can return to total or increment of CPU time by typing, respectively, T or +.
	J-O	Do not display operator jobs. (Operator jobs are those running under project number 1 and programmer number less than 200.)

(continued on next page)

Table 7-3 (Cont.)
VT5DPY and VT50PY Commands

Command Type	Format	Description
Job Status (Cont.)	J+O	Include operator jobs in display.
	J-D	Do not display detached jobs.
	J+D	Include detached jobs in display.
	P	Indicate priority of jobs more exactly than + and - characters.
	-P	Indicate priority of jobs with + (for higher than normal) and - (for lower than normal) characters.
	W	Indicate under STATE column, the last WAIT state rather than actual state.
	-W	Remove last WAIT state and indicate actual state of each job.
	K	Display, under SIZE column, the amount of memory occupied by each job.
	-K	Display, under SIZE column, the amount of memory remaining to each job.
Disk Structure	D Dn -D	Display disk structure statistics. If n is 1, place item first on the screen. Preceding minus sign removes same from the screen.
	L -L	Display, under COMMENTS column, the logical name of each device. Preceding minus sign replaces logical names with standard PUB, PRI, NFS, or LCK notations.
Busy Devices	B Bn -B	Display busy device statistics. If n is 1, place item first on the screen. Preceding minus sign removes same from the screen.
Free Buffer	F Fn -F	Display free buffer statistics. If n is 1, place item first on the screen. Preceding minus sign removes same from the screen.
Message	M Mn -M	Display message receiver statistics. If n is 1, place item first on the screen. Preceding minus sign removes same from screen.
Run Time System	R Rn -R	Display Run Time System data. If n is 1, place data first on the screen. R preceded by a minus sign removes same from the screen.

7.3.2.1 Header Line - The header line looks like the following sample.

```
RSTS V06A SYSTEM #880 STATUS ON 31-JUL-74 11:15 UP 2:22:22
```

The header line contains the system identification, the current date and time of day and the number of hours, minutes and seconds since the start of time sharing operations. The latter item is termed up time.

7.3.2.2 Job Status - The job status consists of job statistics information similar to that of SYSTAT with the addition of a few states and a PR column for running priority.

The additional states which can appear in the STATE column are listed and described below.

Xnn	Job is swapped out and occupies slot number nn in swapping file X. The swapping file is denoted by letters A, B, C, or D to represent respectively files SWAP0.SYS through SWAP3.SYS.
^C	Job is in CTRL/C state, awaiting input to the monitor.
LCK	Job is locked in memory for the current operation.
PK	Job is running on a pseudo keyboard rather than on a physical terminal.
RS	Job is runnable and waiting for memory to be allocated so that the system can swap it in.
SWI	Job is currently being swapped into memory.
SWO	Job is currently being swapped out of memory.

The PR column can display the following abbreviations.

If -P is in effect:

+	Higher than normal priority
-	Lower than normal priority
S	Special run priority
^	CTRL/C temporary priority
K	Keyboard delimiter temporary priority

If P is in effect:

+n	Positive priority n
0	Zero priority
-n	Negative priority n

7.3.2.3 Memory Status - The S command causes the program to print a table indicating the usage of each 1K word portion of memory. This memory status report replaces the job status report on the lefthand half of the screen. To display the job information again, use the J command.

Because of the limitations of the screens, all memory status information may not be displayed simultaneously. The Sn command allows the user to determine the starting 8K section. The display program prints 8 abbreviations per row on the screen. Each abbreviation concerns the status of a 1K section of memory. The starting 1K section is in the leftmost position of the first row of the memory status display table. The number of rows printed, and thus the extent of memory covered, is limited by the terminal. The VT50 screen allows 72 1K sections to be displayed; the VT05 screen allows 168 1K sections to be displayed. The program indicates the starting 1K section by printing a header line in the following format.

MEMORY USAGE (STARTING AT nK)

The value n is the 1K section number.

The memory status report uses the following abbreviations.

MON	Occupied by RSTS/E monitor
n	Occupied by Run Time System n where n is the position in the Run Time System list. (See Section 7.3.2.8 in this document.)
n	Occupied by job number n
nLK	Job number n is locked in this 1K portion
nSO	Job number n is being swapped out of memory
nSI	Job number n is being swapped into memory
NXM	Memory space is locked or nonexistent
END	End of physical memory for user jobs

7.3.2.4 Disk Structure - The disk structure report produced by the D command is the same as that printed by SYSTAT. The COMMENT column can have the following items.

PRI	Cartridge or pack is private.
PUB	Cartridge or pack is public.
NFS	Disk is open as a non-file structured device.
LK	Disk is in locked state.
xxxxxx	Logical name of disk (the identification label or the name associated to the disk by the ASSIGN command). Appears only if user types the L command.

7.3.2.5 Busy Device Statistics - The busy device statistics are the same as the ones generated by SYSTAT.

7.3.2.6 Message Receiver Statistics - The message receiver statistics reflect information on jobs using the send/receive system function call. The program displays the following information.

xxxxxx (PRIV)	The six character logical identification of the receiving job. The designation (PRIV) indicates the job receives messages from privileged sending jobs.
x	Number of the job.
x/x	Lefthand number (decimal) shows number of messages queued for the job and rightmost number (decimal) shows the declared maximum number of messages the job can have queued.

7.3.2.7 Free Buffer Status - The free buffer statistics are the same as those generated by the SYSTAT program.

7.3.2.8 Run Time System Statistics - The statistics for run time systems are displayed on the righthand portion of the screen. Table 7-4 describes the elements of the display.

