RT–11 Installation Guide

AA–H376B–TC

March 1983

This manual tells you how to install the RT–11 operating system. It provides information necessary to bootstrap and preserve the distribution volume(s); create the working system; and customize, preserve, and test the working system.

This manual supersedes the *RT–11 Installation and System Generation Guide*, AA–H376A–TC.

Operating System: RT–11 Version 5.0

digital equipment corporation · maynard, massachusetts

First Printing, March 1983

Printed in U.S.A.

A postage-paid READER'S COMMENTS form is included on the last page of this document. Your comments will assist us in preparing future documentation.

M17500

Contents

Tables

Preface

How to Use This Manual

This manual introduces you to the RT–11 Version 5 software kit and tells you how to install your operating system. The modular structure of this manual allows you to read only the chapters that you need for your particular combination of hardware and software.

Chapter 1 of the manual introduces you to installation and system generation concepts and helps you choose a reading path through the other chapters.

Installation concepts — both planning activities and actual installation procedures — are covered in Chapters 2 through 9. You should read Chapter 2 to plan your installation, then choose the chapter (3, 4, 5, 6, 7, 8, or 9) that describes the installation of a system most closely resembling your own configuration.

Additional information available in the appendixes includes BATCH streams to build system programs and toggle-in software bootstraps.

If you discover that you need to perform the system generation process, refer to the *RT–11 System Generation Guide.*

Before you begin, you should read the *Guide to RT–11 Documentation,* which describes the other software documents associated with the RT–11 operating system. Familiarity with the RT–11 system, as described in the *RT–11 System User's Guide,* is particularly helpful when you perform the procedures in this manual.

If you are a FORTRAN IV, BASIC–11, FMS–11, CTS–300, or other layered product user, you build your FORTRAN IV, BASIC–11, FMS–11, or other system after building the RT–11 system. A layered product is software that is sold separately but requires the RT–11 operating system environment to run. See the appropriate installation manual for instructions on installing such products.

Documentation Conventions

You should become familiar with certain symbolic conventions used in this manual.

1. Examples consist of actual computer output whenever possible. In these examples, user input appears in red where it must be differentiated from computer output.

2. Unless the manual indicates otherwise, all commands or command strings end with a carriage return. The symbol (RET) represents a carriage return, (LF) a line feed, (SP) a space, (ESC) an ESCAPE or ALTMODE, and (TAB) a tab.

3. To produce certain characters in system commands, you must type a combination of keys concurrently. For example, while holding down the CTRL key, type C to produce the CTRL/C character. Key combinations such as this are documented as (CTRL/C), (CTRL/O), and so forth.

4. In descriptions of command syntax, capital letters represent the command name, which you must type. Lowercase letters represent a variable for which you must supply a value.

5. In examples, you must distinguish between the capital letter O and the number 0. Examples in this manual represent these characters as follows:

Letter O:	O
Number 0:	0

6. The sample terminal dialog in this manual contains version numbers where they would normally appear. The version numbers include xx in those fields that can vary from installation to installation. The exact contents of these fields are not of interest in the examples in this manual, as long as appropriate digits appear in the area indicated. The same is true for the FREE BLOCKS messages included in device directories.

   If you submit a software performance report (SPR) to DIGITAL, you must include the complete version number.

7. A decimal point (.) follows a number to indicate that it is a decimal number. A number without a decimal point is an octal number. For example, 128. is 128 (decimal) and 126 is 126 (octal).

Chapter 1
Introduction

Now that your PDP–11 processor has been installed and your software distribution kit has been delivered, you are ready to get started with RT–11. To begin, you must install RT–11. Installation requires bootstrapping and preserving the distribution volumes, installing software updates if necessary, creating the working system from selected components, and customizing, preserving, and testing the working system. In most cases, your system is ready to run once you have installed and tested it. However, if you need certain special features, for example, multiterminal support, you must perform the system generation process. In this case, you must run the SYSGEN program, answer its questions, and assemble your own generated monitor(s) and device handlers. Figure 1–1 summarizes the flow of the installation process. Refer to the *RT–11 System Generation Guide* for specific information about generating specialized monitors.

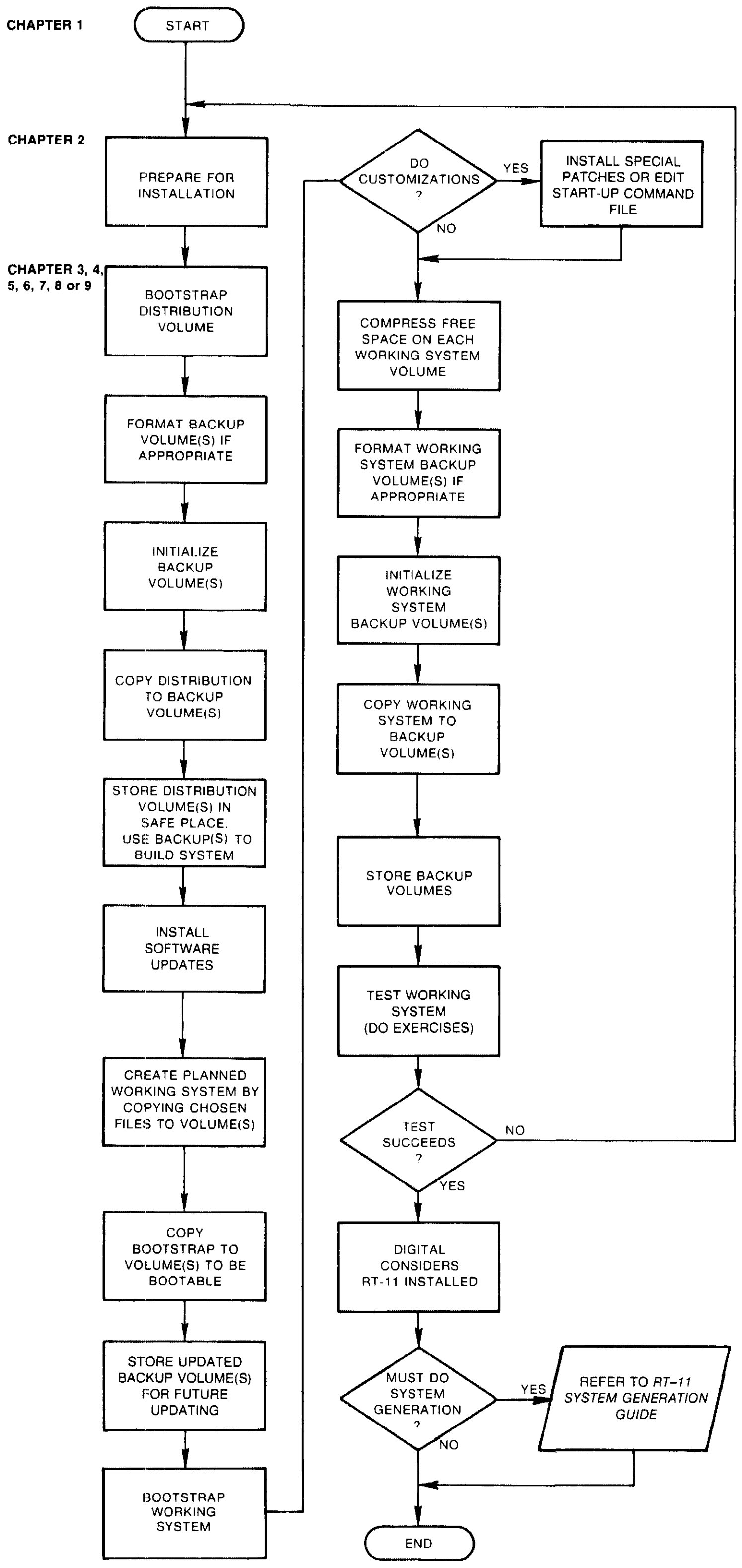
The following sections will help you decide which course of action you should take. Once you have established which steps you must complete, identify the chapters you need to read. You need not read this entire manual.

1.1 Identifying Your Needs

In order to select the procedures you must perform and the reading path you should follow, answer the following questions, using the information in Sections 1.1.1 through 1.1.3.

- What is your hardware configuration?
- Which monitor(s) do you need?
- Which features do you need?

Figure 1–1: RT–11 Installation Process


CHAPTER 1
START
CHAPTER 2
PREPARE FOR INSTALLATION
CHAPTER 3, 4, 5, 6, 7, 8 or 9
BOOTSTRAP DISTRIBUTION VOLUME
FORMAT BACKUP VOLUME(S) IF APPROPRIATE
INITIALIZE BACKUP VOLUME(S)
COPY DISTRIBUTION TO BACKUP VOLUME(S)
STORE DISTRIBUTION VOLUME(S) IN SAFE PLACE. USE BACKUP(S) TO BUILD SYSTEM
INSTALL SOFTWARE UPDATES
CREATE PLANNED WORKING SYSTEM BY COPYING CHOSEN FILES TO VOLUME(S)
COPY BOOTSTRAP TO VOLUME(S) TO BE BOOTABLE
STORE UPDATED BACKUP VOLUME(S) FOR FUTURE UPDATING
BOOTSTRAP WORKING SYSTEM
DO CUSTOMIZATIONS ?
YES
INSTALL SPECIAL PATCHES OR EDIT START-UP COMMAND FILE
NO
COMPRESS FREE SPACE ON EACH WORKING SYSTEM VOLUME
FORMAT WORKING SYSTEM BACKUP VOLUME(S) IF APPROPRIATE
INITIALIZE WORKING SYSTEM BACKUP VOLUME(S)
COPY WORKING SYSTEM TO BACKUP VOLUME(S)
STORE BACKUP VOLUMES
TEST WORKING SYSTEM (DO EXERCISES)
TEST SUCCEEDS ?
NO
YES
DIGITAL CONSIDERS RT-11 INSTALLED
MUST DO SYSTEM GENERATION ?
YES
REFER TO RT–11 SYSTEM GENERATION GUIDE
NO
END

1.1.1 What Is Your Hardware Configuration?

Your hardware configuration is an important consideration for three reasons:

1. Installation procedures vary according to the hardware configuration.
2. System generation cannot be performed on all hardware configurations.
3. System generation procedures vary according to the hardware configuration.

Chapters 3 through 9 contain detailed procedures for installing RT–11 systems on certain common hardware configurations. Unless you have an unusual configuration, read only the chapter intended for your configuration. (See Table 1–5 to identify your reading path.)

1.1.2 Which Monitor(s) Do You Need?

You have to decide which monitor or monitors you need in order to know what steps to follow in getting started. Your distribution kit includes a base-line single-job monitor (BL), a single-job monitor (SJ), a foreground/background monitor (FB), and an extended memory monitor (XM). Sections 1.1.2.1 through 1.1.2.4 describe the various monitors, and Table 1–1 summarizes the features available in the distributed monitors. The sections and the table give you the information you need to make a decision about your monitor requirements.

Table 1–1: Features Available in Distributed Monitors

	Monitor			
Characteristics	**BL**	**SJ**	**FB**	**XM**
Memory limits	16K to 30K words	16K to 30K words	24K to 30K words	32K to 124K words (up to 2M words in Q-bus systems)
Size of resident monitor	1.8K words	2.0K words	5.3K words	6.6K words plus 2.6K words for USR
Size of disk image	71 blocks (approx.)	73 blocks (approx.)	86 blocks (approx.)	95 blocks (approx.)
FG job support	no	no	yes	yes
System job support	no	no	no*	yes*
Timer support (.MRKT and .CMKT — including midnight date/time rollover)	no	no*	yes	yes
Error messages on system I/O errors	no*	yes*	yes	yes

* Feature enabled/disabled by system generation.

(continued on next page)

Table 1–1: Features Available in Distributed Monitors (Cont.)

	Monitor			
Characteristics	**BL**	**SJ**	**FB**	**XM**
Multiterminal support	no	no*	no*	no*
60Hz clock frequency	yes*	yes*	yes*	yes*
50Hz clock frequency instead of 60Hz	no*	no*	no*	no*
Programmable clock instead of line as system clock	no	no*	no*	no*
Start-up command file	yes*	yes*	yes*	yes*
Memory parity support	no	no*	no*	yes*
Month and year date rollover support	no	no*	no*	no*
Fetchable device handlers	yes	yes	yes	yes*
User command linkage support	no	no*	no*	no*
Concise command language	yes	yes	yes	yes
High-speed ring buffer	no	no*	no*	no*
Contains all keyboard monitor commands	yes*	yes*	yes*	yes*
Floating point support	no	yes*	yes*	yes*
Error message on power fail	no	no*	no*	no*
BATCH	no	no*	no*	no*
Error logging	no	no*	no*	no*
MSCP devices	yes†	yes†	yes†	yes†
RK11/RK05	yes†	yes†	yes†	yes†
RL01/RL02	yes† (2 drives)	yes† (2 drives)	yes† (2 drives)	yes† (2 drives)
RJS03/RJS04	yes‡ (RJS03)	yes‡ (RJS03)	yes‡ (RJS03)	no‡ (RJS03)
RK06/RK07	yes†	yes†	yes†	yes†
RF11	yes‡	yes‡	yes‡	no‡
RP11/RPR02/RP03	yes‡	yes‡	yes‡	no‡

* Feature enabled/disabled by system generation.

(continued on next page)

† Refer to Section 2.7.11.

‡ RT–11 no longer supports this device; however, the handler is included in the software kit.

Table 1–1: Features Available in Distributed Monitors (Cont.)

	Monitor			
Characteristics	**BL**	**SJ**	**FB**	**XM**
RX01	yes† (2 drives)	yes† (2 drives)	yes† (2 drives)	yes† (2 drives)
RX02	yes† (2 drives)	yes† (2 drives)	yes† (2 drives)	yes† (2 drives)
DECtape	yes‡	yes‡	yes‡	no‡
DECtape II	yes†	yes†	yes†	yes†
PDT–11 intelligent terminal	yes‡	yes‡	yes‡	no
Virtual memory handler	yes	yes	yes	yes (as data device only)
File-structured magtape — TM11	yes (2 units)	yes (2 units)	yes (2 units)	yes (2 units)
File-structured magtape — TJU16	yes (2 units)	yes (2 units)	yes (2 units)	yes (2 units)
File-structured magtape — TS11	yes (2 units)	yes (2 units)	yes (2 units)	yes (2 units)
Hardware magtape TM11	no§	no§	no§	no§
Hardware magtape TJU16	no§	no§	no§	no§
Hardware magtape TS11	no§	no§	no§	no§
TA11 cassette	yes‡	yes‡	yes‡	no
VT11/VS60	no	no*	no*	no
Line printer	yes	yes	yes	yes
Serial printer	yes	yes	yes	yes
High-speed paper tape reader/punch	yes‡	yes‡	yes‡	no‡
CR11 card reader	yes‡	yes‡	yes‡	no
Null handler	yes	yes	yes	yes
Logical disk handler	yes	yes	yes	yes
Single-line editor	yes	yes	yes	yes
Extra device slots	0*	0*	0*	4*

* Feature enabled/disabled by system generation.
† Refer to Section 2.7.11.
‡ RT–11 no longer supports this device; however, the handler is included in the software kit.
§ Refer to Section 2.7.11.2.

1.1.2.1 Base-Line Single-Job Monitor — The base-line single-job monitor (BL) is a stripped-down version of the single-job monitor (SJ). The BL monitor does not support optional monitor and device functions. It has the smallest residency requirement of any RT–11 monitor and is intended for use in applications that require small monitor size and minimal executive support.

Table 1–2 summarizes the differences between the distributed BL and SJ monitors.

Table 1–2: Base-Line Monitor Versus Single-Job Monitor

BL Monitor	SJ Monitor
Does not support graphic display.	Supports graphic display terminal (VT11/VS60) as ASCII console terminal.
System halts on power failure or system device I/O error.	System prints error message on power failure or system device I/O error.
Supports only one terminal.	Can be generated to support more than one terminal.
Minimal memory and system device requirements for monitor.	Requires slightly larger memory and system device for monitor.
Does not include floating point support.	Includes floating point support.

If your application program is very large and you require the smallest possible monitor, the BL monitor may be your best choice. The BL monitor can perform all the system commands and can run most of the utilities. For highly interactive applications, you should use the SJ monitor. The more complete error processing and device support available in the SJ monitor provide a base that is easier to use and more flexible.

1.1.2.2 Single-Job Monitor — Of all the distributed RT–11 monitors, the single-job monitor (SJ) has the fastest response times at interrupt and keyboard levels and the lowest memory requirements, except for the BL monitor. If your application involves interactive program development, maximum-throughput real-time data acquisition, or continuous execution of a single end-user application program, the SJ monitor is the best choice.

The SJ monitor supports all hardware devices (except memory management hardware) and runs all the system utilities except QUEUE. It runs in any supported configuration with at least 16K words of memory but cannot make use of more than 28K words of memory (30K on an LSI–11 processor).

1.1.2.3 Foreground/Background Monitor — The foreground/background monitor (FB) is the smallest RT–11 monitor that supports multiprogramming. It allows you to execute a completely independent foreground job at a higher software priority level than the background, while you use the remaining system facilities to support the background. The RT–11 foreground job is not intended for a two-user time-sharing system. Rather, it best supports a stable, event-driven real-time or I/O application that can execute with a minimum of user interaction while the bulk of the system's business is conducted in the background.

The background in the foreground/background environment appears just like the SJ monitor; all the facilities available to you as an SJ user are available to you as an FB user in the background.

The FB monitor provides you with more features than the SJ monitor. These include:

1. The ability to run a foreground job
2. Enhanced terminal service
3. Extended executive facilities for real-time applications

The minimum memory requirement for the FB monitor is 16K words. The FB monitor resides in 8K or fewer words of memory and cannot use more than 28K words of memory (30K on an LSI–11 processor).

If your application includes the need for a software priority real-time application to run concurrently with normal system development and data-processing applications, the FB monitor is the suitable choice. If you do not require concurrent real-time execution, you can conserve system resources by using the SJ monitor.

1.1.2.4 Extended Memory Monitor — The extended memory monitor (XM) is the largest and most powerful of the RT–11 executives. It has all the facilities of the FB monitor; in addition, it can support up to 128K words of memory in systems supporting 18-bit architecture and up to 2 megawords of memory in Q-bus systems supporting 22-bit architecture. The XM monitor provides a set of programmed requests that allow advanced applications to make use of additional memory above 28K (using the PDP–11 memory management unit).

The XM monitor is significantly larger than the FB monitor. The XM monitor's requirement that the USR be resident adds even more to the monitor memory overhead. Consequently, if you have no more than 48K words of memory, choose one of the other monitors. Also, if your application involves program development or execution of an end-user application that does not make use of extended memory facilities, you may opt for the FB or SJ monitor.

The XM monitor requires 32K words of memory, a memory management unit (KT11 hardware), and the extended instruction set (EIS) option to operate.

1.1.3 Which Features Do You Need?

The following sections summarize the available RT–11 features, so that you can decide which features you need and determine how to obtain them. You can secure some of the features through simple software customization procedures during a standard installation process. However, other features are available only when you go through the system generation process.

1.1.3.1 Is Installation All You Need to Do? — You need to decide whether installation is all you need to do or whether you will also need to perform the system generation process. Table 1–3 summarizes the features that you can add to your system by performing simple customizations during installation. Section 2.7 describes these software customizations in detail and provides instructions for implementing them. Many users will not need to make any of these customizations.

Table 1–3: Features Available Through Simple Software Customizations

Feature	Section	Procedure
Adding subroutines to SYSLIB.OBJ	2.3.5	To add subroutines to the default system library, SYSLIB.OBJ, use the librarian. The RT–11 linker searches SYSLIB to resolve undefined globals at the end of a link operation. Normally, you should customize SYSLIB to contain the system subroutines, FORTRAN OTS routines, and application-specific subroutines.
Changing characters SRCCOM uses to indicate insertions and deletions	2.7.1	Normally, when you use the SRCCOM utility or DIFFERENCES/CHANGEBAR command to compare two files, SRCCOM uses the vertical bar character to indicate insertions and the bullet (alphabetic o) character to indicate deletions. You can modify SRCCOM to use different characters.
Changing default output device from line printer to terminal	2.7.2	If your configuration does not include a line printer, you can change the default output device (which certain monitor commands use) from line printer to terminal. To make this change, you can edit the start-up command file.
Changing location of VT11/VS60 floating vectors	2.7.3	If you add certain other devices to a configuration that includes a VT11 or VS60, you must modify the monitor to move the VT11 or VS60 vector addresses.
Changing number of directory columns	2.7.4	When you use the DIRECTORY command, the directory prints in two columns. When you use the DIRECTORY/BRIEF command, the directory contains five columns with less information. You can modify DIR to change the default number of columns.
Changing default order of directory listings	2.7.5	By default, DIR lists a directory's contents by physical position. You can modify DIR to change the default to any one of five orders: creation date, file name, file size, file type, or position.

(continued on next page)

Table 1–3: Features Available Through Simple Software Customizations (Cont.)

Feature	Section	Procedure
Changing number of /Q p-sects LINK allows	2.7.6	You can modify LINK to change the number of absolute base-address p-sects that LINK allows. Normally, LINK's /Q option lets you specify the absolute base addresses of up to eight p-sects in your program.
Changing size of LINK's library module list	2.7.7	To change the default size of LINK's list of library modules, you can modify LINK. LINK normally creates a list of 252 (octal) modules to be included from libraries during the link operation. If you are a DIBOL user, you must customize the software.
Changing size of QUEUE work file	2.7.8	You can modify QUEUE to change the size of the work file QUEUE uses to queue files to be sent to a device. If you increase the size of the work file, QUEUE can handle more files at one time.
Modifying EDIT	2.7.9	In certain applications using VT11/VS60 terminals, the text window overflows onto the scrolled editing commands, making a portion of the screen difficult to read. You can alleviate this problem by changing EDIT to reduce the size of the text window. If you have a terminal with nonstandard ESCAPE code (that is, the terminal generates 175 octal or 176 octal rather than 33 octal), you must modify EDIT so that it operates correctly.
Extracting overlay handlers from SYSLIB.OBJ	2.7.10	If you intend to use overlaid programs, you need the overlay handlers that are included in SYSLIB. If you are a MACRO-only user and need only the overlay handlers, you can extract them from SYSLIB for use as a separate library.
Installing other devices	2.7.11	When you boot the system device, the bootstrap installs the devices in your hardware configuration if the appropriate device handler is on the system device and if there are enough device slots. If you want to install a device that the bootstrap does not install, use the REMOVE and INSTALL commands. You can also: *Change LP, LS, DD, RK, DX, DY, DL, DM, or DU addresses:* If any of these devices are installed at nonstandard CSR and vector addresses, you can use the monitor SET command to change the addresses. *Install hardware magtape:* If you need hardware magtape support, you must perform a system generation and then rename the hardware handler to MT.SYS, MM.SYS, or MS.SYS.

(continued on next page)

Table 1–3: Features Available Through Simple Software Customizations (Cont.)

Feature	Section	Procedure
Installing other devices (cont.)		*Set magtape parity and density:* If you need to set magtape parity or density (other than the standard 800 bits/in, odd parity), you can use the monitor SET command for TM11 or TJU16 magtape (but not TS11).
		Change RP02 support to RP03: To use RP03 disk drives, you must think of each RP03 drive as two logical units of 40,000 blocks.
Modifying BATCH to save space	2.7.12	If you need to save disk space, you can store certain system programs on DK: rather than on SY: and cause BATCH to access them there. Modify BATCH so that it invokes system programs with the monitor RUN command (which assumes DK: as the default) rather than with the R command (which always uses SY:).
Modifying LINK to change default SYSLIB device	2.7.13	By customizing the linker, you can change the device on which the default system library, SYSLIB.OBJ, resides.
Modifying help text	2.7.14	To change the help text that prints when you use the HELP command, you must create your own help text file, process that text file with LIBR, and copy the resulting library and the file HELP.EXE to the same volume.
Preventing fatal system errors from causing a reset	2.7.15	You can customize the monitor to prevent it from performing a hard reset of errors when a fatal system error occurs. The normal reset stops I/O, protecting media from being corrupted. Some hardware errors may be more easily diagnosed without this reset.
Running RT–11 in less memory than is available	2.7.16	You can customize your software if your application requires that RT–11 run in less memory than is available.
Forcing 22-bit addressing	2.7.17	You can modify the XM monitor to force PDP–11/23 processors to use the 22-bit addressing mode.
Setting upper limit on file size	2.7.18	If your application requires an upper limit to the size of a file, you can change the monitor to set the limit you require.
Specifying 50-cycle clock rate	2.7.19	To use a 50-cycle clock rate rather than the 60-cycle clock of the standard monitors, you must modify the monitor.

(continued on next page)

Table 1–3: Features Available Through Simple Software Customizations (Cont.)

Feature	Section	Procedure
Using CAPS–11 to load files	2.7.20	If you need to use CAPS–11 to load RT–11 files, you must modify the cassette handler, CT.SYS, to alter header records.
Setting upper limit on memory size	2.7.21	If your PDP–11 does not generate a bus timeout trap, RT–11 does not load into memory properly. You can alter the monitor to cause the bootstrap to never look for more than 28K words of memory.
Suppressing bootstrap message	2.7.22	If you want to prevent the monitor identification message from printing when you bootstrap a monitor, you can modify that monitor.
Suppressing start-up indirect command file	2.7.23	If you want to prevent the start-up command file from executing when you bootstrap a monitor, you can modify that monitor.
Suppressing start-up indirect command file echo	2.7.24	If you want the start-up indirect command file to execute when you bootstrap a monitor but you do not want the command lines in the file to echo (appear) on the terminal, you can modify the monitor.
Changing bootstrap message	2.7.25	If you want to change the monitor identification message that appears when you bootstrap a monitor, you can modify that monitor.
Changing default device for indirect command files	2.7.26	If you want to change the default device for indirect command files, you can modify the monitor. Normally, the monitor looks for the command file on DK:.
Changing default file type for indirect command files	2.7.27	If you want to change the default file type for indirect command files, you can modify the monitor. Normally, indirect command files have the default file type .COM.
Changing default device for FRUN command	2.7.28	If you want to change the default device for the FRUN command, you can modify the monitor. Normally, the monitor looks for the foreground program on device DK: when you type FRUN filnam.
Changing default file type for FRUN command	2.7.29	If you want to change the default file type for the FRUN command, you can modify the monitor. Normally, the default file type for foreground programs is .REL.
Changing default device for EDIT command	2.7.30	If you want to change the default device for the EDIT command, you can modify the monitor. Normally, the monitor looks for EDIT.SAV on device DK:.

(continued on next page)

Table 1–3: Features Available Through Simple Software Customizations (Cont.)

Feature	Section	Procedure
Changing default file name for EDIT command	2.7.31	If you want the monitor to run an editor other than the default editor when you type the EDIT command, you can modify the monitor to change the file name of the default editor.
Using Examine and Deposit above background job	2.7.32	If you want to be able to examine and modify the monitor and the I/O page, you can customize the monitor to remove a restriction on the use of the examine (E) and deposit (D) commands. Normally, the monitor allows you to examine and modify only locations inside the background job's area.
Changing default device for QUEMAN	2.7.33	If you want to send a file to a device other than LP: when you use the PRINT command in an FB system (with QUEUE running as a foreground job), you can modify the monitor to change the default. This modification also changes the default for the DELETE/ENTRY command.
Changing indirect command file nesting depth	2.7.34	If you want to change the depth to which you can nest indirect command files, you can modify the monitor. Normally, RT–11 allows you to nest indirect command files three deep.
Changing threshold for resuming output-stalled jobs	2.7.35	If you have a foreground job that produces much terminal output and stalls frequently (waiting for room in the terminal output buffer), you can modify the monitor to improve throughput.
Changing default number of directory segments	2.7.36	If you want DUP to use a different default number of directory segments when you initialize volumes, you can modify DUP. DUP normally uses the number of directory segments specified in a table in DUP.
Changing width of banner pages from 132 to 80 columns	2.7.37	If you want to change the width of banner page printouts from 132 to 80 columns, you can modify QUEUE.
Modifying listing page length in LINK	2.7.38	If your line printer uses paper that is not 10.5 inches long, or if you do not have a line printer, you can modify LINK to change the listing page length from the standard 60 lines.

(continued on next page)

Table 1–3: Features Available Through Simple Software Customizations (Cont.)

Feature	Section	Procedure
Assigning help file	2.7.39	You can modify the monitor to assign your HELP file to a file or device other than SY:HELP.SAV.
Changing device from which IND is run	2.7.40	You can modify the monitor to change the device from which IND is run.
Replacing bad blocks	2.7.41	You must customize DUP if your user-written device handler supports bad block replacement. One change is required to replace all bad blocks, another to replace only bad sectors.
Including user-written magtape handler	2.7.42	You must customize DUP, PIP, and MDUP if you use a user-written magtape handler.
Producing 1600 bits/in bootable magtape	2.7.43	You can edit DISMT1.COM to replace module MBOOT.BOT with MBOT16.BOT, a primary bootstrap for producing 1600 bits/in, phase-encoded bootable magtape.
Changing default file type of logical disk files	2.7.44	You can change the default file type of logical disk files from .DSK by modifying either the LD.SYS file or the conditional file SYSGEN.CND.

1.1.3.2 Do You Need to Perform the System Generation Procedures? — You must perform the system generation procedures if you want to add to your system any features listed in Table 1–4. You may also need to perform the system generation procedures if you want to remove features to reduce the size of the monitor and improve response time. Refer to the system generation dialog and the descriptive text in the *RT–11 System Generation Guide* for further information about these features.

You should not attempt to perform the system generation procedures unless your hardware configuration meets certain requirements. DIGITAL supports automatic system generation (under license) only on a system with at least 16K words of memory and 2000 free blocks of disk storage. The minimum configuration that DIGITAL recommends for system generation is a system with at least two disk drives and 24K words of memory. DIGITAL also supports system generation on dual-diskette systems (with at least 28K words of memory) if you use the manual method described in the *RT–11 System Generation Guide*. However, DIGITAL does not recommend this very lengthy procedure.

Table 1–4: Features Available Only Through the System Generation Procedures

Feature	Function
Asynchronous terminal status	Provides a program with the updated status of terminals in multiterminal systems. MU BASIC–11 requires this support.
BATCH support	Job control language allows RT–11 to operate unattended. All monitors except BL support BATCH.
Device I/O timeout	Permits device handlers to issue a mark-time programmed request. DECnet applications require this support, but RT–11 does not.
Disable .FETCH request under XM	Disables the use of the .FETCH request under XM, requiring device handlers to be resident in memory.
Double-density only RX02	Permits you to use only double-density RX02 diskettes on the system. This feature makes the RX02 handler smaller.
DZ11 up to 9600 baud	Permits you to initialize lines at specific baud rates up to 9600.
Error logging	Keeps a statistical record of device, memory parity, and memory cache errors. An error logging job is created when you select this support. The EL job retrieves information that is later available to you in summary report format. All monitors except BL support error logging.
Extra device slots	Permits you to add devices to the system after it is built. The number of logical assignments you can make is equal to the number of devices plus the empty device slots in the system.
High-speed ring buffer	Causes character processing and interpretation to be performed at the fork level, allowing short bursts of characters transmitted at very high rates to be received. Use of this feature is recommended with PDTs. All monitors except BL support this feature.
Keyboard monitor command subsets	Allows you to choose one, two, or three subsets of the keyboard monitor commands instead of all the commands.
Memory parity	Causes the system to print an error message when a memory parity error occurs if your configuration includes memory parity hardware. If you have this hardware but do not select this support, the system halts when memory errors occur. All monitors except BL support this feature; it is available in the distributed XM monitor.
Month and year rollover	Adds support that automatically rolls over the date at the end of the month and the end of the year. Normally, you must reset the date and time. This support is useful for applications that run continuously and over a long period of time. All monitors except BL support this feature.

(continued on next page)

Table 1–4: Features Available Only Through the System Generation Procedures (Cont.)

Feature	Function
Multiterminal support	Permits you to use two or more terminals with the SJ, FB, or XM monitor. MU BASIC–11 requires this support.
Multiterminal timeout	Causes the monitor to reset at regular intervals any terminal that goes off line. This support minimizes the impact of static in multiterminal systems. MU BASIC–11 requires this support.
Programmable clock as system clock	Allows you to substitute as the system clock the KW11–P programmable clock for the usual line clock. However, the programmable clock would not then be available for program use. All monitors except BL support this feature.
Ring buffer size	Allows you to change the size of the input and output ring buffers. The input ring is a buffer in the monitor that holds characters you type at a terminal until a program requests them. The output ring is a buffer in the monitor that holds characters until the terminal can print them. The default input ring buffer size is 134 (decimal) characters, and the default output ring buffer size is 40 (decimal) characters.
Second RX01, RX02, or TU58 controller	Adds support for a second RX01, RX02, or TU58 controller, allowing a total of four units in the configuration instead of the usual two.
SJ message on system I/O errors	Causes the SJ monitor to issue an error message instead of simply halting. This feature helps to reduce confusion when an error occurs. The FB and XM monitors always issue error messages.
SJ timer	Configures the SJ monitor to support mark-time and cancel mark-time programmed requests. Otherwise, only the FB and XM monitors support these requests, which provide timer capabilities.
.SPCPS programmed request	The save/set main-line PC and PS (.SPCPS) programmed request changes the flow of control of main-line code by saving the PC and PS and changing the main-line PC to a new value. This support can be generated for only the FB and XM monitors. .SPCPS is especially useful to control the switching context among users in multiuser applications.
System jobs	Assembles the FB or XM monitor to support as many as eight simultaneously active jobs instead of the usual two. Both the error logging subsystem and the device queue program (QUEUE) can run as system jobs. This feature is available in the distributed XM monitor.
User command linkage	Permits users to create their own commands in a format similar to supplied DCL commands. All monitors except BL support this feature.

1.2 Choosing a Reading Path

The modular organization of this manual limits your reading to those chapters that answer your needs. Table 1–5 gives you the information you need to select these chapters. To use the table, find your distribution medium and your target system device (the device that you will want to bootstrap and that will contain the monitors and system components). Then find in the rightmost column the relevant chapters for your configuration. If you must perform a system generation, refer to the *RT–11 System Generation Guide* after you have installed your system.

Also skim the table of contents of this manual for an overview of the additional information offered in the appendixes.

NOTE

> If you are unfamiliar with RT–11 software, you should probably use the distributed monitors uncustomized for a while. Once you have gained familiarity with the system, you will be better equipped to perform customizations or system generation procedures. Be aware, however, that some customizations may be essential for your particular application, and, consequently, you may experience problems when you use software without needed support. If you are an inexperienced user, be sure to study Tables 1–3 and 1–4, but you may want to skip Section 2.7 (Software Customizations) until you have gained some experience.

Table 1–5: Chapters You Should Read

Distribution Medium	System Device in Target System	Chapters
Single-Density Diskette RX01	Single-Density Diskette or DECtapeII	1, 2, 3
	Hard Disk RL01 RL02 RK05 RK06 RK07	1, 2, 4
Hard Disk RL01 RL02 RK05 RC25	Hard Disk RL01 RL02 RK05 RK06 RK07 RC25	1, 2, 5
	Single-Density or Double-Density Diskette	1, 2, 6
Double-Density Diskette RX02	Double-Density Diskette RX02	1, 2, 7
Magtape TM11 TJU16	Hard Disk RL01 RL02 RK05 RK06 RK07	1, 2, 8
Diskette RX50	RD51	1, 2, 9

Chapter 2
Preparing for Installation

You should take certain steps, summarized in the following list, before you actually begin to install your system.

1. Survey the general installation procedures.
2. Study the contents of your software kit.
3. Select the components you need in your working system.
4. Plan the arrangement of components.
5. Install any software updates.
6. Acquire sufficient media.
7. Choose the software customizations you need to make.

You can use the worksheet (Figure 2–2) at the end of this chapter to record the components you select, the arrangement of files you plan, and any other information you need in order to perform the installation procedures.

2.1 Surveying Installation Procedures

Although the specific steps you must perform to install RT–11 depend on your configuration, all users must perform some general procedures.

First, you must bootstrap the distribution volume and preserve the volume (or volumes) by making backup copies. Then you must install software updates if any have been published, in all affected components. You then create the working system from chosen components (eliminating components you do not need) and install the bootstrap on the working system volume(s). At this point, you make software customizations, which do not require performing the system generation process. When you have customized the system, you should compress the working system volume or volumes and preserve them on backup volumes. Once you are satisfied with the working system you have created, you should test it to make sure that it works properly.

2.2 Studying Software Kit Contents

Familiarity with the contents of the software kit you received helps to make the installation procedures go smoothly. You should learn which files are included in the kit and what each file does.

Study Table 2–1 for a summary of the RT–11 components distributed in the kits, and refer to the *RT–11 System Utilities Manual* for a thorough description of each utility. Maps of the specific files on each volume in the various software distribution kits appear in the *RT–11 Release Notes*. RT–11 is distributed on RC25, RL01, RL02, and RK05 disks, on RX01, RX02, and RX50 diskettes, and on magtape. The organization of files on volumes depends on the distribution medium.

Table 2–1: RT–11 Software Components

Type of Software	Description
Monitors	
RT11AI.SYS	Automatic installation monitor
RT11BL.SYS	Base-line single-job monitor
RT11SJ.SYS	Single-job monitor
RT11FB.SYS	Foreground/Background monitor
RT11XM.SYS	Extended memory monitor
Device Handlers	
CR.SYS	Card reader handler for SJ, FB monitors
CT.SYS	TA11 cassette handler for SJ, FB monitors
DD.SYS	DECtape II handler for SJ, FB monitors
DDX.SYS	DECtape II handler for XM monitor
DL.SYS	RL11/RL01/RL02 handler for SJ, FB monitors
DLX.SYS	RL11/RL01/RL02 handler for XM monitor
DM.SYS	RK611/RK06/RK07 handler for SJ, FB monitors
DMX.SYS	RK611/RK06/RK07 handler for XM monitor
DP.SYS	RP11/RP02/RP03 handler for SJ, FB monitors
DS.SYS	RJS03/04 handler for SJ, FB monitors
DT.SYS	DECtape handler for SJ, FB monitors
DU.SYS	MSCP device handler for SJ, FB monitors
DUX.SYS	MSCP device handler for XM monitor
DX.SYS	RX11/RX01 single-density diskette handler for SJ, FB monitors
DXX.SYS	RX11/RX01 single-density diskette handler for XM monitor
DY.SYS	RX211/RX02 handler for SJ, FB monitors

(continued on next page)

Table 2–1: RT–11 Software Components (Cont.)

Type of Software	Description
DYX.SYS	RX211/RX02 handler for XM monitor
LD.SYS	Logical disk subsetting handler for SJ, FB monitors
LDX.SYS	Logical disk subsetting handler for XM monitor
LP.SYS	Line printer handler for SJ, FB monitors
LPX.SYS	Line printer handler for XM monitor
LS.SYS	Serial printer handler for SJ, FB monitors
LSX.SYS	Serial printer handler for XM monitor
MM.SYS	File-structured TJU16 handler for SJ, FB monitors
MMX.SYS	File-structured TJU16 handler for XM monitor
MS.SYS	File-structured TS11 handler for SJ, FB monitors
MSX.SYS	File-structured TS11 handler for XM monitor
MT.SYS	File-structured TM11 handler for SJ, FB monitors
MTX.SYS	File-structured TM11 handler for XM monitor
NL.SYS	Null handler for SJ, FB monitors
NLX.SYS	Null handler for XM monitor
PC.SYS	High-speed paper tape handler for SJ, FB monitors
PD.SYS	PDT–11 skeleton handler for SJ, FB monitors
RF.SYS	RF11 handler for SJ, FB monitors
RK.SYS	RK11/RK05 handler for SJ, FB monitors
RKX.SYS	RK11/RK05 handler for XM monitor
TT.SYS	Terminal handler for SJ monitor
VM.SYS	Virtual disk handler for SJ, FB monitors
VMX.SYS	Virtual disk handler for XM monitor
Other System Files	
SL.SYS	Single-line editor pseudohandler for SJ, FB monitors
SLX.SYS	Single-line editor pseudohandler for XM monitor
SWAP.SYS	External monitor swap blocks
Utility Programs	
BINCOM.SAV	Binary compare utility
BUP.SAV	Backup utility
CREF.SAV	Cross-reference utility
DIR.SAV	Directory utility
DUMP.SAV	File dump utility

(continued on next page)

Table 2–1: RT–11 Software Components (Cont.)

Type of Software	Description
DUP.SAV	Disk maintenance utility
EDIT.SAV	Text editor
FILEX.SAV	Foreign file exchange utility
FORMAT.SAV	Disk formatting utility
HELP.SAV	Help utility
IND.SAV	Indirect control file processor
KED.SAV	Keypad editor for VT100 terminal
K52.SAV	Keypad editor for VT52 terminal
LIBR.SAV	Librarian
LINK.SAV	Linker
MACRO.SAV	MACRO assembler
MDUP.MM	Magtape bootstrap utility for TJU16
MDUP.MS	Magtape bootstrap utility for TS11
MDUP.MT	Magtape bootstrap utility for TM11
MDUP.SAV	Magtape utility
PAT.SAV	Patching utility for program object modules
PIP.SAV	File transfer utility
QUEMAN.SAV	User interface with QUEUE utility
QUEUE.REL	Device queue utility
RESORC.SAV	System resource display utility
SIPP.SAV	Save image patch utility
SLP.SAV	Source language patch utility
SRCCOM.SAV	Source compare utility
SYSMAC.SML	System macro library
Miscellaneous	
BATCH.SAV	BATCH processor
DISMT1.COM	Copy and write bootstrap to distribution magtape 1
DISMT2.COM	Copy distribution magtape 2
ERROUT.SAV	Error log program
HELP.EXE	Help program — executable image
HELP.TXT	Help text
RTBL.MAP	Distributed BL monitor link map

(continued on next page)

Table 2–1: RT–11 Software Components (Cont.)

Type of Software	Description
RTSJ.MAP	Distributed SJ monitor link map
RTFB.MAP	Distributed FB monitor link map
RTXM.MAP	Distributed XM monitor link map
STARTF.COM	FB start-up command file
STARTS.COM	SJ start-up command file
STARTX.COM	XM start-up command file
V5USER.TXT	Distribution kit message text file
System Generation Files	
SYSGEN.COM	SYSGEN command file
BL.ANS	Answer file to build distributed BL monitor
SJFB.ANS	Answer file to build distributed SJ, FB monitors
XM.ANS	Answer file to build distributed XM monitor
Automatic Installation Files	
AI.COM	Automatic installation command file
DUPAI.SAV	Device utility program for automatic installation
STARTA.COM	Start-up file for automatic installation
TERMID.SAV	Console terminal identification program
Debuggers	
ODT.OBJ	Debugging aid
VDT.OBJ	Debugging aid for virtual and multiterminal jobs
Graphics Software	
VTHDLR.OBJ	VT11/VS60 display handler
VTMAC.MAC	Display handler macro file
Libraries and Subroutines	
GETSTR.FOR	FORTRAN subroutine source
PUTSTR.FOR	FORTRAN subroutine source
SYSLIB.OBJ	System FORTRAN-callable subroutines
SYSMAC.MAC	System macro library source
Bootstraps	
MBOOT.BOT	Magtape primary bootstrap for 8-track tape
MBOT16.BOT	Magtape primary bootstrap for 16-track tape
MSBOOT.BOT	Magtape secondary bootstrap

(continued on next page)

Table 2–1: RT–11 Software Components (Cont.)

Type of Software	Description
Demonstration Programs	
DEMOBG.MAC	Demonstration source — SJ, FB macro program
DEMOED.TXT	Demonstration source
DEMOFG.MAC	Demonstration source — FB macro program
DEMOF1.FOR	Demonstration source — FB FORTRAN program
DEMOX1.MAC	Demonstration source
Source Files	
BA.MAC	BATCH handler source file for system generation
BSTRAP.MAC	Bootstrap source file for system generation
CR.MAC	Card reader handler source file for system generation
CT.MAC	Cassette handler source file for system generation
DD.MAC	DECtape II handler source file for system generation
DL.MAC	RL01/02 handler source file for system generation
DM.MAC	RK06/07 handler source file for system generation
DP.MAC	RP11 handler source file for system generation
DS.MAC	RJS03/04 handler source file for system generation
DT.MAC	DECtape handler source file for system generation
DU.MAC	MSCP device handler source file for system generation
DX.MAC	RX01 handler source file for system generation
DY.MAC	RX02 handler source file for system generation
EDTGBL.MAC	Monitor edit log and global definition file for system generation
EL.MAC	Error logger handler source file for system generation
ELCOPY.MAC	Error log job source file for system generation
ELINIT.MAC	Error log job source file for system generation
ELTASK.MAC	Error log job source file for system generation
ERROUT.OBJ	Error log job source file for system generation
ERRTXT.MAC	Error log job source file for system generation
FB.MAC	FB conditional source file for system generation
FSM.MAC	Magtape file support source file for system generation
KMON.MAC	Keyboard monitor source file for system generation
KMOVLY.MAC	Keyboard monitor overlay source file for system generation
LD.MAC	Logical disk subsetting handler source file for system generation

(continued on next page)

Table 2–1: RT–11 Software Components (Cont.)

Type of Software	Description
LP.MAC	Line printer handler source file for system generation
LS.MAC	Serial printer handler source file for system generation
MTTEMT.MAC	Multiterminal programmed request source file for system generation
MTTINT.MAC	Multiterminal interrupt service source file for system generation
NL.MAC	Null handler source file for system generation
PC.MAC	High-speed paper tape handler source file for system generation
PD.MAC	PDT–11 handler source file for system generation
RF.MAC	RF11 handler source file for system generation
RK.MAC	RK05 handler source file for system generation
RMONFB.MAC	FB/XM resident monitor source file for system generation
RMONSJ.MAC	SJ resident monitor source file for system generation
SJ.MAC	SJ conditional source file for system generation
TJ.MAC	TJU16 handler source file for system generation
TM.MAC	TM11 handler source file for system generation
TRMTBL.MAC	Multiterminal table source file for system generation
TS.MAC	TS11 handler source file for system generation
TT.MAC	TT.SYS source file for system generation
USR.MAC	USR source file for system generation
VM.MAC	VM.SYS source file for system generation
XM.MAC	XM conditional source file for system generation
XMSUBS.MAC	XM monitor subroutines for system generation
Unsupported Software	
CONSOL.MAC	Change boot-time console terminal
CUSTOM.TXT	Customization symbol location file
DATIME.MAC	Date and time source file for system generation
DATIME.SAV	Date and time utility
MTYSET.MAC	Autobaud terminal/set characteristics
RTMON.REL	System activity monitor
SEARCH.TEC	TECO macro
SORT.TEC	TECO macro
TECO.SAV	TECO editor

(continued on next page)

Table 2–1: RT–11 Software Components (Cont.)

Type of Software	Description
TTYSET.SAV	Set terminal characteristics
TYPE.TEC	TECO macro
V5NOTE.TXT	RT–11 version 5 release notes
VTMAC.MAC	VT11 macro library

2.3 Selecting Components for Your Working System

Because space on volumes is limited, you should include in your system only those components that are essential to your application. To decide which components these should be, study the space limitations of your particular device, then consider the suggestions given for each software component (Sections 2.3.1 through 2.3.14).

Before selecting components, you should become familiar with your medium's space limitations. (See Table 2–2.) In addition, examine the software kit maps in the *RT–11 Release Notes* to establish how many blocks are occupied by the components residing on each volume and how much free space is available on the volume.