Table 7-4
Run Time System (RTS) Statistics

Item	Format	Meaning
Name	BASIC	Denotes the BASIC-PLUS compiler and Run Time System.
	RTSLIB	Denotes the auxiliary RTS supplied by DIGITAL.
Size	n	The number of 1K sections occupied by the Run Time System.
Number of Users	n	The number of user jobs currently running under control of the Run Time System.
Comments	R-O	The Run Time System allows read only access.
	TEMP	The Run Time System occupies user job space temporarily. Module is unloaded if no job is running under its control.
	R-W	The Run Time System allows read and write access.
	LOADING	The Run Time System is being loaded into memory.

7.4 DETERMINING TERMINAL AND REMOTE LINE CHARACTERISTICS - TTYSET

The RSTS/E system operates with a variety of terminals. During system generation, the system manager specifies the numbers and types of terminal interfaces as part of the hardware configuration. The types of terminals are intentionally not specified. Since many terminals can operate in Teletype mode at a speed of 110 baud, the system automatically sets the default characteristics of all line interfaces as those of the ASR-33 device. The system manager or privileged user must set the characteristics of other types of terminals.

The TTYSET system program sets terminal characteristics. As described in Section 4.5 of the RSTS-11 System User's Guide, a user can set the characteristics of his own terminal by TTYSET commands. The system manager or privileged user can set the characteristics of terminals other than his own by the privileged TTYSET KBN: command. The standard method of setting characteristics of local lines is by the TTYSET commands in the START.CTL file. This procedure sets up all local lines automatically at the start of each time sharing session.

Two methods exist to set characteristics of remote lines. First, all remote lines can have the default characteristics of the ASR-33 device. Consequently a remote user must log into the system at a speed of 110 baud and then run TTYSET to set the characteristics of his terminal. If the remote line is connected to a DC11 single line interface or to a DH11 multiplexer line, the user can type the TTYSET SPEED command to change the baud rate. The terminal characteristics revert to those of the ASR-33 device when the user hangs up the line.

The second method for setting characteristics of remote lines involves setting the so called ring characteristics. The system manager or privileged user can run TTYSET to set the ring characteristics which cause the system to automatically use the characteristics so set every time a call is answered on a particular remote line. By setting ring characteristics for a remote line, the system manager can establish certain lines for alphanumeric display terminals running at 300 baud, for 2741 communications terminals, or for other types of terminals. The ring characteristics remain in effect for the current time sharing session unless changed again by TTYSET. TTYSET commands in the START.CTL file can also establish ring characteristics.

7.4.1 Establishing the Terminal Speed Characteristics File - TTYSET.SPD

The TTYSET system program has a command which sets the baud rate of a variable speed line so that a variable speed terminal can operate at any of its legal speeds. The command is the SPEED command. The proper functioning of the SPEED command requires that the system manager create a file called TTYSET.SPD and store it under the system library account [1,2]. The TTYSET.SPD file contains entries in ASCII format which specify the values for the speeds (baud rates) that a given terminal line can handle. Currently, the DC11 remote line interface and the DH11 multiplexer interface are the only RSTS/E supported devices which support programmable baud rates.

The system manager creates the TTYSET.SPD file under the system library account [1,2] by running PIP and specifying values as the following sample format demonstrates.

```
RUN $PIP
PIP - RSTS V05B-24 SYSTEM #219
#SY0:$TTYSET.SPD<KB:/FA
11,  0,50, 75,110,-1,150,200,300,600,1200,1800,2400,4800,9600,-1,-1
12,110,-1,150,300,-1, -1, -1, -1, -1,  -1,  -1,  -1,  -1,-1,-1
13,  0,-1, -1,110,-1,150, -1,300,600,1200,  -1,2400,  -1,  -1,-1,-1
^Z
#^Z

READY
```

Each line of the file TTYSET.SPD contains 17 entries separated by commas. The first entry is the keyboard number of the line interface which allows programmable baud rates. Each of the remaining 16 entries in the line represents the baud rate corresponding to a valid speed setting of 0 through 15. Speed settings that are not used must have a value of -1.

The conventional baud rates are shown in the sample format above at the line beginning with keyboard line number 11. The sixth, sixteenth, and seventeenth entries in the line are not used by TTYSET and must contain -1. The entries are, on the DH11 multiplexer, 134.5 baud and two externally controlled baud rates.(1) If a baud rate is allowed on a keyboard line, its value must appear as an entry in the line for that keyboard number.

Take, for demonstration purposes, the line beginning with keyboard line number 12 above. The entry for KB12 represents values that would be entered for a typical DC11 remote line interface which is established to handle the valid, variable speeds of a VT05A display terminal or LA30S serial DECwriter. The DC11 interface supports four different speeds. The four entries in the line after the entry for the keyboard line number are used to define the speeds allowed on that DC11 line. The VT05A terminal and LA30S DECwriter can handle speeds of 110, 150, and 300 baud. The entries for those speeds are made in the relative positions of the line for KB12. The entry for 134.5 (between 110 and 150) is not used by TTYSET(1) and must contain a -1. All other entries in a line for a DC11 must contain a -1.

The line beginning with keyboard line number 13 in the sample format above represents values that would be entered for a DH11 multiplexer line to handle the valid speeds of a VT05B display terminal. The DH11 interface supports up to 16 different speeds including 0 baud. (A speed of zero effectively turns the line off.) The VT05B terminal can handle speeds of 110, 150, 300, 600, 1200, and 2400 baud; 0 baud and split speeds are also allowable. (Split speeds means that the transmit rate differs from the receiver rate.) The entries for those speeds are made in the proper positions of the line for KB13. For those speeds not supported by or not allowed for the VT05B terminal on the KB13 line, the user must indicate by making a -1 entry.