Table 2–2: Device Size

Device	Device Name	Size in Blocks
RX01 Diskette	DX	486
RX02 Diskette	DY	974
RK05 Disk	RK	4.8K
RL01/02 Disk	DL	10.2/20.4K
RK06/07 Disk	DM	27.1/53.7K
RX50 Diskette	DU	786
RD51 Disk	DU	19.6K
RC25 Disk	DU	43K (approximate)
RA80 Disk	DU	220K (approximate)
Virtual Memory	VM	Variable, depending on physical memory available

You should keep in mind the number of blocks various components occupy when you select the components for your working system. Also keep them in mind when you decide how to arrange these components on volumes. In addition, remember that you may need space for data storage on your system volume and other volumes in the working system.

Although you will probably find it more efficient to select components for the working system before actually starting installation, you can also examine volume directories during the installation process to get the component size information. Once you have booted the system, begun the installation process, and backed up the distribution media, you can examine a backup volume's directory (or the directories of each backup volume if there are more than one). To examine a directory, you will use the DIRECTORY command. The following is a sample directory.

```
 17-SEP-82
SWAP  .SYS    26  26-AUG-82     RT11FB.SYS     86  26-AUG-82
DIR   .SAV    18  26-AUG-82     RK    .SYS      3  26-AUG-82
DX    .SYS     3  26-AUG-82     DL    .SYS      4  26-AUG-82
PIP   .SAV    29  26-AUG-82     DUP   .SAV     43  26-AUG-82
FORMAT.SAV    20  26-AUG-82     RESORC.SAV     20  26-AUG-82
RT11FB.SYS    86  26-AUG-82
 10 FILES, 239 BLOCKS
 735 FREE BLOCKS
```

You can also find the total free space on an initialized blank volume by obtaining a directory of the volume.

Once you have examined the software kit maps, read the following sections. The suggestions given in these sections may help you make your selections. The following elements are described.

1. Monitor
2. SWAP.SYS File
3. System Device Handler
4. Other Device Handlers
5. Default System Library
6. Exercises
7. Help Package
8. Line Printer Handler
9. MACRO Assembler
10. Queue Package
11. Source Files
12. Start-up Command File
13. Text Editors
14. Utility Programs

As you make selections, record your choices on the worksheet at the end of the chapter (Figure 2–2).

2.3.1 Monitor

In general, you need only one monitor on a working system. If you do need more than one monitor, build a different system volume for each monitor. When your system device is a large disk, you may have room for several monitors.

2.3.2 SWAP.SYS File

You need the file SWAP.SYS on a system volume to serve as temporary storage for part of a program in memory when KMON, the USR, or both must swap over that program. When KMON or the USR are no longer needed, the system reads this external swap file back into main memory. The keyboard command, SET EXIT NOSWAP, precludes the necessity of using SWAP.SYS once you have bootstrapped the system. However, you must have the SWAP.SYS file on your distribution disk in order to bootstrap your system successfully.

2.3.3 System Device Handler

You need a system device handler on each system volume. For example, if you build an FB system with RL02 as the system device, the file DL.SYS must be on the system disk.

2.3.4 Other Device Handlers

In addition to the system device handler, you need the device handlers for the other peripheral devices in your configuration. You do not need handler files for any devices you do not have. You must have TT.SYS (the terminal handler) on your system volume if you plan to use a BL monitor or a non-multiterminal SJ monitor. However, the FB, XM, and multiterminal SJ monitors each contain an integral, resident TT handler, so you need not have TT.SYS on your system volume if you plan to use any of those monitors.

2.3.5 Default System Library

To use the LINK utility program, you may need the file SYSLIB.OBJ, the default system library, which the RT–11 linker searches to resolve any undefined globals at the end of a linking operation.

Generally, SYSLIB for your application should contain the system subroutines (the file SYSLIB.OBJ found in the software kit), installation-specific libraries of application subroutines, and the FORTRAN OTS routines. If SYSLIB must contain application subroutines and language routines, you must customize it to include these routines. If you intend to link overlaid files, you need SYSLIB, because it contains the overlay handlers. If you are a MACRO-only user, requiring only the overlay handlers in SYSLIB (and not the other routines), you can create a separate library containing only the overlay handlers by using the EXTRACT option with the LIBRARY command. You can name the new, smallerr library SYSLIB. Section 2.7.10 describes the procedure for creating such a library. Section 2.7.10 also describes adding the overlay handlers to another library (for example, FORLIB or the DIBOL library) when the other library already exists.

To add modules to SYSLIB from a file xxxxxx.OBJ, use the following command, in which you need both the /REMOVE and /INSERT options. /INSERT inserts the new library or module in the old one; /REMOVE removes the duplicate global, $OVRH, from the library directory. You must remove $OVRH from the library directory for SYSLIB to function properly. This global appears in both overlay handlers, OHANDL and VHANDL, because VHANDL includes the program code found in OHANDL. VHANDL processes both unmapped and virtual overlays, while OHANDL processes only unmapped overlays.

```
.LIBRARY/INSERT/REMOVE SYSLIB.OBJ xxxxxx.OBJ(RET)
Global? $OVRH(RET)
Global? $ERRS(RET)
Global? $ERRTB(RET)
Global? (RET)
.
```

Note that you must also remove the globals $ERRS and $ERRTB from the library directory if you have included the FORTRAN library in SYSLIB.

Refer to the *RT–11/RSTS/E FORTRAN IV User's Guide* for instructions on creating in-line code versions of GETSTR and PUTSTR and replacing threaded code modules in a library with in-line code modules.

2.3.6 Exercises

If you intend to perform the exercises in *Introduction to RT–11*, you need the following components on the system volume:

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS or LS.SYS (if appropriate)
Other handlers if appropriate
DIR.SAV
PIP.SAV
DUP.SAV
LINK.SAV
EDIT.SAV
SRCCOM.SAV
RESORC.SAV
LIBR.SAV
MACRO.SAV
CREF.SAV
SYSMAC.SML
ODT.OBJ
DEMOED.TXT
DEMOX1.MAC
DEMOBG.MAC
DEMOFG.MAC

To do FORTRAN IV exercises in *Introduction to RT–11*, you also need:

FORTRA.SAV
SYSLIB.OBJ (with FORLIB.OBJ included in it if possible)
FORLIB.OBJ (if not included in SYSLIB.OBJ)
DEMOF1.FOR

To do BASIC–11 exercises in *Introduction to RT–11*, you also need:

BASIC.SAV

If the system volume is a single-density diskette, you will not have room for all components on the system volume. See Section 2.4.5 for the file arrangement you should use.

2.3.7 Help Package

Your distribution kit includes three help files: HELP.SAV, HELP.TXT, and HELP.EXE. HELP.EXE is the help program. To run, the HELP program requires the help text file, HELP.TXT. HELP.SAV consists of HELP.EXE and HELP.MLB (resulting file after the librarian, LIBR, processes HELP.TXT) merged into a ready-to-use utility. Unless you want to change the help text, HELP.SAV is the only file you need. You can delete the other help files from the working system.

Section 2.7.14 describes how to customize the help text to your specific needs.

2.3.8 Line Printer Handlers

The software kit includes the line printer handler, LP.SYS, and the serial printer handler, LS.SYS. If your hardware configuration includes a serial printer instead of the usual line printer, you should include only LS.SYS in your working system.

You can use the serial printer in the same way you use a line printer. On your working system volume, rename the original LP.SYS file to something else in order to save it. Then, rename LS.SYS to LP.SYS.

```
.RENAME LS.SYS LP.SYS(RET)
.BOOT SY:(RET)
```

You must reboot the system after renaming the handler so the proper handler will be installed in the device table. Then, when you use the PRINT command, the system sends output to the serial printer. Since you will store both the distribution medium and the backup volume(s) you make during the installation process, you will always have copies of both handlers.

If your serial printer is installed at nonstandard vector and control status register addresses, you can use the SET command to change the addresses. (See Section 2.7.11.1.) Note, however, that once you have renamed LS.SYS to LP.SYS, you must use the device name LP: in a SET command. The available SET options would still be those for the LS handler.

2.3.9 MACRO–11 Assembler

If you intend to use the MACRO–11 assembler, you need the files MACRO.SAV and SYSMAC.SML (the system macro library) on the system volume. See the *RT–11 System Utilities Manual* for a description of the assembly process.

2.3.10 Queue Package

The queue package runs with the FB or XM monitors only. It sends files to any RT–11 device but is particularly useful in queuing files for printing.

To use the queue package, you need the files QUEUE.REL (which queues and prints the files you specify) and QUEMAN.SAV (which processes command lines and sends the information to QUEUE.REL). When you run QUEUE, it creates a temporary work file (QUFILE.WRK) on the system disk, which contains the queue of files.

QUEUE runs as a foreground job. If you want to use QUEUE as a system job under FB, you must perform the system generation process to generate support for system jobs under the FB monitor. Refer to the *RT–11 System Utilities Manual* for more information about the queue package; refer to the *RT–11 System Generation Guide* for information about the system generation process. Section 2.7.8 in this manual describes how to increase the size of the queue work files.

2.3.11 Source Files

Normally, you do not need any source files in your working system except demonstration sources. Otherwise, the source files (file type .MAC) are required only for the system generation process.

You need the demonstration sources in the working system to perform the exercises in Chapters 3 through 9 of this manual and in *Introduction to RT–11*. Once you have finished these exercises, the demonstration source files are no longer useful and can be deleted from your working disk.

2.3.12 Start-Up Command File

The standard RT–11 monitors look for a start-up command file (STARTx.COM) whenever you bootstrap the system. If they find one, the monitors execute its commands (to print a message, assign devices, and so on). If the monitors do not find a start-up command file, they print a message indicating that the start-up file has not been found. You do not need a start-up command file unless you want one; you can delete the distributed start-up files or create your own. Refer to the *RT–11 System User's Guide* for more information about start-up command files.

2.3.13 Text Editors

RT–11 gives you a choice of three text editors. EDIT is a character-oriented text editor that you can use with either a hard-copy terminal or a video terminal. The Keypad Editor (KED or K52) is an easy-to-use, cursor-oriented editor that you can use only on a VT100-compatible or VT52 video terminal. TECO is a powerful (although sometimes difficult-to-use) editor that DIGITAL includes in the software kit but does not support.

Generally, you need only one editor in your working system, although you may want to try out one or two of these editors before choosing the one you prefer. Refer to the *RT–11 System User's Guide*, the *PDP–11 Keypad Editor User's Guide*, and the *PDP–11 TECO User's Guide* for descriptions of these editors.

You need the file EDIT.SAV if you choose EDIT as your text editor.

If you choose the keypad editor, the file you need depends on the type of terminal you have. You need the file KED.SAV if your terminal is a VT1xx; you need K52.SAV if your terminal is a VT52.

If you choose TECO, you need TECO.SAV.

2.3.14 Utility Programs

You do not need any system utility programs you do not intend to use, but remember that most of the keyboard monitor commands need certain system programs in order to work. For example, PIP, DUP, and DIR are necessary for most of the keyboard commands to function. Appendix A of the *RT–11 System User's Guide* summarizes the keyboard monitor commands and lists the system utility programs that each command requires.

Refer to the *RT–11 System Utilities Manual* for descriptions of the functions of the utility programs. If your system device is a large disk, you will probably find it useful to have most of the utility programs on your working system device. On the other hand, if you have a small system device, you will not have room for all the utilities.

In addition to PIP, DUP, and DIR, you probably need to use an editor, LINK, FORMAT (for certain devices), and HELP frequently, and you may want them on the system volume. However, the utilities you use less frequently can occupy a second volume, which you can insert in and run from Unit 1.

2.4 Planning the Arrangement of Components

Planning the arrangement of components will enable you to use your system most efficiently. If your system volume is one of the smaller devices, this planning is especially important. Therefore, consider the following suggestions before you build a working system; they can help you create an efficient system.

1. Assign the default device to the data device.
2. Create a separate utilities volume.
3. Create bad blocks on TU58 DECtape II cartridges to avoid excessive rewinds.

4. Create several system volumes.
5. Create volumes for use with *Introduction to RT–11*.
6. Limit components on the system volume.

More details on each of these suggestions follow. (You can use the worksheet at the end of this chapter, Figure 2–2, to record the arrangement as you plan it.)

2.4.1 Assigning the Default Device to the Data Device

Once you have installed the system and it is running, you can assign the default device DK: to the data device (Unit 1). Most temporary files, data files, and others default to the second unit, minimizing demand for system device capacity.

The command to assign the default device to Unit 1 is as follows:

```
.ASSIGN xx1: DK:(RET)
```

You can include this command in your start-up command file to assign the default device to xx1: whenever you boot the system. The procedures in this manual assume DK: is the system device unless indicated otherwise. (See the *RT–11 System User's Guide.*)

Be sure to make any adjustments in your procedures if you assign the default to the data device.

2.4.2 Creating a Separate Utilities Volume

Create a separate utilities volume for the utility programs you expect to use infrequently. This technique will provide you with a system volume containing all the components necessary to execute the majority of keyboard commands and perform common program preparation functions. When you need a seldom-used utility, you can insert the utilities volume in Unit 1 instead of the data volume. You can then run a non-overlaid utility directly from the utilities volume (or you can copy the utility temporarily to the system volume).

NOTE

> The PIP and DUP utilities must always reside on the system volume.

To run a non-overlaid utility from the utilities volume, use the following commands, where xx is the physical device name, and aaaaaa is the utility program's name.

```
.RUN xx1:aaaaaa(RET)
*
```

Replace the utilities volume in Unit 1 with the data volume, and issue the appropriate commands to the utility.

```
*(CTRL/C)
.
```

However, if you run an overlaid utility from Unit 1, the volume containing that utility must remain in Unit 1 at all times. Therefore, you should generally include the overlaid utilities on your system volume. The overlaid components are PIP, DUP, MACRO, LINK, LIBR, KED, K52, FORMAT, HELP, IND, BUP, and TECO.

ODT.OBJ is also useful on the system volume to debug MACRO–11 programs.

An example of this kind of arrangement of volumes into a system volume and a utilities volume follows:

System Volume	Utilities Volume
SWAP.SYS	FILEX.SAV
RT11FB.SYS	SIPP.SAV
DX.SYS	BINCOM.SAV
LP.SYS	SRCCOM.SAV
IND.SAV	DUMP.SAV
PIP.SAV	RESORC.SAV
DUP.SAV	PAT.SAV
DIR.SAV	FORMAT.SAV
KED.SAV	BUP.SAV
MACRO.SAV	
LINK.SAV	
LIBR.SAV	
ODT.OBJ	
HELP.SAV — required for the HELP command	

2.4.3 Creating Bad Blocks on TU58 DECtape II Cartridges

If your volumes are TU58 DECtape II cartridges, you may encounter slow response time due to excessive rewinds of the tape. Because of the way DECtape II stores data in records, you can actually improve system performance by creating dummy bad blocks in strategic locations. DECtape II writes data records in a specific sequence and pattern; to write an entire cartridge, for example, it performs the following actions.

1. Writes alternate data records on the first track.
2. Rewinds to return to the beginning-of-tape (BOT) mark.
3. Writes data records skipped on the first pass of the first track.
4. Rewinds the tape.
5. Writes alternate data records on the second track.
6. Rewinds the tape.
7. Writes data records skipped on the first pass of the second track.

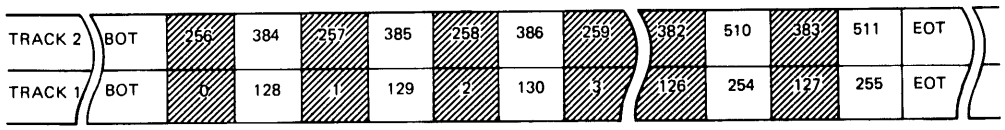
Performance degradation occurs when a file (particularly a monitor file) overlaps from the end-of-tape to the beginning-of-tape; for example, it extends from the last portion of the second pass on track 1 to the first portion of the first pass on track 2.

You can create dummy bad blocks in three locations:

- Block 128 (decimal), at the beginning of the second pass on track 1
- Block 256 (decimal), at the beginning of the first pass on track 2
- Block 384 (decimal), at the beginning of the second pass on track 2

This arrangement prevents the system from writing across rewinds, since RT–11 requires contiguous free space in which to write files. However, this technique prevents you from creating any file over 127 (decimal) blocks long and also increases fragmentation. Figure 2–1 illustrates the location of blocks on tape.

Figure 2–1: Block Locations on DECtape II



To create these dummy bad blocks, insert an initialized blank volume (write-enabled) in Unit 1 and type the following commands:

```
.CREATE/START:128. xx1:FIL1.BAD(RET)
.CREATE/START:256. xx1:FIL2.BAD(RET)
.CREATE/START:384. xx1:FIL3.BAD(RET)
.
```

Repeat this procedure on all the cartridges for your working system. Then, when you build your system, use the volumes on which you have created these bad blocks.

NOTE

If you create these dummy bad blocks, you should consider them a permanent part of the cartridge (unless you reinitialize it). You can use the DELETE command to remove dummy bad blocks only if you have not compressed the cartridge with the SQUEEZE command. SQUEEZE renames bad blocks in such a way that you cannot type the file name to delete the file.

2.4.4 Creating Several System Volumes

Create several system volumes, each devoted to a particular function. You can then change the system volume as normal job flow changes the functions you need. To change system volumes, wait for a logical stopping point in the job flow; do not arbitrarily remove the system volume in the middle of an operation. Be sure to copy the bootstrap record to each system volume.

2.4.5 Creating Volumes for Use with *Introduction to RT–11*

If you intend to perform the exercises in *Introduction to RT–11*, you need certain components on your working system. If your system device is one of the small devices (RX01 or RX02), you need to build four volumes, which are listed below with the files they contain. Be sure to copy the bootstrap to each volume.

System Volume

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS or LS.SYS (if appropriate)
PIP.SAV
DUP.SAV
DIR.SAV
LINK.SAV
EDIT.SAV
SRCCOM.SAV
RESORC.SAV
LIBR.SAV
IND.SAV
ODT.OBJ
MACRO.SAV
SYSMAC.SML
CREF.SAV
DEMOED.TXT
DEMOF1.FOR
DEMOX1.MAC
DEMOBG.MAC
DEMOFG.MAC

FORTRAN IV Language Volume

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS or LS.SYS (if appropriate)
PIP.SAV
DUP.SAV
DIR.SAV
EDIT.SAV
FORTRA.SAV
DEMOF1.FOR

LINK Volume

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS or LS.SYS (if appropriate)
PIP.SAV
DUP.SAV
DIR.SAV
LINK.SAV
SYSLIB.OBJ (including FORLIB.OBJ)
FORLIB.OBJ (if not included in SYSLIB.OBJ)

BASIC–11 Language Volume

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS or LS.SYS (if appropriate)
PIP.SAV
DUP.SAV
DIR.SAV
LINK.SAV
EDIT.SAV
BASIC.SAV
SYSLIB.OBJ

2.4.6 Limiting Components on the System Volume

Limit the system volume to necessary and frequently used system components. Place on the system volume only one monitor file and only the handler files for devices in your configuration. For example, a typical RX01 system volume need have only:

SWAP.SYS
RT11FB.SYS
DX.SYS (RX01 device handler)
LP.SYS (if your system includes a line printer)

The device handler is not included in the monitor file, so you must include the system device handler on any system volume. If you use the BL monitor or the SJ monitor without multiterminal support, you do need TT.SYS, because the terminal service is not resident in those monitors.

However, if you intend to use the keyboard monitor commands, you need to include at least the utility programs PIP, DUP, and DIR, which are necessary for most of the keyboard commands to function. The programs EDIT, KED, LINK, and HELP are also used frequently.

2.5 Installing Software Updates

To make sure that RT–11 operates correctly, you must install software updates. Software updates are critical to system or component operation.

Software updates are performed to provide normal maintenance; for example, to correct documentation or software errors detected since the last software release. These updates also introduce new software functions that are not described in the documentation; for example, an update would be required to support new hardware, to release a bundled layered product, or to support a released layered product.

Software updates are distributed, not as source modules or binary patches, but as complete software module replacements. For example, if a software error exists in the PIP utility program, the update kit will contain the complete PIP software module. The correction will already have been installed and fully tested. Your only requirement is to copy the new module from the distribution medium to your system, replacing the old software module with the new. After completing the update, you may be required to perform part of the installation procedure, for example, to rebuild your working system.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to the problems until the module in error is replaced.

Each software update kit includes:

- A letter from Software Product Services
- Release notes — identify the contents of the update kit, list the known problems that the kit fixes, and provide any special information about the update.
- User documentation — provides instructions on how to update the system.
- Distribution medium — contains new software modules and IND procedures to automate the updating process.

The distribution medium is the same as the medium on which you received the base system. It contains modules for both base system and layered products.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

2.6 Acquiring Sufficient Media

When you perform the installation procedures, you need blank media on which to make backup copies of the distribution, create a working system, and make backup copies of the working system. For example, if you receive RT–11 on nine RX01 diskettes, you need at least 11 blank diskettes, for the following purposes.

9	to back up distribution
1 (or more)	for working system
1 (an equal number)	to back up working system
?	for your application's data storage requirements
11+	total

Before you begin installation, read over the installation chapter that pertains to your configuration and establish how many cartridges, diskettes, disks, or magtapes you need. Then you can secure additional media, if required.

2.7 Choosing Software Customizations

Although the RT–11 components as distributed need no alteration for most applications, you can alter them in some ways. Many alterations require your going through the system generation process. Others require less time-consuming procedures, such as modifying the distributed monitors. Compare Tables 1–3 and 1–4, and read the following sections to establish what you need to do. Identify any non-system-generation customizations you need to make, and record them on the worksheet at the end of this chapter (Figure 2–2). You can perform the procedure for each selected customization during the installation process (Chapters 3 through 9). System generation is necessary only if you need a customization that cannot be achieved with one of the procedures described in the following sections.

NOTE

> Refer to Appendix D in the *RT–11 System Generation Guide* for additional modifications that further customize specially generated monitors.

DIGITAL strongly recommends that you use the SIPP utility to install software customizations. Also, use the feature of SIPP that creates an indirect command file when it installs the customization. In this way, you can store copies of the indirect command files, so that you can easily install the customizations again if necessary. When you invoke SIPP and it responds with an asterisk, enter the following:

```
.RUN SIPP(RET)
*filnam.COM=filnam.typ/L(RET)
```

The equal sign and file name cause SIPP to create the indirect file, and /L causes SIPP not to customize the input file. You can use the indirect file to customize whenever necessary.

In order to customize your software, you must:

1. Determine which customization(s) you wish to make.
2. Locate the symbol of the software component you wish to modify. The values of symbols for monitor customizations may be obtained from the link maps distributed with RT–11 as the files RTBL.MAP, RTSJ.MAP, RTFB.MAP, and RTXM.MAP. The values of symbols for utility customizations may be found in the file CUSTOM.TXT.
3. Follow the instructions to modify the software component, substituting the actual address value of the symbol in place of the symbol in the update.

NOTE

> In the software customizations reproduced in this manual, lowercase alphabetic x represents unknown characters. These characters vary according to the specific software component. All numeric input values are octal.

2.7.1 Changing Characters That Indicate Insertion/Deletion

You can modify SRCCOM to change the default characters that SRCCOM uses to indicate insertions and deletions on listings. Normally, when you use the DIFFERENCES/CHANGEBAR command or SRCCOM's /D option to compare two files, SRCCOM places vertical bars next to each line that has been added to the new file and bullets (lowercase alphabetic o) next to lines that have been deleted. If you want to use characters other than the vertical bar and bullet characters, you can modify SRCCOM.