---

(1)Although the 134.5 entry must not appear in the TTYSET.SPD file, the speed 134.5 is used for 2741 terminals. When the user types the macro command 2741, the program sets speed 4 (134.5) on the DH11 line or speed 2 on the DC11 line but does not check the entry in the TTYSET.SPD file.

The system manager ensures that the file TTYSET.SPD is stored in the system library account [1,2] on the system disk when he types the SY0: device specification and the dollar sign character ($) preceding the filename in the PIP command string. The CTRL/Z combination (^Z) typed on the line following the text for KB13 terminates the entry of text to the ASCII file TTYSET.SPD and returns control to PIP, signalled by the # character being printed. The CTRL/Z combination typed on the line with the # character terminates the PIP run and returns control to the RSTS/E Monitor. It is recommended that, after creating the TTYSET.SPD file, the system manager run the TTYSET program and execute the SPEED command for each line.

7.4.2 TTYSET Privileged Feature - KBn: Command

The system manager sets the characteristics of other terminals in the RSTS/E system by use of the KBn: command. The system manager, while logged into the system under his privileged account, calls the TTYSET system program as follows.

```
RUN $TTYSET
'TTYSET' TERMINAL CHARACTERISTICS PROGRAM
?
```

The program responds with a header line and a question mark character (?), which indicates that the program is ready to accept commands. If the system manager wishes to set the characteristics of a VT05A terminal at keyboard 3, he types the following commands.

```
?KB3:
FOR KB3:?  VT05
FOR KB3:?
```

In response to the system manager typing the KB3: command, TTYSET prints the FOR KB3: prompting message. This message indicates that the commands typed apply to keyboard number 3. The VT05 command immediately sets the characteristics of the line to those of a VT05A alphanumeric display terminal. TTYSET prints the prompting message again. Another KBn: command can be typed to set characteristics of another keyboard.

The system manager can also change discrete characteristics of a terminal. If he desires to limit the line length of the terminal at keyboard unit 4, he types the following commands.

```
FOR KB3:?  KB4:
FOR KB4:?  WIDTH 60
FOR KB4:?  EXIT

EXIT
```

As a result of the execution of the above commands, whenever 60 characters are printed on a line at keyboard unit 4, a carriage return and line-feed operation is performed. The EXIT command terminates TTYSET.

7.4.3 Automatic Setting of Terminal Characteristics

The setting of terminal characteristics in the RSTS/E system applies only to the current time sharing session. This condition allows for replacement of an ASR-33 type terminal with another terminal without having to change the system configuration and, thus, rebuild the system. Since it is quite bothersome for the system manager or a user to set terminal characteristics each time the system is initialized, the INIT system program can automatically set both local and remote terminal characteristics by commands in the START.CTL and CRASH.CTL files. Refer to the sample START.CTL and CRASH.CTL files in Section 3.1.

7.4.4 Setting Terminal Characteristics of Remote Lines - /RING

A remote line user whose terminal is other than an ASR-33 type terminal must set the characteristics of his terminal each time he logs into the system if he wants to ensure recognition of the characteristics of that terminal by the RSTS/E system. The characteristics discussed above remain set until either the telephone is hung up, the line is disconnected or the system shuts down or restarts.

The /RING option relieves the user from the necessity of setting the characteristics each time he logs into the system on a certain remote line. For example, to set the characteristics of the remote line on keyboard 14 for the current time sharing session, the system manager can run TTYSET under a privileged account and type commands as follows.

```
?KB14:/RING
FOR KB14:(RING)?LA36
FOR KB14:(RING)?
```

When the system manager indicates the /RING option with the KBn: command, TTYSET prints the prompting message FOR KBn:(RING) after which any TTYSET command can be typed. The command takes effect immediately and TTYSET prints the prompting message again. If the user is not logged into the system under a privileged account or the characteristics of the line conflict with the command, TTYSET prints an error message and the prompting message. If TTYSET is successful, the characteristics for the remote line are set for the duration of the current time sharing session. Each time a user dials the particular line, the system uses the characteristics for that terminal.

One caution is in order. The DL11E-type interface and the individual local interfaces (KL11, DL11-A through DL11-E, and LC11) do not have programmable baud rates. Therefore, the user must not execute commands to change baud rates on a keyboard line having any of those interfaces.

7.5 INITIALIZING A DISK DURING TIME SHARING - DSKINT

The system manager or a privileged user can initialize a disk cartridge with the DSKINT system program rather than the DSKINT initialization option. DSKINT writes a minimal file structure on a previously formatted disk. All user files on the disk are destroyed. All disks must be initialized before they can be used on a RSTS/E system.

System disk initialization is a unique process performed by the system. The procedures described in this section apply only to non-system disks. Each user can initialize his disk without affecting other users on the system.

DSKINT is called by using the RUN command as shown below.

```
RUN $DSKINT
```

DSKINT responds by printing questions requesting specific information. Refer to Chapter 3 of the RSTS/E System Generation Manual for detailed descriptions of these questions. Shown below is a sample dialog.

```
RUN $DSKINT

DISK ? RK
UNIT ? 1
PACK ID ? MARK
PACK CLUSTER SIZE ? 16
MFD PASSWORD ? LEVY
MFD CLUSTER SIZE ? 16
PUB OR PRI ? PRI

READY
```

Because the DSKINT system program does not perform any pattern checks for bad blocks and cannot format a disk, it is not as versatile as the DSKINT initialization option. Furthermore, the program does not create the storage allocation table. After the disk is initialized by the DSKINT system program, the user must build the SAT by executing the CLEAN command of the UTILTY system program. See Section 7.1.2 for a description of the CLEAN command.

7.6 OPTIMIZING DISK DIRECTORIES - REORDR

The REORDR program reorders blocks in a specified directory on a RSTS/E file structured disk to give optimum structure for fast access. The program is stored on the system disk with protection code <124>. The disk to be employed must be logically mounted. The directories being reordered must not be accessed while REORDR runs.

REORDR accepts a device and an account specification or several specifications separated by commas. The device specification must be explicit. Logical names are acceptable but SY: is not acceptable. The account specification can contain the wild card specification character (*). The system account [0,1] and the MFD account [1,1] are never reordered.

It is suggested that the REORDR program be run once a week or whenever directory accesses appear to degrade system operation. The safest way to run the program is from a special INIT control file at system start up time. The following is a sample control file.

```
LOGINS
LOGIN KB1: [1,2]
FORCE KB1: RUN $UTILTY
FORCE KB1: NO LOGINS
FORCE KB1: EXIT
SEND REORDR IS RUNNING
SEND PLEASE STAND BY
MOUNT DK1:PRIV1
MOUNT DK2:PRIV2
FORCE KB1: RUN $REORDR
FORCE KB1: DK1:[*,*],DK2:[*,*]
FORCE KB1: ^
FORCE KB1: LOGINS
FORCE KB1: BYE/F
@START.CTL
END
```

The commands in the control file mount the disks to be employed and log a job into the system to run the REORDR program. The UTILTY program is run to disable further logins and thus prevent users from accessing the disks during the reordering. A message broadcast to all terminals can inform users of the action. Following the commands to reorder all directories on the disk and terminate REORDR, INIT enables logins again, logs keyboard 1 off the system and transfers to the standard control file to perform a normal system start up.

By following the recommended procedures, little chance exists for users to access a directory while it is being reordered. Thus, REORDR is always run by INIT from a specially tailored control file. The following dialogue shows the procedure.

```
SYSTEM INITIALIZATION PROGRAM
CONTROL FILE NAME?  REORDR.CTL
```

INIT then uses the tailored control file REORDR.CTL which contains the commands to run REORDR.

7.7 PROCESSING USER COMMENTS - GRIPE

The RSTS/E system includes a program to allow users of the system to communicate comments to the system manager. Comments are entered, under the control of the GRIPE system program, to a common file named GRIPE.TXT file. The file, GRIPE.TXT, which retains the user comments for inspection by the system manager, is created, expanded, and deleted on an as-needed basis under the system library account [1,2]. As an aid in identifying the user who entered the comment, the GRIPE system program uses a name item supplied in the individual user's account information in the ACCT.SYS file, also stored in the system library account [1,2]. However, the name item and entry for the account in the ACCT.SYS file are not required for GRIPE to run. See Section 4.1.3 for the ACCT.SYS file format.

The system manager or a privileged user invokes the GRIPE system program in the same manner as the general user. (GRIPE is described in Chapter 4 of the RSTS-11 System User's Guide.) Once GRIPE prints its query line, the system manager can then examine the contents of GRIPE.TXT or can clear the contents of GRIPE.TXT.

The *LIST command is used in the following manner to examine the contents of the GRIPE.TXT file.

```
RUN $GRIPE
YES?  (END WITH ESCAPE)
*LIST$ OUTPUT? LP:

READY
```

The system manager types *LIST and the ESCAPE or ALT MODE key immediately after the query line. (Typing the ESCAPE or ALT MODE key on a separate line causes *LIST to be entered as text into the GRIPE.TXT file.) If the GRIPE.TXT file is empty, the message NO GRIPES FOUND is printed, followed by the READY message. Otherwise, the GRIPE program requests an output device on which to list the contents of the GRIPE.TXT file. The system manager can type the RETURN key to have the comments listed at the keyboard printer or can type a device designator, such as LP: shown in the example above. The output for each user comment in the GRIPE.TXT file consists of an identification line (including the account entering the comment, the date and time it was entered, and an account name taken from the ACCT.SYS file) and the text of the comment. The program run is automatically terminated upon completion of the output. Control is returned to BASIC-PLUS. This action is signalled by printing of the READY message.

The system manager clears the contents of the GRIPE.TXT file by using the *RESET command after invoking GRIPE. The following example demonstrates the use of *RESET.

```
RUN $GRIPE
YES? (END WITH ESCAPE)
*RESET$

READY
```

The system manager must type *RESET immediately followed by the ESCAPE or ALT MODE key. The clearing of the GRIPE.TXT file and termination of the GRIPE run is signalled by the READY message.

7.8 COMMUNICATING WITH OTHER TERMINALS - PLEASE AND TALK

A user can communicate with the system console terminal or with another user's terminal by using the system programs, PLEASE and TALK, respectively. These programs are discussed individually in the sections that follow.

Both programs have a protection code of <232>, permitting all users to run them. The system manager can change the protection code to <124> so only privileged users can run these programs.

7.8.1 Sending a Message to the Console Terminal - PLEASE

PLEASE enables the user to send a message to the system console terminal (KB0:). This message, along with the user's job number and keyboard number, is broadcast directly to the system console terminal.

PLEASE is called as follows:

```
RUN $PLEASE
```

PLEASE prints the query line shown below:

```
YES!!MAY I HELP YOU?  (END WITH CONTROL/Z)
```

Characters are not transmitted until the CR key is typed, at which point the entire line is broadcast. So type the CR key after each line.

Since each line of the message is broadcast immediately, no prompting characters are printed on the sending terminal between the lines of the message. One line of the message can be typed into the sending terminal while simultaneously another line is being printed out on the system console terminal. If the system console terminal is in use, each line of the message is stored in one of that terminal's buffers and printed at a later time.

Type the CTRL/C or CTRL/Z combination to terminate the program after the last line of the message has been broadcast.