In the following customization, n is the ASCII code for the character you want to use to indicate insertions, and m is the ASCII code for the character you want to use to indicate deletions.

```
.RUN SIPP(RET)
*SRCCOM.SAV(RET)
Base?      0(RET)
Offset?    1000(RET)
     Base      Offset       Old       New?
      000000    001000    076157      ;A(RET)
      000000    001000       <o>      ;Am(RET)
      000000    001001       <|>      ;An(RET)
      000000    001002      <^C>      (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.2 Changing Default Output Device from Line Printer to Terminal

If your configuration does not have a line printer, you can cause monitor commands to default to the terminal (TT) instead of the line printer (LP). Since several monitor commands default the output device to LP, you should edit the start-up command file to cause all system references to the device LP: to use the terminal. To change the defaults of such commands (for example, DUMP and PRINT), you need to add an ASSIGN TT: LP: command to the start-up file. Then, every time you bootstrap the system, the reassignment of the default device takes place.

2.7.3 Changing Location of VT11/VS60 Floating Vectors

Under certain circumstances, you may need to change the VT11/VS60 vector address. VT11/VS60 display processor vectors are normally located at 320 to 332. However, the floating vector region on the PDP–11 is situated in locations 300 to 476. Therefore, you may have to move the VT11/VS60 vectors if you add other devices.

If the vectors for your VT11/VS60 change, install the following customization to modify the monitor for a different VT11/VS60 vector address. Once you have made this change, all DIGITAL supplied software that accesses the display will function properly on the system without further alterations.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, $RMON is the value of that symbol from the appropriate monitor link map, and nnn is the location of the first VT11/VS60 vector on your system. Find the monitor link map for the monitor you want to alter, and use the value of $RMON in the update. Note that nnn must be an even octal value between 70 and 464.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?     $RMON(RET)
Offset?   354(RET)

      Base      Offset      Old       New?
       $RMON     000354   000320     nnn(RET)
       $RMON     000356   000000     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.4 Changing the Number of Directory Columns

You can modify DIR to change the number of columns in the directory listing that prints when you use the DIRECTORY command. Normally, the directory contains two sets of three columns, with each set listing file names and types, file size in blocks, and date of creation. If you use the /FAST or /BRIEF option with DIRECTORY, DIR lists only file names and types in five columns. You can also use the /COLUMNS:n option to specify the number of columns in the directory listing. However, if you want to change the default number of columns in the directory listing, install the following change.

In the customization, ffffff is an octal number (in the range 1 to 11) for the number of directory columns to be displayed when you use the DIRECTORY/FAST command. The value nnnnnn is an octal number (in the range 1 to 11) for the number of columns to be displayed when you use the DIRECTORY command.

```
.RUN SIPP(RET)
*DIR.SAV(RET)
Base?      0(RET)
Offset?    1000(RET)

      Base      Offset      Old       New?
      000000    001000    000005     ffffff(RET)
      000000    001002    000002     nnnnnn(RET)
      000000    001004    000005     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.5 Changing the Default Order of Directory Listings

You can change the default order in which directory entries are listed in a directory listing. The current default order is by position in the directory.

In the following customization, SWS is the value of that symbol from the CUSTOM.TXT file. The variable nnnnnn represents the address of the value 6174 that SIPP returns in response to the search command. The variable yyyyyy represents the contents of nnnnnn. The variable zzzzzz requests the contents of SWS + 2. The variable xxx represents one of the following default ordering options:

DAT	Order by date (earliest to latest)
NAM	Alphabetical order by file name
POS	Order by position in directory (current default)
SIZ	Order by size (smallest to largest)
TYP	Alphabetical order by file type

```
.R SIPP(RET)
*DIR.SAV(RET)
Base?      0(RET)
Offset?   ;S(RET)
Search for?        SWS(RET)
Start?     (RET)
End?       (RET)
Found at nnnnnn
Offset?  nnnnnn(RET)

         Base          Offset       Old     New?
          000000        nnnnnn        SWS    SWS+2(RET)
          000000        nnnnnn+2   yyyyyy      (CTRL/Z)(RET)
       Offset?   SWS(RET)

         Base          Offset       Old     New?
          000000        SWS       000000     ;Rxxx(RET)
          000000        SWS+2     zzzzzz     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.6 Changing the Number of /Q Program Sections LINK Allows

You can alter LINK to change the number of absolute base address p-sects (/Q p-sects) that LINK allows. Normally, the /Q option to LINK lets you specify the absolute base addresses of up to eight p-sects in your program. You need absolute base address p-sects to prepare programs in absolute loading format for use in read only memory (ROM) storage. Refer to the *RT–11 System Utilities Manual* for more information about absolute base address p-sects and about LINK in general.

The limit of eight such p-sects, however, is the default number, and you can change it by altering LINK. LINK uses the number of p-sects to set up the /Q buffer area and to establish how many times it should ask the question:

```
Load section: address?
```

Note, however, that LINK allocates the buffer space even if you do not use the /Q option when you perform the link. LINK calculates the size of the buffer to be three times the contents of QSWNUM.

To change the number of /Q p-sects LINK allows, use SIPP to change LINK.SAV as follows. QSWNUM is the value of that symbol from the CUSTOM.TXT file. The number nnn must be in the range 1 to 177 (octal); it represents the number of p-sects you want.

```
.RUN SIPP(RET)
*LINK.SAV(RET)
Segment? 0(RET)
Base?    0(RET)
Offset?  QSWNUM(RET)
Segment    Base       Offset        Old    New?
000000     000000     QSWNUM         ?     nnn(RET)
000000     000000     QSWNUM+2       ?     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.7 Changing the Size of LINK's Library Module List

You can modify LINK to change the default size of LINK's list of library modules. LINK creates a list of 252 (octal) modules to be included from libraries during the link operation. Because the size of each entry in this list is larger in RT–11 Version 5 than in previous versions, the list may not be large enough for your application. You can use the LINK /P option at link time to increase the size of this list. If you want to change the default size of the list (and avoid using the /P option), you can modify the linker. You can still override the new default at link time by using /P.

Note that if you increase the default size of the list, the maximum number of global symbols allowed in the link will be reduced.

NOTE

> You must install this change if you use DIBOL. Make the default number of modules LINK holds 400 (octal) or greater.

In the following customization, LMLSIZ is the value of that symbol from the CUSTOM.TXT file, and nnnnnn is the number of modules the list should hold.

```
.RUN SIPP(RET)
*LINK.SAV(RET)
Segment? 0(RET)
Base?    0(RET)
Offset?  LMLSIZ(RET)

Segment     Base      Offset      Old   New?
000000     000000    LMLSIZ        ?    nnnnnn(RET)
000000     000000    LMLSIZ+2      ?    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.8 Changing the Size of the QUEUE Work File

QUEUE, the device queue foreground program (or system job), uses a work file five blocks long. This work file allows you to queue approximately 127 (decimal) files at once. If your application requires larger queues, you can modify QUEUE.REL to change the default size of the work file.

In the following customization, QSIZE and QCBLK are the value of those symbols from the CUSTOM.TXT file, and nnn is the default size of the work file in octal blocks. To compute the approximate size of the work file that would be required for the number of files you need queued at once, use the formula:

nnn = (max no. of file specs in queue at one time + 1)/32 + 1

```
.RUN SIPP(RET)
*QUEUE.REL/A(RET)
Base?     0(RET)
Offset?   QSIZE(RET)

      Base     Offset     Old    New
      000000   QSIZE     000005  nnn(RET)
      000000   QSIZE+2   000000  (CTRL/Z)(RET)

Offset?  QCBLK(RET)

      Base     Offset     Old    New?
      000000   QCBLK     000004  (nnn-1)(RET)
      000000   QCBLK+2   000200  (nnn-1)*32(RET)
      000000   QCBLK+4   000000  (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.9 Modifying EDIT

To customize the editor, EDIT, you can reduce the size of the editor's text window and you can cause the editor to operate correctly on terminals with nonstandard ESCAPE codes.

2.7.9.1 Size of the Text Window — If your configuration includes a VT11/VS60 display processor, you may need to reduce the size of the editor's text window to correct an overflow problem. The editor works in such a way that when you use a VT11 or VS60, the window into the buffer and the scrolled command lines are separate "pictures". On rare occasions, if the text window around the cursor contains long lines and several line feed (or form feed) characters, the window may overflow onto the scrolled editing commands, making that portion of the screen difficult to read. While this problem does not usually occur, if it does, you can make the obscure lines clear by advancing the cursor several lines.

However, if the problem is troublesome for your particular application, you can remove it by reducing the size of the display window. Use the following customization to make this change, where DSARG is the value of that symbol from CUSTOM.TXT file, and nnn is the number of lines to be displayed above and below the cursor. To eliminate the problem, make nnn smaller than 12 (octal).

```
.RUN SIPP(RET)
*EDIT.SAV(RET)
Base?      0(RET)
Offset?    DSARG(RET)

      Base          Offset        Old    New?
       000000        DSARG     000012   nnn(RET)
       000000        DSARG+2   010200   (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.9.2 Terminals with Nonstandard ESCAPE Codes — You can modify the editor to allow it to operate correctly on terminals with nonstandard ESCAPE codes. Certain older terminals generate 175 or 176 (octal), rather than the standard 33 (octal), when you type the ESCAPE or ALTMODE key. Because codes 175 and 176 represent legitimate characters on more modern terminals, EDIT does not recognize ESCAPE code as the command terminator in the older terminals.

If you have an older terminal, you can correct the problem by making the following change, so that you can use the ESCAPE code as documented in the EDIT chapter of the *RT–11 System Utilities Manual.* In the change, ALTMDE is the value of that symbol from the CUSTUM.TXT file, and nnn represents the octal code that your terminal generates when you type the ESCAPE key on it.

```
.RUN SIPP(RET)
*EDIT.SAV(RET)
Base?      0(RET)
Offset?    ALTMDE(RET)
```

```
      Base       Offset      Old        New
       000000     ALTMDE  xxxxxx        (RET)

      Base       Offset      Old        New?
       000000     ALTMDE      033       nnn(RET)
       000000     ALTMDE+1    016       (CTRL/Y)(RET)
*(CTRL/C)
.
```

The character $ echoes on the terminal, regardless of the octal value used for the ESCAPE code. However, EDIT recognizes only the ESCAPE code you specify, not both.

2.7.10 Extracting the Overlay Handlers from SYSLIB

You can extract the overlay handlers from the default system library, SYSLIB, if those are the only components of SYSLIB you need. Remember that you need the overlay handlers that are included in SYSLIB if you intend to use overlaid programs.

Create a separate library containing only the overlay handlers by using the /EXTRACT option to the LIBRARY command, as described in the following procedure:

1. Extract the unmapped overlay handler (for LINK/O overlays) from SYSLIB:

```
.LIBRARY/EXTRACT SYSLIB OHANDL(RET)
Global? $OVRH(RET)
Global? (RET)
.
```

2. Extract the virtual overlay handler (for LINK/V overlays) from SYSLIB:

```
.LIBRARY/EXTRACT SYSLIB VHANDL(RET)
Global? $OVRHV(RET)
Global? (RET)
.
```

3. Combine the files you extracted in a new library (which can have any name, including SYSLIB). You must combine the files in the indicated order:

```
.LIBRARY/REMOVE/CREATE newlib VHANDL,OHANDL(RET)
Global? $OVRH(RET)
Global? (RET)
.
```

   In the command, you need both the /REMOVE and /CREATE options. /CREATE creates the new library, and /REMOVE removes a duplicate global, $OVRH, from the library directory. This global appears in both OHANDL and VHANDL, because VHANDL includes the program code found in OHANDL. Remember, VHANDL processes both unmapped and virtual overlays.

You can also put either handler in a library by itself. OHANDL handles only unmapped overlays. VHANDL handles both unmapped and extended memory overlays, but it is larger than OHANDL.

In the following command, which creates a library from one of the overlay handlers, x is O for OHANDL or V for VHANDL.

```
.LIBRARY/CREATE newlib xHANDL(RET)
.
```

To add the overlay handlers to another library or module (for example, FORLIB or the DIBOL library), you can combine the distributed SYS-LIB.OBJ with the library or module. Use the LIBR utility, but remove the global $OVRH from the new library. You must remove this global if you use as input to LIBR any library that includes both overlay handlers (as does the distributed SYSLIB). In the following command to combine libraries, the /REMOVE option removes $OVRH. Note that newfile is a single module or a library, and gbl is any global that must be removed from newfile. If you combine FORLIB with SYSLIB, you must also remove the FORTRAN IV globals. (Refer to the FORTRAN library generation procedures.)

```
.LIBRARY/REMOVE/INCLUDE SYSLIB newfile(RET)
Global?   $OVRH(RET)
Global?   gbl(RET)
Global?   (RET)
.
```

2.7.11 Installing Other Devices

You may need to install device handlers that are available but are not installed in the standard monitors. Installing a device handler adds information to the monitor device tables, so that you can use the device. Many devices are available in the standard monitors. (Refer to the *RT–11 System User's Guide* for a list of available devices.) You can perform the system generation process to create monitors and handlers that support non-standard devices. The *RT–11 Software Support Manual* describes how to write your own device handler.

When you bootstrap RT–11, the bootstrap routine locates the system device handler and installs it. Then the bootstrap looks at the rest of the device handler files on the system device and tries to install the device handler for each device it finds in the configuration. It does not try to install any handlers for which there is no hardware. If there are more handlers than device slots, the bootstrap uses a certain priority scheme to establish which handlers to install. Refer to the *RT–11 Software Support Manual* for a description of these priorities.

To ascertain which device handlers have been installed, use the keyboard monitor SHOW command, which shows you which devices are installed and whether any empty device slots are available.

```
.SHOW(RET)
TT
RK   (Resident)
     RK0 = SY , DK
LD
DX
DT
DD
CT
LS
PC
BA
NL
14 free slots
```

If the bootstrap did not install a device when you booted the system, it did not have enough device slots when it encountered the handler, the hardware was not present, or the device handler was not present. To install a device ensure that the correct handler is on the system device and that the hardware is present. If there are no free slots, use the REMOVE command to remove an unneeded device and the INSTALL command to install the device you need.

```
.REMOVE LS:(RET)
.INSTALL LP:(RET)
```

The standard (distributed) monitors provide a total of 16 (decimal) device slots. If your application requires more than 16 device slots simultaneously, you must perform the system generation process to create your own monitor and device handlers.

To control which handlers the bootstrap installs, place on the system device for your working system only the handlers for the devices you will be using. Do not include in your working system a handler for a device you will not be using.

You can keep a handler from being installed at boot time by giving it a name that does not correspond to the naming conventions for the monitor being booted. A device handler is named yy.SYS for SJ and FB monitors and yyX.SYS for XM monitors (where yy is the device name). Use the RENAME command to rename a handler.

The bootstrap cannot install support for some devices. Thus, different procedures are required. The following sections describe how you can change the control status register (CSR) and vector addresses for the line printer, serial printer, DECtape II, RX01 and RX02 diskettes, and MSCP disks, and how you can install hardware magtape support, set magtape parity and density, and change RP02 support to RP03.

2.7.11.1 CSR and Vector Addresses for Line Printers, DECtape II, and MSCP Disks — With the SET command, you can change the CSR and vector addresses of six devices: line printer, serial printer, DECtape II, RX01 and RX02 diskettes, and MSCP disks. You need to change the addresses if the controller is installed at nonstandard addresses. When using the SET command, enter the following commands, where dd is the device mnemonic (LP, LS, DD, DX, DY, or DU), aaaaaa is the CSR address, and bbb is the vector address.

```
.SET dd: CSR2=aaaaaa(RET)
.SET dd: VEC2=bbb(RET)

.SET dd: CSR=aaaaaa(RET)
.SET dd: VECTOR=bbb(RET)
```

In addition, the following commands are valid for MSCP disks:

```
.SET DU: CSR3=aaaaaa(RET)
.SET DU: VEC3=bbb(RET)

.SET DU: CSR4=aaaaaa(RET)
.SET DU: VEC4=bbb(RET)
```

You can also partition MSCP disks and assign unit numbers to the ports. Use the following commands:

```
.SET DUn: PART=x(RET)
```

where n = unit number
x = partition number

```
.SET DUn: PORT=x(RET)
```

where n = unit number
x = port number

See the description of the SET command in the *RT–11 System User's Guide* and the *RT–11 Software Support Manual* for more information on partitioning MSCP disks and assigning ports.

These commands permanently alter the handler .SYS file. That is, the CSR and vector addresses you specify remain in effect even if the system is rebooted. If you want to change the addresses again, you can use the SET command again.

2.7.11.2 Hardware Magtape Support — You can choose to install hardware magtape support, rather than use file-structured magtape support. File-structured handlers include hardware magtape handler features; however, hardware magtape handlers are smaller.

File-structured magtape handlers are distributed as part of the RT–11 operating system; hardware magtape handlers are not. If you want to use hardware magtape handlers, you must perform a system generation, using a new disk for output. System generation will produce either file-structured or hardware magtape handlers, but not both at the same time.

The files MT.SYS, MTX.SYS, MM.SYS, MMX.SYS, MS.SYS, and MSX.SYS are distributed file-structured handlers for TM11, TJU16, and TS11 magtape devices. The files MTHD.SYS, MTHDX.SYS, MMHD.SYS, MMHDX.SYS, MSHD.SYS, and MSHDX.SYS are the system-generated hardware magtape handlers for the same devices. There are two sets of handlers: one for SJ and FB monitors; the other for XM monitors. The file-name suffix X indicates device support for XM monitors.

There are two ways to install a hardware magtape handler in place of the file-structured handler:

1. Rename the distributed file-structured handler to save it, rename the corresponding hardware magtape handler to the original name of the file-structured handler, and reboot the system to let the bootstrap install the handler.
2. Use the INSTALL and REMOVE commands:
   - Make sure the hardware magtape handler for your monitor and device is on the system disk.
   - Rename the distributed file-structured handler to save it.
   - Remove the file-structured handler from the system volume.
   - Rename the corresponding hardware magtape handler to the name of the distributed file-structured handler.
   - Install the renamed hardware magtape handler.

For example, to install TM11 hardware support in place of TM11 file-structured support, first make sure that the hardware magtape handler MTHD.SYS is on the system volume, and then do the following:

1. Rename and remove the file-structured handler. (You must remove the old handler before you can install the new one.)

```
.RENAME MT.SYS MT.FIL(RET)
.REMOVE MT:(RET)
.
```

2. Rename and install the hardware magtape handler.

```
.RENAME MTHD.SYS MT.SYS(RET)
.INSTALL MT:(RET)
.
```

The handler you have named MT.SYS will be installed in the device table when the system is bootstrapped.

Table 2–3 identifies hardware handlers that you must rename, if you are using an SJ or FB monitor. Make sure you save the distributed (file-structured) handler.

Table 2–3: File-Structured and Hardware Magtape Handlers for SJ and FB Monitors

Device	File-Structured Handler	Hardware Handler	Renamed Hardware Handler Name
TM11	MT.SYS	MTHD.SYS	MT.SYS
TJU16	MM.SYS	MMHD.SYS	MM.SYS
TS11	MS.SYS	MSHD.SYS	MS.SYS

Table 2–4 identifies hardware handlers that you must rename, if you are using an XM monitor. Make sure you save the distributed (file-structured) handler.

Table 2–4: File-Structured and Hardware Magtape Handlers for XM Monitors

Device	File-Structured Handler	Hardware Handler	Renamed Hardware Handler Name
TM11	MTX.SYS	MTHDX.SYS	MTX.SYS
TJU16	MMX.SYS	MMHDX.SYS	MMX.SYS
TS11	MSX.SYS	MSHDX.SYS	MSX.SYS

2.7.11.3 Magtape Parity and Density — If you need to operate TM11 or TJU16 magtape at parity or density settings different from standard support, you can use the monitor SET command as described in the *RT–11 System Utilities Manual* to set them. The distributed monitors support seven- and nine-track TM11 and TJU16 magtape at 800 bits/in and odd parity. You cannot set the density of TS11 magtape, since it is 1600 bits/in odd parity only.

2.7.11.4 RP03 Support — RT–11 can accommodate the 40000 (decimal) block RP02 cartridge as a single logical unit but not the RP03, because the RT–11 file structure can accommodate only a maximum of 65536 (decimal) blocks. Therefore, to use RP03, think of each RP03 drive as two logical units of 40000 blocks each. (A single RP03 looks like two RP02 drives to the system.) Access the cartridge on physical unit n as logical DPn+4; thus, refer to drive 0 as DP0: and DP4:, drive 1 as DP1: and DP5:, and so on. Each logical unit has its own directory and data space. The system can support up to four RP03s. You can mix RP02s and RP03s as long as the total number of units (physical drives) on the system does not exceed four. The system can support as many as eight RP02s.

2.7.12 Modifying BATCH to Save Space

To minimize space demands on your system device, you can modify BATCH to access certain system programs on DK: rather than on SY:. These customizations allow you to store certain system programs on DK: rather than on SY: and let BATCH access them there.

You can make any or all of the customizations. Modify BATCH for the system programs you need to remove from the system device. Copy the programs for which you install changes to the device on which you want them to reside; then, delete them from the system device (SY:). Finally, use the ASSIGN command to assign the logical name DK: to the device to which you copied the system programs. Then, run BATCH as usual.

The following change to BATCH makes DK: the default storage volume for one of the specified programs. Use the octal value of nnnnnn (from the table) that corresponds to the program you want to affect.

Program	**Octal Value of nnnnnn**
DIR	1367
MACRO	2007
FORTRAN	2020
LINK	2037
PIP	2052
BASIC	2071

```
.RUN SIPP(RET)
*BATCH.SAV(RET)
Base?    BASE(RET)
Offset?  nnnnnn(RET)
  Base        Offset       Old    New
    BASE       nnnnnn     xxxxxx  (RET)

  Base        Offset       Old    New
    BASE       nnnnnn       040   125(RET)
    BASE       nnnnnn+1     040   (CTRL/Y)(RET)
*(CTRL/C)
.
```

Once you assign DK: to a device other than SY:, the new device becomes the default input and output storage device for most system programs. You may need to modify BATCH jobs to reference certain files on SY: explicitly, since that is no longer the same device as DK:. You can keep .BAT and .CTL files on SY: by invoking BATCH as follows:

```
.RUN BATCH(RET)
*SY:myJob=SY:myJob(RET)
```

2.7.13 Modifying LINK to Change the Default SYSLIB Device

You can modify the linker to make it look for the default system library (SYSLIB.OBJ) on the device you choose instead of on the system device (SY:). This change may be useful if you have space problems on your system device, because you can then place SYSLIB on the device you have specified to LINK.

To change the device on which SYSLIB.OBJ resides, make the following change to LINK.SAV. In the customization, dev is the name of the device on which you want to place SYSLIB.

```
.RUN SIPP(RET)
*LINK.SAV(RET)
Segment? 0(RET)
Base?    0(RET)
Offset?  SYSLIB(RET)

Segment     Base      Offset      Old    New?
000000     000000     SYSLIB       ?     ;Rdev(RET)
000000     000000     SYSLIB+2     ?     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.14 Modifying the Help Text

To change the help text that prints when you use the HELP command, you must create your own help text file, process that file with LIBR, and copy the resulting library and the file HELP.EXE to the same volume.

The distribution kit includes three help files: HELP.SAV, HELP.TXT, and HELP.EXE. HELP.EXE is the help program. To run, the help program requires a help text file. DIGITAL supplies HELP.TXT, which is such a text file. HELP.SAV consists of HELP.EXE and HELP.MLB (the resulting file after the librarian, LIBR, processes HELP.TXT) merged into a ready-to-use utility.

HELP.SAV is the only file you need if you do not want to change the help text. However, if you do want to change the text that prints when you use the HELP command, you must perform the following procedure.

First, edit the file HELP.TXT in your working system. Make sure that this file (as well as the rest of the distribution) is safely backed up and the actual distribution media are stored away. Add any explanations your application requires and delete any explanations that do not apply to your application.