NOTE

> Do not type the CTRL/C or CTRL/Z combination to broadcast the last line. Using either of these combinations before typing the CR key terminates the program and the last line is not broadcast.

PLEASE prints THANK YOU. to indicate all lines have been broadcast.

Here is an example of a user broadcasting a message to the console terminal.

```
RUN $PLEASE
YES!! MAY I HELP YOU? (END WITH CONTROL/Z)
```

```
I WILL BE OUT OF TOWN FOR THE NEXT TWO WEEKS.
WILL LET YOU KNOW WHEN I RETURN.
                -- JOHN SMITH

^Z
THANK YOU.

READY
```

The message printed on the system console terminal automatically includes the sender's job and keyboard numbers as shown below.

```
JOB 9 KB 15: I WILL BE OUT OF TOWN FOR THE NEXT TWO WEEKS.
JOB 9 KB 15: WILL LET YOU KNOW WHEN I RETURN.
JOB 9 KB 15:                 --JOHN SMITH
```

7.8.2 Sending a Message to Another Terminal - TALK

TALK enables users to broadcast messages, line by line, to each other's terminals. Both the sending and the receiving terminal must be on-line, but no user need be logged in on the receiving terminal.

TALK is called as follows:

```
RUN $TALK
```

The first query line printed is:

```
TO WHICH KEYBOARD (KB)?
```

Type the number of the keyboard to which the message is to be sent; then type the CR key. If the receiving terminal is in use, each line of the message is stored in one of that terminal's buffers and printed at a later time.

NOTE

As a general rule, don't broadcast to a terminal in use before consulting with that terminal's operator. An unscheduled message can disrupt an elaborate printout, sometimes destroying hours of work.

Typing the number of a non-existent or off-line terminal does not return an error message. In the case of a nonexistent terminal, the program terminates when the CR key is typed. Messages sent to an off-line terminal, however, are broadcast normally, but never received. In this case, typing the CR key does not terminate the program.

TALK prints the following instructions:

```
YOU MAY PROCEED - CARRIAGE RETURN SENDS THE LINE
'ALTMODE' ('ESCAPE') SENDS AND TERMINATES THE PROGRAM
```

To send a message, type the CR key after typing each line. Characters

are not transmitted until the CR key is typed, at which point the entire line is broadcast. Since each line of the message is broadcast immediately, no prompting characters are printed on the sending terminal between the lines of the message.

The ALT or ESCAPE key, used to terminate the program, can be typed after the last CR key or instead of it. This key prints an ESC character (echoed as $) on the sending terminal. When typed instead of the CR key, the ALT or ESCAPE key also prints an ESC character on the receiving terminal. When this happens, the system does not perform a carriage return operation on the receiving terminal.

Shown below is a typical sending terminal dialog between keyboard 12 (sender) and keyboard 9 (receiver).

```
RUN $TALK

TO WHICH KEYBOARD (KB)?  9
YOU MAY PROCEED - CARRIAGE RETURN SENDS THE LINE
'ALTMODE' ('ESCAPE') SENDS AND TERMINATES THE PROGRAM
GEORGE

PLEASE MOUNT MY DECTAPE WHEN YOU GET A CHANCE.

                                        THANKS...

                                         HERMAN

$
READY
```

The receiving terminal's printout automatically identifies the sending device by enclosing it in asterisks. No ESC character is printed. The actual message is printed next to the identification, as follows:

```
   KB12 ** GEORGE
** KB12 **
** KB12 **    PLEASE MOUNT MY DECTAPE WHEN YOU GET A CHANCE.
** KB12 **
** KB12 **                                THANKS...
** KB12 **
** KB12 **                                 HERMAN
$
```

If at this point, keyboard 9 returns a message, it is done as follows:

```
RUN $TALK

TO WHICH KEYBOARD (KB)? 12
YOU MAY PROCEED - CARRIAGE RETURN SENDS THE LINE
'ALTMODE' ('ESCAPE') SENDS AND TERMINATES THE PROGRAM
OKAY, HERMAN...

YOUR DECTAPE IS MOUNTED ON UNIT #2

SORRY FOR THE DELAY!
                                        GEORGE$

READY
```

Notice that the ESC key was typed instead of the final CR key in the above example. This prints the ESC character on both the sending and receiving terminals (see below).

```
** KB9 ** OKAY, HERMAN...
** KB9 **
** KB9 ** YOUR DECTAPE IS MOUNTED ON UNIT   2.
** KB9 ** SORRY FOR THE DELAY!
** KB9 **
** KB9 **                                 GEORGE$
```

APPENDIX A

HARDWARE BOOTSTRAP PROCEDURES

Bootstrapping a device involves using the central processor unit (CPU) console switches to access and initiate a hardware loader. The hardware loader contains machine instructions for reading a special record from the device. The record, called a bootstrap record, is transferred into memory and executes a specially designed software program. For the bootstrap operation to succeed, the device accessed must be on line and ready; the medium accessed must contain a proper bootstrap record; and the console terminal must be on line.

The console switches and their usage are described in Chapter 11 of the PDP-11/70 Processor Handbook and in Chapter 8 of the PDP-11/45 Processor Handbook and the PDP-11/40 Processor Handbook. The bootstrap procedure to use depends upon the type of hardware bootstrap device on the system. Table A-1 summarizes the addresses needed to bootstrap a device. The detailed procedures to bootstrap a device are presented according to the types of hardware bootstrap devices available.