When you edit HELP.TXT, you must follow a specific format, as follows:

1. Give each topic in the file an alphabetic name.
2. The name you give must be unique within the first six characters.
3. Place each topic on a page, delimited by form feeds. (See the following example.)
4. Place topics in alphabetical order within the file.
5. Leave the dummy topic 999999 at the end of the file.

The following illustrates the format of a topic, properly formatted, on a page.

```
(FF)
.MACRO(TAB)TOPICNAME(RET)(LF)
TOPICNAME(TAB)ONE LINE DESCRIPTION OF THE TOPIC(RET)(LF)
(RET)(LF)
(SP)(SP)SUBTOPICNAME(RET)(LF)
(TAB)TEXT ABOUT THE SUBTOPIC(RET)(LF)
(TAB)MORE TEXT (ANY NUMBER OF LINES) (RET)(LF)
(SP)(SP)(SP)SUBTOPICITEMNAME(RET)(LF)
(TAB)DESCRIPTION OF SUBTOPIC ITEM (USUALLY ONLY FOR(RET)(LF)
(TAB)THE SUBTOPIC 'OPTION'), ANY NUMBER OF LINES(RET)(LF)
[ANOTHER SUBTOPIC, STARTING FROM THE BLANK LINE ABOVE]
.ENDM(RET)(LF)
(FF)
```

Now, create the help text library file by processing HELP.TXT with the librarian. Use the following command:

```
.LIBR/MACRO HELP.MLB HELP.TXT(RET)
.
```

You can leave HELP.MLB as a separate file or merge it with HELP.EXE.

To run HELP with a separate program and text library file or files, copy HELP.EXE to the system volume. When you copy HELP.EXE, name it HELP.SAV.

```
.COPY xxn:HELP.EXE SY:HELP.SAV(RET)
.
```

Make sure that the file HELP.MLB that you create is also on the system volume. You can back up HELP.EXE and HELP.TXT on another volume and delete them from the system volume. Then, to invoke HELP, type:

```
.HELP(RET)
```

If you leave HELP.MLB separate, you can alter it more easily in the future. However, if HELP.MLB is a separate file, the program runs more slowly. This problem is significant on systems that use a small device (for example, diskette or DECtape II) as the system device.

DIGITAL recommends that you merge the two files, since the program runs faster that way. Use the following command to merge the program and text library file or files.

```
.COPY HELP.(EXE+MLB) HELP.SAV(RET)
.
```

In this case too, you can back up HELP.EXE and HELP.TXT on another volume and delete them from the system volume. You can also delete the file HELP.MLB, since you can recreate it from the file HELP.TXT.

2.7.15 Preventing Fatal System Errors from Causing a Reset

Normally, the monitor performs a hard reset when a fatal system error occurs. The reset stops I/O transfers, minimizing the possibility that the error will corrupt media. In some cases, the cause of software errors might still be in memory, and the reset preserves the data, making it possible to analyze the error.

However, in rare cases, the reset may prevent diagnosis of hardware errors. If you prefer to suppress the reset, you can install the following change in the monitor, although doing so increases the risk of corrupting media. DIGITAL does not recommend using a monitor with this customization installed except for diagnostic purposes. Do not use such a monitor for normal operations.

In the customization, monitr.SYS is the name of the FB or XM monitor file you want to modify, and FATAL is the value of that symbol from the monitor link map.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?     0(RET)
Offset?   FATAL(RET)

     Base      Offset        Old     New?
      000000   FATAL       000005    240(RET)
      000000   FATAL+2     010127    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.16 Running RT–11 in Less Memory Than Is Available

If your application requires that RT–11 run in less memory than is available, you can make a customization that allows you to bootstrap the system to run in the lower 12K words or 8K words of a 16K word machine. The BL, SJ, and FB monitors have bootstraps that allow the system to run in less memory than is available. (This cannot be done for XM monitors, because the XM monitor requires 32K memory.) The distributed monitors automatically make use of all available memory, since most applications require that RT–11 do so. However, if your configuration includes a hardware switch register and your application requires less memory, you can make the following customization. In the customization, monitr.SYS is the name of the monitor file that you want to modify.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?     1000(RET)
Offset?   30(RET)

     Base      Offset        Old     New?
      001000    000030     000407    0(RET)
      001000    000032     013704    (CTRL/Y)(RET)
*(CTRL/C)
.
```

If this is the hardware bootable monitor, you must write a new system bootstrap with the COPY/BOOT command after you install this change.

Once you make the change to your monitor, a halt occurs whenever you boot that monitor file. While the system is halted, set the switch register to one of the values from the following table and press the CONTINUE switch; the bootstrap operation then completes for the memory size you specify.

Value	Size in Words
40000	8K
44000	9K
50000	10K
54000	11K
60000	12K
64000	13K
70000	14K
74000	15K
100000	16K

Value	Size in Words
104000	17K
110000	18K
114000	19K
120000	20K
124000	21K
130000	22K
134000	23K
140000	24K
144000	25K
150000	26K
154000	27K

If your configuration does not include a hardware switch register or if you always want the system to boot in a specified amount of memory without halting, you can make the following customization. In the customization, monitr.SYS is the name of the monitor file that you want to modify, and nnnnnn is a value from the preceding table.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      1000(RET)
Offset?    30(RET)

     Base      Offset      Old      New?
      001000    000030    000407    240(RET)
      001000    000032    013704    12704(RET)
      001000    000034    177570    nnnnnn(RET)
      001000    000036    042704    (CTRL/Y)(RET)
*(CTRL/C)
.
```

If this is the hardware bootable monitor, write a new system bootstrap with the COPY/BOOT command. Once you change your monitor, the system will boot in the memory you specify whenever you boot the customized monitor file.

To run RT–11 in exactly 28K words, use the customization in Section 2.7.21.

2.7.17 Forcing 22-Bit Addressing

When the XM monitor is bootstrapped, the type of processor being used determines whether or not 22-bit addressing should be enabled. On Q-bus processors that support the 22-bit addressing mode, for example, the PDP–11/23–PLUS, 22-bit addressing is automatically selected. It is not selected on other processors, such as the PDP–11/23. However, you may customize your monitor to force the 22-bit addressing mode.

In this customization, monitr.SYS is the name of the monitor you wish to modify, and ..FQ22 represents an address found in that monitor's link map. The value of xx in the new value must equal the value of yy in the old value.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      0(RET)
Offset?    ..FQ22(RET)
     Base      Offset       Old     New?
     000000    ..FQ22       10yy    4xx(RET)
     000000    ..FQ22+2   xxxxxx    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.18 Setting Upper Limit on a File Size

If your application requires an upper limit on the size of a file, you can install a customization that changes the maximum size RT–11 allocates in a general .ENTER request. On distributed monitors, the .ENTER programmed request allocates space in such a way that the maximum size of a file is either half the largest space available or the entire second largest space available, whichever is larger. For most applications, this scheme is satisfactory and should be left unchanged. However, if yours is an application that requires an upper limit, you should make the following change.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, $RMON is the value of that symbol from the monitor link map, and nnnnnn is the octal number of blocks that is to be the maximum file size for a general .ENTER.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      $RMON(RET)
Offset?    314(RET)

     Base       Offset    Old      New?
    $RMON       000314   177777    nnnnnn(RET)
    $RMON       000316   xxxxxx    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.19 Specifying 50-Cycle Instead of 60-Cycle Clock Rate

You can modify the monitor so that it causes the TIME command to base calculations on a 50-cycle clock rate rather than a 60-cycle rate, which is the standard rate in RT–11. The 50-cycle clock has specialized uses and is the common frequency in Europe. To alter the rate, you must modify the monitor so that bit 5 is set in the monitor configuration word.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, $RMON is the value of that symbol from the monitor link map, and the new value that you enter is the sum of the old value displayed by SIPP plus 40 (octal).

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      $RMON(RET)
Offset?    300(RET)

     Base      Offset       Old      New?
     $RMON      000300    nnnnnn     nnnnnn+40(RET) (You must add 40.)
     $RMON      000302    000000     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.20 Using CAPS–11 to Load Files

If you use CAPS–11 to load RT–11 files, you must modify the cassette handler. CAPS–11 cassette file headers differ from RT–11 cassette file headers, so that problems occur when you transfer files to cassette. When you load files with the CAPS–11 CABLDR or with the CTLOAD bootstrap, CABLDR and CTLOAD interpret the level byte in the file header as a header continuation byte. If the byte in the cassette header is nonzero, CAPS–11 ignores at least the first data record of the file, since it assumes that the record is an auxiliary header record. To avoid losing any data records, you must modify CT.SYS so that all header records contain a level byte of 0.

```
.RUN SIPP(RET)
*CT.SYS(RET)
Base?      1000(RET)
Offset?    3463(RET)

      Base      Offset      Old     New?
       001000    003463      001     0(RET)
       001000    003464      040     (CTRL/Y)(RET)
*(CTRL/C)
.
```

If you need to use the system generation process to build your own system, you can edit CT.MAC. Use any editor to create a file CT.SLP as follows. Use SLP to edit CT.MAC.

```
\
-/LEVEL:/,,
LEVEL:(TAB).BYTE(TAB)0
/
```

2.7.21 Setting Upper Limit on Memory Size

If your PDP–11 does not generate a bus timeout trap, when the running program accesses location 160000, the RT–11 bootstrap may assume that you have an LSI–11 with the MSV11–DD memory option. The bootstrap assumes that there are 30K words available for the operating system. If this is not the case, RT–11 will not load into memory properly. However, if you install the following customization in your monitor, the bootstrap will never look for more than 28K words of memory. You cannot install this customization in an XM monitor. In the customization, monitr.SYS is the name of the monitor file that you want to modify.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      1000(RET)
Offset?    60(RET)

      Base      Offset        Old     New?
       001000    000060    170000     160000(RET)
       001000    000062    001402     (CTRL/Y)(RET)
*(CTRL/C)
.
```

If the monitor you modify is the hardware bootable monitor, write a new system bootstrap with the COPY/BOOT command.

2.7.22 Suppressing the Bootstrap Message

If you want to prevent the monitor identification message from printing when you bootstrap a monitor, you can modify that monitor. In the customization, monitr.SYS is the name of the monitor file that you want to modify, and ..SLNT is the value of that symbol from the monitor link map.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?       0(RET)
Offset?     ..SLNT(RET)

      Base      Offset        Old     New?
       000000   ..SLNT      000000    1(RET)
       000000   ..SLNT+2    001003    (CTRL/Y)(RET)
*(CTRL/C)
.
```

If the monitor you modify is the hardware bootable monitor, write a new system bootstrap with the COPY/BOOT command.

2.7.23 Suppressing the Start-Up Indirect Command File

If you want to prevent the start-up command file from executing when you bootstrap a monitor, you can modify that monitor. The standard monitors include start-up indirect command file support although you need not select it if you perform the system generation process to create special monitors. In the customization, monitr.SYS is the name of the monitor file that you want to modify, and ..NIND is the value of that symbol from the monitor link map.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?       0(RET)
Offset?     ..NIND(RET)

      Base      Offset        Old     New?
       000000   ..NIND      004000    0(RET)
       000000   ..NIND+2    000044    (CTRL/Y)(RET)
*(CTRL/C)
.
```

If the monitor you modify is the hardware bootable monitor, write a new system bootstrap with the COPY/BOOT command.

2.7.24 Suppressing the Start-Up Indirect Command File Echo

If you want the start-up indirect command file to execute when you bootstrap a monitor but you do not want the command lines in the file to echo (appear) on the terminal, you can modify the monitor. This customization causes the monitor to use the SET TT QUIET mode of operation. In the customization, monitr.SYS is the name of the monitor file that you want to modify, and ..TTQU is the value of that symbol from the monitor link map.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?     0(RET)
Offset?   ..TTQU(RET)

     Base     Offset        Old     New?
      000000  ..TTQU     000000     1(RET)
      000000  ..TTQU+2   001403     (CTRL/Y)(RET)
*(CTRL/C)
.
```

If the monitor you modify is the hardware bootable monitor, write a new system bootstrap with the COPY/BOOT command.

2.7.25 Changing the Bootstrap Message

If you want to change the monitor identification message that appears when you bootstrap a monitor, you can modify that monitor. Run SIPP to modify the monitor file. Place the string (the message) in the monitor image starting at location BSTRNG. End the string with a null byte. The string must not be more than 20 (octal) bytes long, including the null byte. If the monitor you modify is the hardware bootable monitor, write a new system bootstrap with the COPY/BOOT command.

2.7.26 Changing the Default Device for Indirect Command Files

If you want to change the default device for indirect command files, you can modify the monitor. Normally, when you invoke an indirect command file (by typing @filnam), the default device where the monitor looks for the command file is DK:. If you have a special application, you can change this default to any three-character device name.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, ..ATDK is the value of that symbol from the monitor link map, and nnn is the new default device name.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?     0(RET)
Offset?   ..ATDK(RET)

     Base     Offset        Old     New?
      000000  ..ATDK     015270     ;R(RET)
      000000  ..ATDK      <DK >     ;Rnnn(RET)
      000000  ..ATDK+2    <AW1>     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.27 Changing the Default File Type for Indirect Command Files

If you want to change the default file type for indirect command files, you can modify the monitor. Normally, indirect command files have the default file type .COM. When you invoke an indirect command file (by typing @filnam), the monitor looks for the file filnam.COM. If you have a special application, you can change this default to any three-character file type.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, ..ATFX is the value of that symbol from the monitor link map, and nnn is the new default file type.

```
.RUN SIPP[RET]
*monitr.SYS[RET]
Base?      0[RET]
Offset?    ..ATFX[RET]

     Base      Offset        Old     New?
      000000   ..ATFX       012445   ;R[RET]
      000000   ..ATFX        <COM>   ;Rnnn[RET]
      000000   ..ATFX+2      <xxx>   [CTRL/Y][RET]
*[CTRL/C]
.
```

2.7.28 Changing the Default Device for the FRUN Command

If you want to change the default device for the FRUN command, you can modify the monitor. Normally, when you start a foreground program under the FB or XM monitor (by typing FRUN filnam), the default device where the monitor looks for the program file is DK:. If you have a special application, you can change this default device to any three-character device name.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, ..FRDK is the value of that symbol from the monitor link map, and nnn is the new default device name.

```
.RUN SIPP[RET]
*monitr.SYS[RET]
Base?      0[RET]
Offset?    ..FRDK[RET]

     Base      Offset        Old     New?
      000000   ..FRDK       015270   ;R[RET]
      000000   ..FRDK        <DK >   ;Rnnn[RET]
      000000   ..FRDK+2      <xxx>   [CTRL/Y][RET]
*[CTRL/C]
.
```

2.7.29 Changing the Default File Type for the FRUN Command

When you start a foreground program under the FB or XM monitor (by typing FRUN filnam), the default file type for the program file is .REL. If you have a special application, you can change this default file type to any three-character file type.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, ..FRUX is the value of that symbol from the monitor link map, and nnn is the new default file type.

```
.RUN SIPP[RET]
*monitr.SYS[RET]
Base?      0[RET]
Offset?    ..FRUX[RET]

     Base      Offset        Old     New?
      000000   ..FRUX       070524   R[RET]
      000000   ..FRUX        <REL>   ;Rnnn[RET]
      000000   ..FRUX+2      <ANG>   [CTRL/Y][RET]
*[CTRL/C]
.
```

2.7.30 Changing the Default Device for the EDIT Command

If you want to change the default device for the EDIT command, you can modify the monitor. Normally, when you invoke an editor by typing the EDIT command, the default device where the monitor looks for EDIT.SAV is DK:. If you have a special application, you can change this default device to any three-character device name. The customization changes the default device for EDIT and TECO only; it does not change the default device for KED or K52.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, ..EDDV is the value of that symbol from the monitor link map, and nnn is the new default device name.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      0(RET)
Offset?    ..EDDV(RET)

     Base      Offset       Old    New?
      000000   ..EDDV     075250   ;R(RET)
      000000   ..EDDV     <SY >    ;Rnnn(RET)
      000000   ..EDDV+2   <CHU>    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.31 Changing the Default File Name for the EDIT Command

If you want the monitor to run a program other than the default editor when you type the EDIT command, you can modify the monitor to change the default file name. If you have a special application, you can change this file name to any six-character file name.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, $RMON is the value of that symbol from the monitor link map, PROGDF is the value of that symbol from the monitor link map, and EEE is 200 (octal) plus the new byte value of one of the following symbols from the monitor link map:

$$EDIT	Command value for default editor EDIT
$$TECO	Command value for default editor TECO
$$KED	Command value for default editor KED
$$K52	Command value for default editor K52

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      $RMON(RET)
Offset?    PROGDF(RET)

     Base      Offset       Old    New?
      $RMON    PROGDF     103625   \(RET)

     Base      Offset       Old    New?
      $RMON    PROGDF        225   EEE(RET)
      $RMON    PROGDF+1      207   (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.32 Using Examine and Deposit Above the Background Job

If you want to be able to examine and modify the monitor and the I/O page, you can modify the monitor to remove a restriction on the use of the E (Examine) and the D (Deposit) keyboard commands. Normally, the monitor allows you to examine and modify only locations inside the background job's area. You can remove this restriction, but you must be extremely careful when modifying the monitor or I/O page, since you may inadvertently destroy the resident monitor or corrupt a device.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, and ..EMON is the value of that symbol from the monitor link map.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      0(RET)
Offset?    ..EMON(RET)

     Base     Offset        Old     New?
      000000   ..EMON      103041    240(RET)
      000000   ..EMON+2    103007    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.33 Changing the Default Device for QUEMAN

You can change the default device for the queue manager (QUEMAN). Normally, when you use the PRINT or DELETE/ENTRY command (in an FB or XM system with QUEUE running as a foreground or system job) QUEMAN sends the file to the device LP: or deletes the entry from the LP: queue. If you frequently queue to another device (such as the serial printer, LS:), you can change the default device for these commands.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, ..QULP is the value of that symbol from the monitor link map, and dv is the two-character device name that you want as the default.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      0(RET)
Offset?    ..QULP(RET)

     Base     Offset        Old     New?
      000000   ..QULP       050114   ;A(RET)
      000000   ..QULP         <L>    ;Ad(RET)
      000000   ..QULP+1       <P>    ;Av(RET)
      000000   ..QULP+2       <:>    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.34 Changing the Indirect Command File Nesting Depth

You can increase the indirect command file nesting depth. Normally, RT–11 allows you to nest indirect files to a depth of three. A nesting depth of three allows your indirect file to invoke another indirect file, which invokes still another indirect file. If you have a special application that requires more nesting, you can change the maximum nesting depth by modifying the monitor. Note that if you increase the nesting depth, any use of the indirect file feature will use more memory than is usual.

In the customization, monitr.SYS is the name of the monitor file that you want to modify, $RMON is the value of that symbol from the monitor link map, and nnn is the maximum indirect file nesting depth that you want; nnn should be a small integer, and must not be zero.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      $RMON(RET)
Offset?    377(RET)

      Base      Offset      Old     New?
       $RMON     000377      003     nnn(RET)
       $RMON     000400      xxx     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.35 Changing the Threshold for Resuming Output-Stalled Jobs

You can improve FB or XM system throughput by changing the threshold for resuming output-stalled jobs. In an RT–11 FB or XM system, a job is placed in a stalled state whenever it has terminal output to print but there is no room in its terminal output ring buffer. The system restarts the job when room becomes available in that ring buffer. The system's default mode of operation is to restart the job as soon as a single character of space is available. If more than one job is running, this mode of operation can cause the system to spend much time swapping in the context of a job, simply to have it output a single character and then stall again.

You can patch the monitor so that a job that is stalled waiting for room in the terminal output buffer does not resume execution until several characters are available in the ring buffer. If you have a foreground or system job that produces a large amount of terminal output, installing this change can greatly improve system throughput.

In the following customization, monitr.SYS is the name of the monitor file that you want to modify, ..TTON is the value of that symbol from the monitor link map, and nnn is the threshold value for resuming a terminal output stalled job.

The monitor will resume such a job when there are nnn-1 characters left to print in the output ring. The default value, 50 (octal), is the size of the ring; consequently, the monitor resumes a job when 47 (octal) characters are left to print (that is, when only one character position is available in the output ring). You can specify any value from 1 to the size of the output ring buffer (which is normally octal 50, but can be changed at system generation time). Note that a value of 1 will cause the job to stay stalled until its ring buffer is empty, and may cause terminal output to appear unaligned.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?      0(RET)
Offset?    ..TTON(RET)

      Base      Offset       Old      New?
       000000   ..TTON     000050     nnn(RET)
       000000   ..TTON+2  xxxxxx     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.36 Changing the Default Number of Directory Segments

You can change the number of directory segments DUP creates when you initialize a volume. Normally, DUP uses the default number of directory segments, which depends on the size of the volume. (Refer to the *RT–11 System Utilities Manual.*) In other words, when you use the INITIALIZE command to initialize a volume, DUP ascertains the size of the volume and checks a table (within DUP) to establish the number of directory segments to use. This table consists of two-word entries that give a size and the number of segments. DUP searches the table until it finds a size larger than or equal to the size of the volume being initialized and uses the value in the following word as the number of directory segments.

If the default number of directory segments for a volume is unacceptable for your application, you can use the /SEGMENTS:n option with INITIALIZE to initialize a volume with n directory segments. (Refer to the *RT–11 System User's Guide.*) However, if you want DUP to use a specific number, you can modify DUP to change DUP's directory segment table (duplicated in the following).

```
.WORD      1000          ;Volumes with <= 512. blocks
.WORD      1             ;set 1 segment directories
.WORD      4000          ;Volumes with <= 2048. blocks
.WORD      4             ;set 4 segment directories
.WORD      30000         ;Volumes with <= 12288. blocks
.WORD      20            ;set 16. segments
.WORD      177777        ;Volumes with <= 65535. blocks
.WORD      37            ;set 31. segments
.BLKW      10.           ;Expansion space for finer variations
.WORD      0             ;Must be 0.
```

In the customization, nnnnnn is an octal offset value from the table below. The value mmmmmm is an octal number (in the range 1 to 37) for the number of directory segments you want as the default. Find the size of the volume for which you are changing the default number of directory segments in the table. Enter the corresponding offset value in nnnnnn. Then, enter the number of default directory segments you want in mmmmmm.

Size of Volume (Blocks) (decimal)	Offset Value (octal)
<=512	000002
<=2048	000006
<=12288	000012
<=65535	000016

```
.R SIPP(RET)
*DUP.SAV(RET)
Segment?   10(RET)
Base?      SEGTBL(RET)
Offset?    nnnnnn(RET)

Segment      Base      Offset        Old       New?
000010      SEGTBL      nnnnnn     <xxxxxx>  mmmmmm(RET)
000010      SEGTBL      nnnnnn+2  <xxxxxx>  (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.37 Changing the Banner Page Width

You can modify QUEUE to change the width of the banner page on line printer output from 132 positions to 80 positions.

In the customization, ..NB1, ..NB2, and ..NB3 are the values of those symbols from the link map.