Table A-1
Summary of Hardware Bootstrap Addresses

Device to Bootstrap	Bootstrap Type			
	BM873-YA	BM873-YB(1)	MR11-DB	BM792-YB(2)
RF11 disk	773000	773136	773100	777462
RK11 disk cartridge unit 0	773010	773030	773110	777406
RP03 disk pack unit 0	773100	773350	773154	776716
RP04 disk pack unit 0	-	773320	-	-
RK11 disk (unit specified in SR)	-	773032	-	-
RP03 disk (unit specified in SR)	-	773352	-	-
RP04 disk (unit specified in SR)	-	773322	-	-
TM11/TU10 Magtape	773050	773110	773136	See footnote 3.
TM02/TU16 Magtape	-	773150	-	-
TC11/TU56 DECtape	773030	773070	773120	777344

(1)To bootstrap a non-zero disk unit, set the address in the Switch Register, press the LOAD ADRS switch, set the unit number in the switch Register, and press the START switch.

(2)For the BM792-YB loader, set the address 773100 in the Switch Register, depress the LOAD ADRS switch, set the value from the table in the Switch Register, and press the START switch.

(3)To bootstrap a magtape, use the loading routine described in Section A.4.

HARDWARE BOOTSTRAP PROCEDURES

A.1 BM873-YA PROCEDURE

If the BM873-YA Restart/Loader is on the system, perform the following steps.

Move the CPU console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to one of the following values.

773000	for RF11 disk
773010	for RK11 disk cartridge unit 0
773100	for RP03 disk pack unit 0
773050	for TM11/TU10 magtape unit 0
773030	for TC11/TU56 DECtape unit 0

Depress the CPU LOAD ADRS switch.

Depress the CPU START switch.

A.2 BM873-YB PROCEDURE

If the BM873-YB Restart/Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to one of the following values.

773030	for RK11 disk cartridge unit 0
773136	for RF11 disk
773320	for RP04 disk pack unit 0
773350	for RP03 disk pack unit 0
773032	for RK11 disk unit specified in the Switch Register
773322	for RP04 disk unit specified in the Switch Register
773352	for RP03 disk unit specified in the Switch Register
773110	for RM11/TU10 magtape
773150	for RM02/TU16 magtape
773070	for RC11/TU56 DECtape

Depress the CPU LOAD ADRS switch.

If necessary, set the CPU Switch Register to the unit number of the disk drive being bootstrapped.

Depress the CPU START switch.

A.3 MR11-DB PROCEDURE

If the MR11-DB Bulk Storage Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to one of the following values.

773100	for RF11 disk
773110	for RK11 disk cartridge unit 0
773154	for RP03 disk pack unit 0
773136	for TM11/TU10 magtape unit 0
773120	for TC11/TU56 DECtape unit 0

Depress the CPU LOAD ADRS switch.

Depress the CPU START switch.

A.4 BM792-YB PROCEDURE

If the BM792-YB Hardware Loader is on the system, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to 773100.

Depress the CPU LOAD ADRS switch.

Set the CPU Switch Register to one of the following values.

777462	for RF11 disk
777406	for RK11 disk cartridge unit 0
776716	for RP03 disk pack unit 0
777344	for TC11/TU56 DECtape unit 0

Depress the CPU START switch.

To bootstrap a TM11/TU10 magtape from unit 0 when the system has neither the BM873 nor the MR11-DB loader, the user must manually enter a load routine into memory using the CPU console Switch Register and the DEP Switch.

To load the routine, perform the following steps.

Move the CPU Console ENABLE/HALT switch to its HALT position and back to its ENABLE position.

Set the CPU Switch Register to 010000.

Depress the CPU LOAD ADRS switch.

Load the following contents into memory using the Switch Register and DEP switch.

Address	Contents
010000	012700
010002	172524
010004	005310
010006	012740
010010	060011
010012	105710
010014	100376
010016	005710
010020	100767
010022	012710
010024	060003
010026	105710
010030	100376
010032	005710
010034	100777
010036	005007

Set the Console Switch Register to 010000.

Depress the CPU LOAD ADRS switch.

Depress the CPU START switch.

If the system reads the tape but halts at address 010034, the device generated a magtape error. The user can try another drive. If the system appears to take no action and halts, verify the accuracy of the routine by using the CPU Console EXAM switch. Use the Switch Register and the DEP switch to correct any erroneous contents. Rewind the tape to its load point before executing the routine again. If no recovery is successful, it will be necessary to have a DIGITAL Field Service representative check the drive. If the hardware is working properly, it will be necessary to use a new magtape reel.

APPENDIX B

RSTS/E CONSISTENCY ERROR MESSAGES

During the execution of RSTS/E initialization routines, many checks are made to determine the consistency of system structures. The existing structures are compared to their definitions and references as they appear in other parts of the system. The checks must always be successful. If they are not successful, a consistency error has been detected and the system requires correction.

The initialization code is executed in two phases: first, the CILUS phase after the system disk is bootstrapped and before the OPTION: message is printed; and, second, the option phase during which any of the initialization options can be executed.

B.1 CILUS PHASE ERRORS

If an error occurs during the CILUS phase, the initialization code prints a descriptive message and the following text.

```
FATAL RSTS SYSTEM INITIALIZATION ERROR!

THE FATAL ERROR OCCURRED DURING THE CILUS PHASE
OF SYSTEM INITIALIZATION; THERE IS NO RECOVERY.
```

The processor is halted so that, if the CPU console CONT switch is pressed, the code bootstraps the system disk again. Bootstrapping the disk, however, is usually not a worthwhile procedure because most of the errors reflect a serious problem rather than a transient hardware error. Reloading of the system disk (see Section 2.9 of the RSTS/E System Generation Manual) is recommended in some cases but regenerating the system is required in other cases.

The list below gives the descriptive messages followed by a one-word advice (in parentheses) indicating the recovery procedure. The following legend briefly summarizes the recovery procedures.

(reboot)	Bootstrap again. If unsuccessful, reload the system disk.
(reload)	Reload the system disk (see Section 2.9 of the RSTS/E System Generation Manual). If reloading does not eliminate the error condition, regenerate the system.
(regen)	Regenerate the system and load the new CIL. (See Section 2.1 of the RSTS/E System Generation Manual.)
(hard)	A hardware adjustment or addition is required.
(spr)	File a Software Performance Report. There is no recovery.

The following are the descriptive messages.

CHECKSUM ERROR IN CIL INDEX.	(reboot)
CIL LINE IS MISSING FROM CIL INDEX.	(reload)
COMD LINE IN CIL INDEX IS IN INCORRECT FORMAT,	(reload)
DEVICE BOOTED DOES NOT MATCH SYSTEM DEVICE.	(regen)
DEVICE BOOTED IS NOT A LEGAL RSTS SYSTEM DEVICE.	(regen)
DEVICE ERROR WHEN READING CIL INDEX.	(reboot)
DOUBLE OCCURRENCE OF SOME SYSTEM IMAGE.	(regen)
FORMAT ERROR IN CIL INDEX.	(reload)
ILLEGAL BLOCK SIZE IN CILUS BOOTSTRAP PARAMETERS.	(regen)
ILLEGAL SWAPPING DISK DETECTED.	(regen)
INIT ERROR - SECTION I.BOOT TOO SMALL	(spr)
INIT WAS INCORRECTLY ASSEMBLED OR LINKED.	(spr)
*** MONITOR IS TOO BIG ***	(regen)
RSTS/E REQUIRES EIS TO RUN!	(hard)
RSTS/E REQUIRES MEMORY MANAGEMENT HARDWARE!	(hard)
TOO MANY NON-SYSTEM IMAGES IN CIL	(regen)

If the initialization code finds that certain modules are missing, it prints one of the two sets of messages.

SYSTEM MODULE xxx IS MISSING FROM CIL. ONE OR MORE CRITICAL MODULES MISSING FROM CIL.	(regen)
RUN-TIME SYSTEM MISSING FROM CIL. ONE OR MORE MODULES MISSING FROM CIL.	(regen)

The module name replaces xxx in the message.

Under certain conditions, the initialization code prints a warning message followed by the OPTION: message. The user can operate within the restrictions described in the warning. The following are the warning messages possible.

```
WARNING ** THIS PDP-11/70 WILL RUN IN 11/45 MODE
DUE TO THE NUMBER OF UNIBUS NPR DEVICES.

WARNING ** THIS MACHINE DOES NOT HAVE THE CONFIGURED CLOCK!
RSTS/E WILL CRASH IF YOU ATTEMPT TO START TIME-SHARING.

WARNING ** THIS MACHINE DOES NOT HAVE A STACK LIMIT REGISTER!
ALTHOUGH NOT ADVISED, RSTS/E WILL RUN WITHOUT THE OPTION.

WARNING ** THIS RSTS SYSTEM, WHICH WAS NOT BUILT FOR AN 11/70,
WILL CRASH IF PARITY ERRORS OCCUR!

WARNING ** THIS RSTS SYSTEM, WHICH WAS BUILT FOR AN 11/70,
WILL CRASH IF PARITY ERRORS OCCUR!

WARNING ** DBx: IS DUAL PORTED, PROCEED WITH CAUTION!
```

If the message concerning the dual ported RP04 disk is printed, ensure that the Controller Select switch on the related drive is in the correct position (the port connected to RSTS/E).

B.2 OPTION PHASE ERRORS

If an error occurs during the option phase, the initialization code prints a descriptive message and halts. The following are the messages.

ATTEMPT TO ASK FOR OPTION WHEN CILUS PHASE NOT DONE.

BAD DIRECTORY DETECTED DURING CLEAN.

EXISTING SYSTEM FILE EMPTY OR NON-CONTIGUOUS.

FILE [0,1]BADB.SYS MISSING FROM SYSTEM DISK.

INIT BUG - ATTEMPT TO DELETE NONEXISTENT FILE.

INIT BUG - FAILED TO CREATE RSTS.CIL ON 2ND TRY

INIT BUG - FILE EXISTED WHEN TRYING TO CREATE.

INIT BUG - INSUFFICIENT DIRECTORY SPACE FOR CREATE.

INIT BUG - INSUFFICIENT DISK SPACE FOR CREATE.

INIT BUG - SATT.SYS NONEXISTENT AT TIME OF WOMP.

INIT BUG - UNABLE TO REBUILD DISK

INSUFFICIENT DIRECTORY SPACE FOR SYSTEM FILES.

INSUFFICIENT DISK SPACE FOR [0.1] DIRECTORY.

PACK CLUSTER SIZE IS NOT 1, 2, 4, 8 OR 16.

REQUIRED FILE BADB.SYS DOES NOT EXIST.

REQUIRED FILE SATT.SYS FILE DOES NOT EXIST.

REQUIRED LIBRARY ACCOUNT [1,2] DOESN'T EXIST.

RSTS CIL IS NOT ON A CLUSTER BOUNDARY.

SYSTEM DISK SAT SIZE NOT EQUAL TO COMPUTED SIZE.

SYSTEM FILE CONTAINS BAD BLOCKS - CANNOT REFRESH.

APPENDIX C

AUXILIARY SYSTEM PROGRAM FILES

Certain auxiliary files are built during the system library build procedures. The program which builds each file is stored in the library along with the file. If, for any reason, the file is damaged or destroyed, the file can be created by running the related program as described in this appendix.