```
.RUN SIPP(RET)
*QUEUE.REL/A(RET)
Base?      0(RET)
Offset?   ..NB1(RET)

      Base      Offset         Old      New?
      000000   ..NB1           122      15(RET)
      000000   ..NB1+1         124      12(RET)
      000000   ..NB1+2         055      0(RET)
      000000   ..NB1+3         061      (CTRL/Z)(RET)
Offset?    ..NB2(RET)

      Base      Offset         Old      New?
      000000   ..NB2        020040      5015(RET)
      000000   ..NB2+2      064506      (CTRL/Z)(RET)
Offset?    ..NB3(RET)

      Base      Offset         Old      New?
      000000   ..NB3        110024      240(RET)
      000000   ..NB2+2      000766      (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.38 Modifying Listing Page Length in LINK

If you do not use line printer paper of a standard size (10.5 inches long) or if your configuration does not include a line printer, you may need to modify the listing page length in LINK. RT–11 LINK sets the number of lines printed on each listing page at 60. This line count is generally satisfactory only for applications with line printers that use paper 10.5 inches long. However, you may require a listing of a different length. In the customization, LINPPG is the value of that symbol in the link map, and nnn is the desired listing page length (in lines).

```
.RUN SIPP(RET)
*LINK.SAV(RET)
Segment?  0(RET)
Base?     0(RET)
Offset?   LINPPG(RET)

Segment      Base       Offset       Old    New?
000000      000000      LINPPG        ?     nnn(RET)
000000      000000      LINPPG+2      ?     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.39 Assigning the HELP File

If you want to assign your HELP file to a file and/or device other than SY:HELP.SAV, you can modify the monitor.

In the customization, monitr.SYS is the name of the monitor file that you wish to modify, and ..HELF is the value of that symbol from the monitor link map. The RAD50 file specification starts at ..HELF.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?       0(RET)
Offset?     ..HELF(RET)

      Base      Offset         Old     New?
       000000   ..HELF        075250   ;R(RET)
       000000   ..HELF        <SY >    ;Rdev(RET)
       000000   ..HELF+2      <HEL>    ;Rfil(RET)
       000000   ..HELF+4      <P  >    ;Rnam(RET)
       000000   ..HELF+6      <SAV>    ;Rext(RET)
       000000   ..HELF+10     <xxx>    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.40 Changing the Device from Which IND.SAV Is Run

By default, the system runs IND.SAV from SY: when KMON is set to IND and you type @IND. You can modify the monitor to change the device from which the IND.SAV file is run. To change the device, change the contents of ..INDN to the RAD50 device name of the device from which IND will be run. For example, to run IND from the virtual device (VM), open the ..INDN location in the link map and change its contents to VM.

In the customization, monitr.SYS is the name of the monitor file that you wish to modify, and ..INDN is the value of that symbol from the monitor link map.

```
.RUN SIPP(RET)
*monitr.SYS(RET)
Base?       0(RET)
Offset?     ..INDN(RET)

      Base      Offset         Old     New?
       000000   ..INDN        075250   ;R(RET)
       000000   ..INDN        <SY >    ;Rdev(RET)
       000000   ..INDN+2      <IND>    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.41 Supporting Bad Block Replacement in User-Written Handlers

If a user-written handler supports bad block replacement, one of the following customizations must be applied. If all bad blocks are replaceable (such as the RL01/02), apply customization A. If only bad sector errors are replaceable (such as the RK06/07), apply customization B.

Customization A

```
.R SIPP(RET)
*DUP.SAV(RET)
Segment?   1(RET)
Base?      0(RET)
Offset?    ARDPS(RET)

Segment  Base    Offset   Old New?
000001   000030  ARDPS    000 377(RET)
000001   000030  ARDPS+1  000 (CTRL/Y)(RET)
*(CTRL/C)
.
```

Customization B

```
.R SIPP(RET)
*DUP.SAV(RET)
Segment?   1(RET)
Base?      0(RET)
Offset?    SRDPS(RET)

Segment  Base    Offset       Old   New?
000001   000000  SRDPS     000000   \(RET)

Segment  Base    Offset       Old   New?
000001   000000  SRDPS        000   377(RET)
000001   000000  SRDPS+1      000   (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.42 Supporting User-Written Magtape Handlers

If a user-written magtape handler is used, the following three customizations must be made. All three customizations assume that the device code for the device is 377 (octal).

Customization 1

```
.R SIPP(RET)
*DUP.SAV(RET)
Segment?   1(RET)
Base?      0(RET)
Offset?    MTDPS(RET)

Segment  Base    Offset       Old   New?
000001   000000  MTDPS     000000   \(RET)

Segment  Base    Offset       Old   New?
000001   000000  MTDPS        000   377(RET)
000001   000000  MTDPS+1      000   (CTRL/Y)(RET)
*(CTRL/C)
.
```

Customization 2

```
.R SIPP(RET)
*PIP.SAV(RET)
Segment?   1(RET)
Base?      0(RET)
Offset?    PIPMT(RET)

Segment  Base    Offset       Old   New?
000001   000000  PIPMT        000   \(RET)

Segment  Base    Offset       Old   New?
000001   000000  PIPMT        000   377(RET)
000001   000000  PIPMT+1      000   (CTRL/Y)(RET)
*(CTRL/C)
.
```

Customization 3

```
.R SIPP(RET)
*MDUP.SAV(RET)
Base?      0(RET)
Offset?    MDUPMT(RET)

          Base     Offset      Old   New?
          000000   MDUPMT      000   \(RET)

          Base     Offset      Old   New?
          000000   MDUPMT      000   377(RET)
          000000   MDUPMT+1    000   (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.43 Replacing Magtape Bootstrap in DISMT1.COM

Module MBOT16.BOT is a primary bootstrap for producing a 1600 bits/in, phase-encoded, bootable magtape. It is usable on either MS or MM drives. To use this bootstrap, edit DISMT1.COM by replacing MBOOT.BOT with MBOT16.BOT in the INITIALIZE command line. This line should then read INITIALIZE/NOQUERY/VOLUMEID/FILE:DIS:MBOT16.BOT TAP:.

2.7.44 Changing Default File Type of Logical Disk Files

By default, the file type of logical disk files is .DSK. You can alter the default file type in one of two ways. (Note that the same procedures also apply to LDX.SYS for the XM monitor.)

Customization 1

Edit the conditional file SYSGEN.CND and add the following line:

```
DEF$LD = ^Rtyp          ; define default for LD filetype
```

where typ is the desired default file type. Then, reassemble and relink LD.SYS using the conditional file SYSGEN.CND.

Customization 2

```
.UNPROTECT SY:LD.SYS(RET)
.R SIPP(RET)
*SY:LD.SYS/A(RET)
Base?      ..LDEX(RET)
Offset?    0(RET)

          Base     Offset      Old        New?
          000000   ..LDEX      ??????     ;R(RET)
          000000   ..LDEX      <DSK>      ;Rtyp(RET)
          000000   ..LDEX+2    <DSK>      ;Rtyp(RET)
          000000   ..LDEX+4    <DSK>      ;Rtyp(RET)
          000000   ..LDEX+6    <DSK>      ;Rtyp(RET)
          000000   ..LDEX+10   ??????     (CTRL/Y)(RET)
*(CTRL/C)
.
```

Figure 2–2: Installation Worksheet

SELECTED COMPONENTS	SIZE IN BLOCKS
Monitors	
☐ RT11BL.SYS	________
☐ RT11SJ.SYS	________
☐ RT11FB.SYS	________
☐ RT11XM.SYS	________
System Device Handler	
________	________
Other Device Handlers	
________	________
________	________
________	________
________	________
________	________
☐ SWAP.SYS	________
Utility Programs	
☐ BATCH.SAV	________
☐ BINCOM.SAV	________
☐ BUP.SAV	________
☐ CREF.SAV	________
☐ DIR.SAV	________
☐ DUMP.SAV	________
☐ DUP.SAV	________
☐ EDIT.SAV	________
☐ ERROUT.SAV	________
☐ FILEX.SAV	________
☐ FORMAT.SAV	________
☐ HELP.SAV	________
☐ IND.SAV	________
☐ LIBR.SAV	________
☐ LINK.SAV	________
☐ MDUP.SAV	________
☐ PAT.SAV	________
☐ PIP.SAV	________
☐ QUEMAN.SAV	________
☐ QUEUE.REL	________
☐ RESORC.SAV	________
☐ SIPP.SAV	________
☐ SLP.SAV	________
☐ SRCCOM.SAV	________
Help Files	
☐ HELP.EXE	________
☐ HELP.TXT	________
________	________
________	________
Libraries	
☐ SYSLIB.OBJ	________
☐ SYSMAC.SML	________
________	________
________	________
Assemblers	
☐ MACRO	________
	________
________	________
Source Files	
________	________
________	________
________	________
________	________
Other Files	
________	________
________	________
________	________

ARRANGEMENT

System Volume

____
____
____
____
____
____
____
____
____
____
____

Volume 2

____
____
____
____
____

Volume 3

____
____
____
____
____

Volume 4

____
____
____
____
____

Volume 5

____
____
____
____
____

Customizations

________	PAGE	________
________	PAGE	________
________	PAGE	________
________	PAGE	________
________	PAGE	________

Chapter 3
Installing a System Distributed on RX01 to Run on a Small Device

If RT–11 was distributed to you on single-density diskettes, and you intend to build a system to run on single-density diskette or DECtape II, perform the procedures described in this chapter.

These procedures cover minor variations that depend on the specific device you have (diskette or cartridge). The word "volume" in this chapter refers to either diskette or cartridge, whichever is appropriate.

NOTE

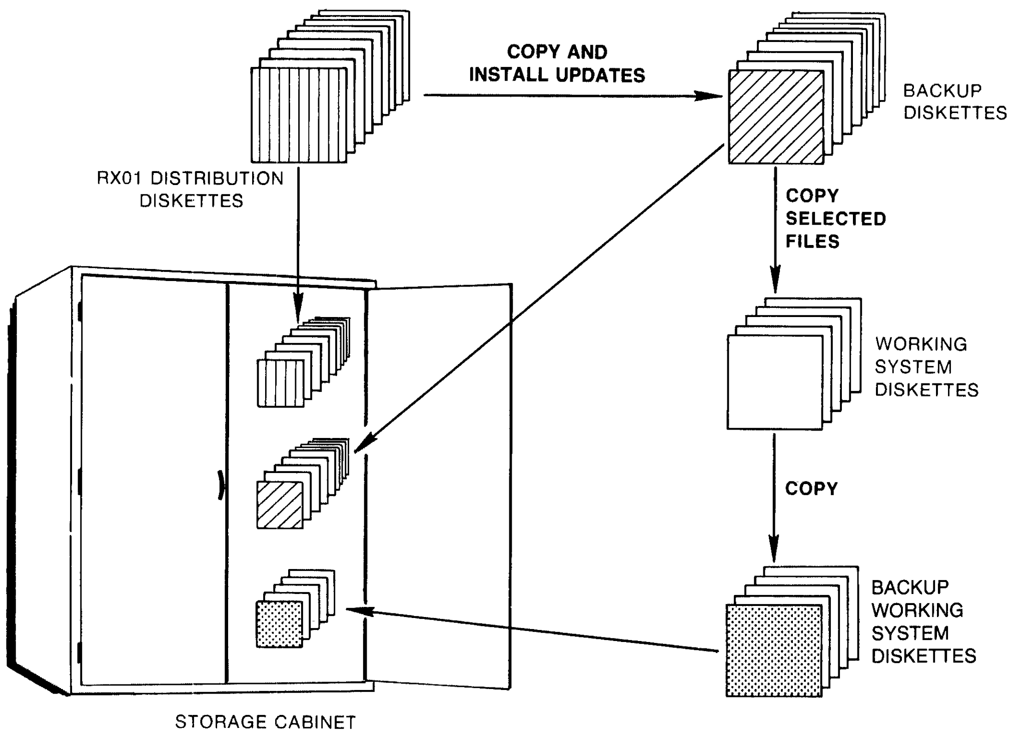
> If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

To install your system, you will have to perform the steps summarized in the following list. Sections 3.1 through 3.10 describe the procedures for each step. Figure 3–1 shows the various backup volumes you create when you install RT–11.

1. Bootstrap the distribution volume.
2. Preserve the distribution volumes.
3. Install software updates.
4. Create the working system from chosen components.
5. Install the bootstrap on volumes that need to be bootable.
6. Customize the system.
7. Compress each volume.
8. Preserve the working system.
9. Test the working system.
10. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 3–1: Sample Backup Volumes



3.1 Bootstrapping the Distribution Volume

The first procedure you perform when installing RT–11 is to bootstrap the distribution volume.

Begin by making sure that the processor is powered up but not running. Insert distribution volume 1 in RX01 Unit 0. For RX01 diskette, the device name is DX:.

Use the hardware bootstrap to boot the volume. (If your configuration does not include a hardware bootstrap, see Appendix B for toggle-in software bootstraps.)

RT–11 should respond with the following message if you have successfully bootstrapped the volume:

```
RT-11FB V05.00

.SET TT NO QUIET

.TYPE V5USER.TXT

Welcome to RT-11 Version 5. RT-11 V5 provides new hardware support and
some major enhancements over Version 4. Among the new features are:

     ● 22-bit addressing support for Q-bus
     ● Virtual Disk facility (VM)
     ● Logical Disk Subsetting handler (LD)
     ● Single line, keypad command editor (SL)
     ● MSCP disk handler for RD51, RC25 and RA80 disks (DU)
     ● New multi-volume Backup/Restore utility (BUP)
     ● New Control File processor (IND)
     ● Concise Command Language (CCL)
     ● User Command Linkage (UCL)
     ● Many new and enhanced keyboard commands and options
     ● Many enhancements to existing RT-11 file utilities

Please use the HELP command; it describes the new commands and options
and the proper syntax with which to invoke them.

.
```

After you boot the system, you can create a start-up file containing any commands or sequence of commands that you would normally want to execute at the start of each terminal session. For example, if your console device is a video terminal, you may wish to execute a start-up file containing the command

```
SET TT SCOPE
```

each time that you use the system. This command allows you to use the DELETE key to backspace and erase characters from the screen.

You can update your start-up file at any time to change, add, or delete commands. When you first create the file, or after you update it, you can execute it with the @ command, so that the commands it contains become effective.

3.2 Preserving the Distribution Volumes

The first procedure you perform with the running RT–11 system is to copy all the distribution volumes for backup, as a safety measure in case of machine failure or human error.

NOTE

If your volumes are diskettes, you can ignore instructions to write-enable or write-protect a volume. Diskettes are always write-enabled. DECtape II provides a write-protect feature, but single-density diskette does not.

Insert a blank volume (write-enabled) in Unit 1. Then use the INITIALIZE command to initialize the blank volume (procedure follows). Use the /BADBLOCKS option with INITIALIZE to cover any bad blocks that may be on your volume. If the volume contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal.

NOTE

> DIGITAL recommends that you use only volumes that do not have bad blocks when you back up distribution volumes and build a working system. To ascertain whether an already initialized volume has bad blocks, use the command DIRECTORY/BAD xxn:. You can use volumes with bad blocks later, for working or data volumes. However, there is an exception to this rule. If your volumes are TU58 tape cartridges, you can use the procedure in Section 2.4.3 when you build your working system but not when you back up the distribution volumes. The procedure in Section 2.4.3 creates dummy bad blocks on cartridges to improve response time.

```
.INITIALIZE/BADBLOCKS xx1:RET
xx1:/Initialize; Are you sure? YRET
?DUP-I-No bad blocks detected xx1:
```

There may be a significant delay (as much as 8 minutes) while the system scans the volume for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Now, remove the newly initialized volume and initialize eight additional blank volumes, leaving a write-enabled and initialized blank volume inserted in Unit 1.

The next step in the preservation process is to copy all the files from distribution volume 1 to the initialized blank volume, which becomes backup volume 1.

As long as the volume you intend to copy is bootable and contains certain system utility programs, you can boot RT–11 from that volume and copy the files from that volume to another volume in Unit 1. A bootable volume has the appropriate monitor file and system device handler and a bootstrap. The SQUEEZE/OUTPUT:xxn: command transfers all the files from one volume in Unit 0 to another one in Unit 1. At the same time, the command consolidates all the empty space at the end of the volume. Distribution volume 1 is bootable, but the other volumes in your kit are not bootable because they lack the necessary components. Thus, these volumes require a different copy procedure. For volume 1, however, the procedure is straightforward. Type the following command, where xx is the physical device name.

```
.SQUEEZE/OUTPUT:xx1: xx0:RET
.
```

NOTE

> Both the SQUEEZE command with the /OUTPUT:xxn: option and the COPY command transfer files from one device to another. SQUEEZE/OUTPUT:xxn: consolidates free space on the device at the same time. The procedures in this manual use the command that is most appropriate and most efficient for an individual operation. For a better understanding of all RT–11 keyboard monitor commands, refer to the *RT–11 System User's Guide.*

There is a delay while the file transfer operation takes place. Then copy the bootstrap on the volume.

```
.COPY/BOOT xx1:RT11FB.SYS xx1:(RET)
.
```

Remove the newly created backup volume from Unit 1, write-protect it, and label it "Backup RT–11 V05 1/9". (Use a soft-tipped pen when you label diskettes.)

To copy the remaining distribution volumes, which lack certain system components, run RT–11 from distribution volume 1 in Unit 0.

Then, type:

```
.SQUEEZE/WAIT/OUTPUT:xx1: xx0:(RET)
Mount output volume in xx1:; Continue?
```

Insert an initialized blank volume (write-enabled) in Unit 1.

```
Y(RET)

Mount input volume in xx0:; Continue?
```

Replace the volume in Unit 0 with distribution volume 2 (write-protected).

```
Y(RET)
Mount system volume in xx0:; Continue?
```

Replace the volume in Unit 0 with distribution volume 1 (write-protected).

```
Y(RET)
.
```

Remove the volume from Unit 1, write-protect it, and label it "Backup RT–11 V05 2/9".

Again type the command:

```
.SQUEEZE/WAIT/OUTPUT:xx1: xx0:(RET)

Mount output volume in xx1:; Continue?
```

Insert another initialized blank volume in Unit 1.

```
Y(RET)
Mount input volume in xx0:; Continue?
```

Replace the volume in Unit 0 with distribution volume 3 (write-protected).

```
Y(RET)
Mount system volume in xx0:; Continue?
```

Replace the volume in Unit 0 with distribution volume 1 (write-protected).

```
Y(RET)
.
```

Remove the volume from Unit 1, label it "backup RT–11 V05 3/9", write-protect it, and insert another initialized blank volume in Unit 1. Repeat these procedures to copy the rest of the distribution volumes.

Now halt the processor. Replace distribution volume 1 with backup volume 1 (write-protected) in Unit 0. Write-protect the distribution volumes, and store them. You will use the backup copies to build a working system.

Use the hardware bootstrap to boot backup volume 1.

```
RT-11FB V05.00
```
(Followed by any start-up file commands.)
```
.
```

NOTE

> If the backup volume does not boot, repeat the procedure to create the backup volume.

Next, remove the protection from all the files on the backup volumes. The files on the distribution volumes have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) When you copied the files to the backup volumes, RT–11 also copied the protection (note the P that prints next to the file size in the directory). For the rest of these procedures, you need to remove the protection from the backup volumes. Type the following command to remove it from the files on backup volume 1. The /SYSTEM option is required to remove protection from system files (.SYS) if wildcards are used in the file specification.

```
.UNPROTECT/SYSTEM *.*(RET)
 Files unprotected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
     .
     .
     .
DK:zzzzzz.ttt
.
```

Insert backup volume 2 in Unit 1, and type the following command:

```
.UNPROTECT/SYSTEM xx1:*.*(RET)
 Files unprotected:
xx1:aaaaaa.ttt
xx1:bbbbbb.ttt
xx1:cccccc.ttt
xx1:dddddd.ttt
     .
     .
     .
xx1:zzzzzz.ttt
.
```

Replace backup volume 2 with backup volume 3 in Unit 1 and repeat this command. In the same way, remove protection from the files on the rest of the backup volumes.

3.3 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Updated modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide.* Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to problems, until the module containing an error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

3.4 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3) and have planned the best arrangement of them on volumes (Section 2.4), you can create the working system by copying selected components to initialized, blank volumes.

Start by initializing a number of blank volumes. Follow the same procedure that you used in Section 3.2. Insert a write-enabled, blank volume in Unit 1 (with the system booted from Unit 0), and type INITIALIZE/BADBLOCKS xx1:. Repeat the process to create as many initialized blank volumes as you need for the system that you have planned.

NOTE

> If you want to create dummy bad blocks on cartridges to avoid excessive rewinds (as described in Section 2.4.3), do so at this point.

Then, use the COPY command to copy selected files from backup volume 1 to the volume that becomes your working system volume. The /SYSTEM option is required for copying .SYS files only if wildcards are used in the input file specification.

```
.COPY xx0:filnam.typ xx1:filnam.typ(RET)
.
```

You can use the following command to avoid typing numerous file specifications.

```
.COPY/QUERY/SYSTEM xx0: xx1:(RET)
  Files copied:
xx0:aaaaaa.ttt to xx1:aaaaaa.ttt? Y(RET)       (to include a specific file)
xx0:bbbbbb.ttt to xx1:bbbbbb.ttt? N(RET)       (to exclude a specific file)
(and so on)
```

To copy files from nonbootable volumes, alternate volumes.

Use the SET command to set the USR to NOSWAP.

```
.SET USR NOSWAP(RET)
.
```

Type the following command, where filnam.typ is the name of the file you want to copy. In this case, you cannot use the /QUERY option; you must specify individual files.

```
.COPY/WAIT xx1:filnam.typ xx0:filnam.typ(RET)
Mount input volume in xx1:; Continue?
```

Place the volume containing the file you want to copy in Unit 1.

```
Y(RET)

Mount output volume in xx0:; Continue?
```

Replace the system volume in Unit 0 with the volume to which you want to copy filnam.typ.

```
Y(RET)

Mount system volume in xx0:; Continue?
```

Replace the volume in Unit 0 with backup volume 1 (write-protected).

```
Y(RET)
.
```

Repeat this procedure to copy all the files for the working system volume. When you have copied all the files you have planned for the working system volume, label it "RT–11 Working System V05 1/x" (where x is the number of volumes in your working system). Repeat these procedures to create the other volumes in the working system.

When you have created and labeled all the working system volumes, you can permit the USR to swap again.

```
.SET USR SWAP(RET)
.
```

3.5 Installing the Bootstrap on Any Volumes That Need to Be Bootable

Once you have created your system, you need to install the bootstrap on any volumes that must be bootable (that is, that you can use as the system volume). Generally, any volume that includes a monitor file and system device handler should be bootable (but remember that the volume would need SWAP.SYS and, for the SJ monitor, TT.SYS).

Insert in Unit 1 the volume on which you need to install the bootstrap. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xx1:RT11aa.SYS xx1:(RET)
.
```

In this command, you need to identify the device on which the monitor that contains the bootstrap information resides, the name of that monitor file, and the device on which you need to install the bootstrap. This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same volume.

Then, insert working system volume 1 in Unit 0, and use the hardware bootstrap to boot your working system.

```
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

Store the updated backup volumes for future updating purposes.

3.6 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 3.10.)

3.7 Compressing Each Volume

DIGITAL recommends that you compress each working system volume to make its free space contiguous. Consolidating free space allows you to use space on the volume that would otherwise be too fragmented to be usable. However, if your volumes are TU58 tape cartridges and you create bad blocks to avoid excessive rewinds, the amount of contiguous free space possible is limited. (Refer to Section 2.4.3.)

Continue to run RT–11 from Unit 0, and use the SQUEEZE command to compress free space. (The volume must be write-enabled.) The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE xx0:(RET)
xx0:/Squeeze; Are you sure? Y(RET)
```

There will be a delay as the system compresses the volume.

```
RT-11aa V05.00
```
(Followed by any start-up file commands.)
```
.
```

The system automatically reboots when you compress a system volume.

Then insert the next volume that you need to compress (write-enabled) in Unit 1.