C.1 CHARACTER GENERATION FILE - CHARS.QUE

The line printer spooling program SPOOL requires the character generation file CHARS.QUE. The file is a virtual core array and is stored on the system disk during system generation by commands in the SPLER.CTL file. To create the CHARS.QUE file, first ensure that the old copy is deleted from the system library. The CHARS program terminates with an error if a file named CHARS.QUE exists in the system library directory. Next, run the CHARS program by typing the following command.

```
RUN $CHARS
READY
```

After terminating, CHARS returns control to BASIC-PLUS as indicated by the READY message.

C.2 BATCH COMMAND DECODING FILE - BATCH.DCD

The BATCH system program requires the command decoding file BATCH.DCD. The file is a virtual core array and is created during system generation by commands in the SPLER.CTL file. To create the BATCH.DCD file, first ensure that the old copy is deleted from the system library. Run the BATDCD program while logged into the system under a privileged account. The program terminates with an error if a file named BATCH.DCD exists in the system library directory. The following sample dialogue shows the proper procedure.

```
RUN $BATDCD
READY
```

After terminating, BATDCD returns control to BASIC-PLUS as indicated by the READY message.

APPENDIX D

NUMBER CONVERSION

Many applications require a number based on bit values in a PDP-11 word. The following list shows the octal and decimal values for each bit in the PDP-11 word.

Bit Number	Octal Value	Decimal Value
0	1	1
1	2	2
2	4	4
3	10	8
4	20	16
5	40	32
6	100	64
7	200	128
8	400	256
9	1000	512
10	2000	1024
11	4000	2048
12	10000	4096
13	20000	8192
14	40000	16384
15	100000	32768(32767+1)

HOW TO OBTAIN SOFTWARE INFORMATION

SOFTWARE NEWSLETTERS, MAILING LIST

The Software Communications Group, located at corporate headquarters in Maynard, publishes software newsletters for the various DIGITAL products. Newsletters are published monthly, and keep the user informed about customer software problems and solutions, new software products, documentation corrections, as well as programming notes and techniques.

There are two similar levels of service:

- The Software Dispatch
- The Digital Software News

The Software Dispatch is part of the Software Maintenance Service. This service applies to the following software products:

PDP-9/15
RSX-11D
DOS/BATCH
RSTS/E
DECsystem-10

A Digital Software News for the PDP-11 and a Digital Software News for the PDP-8/12 are available to any customer who has purchased PDP-11 or PDP-8/12 software.

A collection of existing problems and solutions for a given software system is published periodically. A customer receives this publication with his initial software kit with the delivery of his system. This collection would be either a Software Dispatch Review or Software Performance Summary depending on the system ordered.

A mailing list of users who receive software newsletters is also maintained by Software Communications. Users must sign-up for the newsletter they desire. This can be done by either completing the form supplied with the Review or Summary or by writing to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

SOFTWARE PROBLEMS

Questions or problems relating to DIGITAL's software should be reported as follows:

North and South American Submitters:

Upon completion of Software Performance Report (SPR) form remove last copy and send remainder to:

Software Communications
P.O. Box F
Maynard, Massachusetts 01754

The acknowledgement copy will be returned along with a blank SPR form upon receipt. The acknowledgement will contain a DIGITAL assigned SPR number. The SPR number or the preprinted number should be referenced in any future correspondence. Additional SPR forms may be obtained from the above address.

All International Submitters:

Upon completion of the SPR form, reserve the last copy and send the remainder to the SPR Center in the nearest DIGITAL office. SPR forms are also available from our SPR Centers.

PROGRAMS AND MANUALS

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center.

Digital Equipment Corporation
Software Distribution Center
146 Main Street
Maynard, Massachusetts 01754

Digital Equipment Corporation
Software Distribution Center
1400 Terra Bella
Mountain View, California 94043

Outside of the United States, orders should be directed to the nearest Digital Field Sales Office or representative.

USERS SOCIETY

DECUS, Digital Equipment Computers Users Society, maintains a user exchange center for user-written programs and technical application information. The Library contains approximately 1,900 programs for all DIGITAL computer lines. Executive routines, editors, debuggers, special functions, games, maintenance and various other classes of programs are available.

DECUS Program Library Catalogs are routinely updated and contain lists and abstracts of all programs according to computer line:

- PDP-8, FOCAL-8, BASIC-8, PDP-12
- PDP-7/9, 9, 15
- PDP-11, RSTS-11
- PDP-6/10, 10

Forms and information on acquiring and submitting programs to the DECUS Library may be obtained from the DECUS office.

In addition to the catalogs, DECUS also publishes the following:

DECUSCOPE	-The Society's technical newsletter, published bi-monthly, aimed at facilitating the interchange of technical information among users of DIGITAL computers and at disseminating news items concerning the Society. Circulation reached 19,000 in May, 1974.
PROCEEDINGS OF THE DIGITAL EQUIPMENT USERS SOCIETY	-Contains technical papers presented at DECUS Symposia held twice a year in the United States, once a year in Europe, Australia, and Canada.
MINUTES OF THE DECsystem-10 SESSIONS	-A report of the DECsystem-10 sessions held at the two United States DECUS Symposia.
COPY-N-Mail	-A monthly mailed communique among DECsystem-10 users.
LUG/SIG	-Mailing of Local User Group (LUG) and Special Interest Group (SIG) communique, aimed at providing closer communication among users of a specific product or application.

Further information on the DECUS Library, publications, and other DECUS activities is available from the DECUS offices listed below:

DECUS
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts 01754

DECUS EUROPE
Digital Equipment Corp. International
(Europe)
P.O. Box 340
1211 Geneva 26
Switzerland

RSTS/E System Manager's Guide
DEC-11-ORSMD-A-D

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form (see the HOW TO OBTAIN SOFTWARE INFORMATION page).

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you do not require a written reply, please check here. ☐

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754