```
.SQUEEZE xx1:(RET)
xx1:/Squeeze; Are you sure? Y(RET)
.
```

Replace the volume in Unit 1 with the next one you need to compress, and repeat this procedure for all the volumes you need to compress.

NOTE

> When you compress a volume with system files (.SYS), PIP warns you to reboot. Do reboot as advised. When you compress a system volume, the system automatically reboots.

3.8 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files in the working system and preserve the system on backup volumes.

Use the following command to protect all the files on the system volume:

```
.PROTECT/SYSTEM *.*(RET)
 Files protected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
     .
     .
     .
DK:zzzzzz.ttt
.
```

To protect files on other volumes in the working system, insert each volume in Unit 1 and use the following command:

```
.PROTECT/SYSTEM xx1:*.*(RET)
 Files protected:
xx1:aaaaaa.ttt
xx1:bbbbbb.ttt
xx1:cccccc.ttt
xx1:dddddd.ttt
     .
     .
     .
xx1:zzzzzz.ttt
.
```

Next, copy the working system to backup volumes. Insert a blank volume (write-enabled) in Unit 1 with RT–11 still booted from Unit 0. Use the INITIALIZE/BADBLOCKS command to initialize the blank volume. Then, repeat the process to initialize the appropriate number of volumes.

Copy all the files in your working system. You can use the SQUEEZE/OUTPUT:xxn: command to copy any bootable volumes. Remember that you must use SQUEEZE/WAIT/OUTPUT:xxn: and change volumes to copy the volumes that are not bootable. (Refer to Section 3.2.) Also remember to copy the bootstrap to any volumes that need to be bootable.

Write-protect the backup working system volumes, and store them. If you ever need to restore the working system, you can make copies of the backup working system volumes.

3.9 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, your working system must include at least the following components:

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP  .SYS
RT11FB.SYS
LP    .SYS
DX    .SYS
EDIT  .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
     .
     .
     .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to a scope and the directory scrolls by too quickly to read, type CTRL/S to stop the display and CTRL/Q to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

3.9.1 Preparing the Background Demonstration Program

3.9.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

3.9.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

> If your configuration does not include a line printer, use the console terminal.
>
> ```
> .ASSIGN TT: LST:(RET)
> .
> ```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG(RET)
```

(See Figure 3–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 3–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

      1                                             .TITLE  DEMOBG
      2                                             .IDENT  /V05.00/
      3                                     ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
      4                                     ; RING BELL IF FG JOB SENDS A MESSAGE.
      5
      6                                             .MCALL  .RCVDC,.PRINT
      7
      8 000000                              START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
      9 000034                                      .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
     10 000042  000777                              BR      .                         ;AND LOOP
     11
     12                                     ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE
     13
     14 000044                              MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
     15 000052                                      .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
     16 000106  000207                              RETURN                            ;AND RETURN FROM COMPLETION ROUTINE
     17
     18                                     ;       ASCII MESSAGES
     19                                             .NLIST BEX
     20 000110     007     200              BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL
     21
     22 000112     122     124     055      MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111     106     040              .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24                                     ;       .ASCII  /WELL DONE./
     25 000225     000                              .BYTE   0
     26
     27 000226                              AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
     28 000242                              BUFFER:                                   ;RCVDC MESSAGE AREA
     29         000000'                             .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R         BUFFER  000242R         MSGIN   000044R         ...V1 = 000003          ...V2 = 000027
BELL    000110R         MSG     000112R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG


.
```

3.9.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

3.9.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

3.9.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

3.9.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your system volume.

3.9.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

3.9.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```
(The bell stops ringing.)
```
.DIRECTORY(RET)
 dd-mmm-yy
```
(The directory of the device DK prints on the terminal.)
```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
(CTRL/C)
(CTRL/C)
.
```

(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
(CTRL/F)
F>
(CTRL/C)
(CTRL/C)
B>
UNLOAD F(RET)
.
```

NOTE

> If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

3.10 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Chapter 4
Installing a System Distributed on RX01 to Run on a Hard Disk

If RT–11 was distributed to you on single-density diskette, and you intend to build a system to run on hard disk, perform the procedures described in this chapter. These procedures cover minor variations that depend on the specific device you have and assume that your configuration includes two hard disk drives.

NOTE

> If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

To install your system, perform the steps summarized in the following list. Sections 4.1 through 4.11 describe the procedures involved in each step. Figure 4–1 shows the various backup volumes you create when you install RT–11.

1. Bootstrap the distribution volume.
2. Copy the distribution volumes to the disk.
3. Preserve the distribution volumes.
4. Install software updates.
5. Create the working system from chosen components.
6. Install the bootstrap on the disk.
7. Customize the system.
8. Compress the disk.

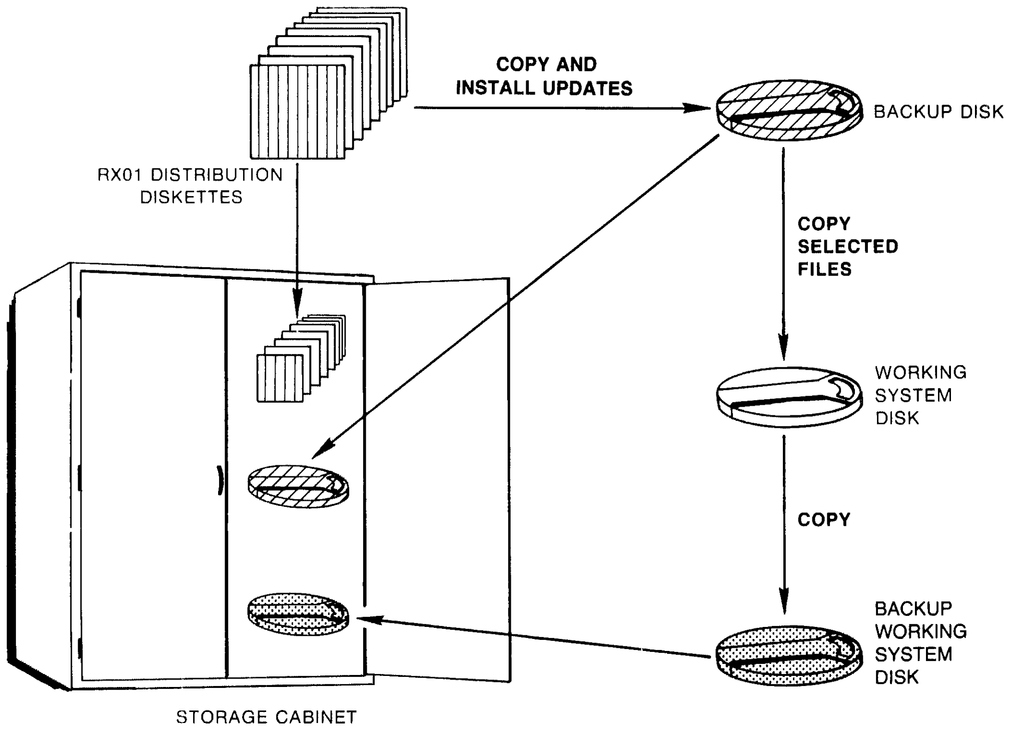
9. Preserve the working system.

10. Test the working system.

11. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 4–1: Sample Backup Disks



4.1 Bootstrapping the Distribution Volume

The first procedure you perform when installing RT–11 is to bootstrap the distribution volume.

Begin by making sure that the processor is powered up but not running. Insert the distribution volume labeled 1/9 (write-protected) in RX01 diskette Unit 0. For RX01 diskette, the device name is DX:.

Use the hardware bootstrap to boot the volume. (If your configuration does not include a hardware bootstrap, see Appendix B for toggle-in software bootstraps.)

RT–11 should respond with the following message if you have successfully bootstrapped the volume:

```
RT-11FB V05.00

.SET TT NO QUIET

.TYPE V5USER.TXT

Welcome to RT-11 Version 5. RT-11 V5 provides new hardware support and
some major enhancements over Version 4. Among the new features are:

     ● 22-bit addressing support for Q-bus
     ● Virtual Disk facility (VM)
     ● Logical Disk Subsetting handler (LD)
     ● Single line, keypad command editor (SL)
     ● MSCP disk handler for RD51, RC25 and RA80 disks (DU)
     ● New multi-volume Backup/Restore utility (BUP)
     ● New Control File processor (IND)
     ● Concise Command Language (CCL)
     ● User Command Linkage (UCL)
     ● Many new and enhanced keyboard commands and options
     ● Many enhancements to existing RT-11 file utilities

Please use the HELP command; it describes the new commands and options
and the proper syntax with which to invoke them.

.
```

After you boot the system, you can create a start-up file containing any commands or sequence of commands that you would normally want to execute at the start of each terminal session. For example, if your console device is a video terminal, you may wish to execute a start-up file containing the command

```
SET TT SCOPE
```

each time that you use the system. This command allows you to use the DELETE key to backspace and erase characters from the screen.

You can update your start-up file at any time to change, add, or delete commands. When you first create the file, or after you update it, you can execute it with the @ command, so that the commands it contains become effective.

4.2 Copying the Distribution Volumes to the Disk

With the RT–11 system running, copy all the distribution volumes to the disk that will serve as your working system disk.

If the disk is an RK05 disk, mount the disk in Unit 0, format the disk, and then initialize it and cover any bad blocks on it.

If the disk is another type, initialize the disk and replace or cover any bad blocks. You have a choice of replacing or covering bad blocks if your disk is an RK06, RK07, RL01, or RL02. If your disk is another type, you should cover any bad blocks.

To replace bad blocks, use the INITIALIZE/REPLACE command, and to cover bad blocks, use the INITIALIZE/BADBLOCKS command (procedures follow). In the commands, xx is the permanent device name for your disk. If the disk contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal, but you can use the disk.

RK05 disk

```
.FORMAT RK0:(RET)
RK0:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS RK0:(RET)
RK0:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RK0:
```

Other type disk

```
.INITIALIZE/BADBLOCKS xx0:(RET)
xx0:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx0:
```

or

```
.INITIALIZE/REPLACE xx0:(RET)
xx0:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx0:
```

There is a delay as the system scans the disk for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

The next step in the preservation process is to copy all the files from distribution volume 1 to the initialized disk. The following command transfers files from the distribution volume to the disk and consolidates all the empty space at the end of the disk. In the command, xx is the permanent device name for your disk, and yy is the device name for the distribution device.

```
.SQUEEZE/OUTPUT:xx0: yy0:(RET)
```

NOTE

> Both the SQUEEZE command with the /OUTPUT:xxn: option and the COPY command transfer files from one device to another. SQUEEZE/OUTPUT:xxn: consolidates free space on the device at the same time. The procedures in this manual use the command that is most appropriate and most efficient for an individual operation. For a better understanding of all RT–11 keyboard monitor commands, refer to the *RT–11 System User's Guide.*

There may be a delay while the system performs this operation.

You must also copy the rest of the distribution volumes to the disk. Insert distribution volume 2 in RX01 Unit 1. In the following command, xx is the permanent device name for your disk, and yy is the name for your distribution device. The /SYSTEM option is required for copying .SYS files only if wildcards are used in the input file specification.

```
.COPY/SYSTEM yy1: xx0:(RET)
 Files copied:
yy1:aaaaaa.ttt to xx0:aaaaaa.ttt
yy1:bbbbbb.ttt to xx0:bbbbbb.ttt
          .
          .
          .
yy1:zzzzzz.ttt to xx0:zzzzzz.ttt
.
```

Insert distribution volume 3 in RX01 Unit 1, and repeat this command. Copy all the distribution volumes to your disk in this way.

NOTE

For optimal system performance, copy the distribution volumes in numerical order.

Finally, copy the bootstrap to the disk and compress the disk. Use the following commands, where xx is the permanent device name for your disk.

```
.COPY/BOOT xx0:RT11FB.SYS xx0:(RET)
.SQUEEZE xx0:(RET)
xx0:/Squeeze; Are you sure? Y(RET)
.
```

4.3 Preserving the Distribution Volumes

Halt the processor, and remove the distribution volumes. Then store them, as a safety measure in case of machine failure or human error. Use the disk to build a working system. Leave the write-enabled disk in Unit 0. Use the hardware bootstrap to boot RT–11 from your disk.

```
RT-11FB V05.00
```
(Followed by any start-up file commands.)
```
.
```

NOTE

If the disk does not boot, repeat the procedures to this point.

Next, remove the protection from all the files on the backup disk. The files on the distribution volumes have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) When you used the SQUEEZE command to copy the files to the backup disk, RT–11 also copied the protection. (Note the P that prints next to the file size in the directory.) For the rest of these procedures, you need to remove the protection from the backup disk. Use the following command:

```
.UNPROTECT/SYSTEM *.*(RET)
 Files unprotected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

4.4 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Updated modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide*. Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to problems, until the module containing the error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

4.5 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3) and have planned the best arrangement of them on a disk (Section 2.4), you can create the working system by copying selected components to another disk or disks.

Mount a blank disk in disk Unit 1 and initialize it. If the disk is an RK05 disk, format it before you initialize it.

RK05 disk

```
.FORMAT RK1:(RET)
RK1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS RK1:(RET)
RK1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RK1:
```

Other type disk

```
.INITIALIZE/BADBLOCKS xx1:(RET)
xx1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx1:
```

or

```
.INITIALIZE/REPLACE xx1:(RET)
xx1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx1:
```

Copy the files you have selected from the disk in Unit 0 to the disk in Unit 1, which will become your working system disk. If you use the following command, RT–11 queries you about all the files on the disk in Unit 0, and you can choose the files it copies.

```
.COPY/SYSTEM/QUERY xx0: xx1:(RET)
  Files copied:
xx0:aaaaaa.ttt to xx1:aaaaaa.ttt? Y(RET)          (to include a specific file)
xx0:bbbbbb.ttt to xx1:bbbbbb.ttt? N(RET)          (to exclude a specific file)
(and so on)
```

4.6 Installing the Bootstrap on the Disk

Once you have created your system, you need to install the bootstrap on the system disk. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xx1:RT11aa.SYS xx1:(RET)
.
```

In this command, you identify the device on which the monitor that contains the bootstrap information resides (your disk), the name of that monitor file (RT11BL, RT11SJ, RT11FB, or RT11XM), and the device on which you need to install the bootstrap (your disk). This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same disk.

Remove the disk from Unit 0 and store it for future updates. Mount the new working system disk in Unit 0, and use the hardware bootstrap to boot the working system disk.

```
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

4.7 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 4.11.)

4.8 Compressing the Disk

DIGITAL recommends that you compress the working system disk to make its free space contiguous. Consolidating free space allows you to use space on the disk that would otherwise be too fragmented to be usable.

Use the SQUEEZE command to compress free space. (The volume must be write-enabled.) The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE xx0:(RET)
xx0:/Squeeze; Are you sure? Y(RET)
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system disk.

4.9 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files in the working system and back it up.

Use the following command to protect all the files on the system disk:

```
.PROTECT/SYSTEM *.*(RET)
 Files protected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

Now preserve the working system on the backup medium of your choice.

4.9.1 Backing Up the System on RX01 Diskettes

Insert a blank RX01 diskette in Unit 1 (with RT–11 still booted from disk Unit 0). Use the INITIALIZE/BADBLOCKS command to initialize the blank diskette. Then, repeat the process to initialize the appropriate number of diskettes.

Copy all the files in your working system, using the COPY/SYSTEM/QUERY command. In the command, xx is the permanent device name for your backup device, and yy is the name for your disk.

```
.COPY/SYSTEM/QUERY yy0: xx1:(RET)
  Files copied:
yy0:aaaaaa.ttt to xx1:aaaaaa.ttt? Y(RET)        (to include a specific file)
yy0:bbbbbb.ttt to xx1:bbbbbb.ttt? N(RET)        (to exclude a specific file)
(and so on)
```

If not enough space exists on the diskette to contain all the files, remove the diskette when it is full, insert a blank, initialized diskette in the drive, and issue the COPY command again. Answer N to all files that have already been copied to a backup diskette.

Then, copy the bootstrap to the backup diskette that will be bootable. Remember that a bootable volume needs a monitor file, a system device handler, the file SWAP.SYS, and for the SJ monitor, TT.SYS.

```
.COPY/BOOT xx1:RT11aa.SYS xx1:(RET)
.
```

Store the backup diskettes. If you ever need to restore the working system, you can copy the backup diskettes to the disk.

4.9.2 Backing Up the System on Another Disk

To back up your system on another disk, initialize a blank disk and copy all the files from the working system disk to the backup disk. In the following commands, xx is the permanent device name for your disk.

RK05 disk

```
.FORMAT RK1:(RET)
RK1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS RK1:(RET)
RK1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RK1:
```

Other type disk

```
.INITIALIZE/BADBLOCKS xx1:(RET)
xx1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx1:
.
```

or

```
.INITIALIZE/REPLACE xx1:(RET)
xx1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx1:
.
```

Copy all the files in your working system.

```
.SQUEEZE/OUTPUT:xx1: xx0:(RET)
.
```

Copy the bootstrap to the backup working system disk. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xx1:RT11aa.SYS xx1:(RET)
.
```

Store the backup working system disk. If you ever need to restore the working system, you can make a copy of the backup working system disk.

4.10 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, your working system must include at least the following components.

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP  .SYS
RT11FB.SYS
LP    .SYS
DX    .SYS
EDIT  .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
     .
     .
     .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

> If you have shifted output to a scope and the directory scrolls by too quickly to read, type (CTRL/S) to stop the display and (CTRL/Q) to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

4.10.1 Preparing the Background Demonstration Program

4.10.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

4.10.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

> If your configuration does not include a line printer, use the console terminal.
>
> ```
> .ASSIGN TT: LST:(RET)
> .
> ```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG(RET)
```
(See Figure 4–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 4–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

                                              .TITLE  DEMOBG
                                              .IDENT  /V05.00/
                                      ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
                                      ; RING BELL IF FG JOB SENDS A MESSAGE.

                                              .MCALL  .RCVDC,.PRINT

 000000                               START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
 000034                                       .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
 000042   000777                              BR      .                         ;AND LOOP

                                      ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE

 000044                               MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
 000052                                       .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
 000106   000207                              RETURN                            ;AND RETURN FROM COMPLETION ROUTINE

                                      ;       ASCII MESSAGES
                                              .NLIST BEX
 000110      007      200             BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL

 000112      122      124      055    MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
 000147      111      106      040            .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
                                      ;       .ASCII  /WELL DONE./
 000225      000                              .BYTE   0

 000226                               AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
 000242                               BUFFER:                                   ;RCVDC MESSAGE AREA
             000000'                          .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R         BUFFER  000242R         MSGIN   000044R         ...V1 = 000003          ...V2 = 000027
BELL    000110R         MSG     000112R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG

.
```

4.10.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

4.10.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

4.10.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

4.10.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your system volume.

4.10.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

4.10.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```
(The bell stops ringing.)
```
.DIRECTORY(RET)
 dd-mmm-yy
```
(The directory of the device DK prints on the terminal.)
```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
(CTRL/C)
(CTRL/C)
.
```
(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
CTRL/F
F>
CTRL/C
CTRL/C
B>
UNLOAD F RET
.
```

NOTE

> If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

4.11 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Chapter 5
Installing a System Distributed on Hard Disk to Run on Hard Disk

If RT–11 was distributed to you on hard disk (RL01/RL02, RK05, or RC25), and you intend to build a system to run on hard disk, perform the procedures described in this chapter. These procedures assume that your configuration includes two disk drives (not necessarily of the same type).

NOTE

> If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

To install your system, perform the steps summarized in the following list. Sections 5.1 through 5.10 describe the procedures involved in each step. Figure 5–1 shows the various backup disks you create when you install RT–11 on a hard disk other than RC25.

Figure 5–2 illustrates the backup procedure for installing RT–11 on an RC25 disk. Notice that files are deleted from the backup disk (a fixed device) to create the working system disk.

1. Bootstrap the distribution disk.
2. Preserve the distribution disk.
3. Install software updates.
4. Create the working system from chosen components.
5. Install the bootstrap on the disk.
6. Customize the system.
7. Compress the disk.
8. Preserve the working system.

9. Test the working system.

10. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 5–1: Sample Backup Disks (RL01, RL02, RK05)

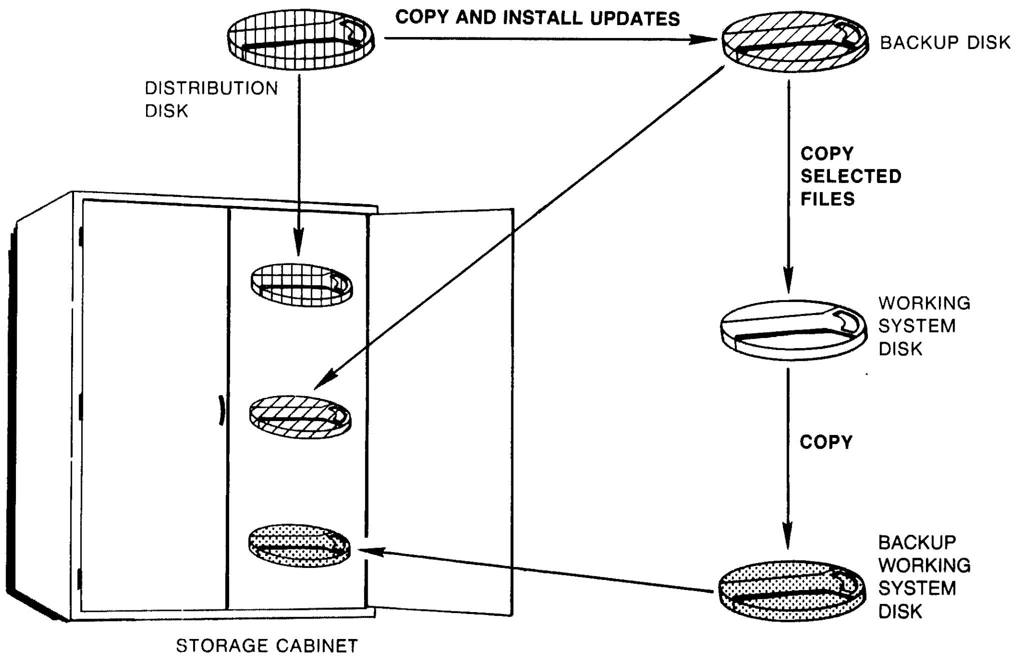
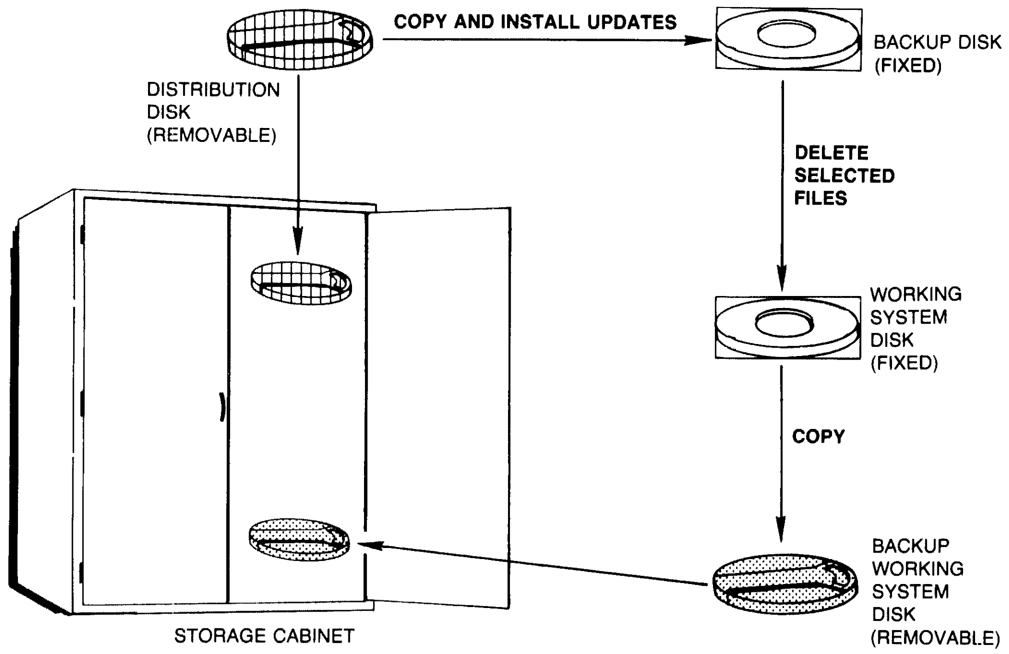


Figure 5–2: Sample Backup Disks (RC25)



COPY AND INSTALL UPDATES
BACKUP DISK (FIXED)
DISTRIBUTION DISK (REMOVABLE)
DELETE SELECTED FILES
WORKING SYSTEM DISK (FIXED)
COPY
BACKUP WORKING SYSTEM DISK (REMOVABLE)
STORAGE CABINET

5.1 Bootstrapping the Distribution Disk

The first procedure you perform when installing RT–11 is bootstrapping the distribution disk.

Begin by making sure that the processor is powered up but not running. Mount the distribution disk (write-protected) in disk Unit 0. The device name is DL, RK, or DU, depending on the type of disk device. Use the hardware bootstrap to boot the disk. (If your configuration does not include a hardware bootstrap, refer to Appendix B for the toggle-in software bootstrap.)

RT–11 should respond with the following message, if you have successfully bootstrapped the disk:

```
RT-11FB V05.00

.SET TT NO QUIET

.TYPE V5USER.TXT

Welcome to RT-11 Version 5. RT-11 V5 provides new hardware support and
some major enhancements over Version 4. Among the new features are:

      ● 22-bit addressing support for Q-bus
      ● Virtual Disk facility (VM)
      ● Logical Disk Subsetting handler (LD)
      ● Single line, keypad command editor (SL)
      ● MSCP disk handler for RD51, RC25 and RA80 disks (DU)
      ● New multi-volume Backup/Restore utility (BUP)
      ● New Control File processor (IND)
      ● Concise Command Language (CCL)
      ● User Command Linkage (UCL)
      ● Many new and enhanced keyboard commands and options
      ● Many enhancements to existing RT-11 file utilities

Please use the HELP command; it describes the new commands and options
and the proper syntax with which to invoke them.

.
```

After you boot the system, you can create a start-up file containing any commands or sequence of commands that you would normally want to execute at the start of each terminal session. For example, if your console device is a video terminal, you may wish to execute a start-up file containing the command

```
SET TT SCOPE
```

each time that you use the system. This command allows you to use the DELETE key to backspace and erase characters from the screen.

You can update your start-up file at any time to change, add, or delete commands. When you first create the file, or after you update it, you can execute it with the @ command, so that the commands it contains become effective.

5.2 Preserving the Distribution Disk

The first procedure you perform with the running RT–11 system is to copy the distribution disk for backup, as a safety measure in case of machine failure or human error. You can then use the backup copy of the distribution disk to build your working system.

Mount a blank disk in the other disk drive, if necessary. (The disk in Unit 1 of RC25 devices is permanently mounted.) If the disk is an RK05 disk, format it, and then initialize it and cover any bad blocks on it.

If the disk is another type, initialize the disk and replace bad blocks (using the command INITIALIZE/REPLACE) or cover bad blocks (using the command INITIALIZE/BADBLOCKS). You have a choice of replacing or covering bad blocks if your disk is an RK06, RK07, RL01, or RL02. (Refer to the *RT–11 System User's Guide.*) If your disk is another type, you should use INITIALIZE/BADBLOCKS to cover any bad blocks. If the disk contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal, but you can use the disk. In the commands, xx is the permanent device name and n is the unit number for your disk.

RK05 disk

```
.FORMAT RKn:(RET)
RKn:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS RKn:(RET)
RKn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RKn:
```

Other type disk

```
.INITIALIZE/BADBLOCKS xxn:(RET)
xxn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xxn:
```

or

```
.INITIALIZE/REPLACE xxn:(RET)
xxn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xxn:
```

The system scans the disk for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

The next step in the preservation process is to copy all the files from the distribution disk to the initialized blank disk.

The following command transfers files from the distribution to the backup disk and consolidates all the empty space at the end of the disk. In the command, yyn is the device name and unit number of your distribution disk, and xxn is the device name and unit number of your backup disk.

```
.SQUEEZE/OUTPUT:xxn: yyn:(RET)
.
```

NOTE

> Both the SQUEEZE command with the /OUTPUT:xxn: option and the COPY command transfer files from one device to another. SQUEEZE/OUTPUT:xxn: consolidates free space on the device at the same time. The procedures in this manual use the command that is most appropriate and most efficient for an individual operation. For a better understanding of all RT–11 keyboard monitor commands, refer to the *RT–11 System User's Guide.*

Next, copy the bootstrap to the backup disk.

```
.COPY/BOOT xxn:RT11FB.SYS xxn:(RET)
.
```

Now halt the processor, remove the distribution disk from Unit 0, and store it. For RC25 devices, use the hardware bootstrap to boot the backup disk on disk Unit 1; for all other devices, mount the backup disk on Unit 0, and use the hardware bootstrap to boot the backup disk.

```
RT-11FB V05.00
(Followed by any start-up file commands.)
.
```

NOTE

> If the backup disk does not boot, repeat the procedures to this point.

Next, remove the protection from all the files on the backup disk. The files on the distribution disk have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) When you copied the files to the backup disk, RT–11 also copied the protection. (Note the P that prints next to the file size in the directory.) For the rest of these procedures, you need to remove the protection from the backup disk. Use the following command:

```
.UNPROTECT/SYSTEM *.*(RET)
 Files unprotected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
     .
     .
     .
DK:zzzzzz.ttt
.
```

5.3 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Updated modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide.* Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to problems, until the module containing the error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

5.4 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3) and have planned the best arrangement of them on disks (Section 2.4), you can create the working system.

If you are using an RC25 disk, you create your working system by deleting selected components from the backup disk. Use the following command to delete unnecessary components:

```
.DELETE/SYSTEM/QUERY DU1:(RET)
  Files deleted?
DU1:aaaaaa.ttt? Y(RET)                    (to delete a specific file)
DU1:bbbbbb:ttt? N(RET)                    (to keep a specific file)
(and so on)
```

For all other disks, you create your working system by copying selected components to one or more disks.

Mount a blank disk in another disk unit and initialize it. If your disk is an RK05, format the disk before you initialize it.

RK05 disk

```
.FORMAT RKn:(RET)
RKn:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete
.

.INITIALIZE/BADBLOCKS RKn:(RET)
RKn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RKn:
.
```

Other type disk

```
.INITIALIZE/BADBLOCKS xxn:RET
xxn:/Initialize; Are you sure? YRET
?DUP-I-No bad blocks detected xxn:
.
```

or

```
.INITIALIZE/REPLACE xxn:RET
xxn:/Initialize; Are you sure? YRET
?DUP-I-No bad blocks detected xxn:
.
```

Copy the files you have selected from the backup disk to the disk that will become your working system disk. If you use the following command, RT–11 queries you about all the files on the backup disk, and you can choose the files it copies. In the command, yyn is the device name and unit number of your backup disk, and xxn is the device name and unit number of the disk to be your working system disk.

```
.COPY/SYSTEM/QUERY yyn: xxn:RET
 Files copied:
yyn:aaaaaa.ttt to xxn:aaaaaa.ttt? YRET          (to include a specific file)
yyn:bbbbbb.ttt to xxn:bbbbbb.ttt? NRET          (to exclude a specific file)
(and so on)
```

If you wish to create multiple working system disks, mount one blank disk for each working system you wish to create, format blank RK05 disks, and initialize the disks. Then, use the preceding COPY command to copy selected files to the appropriate disks.

5.5 Installing the Bootstrap on the Disk

Once you have created your working system disk(s), you need to install the bootstrap on all disks, except RC25. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xxn:RT11aa.SYS xxn:RET
```

In this command, you identify the device on which the monitor that contains the bootstrap information resides, the name of that monitor file (RT11BL, RT11SJ, RT11FB, or RT11XM), and the device on which you need to install the bootstrap. This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same disk.

Remove the backup disk from Unit 0 and store it for future updates. Remove the working system disk from Unit n and mount it on Unit 0.

Use the hardware bootstrap to boot the new working system disk.

```
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

5.6 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 5.10.)

5.7 Compressing the Disk

DIGITAL recommends that you compress the working system disk to make its free space contiguous. Consolidating free space allows you to use space on the disk that would otherwise be too fragmented to be usable.

Continue to run RT–11 from Unit 0 (Unit 1 for RC25 disk), and use the SQUEEZE command to compress free space. (The disk must be write-enabled.) The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE xxn:(RET)
xxn:/Squeeze; Are you sure? Y(RET)
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system disk.

5.8 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you preserve it on the backup medium of your choice. The following sections describe backing up on disk. If you want to back up the system on another device, the procedure may appear elsewhere in this manual; refer to the table of contents.

Insert a blank disk (write-enabled) in another disk unit and initialize the blank disk. If your disk is an RK05, format the disk before initializing it. In the following commands, xxn is the device name and unit number for the backup working system disk.

RK05 disk

```
.FORMAT RKn:(RET)
RKn:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS RKn:(RET)
RKn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RKn:
```

Other type disk

```
.INITIALIZE/BADBLOCKS xxn:(RET)
xxn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xxn:
.
```

or

```
.INITIALIZE/REPLACE xxn:(RET)
xxn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xxn:
.
```

Copy all the files in your working system. In the following command, yyn is the device name and unit number for your working system disk, and xxn is the device name and unit number for the backup working system disk.

```
.SQUEEZE/OUTPUT:xxn yyn:(RET)
.
```

Copy the bootstrap to the backup working system disk. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xxn:RT11aa.SYS xxn:(RET)
.
```

Store the backup working system disk. If you ever need to restore the working system, you can make a copy of the backup working system disk.

5.9 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, your working system must include at least the following components.

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

> If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP  .SYS
RT11FB.SYS
LP    .SYS
RK    .SYS
EDIT  .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
     .
     .
     .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

> If you have shifted output to a scope and the directory scrolls by too quickly to read, type (CTRL/S) to stop the display and (CTRL/Q) to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

5.9.1 Preparing the Background Demonstration Program

5.9.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

5.9.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

If your configuration does not include a line printer, use the console terminal.

```
.ASSIGN TT: LST:(RET)
.
```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG(RET)
```

(See Figure 5–3.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 5–3: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

      1                                         .TITLE  DEMOBG
      2                                         .IDENT  /V05.00/
      3                                 ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
      4                                 ; RING BELL IF FG JOB SENDS A MESSAGE.
      5
      6                                         .MCALL  .RCVDC,.PRINT
      7
      8 000000                          START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
      9 000034                                  .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
     10 000042  000777                          BR      .                         ;AND LOOP
     11
     12                                 ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE
     13
     14 000044                          MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
     15 000052                                  .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
     16 000106  000207                          RETURN                            ;AND RETURN FROM COMPLETION ROUTINE
     17
     18                                 ;       ASCII MESSAGES
     19                                         .NLIST BEX
     20 000110     007     200          BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL
     21
     22 000112     122     124     055  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111     106     040          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24                                 ;       .ASCII  /WELL DONE./
     25 000225     000                          .BYTE   0
     26
     27 000226                          AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
     28 000242                          BUFFER:                                   ;RCVDC MESSAGE AREA
     29         000000'                         .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R      BUFFER  000242R      MSGIN   000044R      ...V1 = 000003      ...V2 = 000027
BELL    000110R      MSG     000112R      START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG
```

5.9.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

5.9.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

5.9.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

5.9.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your system volume.

5.9.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

5.9.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
(CTRL/C)
(CTRL/C)
.
```

(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
(CTRL/F)
F>
(CTRL/C)
(CTRL/C)
B>
UNLOAD F(RET)
.
```

NOTE

If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

5.10 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Chapter 6
Installing a System Distributed on Hard Disk to Run on a Small Device

If RT–11 was distributed to you on hard disk (RL01/RL02, RK05, or RC25), and you intend to build a system to run on a small device (RX01, RX02, or RX50), perform the procedures described in this chapter. These procedures assume that your configuration includes two hard disk drives (not necessarily of the same type).

NOTE

> If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

To install your system, perform the steps summarized in the following list. Sections 6.1 through 6.10 describe the procedures involved in each step. Figure 6–1 shows the backup disk and diskettes you create when you install RT–11.

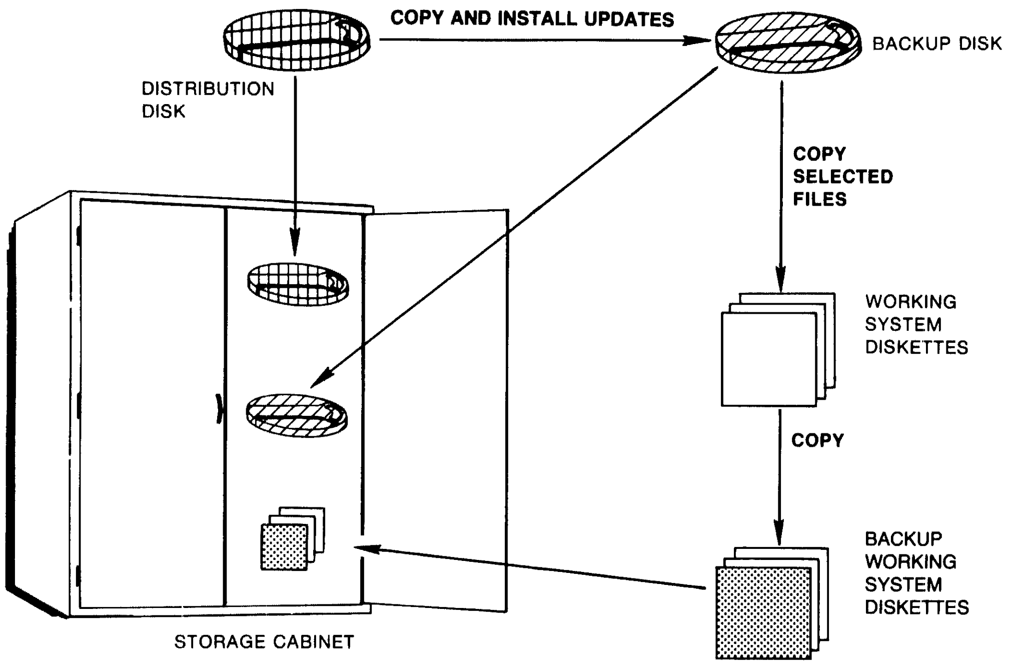
1. Bootstrap the distribution disk.
2. Preserve the distribution disk.
3. Install software updates.
4. Create the working system from chosen components.
5. Install the bootstrap on diskettes that need to be bootable.
6. Customize the system.
7. Compress each diskette.
8. Preserve the working system.

9. Test the working system.

10. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 6–1: Sample Backup Disk and Diskettes



6.1 Bootstrapping the Distribution Disk

The first procedure you perform when installing RT–11 is to bootstrap the distribution disk.

Begin by making sure that the processor is powered up but not running. Mount the distribution disk (write-protected) in disk Unit 0. The device name is DL, RK, or DU, depending on the type of disk device. Use the hardware bootstrap to boot the disk. (If your configuration does not include a hardware bootstrap, refer to Appendix B for the toggle-in software bootstrap.)

RT–11 should respond with the following message if you have successfully bootstrapped the disk:

```
RT-11FB V05.00

.SET TT NO QUIET

.TYPE V5USER.TXT

Welcome to RT-11 Version 5. RT-11 V5 provides new hardware support and
some major enhancements over Version 4. Among the new features are:

     ● 22-bit addressing support for Q-bus
     ● Virtual Disk facility (VM)
     ● Logical Disk Subsetting handler (LD)
     ● Single line, keypad command editor (SL)
     ● MSCP disk handler for RD51, RC25 and RA80 disks (DU)
     ● New multi-volume Backup/Restore utility (BUP)
     ● New Control File processor (IND)
     ● Concise Command Language (CCL)
     ● User Command Linkage (UCL)
     ● Many new and enhanced keyboard commands and options
     ● Many enhancements to existing RT-11 file utilities

Please use the HELP command; it describes the new commands and options
and the proper syntax with which to invoke them.

.
```

After you boot the system, you can create a start-up file containing any commands or sequence of commands that you would normally want to execute at the start of each terminal session. For example, if your console device is a video terminal, you may wish to execute a start-up file containing the command

```
SET TT SCOPE
```

each time that you use the system. This command allows you to use the DELETE key to backspace and erase characters from the screen.

You can update your start-up file at any time to change, add, or delete commands. When you first create the file, or after you update it, you can execute it with the @ command, so that the commands it contains become effective.

6.2 Preserving the Distribution Disk

The first procedure you perform with the running RT–11 system is to copy the distribution disk for backup, as a safety measure in case of machine failure or human error. You can then use the backup copy of the distribution to build your working system.

Mount a blank disk in the other disk drive, if necessary. (The disk in Unit 1 of RC25 devices is permanently mounted.) If the disk is an RK05 disk, format the disk, and then initialize it and cover any bad blocks on it.

If the disk is another type, initialize the disk and replace bad blocks (using the command INITIALIZE/REPLACE) or cover bad blocks (using the command INITIALIZE/BADBLOCKS). You have a choice of replacing or covering bad blocks if your disk is an RL01, RL02, RK06, or RK07. (Refer to the *RT–11 System User's Guide.*) If your disk is another type, for example, RC25, use INITIALIZE/BADBLOCKS to cover any bad blocks. If the disk contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal, but you can still use the disk. In the commands, xx is the permanent device name, and n is the unit number for your disk.

RK05 disk

```
.FORMAT RKn:(RET)
RKn:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS RKn:(RET)
RKn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected RKn:
```

Other type disk

```
.INITIALIZE/BADBLOCKS xxn:(RET)
xxn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xxn:
```

or

```
.INITIALIZE/REPLACE xxn:(RET)
xxn:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xxn:
```

The system scans the disk for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

The next step in the preservation process is to copy all the files from the distribution disk to the initialized blank backup disk.

The following command transfers files from the distribution disk to the backup disk and consolidates all the empty space at the end of the disk. In the command, yyn is the device name and unit number of your distribution disk, and xxn is the device name and unit number of your backup disk.

```
.SQUEEZE/OUTPUT:xxn: yyn:(RET)
.
```

NOTE

Both the SQUEEZE command with the /OUTPUT:xxn: option and the COPY command transfer files from one device to another. SQUEEZE/OUTPUT:xxn: consolidates free space on the device at the same time. The procedures in this manual use the command that is most appropriate and most efficient for an individual operation. For a better understanding of all RT–11 keyboard monitor commands, refer to the *RT–11 System User's Guide.*

Next, copy the bootstrap to the backup disk.

```
.COPY/BOOT xxn:RT11FB.SYS xxn:(RET)
.
```

Now halt the processor, remove the distribution disk from Unit 0, and store it. For RC25 devices, use the hardware bootstrap to boot the backup disk on disk Unit 1. For all other devices, mount the backup disk on Unit 0, and use the hardware bootstrap to boot the backup disk.

```
RT-11FB V05.00
```
(Followed by any start-up file commands.)
```
.
```

NOTE

> If the backup disk does not boot, repeat the procedures to this point.

Next, remove the protection from all the files on the backup disk. The files on the distribution disk have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) When you copied the files to the backup disk, RT–11 also copied the protection (note the P that prints next to the file size in the directory). For the rest of these procedures, you need to remove the protection from the backup disk. Use the following command:

```
.UNPROTECT/SYSTEM *.*(RET)
 Files unprotected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

6.3 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Updated modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide.* Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to problems, until the module containing the error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

6.4 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3) and have planned the best arrangement of them on diskettes (Section 2.4), you can create the working system by copying selected components to blank diskettes.

6.4.1 Initializing RX50 Diskettes

If you wish to create your working system on RX50 diskettes, insert a blank RX50 diskette in Unit 1 or 2 so you can initialize it. You can use the INITIALIZE command with the /BADBLOCKS option to initialize the diskette and to detect any bad blocks that may be on it. If the diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal. In the command, x is the device unit number.

NOTE

> RX50 diskettes are not formattable. DIGITAL recommends that you use only diskettes that have no bad blocks. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DUn:. If bad blocks exist on a diskette, copy the contents of the diskette to an error-free diskette, and dispose of the diskette with bad blocks.

```
.INITIALIZE/BADBLOCKS DUx:(RET)
DUx:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DUx:
```

There may be a delay of up to 1 minute while the system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Now, remove the newly initialized diskette and initialize an adequate number of blank diskettes, leaving one initialized, blank, write-enabled (write-protect notch uncovered) diskette inserted in Unit 1 or 2.

6.4.2 Formatting and Initializing RX01/RX02 Diskettes

If you plan to create your working system on RX01 diskettes, initialize a number of blank RX01 diskettes. For RX02 diskettes, format and initialize the diskettes. Insert a blank diskette in Unit 1. Diskettes are available in single-density but not double-density format. Therefore, to create RX02 diskettes, you must format all your blank diskettes as double-density diskettes. You can use the FORMAT utility program to format the blank diskette and the INITIALIZE command to initialize it. Use the /BADBLOCKS option with INITIALIZE to cover any bad blocks that may be on your diskette. If the diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal.

NOTE

> DIGITAL recommends that you use only diskettes that do not have bad blocks when you build a working system. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DYn:. You can use diskettes with bad blocks later, for working or data volumes.

For RX02 diskettes:

```
.FORMAT DY1:(RET)
DY1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DY1:(RET)
DY1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DY1:
```

For RX01 diskettes, just initialize the diskette:

```
.INITIALIZE/BADBLOCKS DX1:(RET)
DX1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DX1:
```

The system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Now, remove the newly formatted and initialized diskette. Repeat the process to create as many initialized blank diskettes as you need for the system that you have planned. Leave one diskette in Unit 1.

6.4.3 Copying the Selected Files onto Diskette

Use the COPY command with the /SYSTEM option to copy selected files from the backup disk to the diskettes that will become your working system diskettes. The /SYSTEM option is required for copying .SYS files only if wildcards are used in the input file specification. In the command, yyn is the device name and unit number of your backup disk, and xxn is the device name and unit numbers of the diskettes to be your working system diskettes.

```
.COPY yyn:filnam.typ xxn:filnam.typ(RET)
.
```

You can use the following command to avoid typing numerous file specifications. RT–11 queries you about all the files on the diskette, and you choose the files it copies.

```
.COPY/SYSTEM/QUERY yyn: xxn:(RET)
  Files copied:
yyn:aaaaaa.ttt to xxn:aaaaaa.ttt? Y(RET)          (to include a specific file)
yyn:bbbbbb.ttt to xxn:bbbbbb.ttt? N(RET)          (to exclude a specific file)
(and so on)
```

When you have copied all the files you planned for the working system diskette, label the diskette "RT–11 Working System V05 x/y" (where x is the diskette number, and y is the number of diskettes in your working system).

6.5 Installing the Bootstrap on Any Diskettes That Need to Be Bootable

Once you have created your working system diskettes, you need to install the bootstrap on any diskettes that must be bootable (that is, that you can use as system diskettes). Generally, any diskette that includes a monitor file and system device handler should be bootable (but remember that the diskette would need SWAP.SYS and, for the SJ monitor, TT.SYS).

Insert the diskette on which you need to install the bootstrap. In the command, xxn represents the device name and unit number of your diskette, and aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xxn:RT11aa.SYS xxn:(RET)
.
```

In this command, you need to identify the device on which the monitor that contains the bootstrap information resides, the name of that monitor file, and the device on which you need to install the bootstrap. This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same diskette.

Then, insert working system diskette 1 in Unit 0, and boot your working system. Use the hardware bootstrap if possible. If not, type the following command; xxn represents the diskette to boot.

```
.BOOT xxn:(RET)

RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

Store your updated backup distribution disk for future updates.

6.6 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 6.10.)

6.7 Compressing Each Diskette

DIGITAL recommends that you compress each working system diskette to make its free space contiguous. Consolidating free space allows you to use space on the diskette that would otherwise be too fragmented to be usable.

Continue to run RT–11 from Unit 0, and use the SQUEEZE command to compress free space. The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE xx0:RET
xx0:/Squeeze; Are you sure? YRET
```

There is a delay while the squeeze operation takes place.

```
RT-11aa V05.00
```
(Followed by any start-up file commands.)
```
.
```

The system automatically reboots when you compress a system diskette.

Then, insert the next diskette that you need to compress in Unit 1.

```
.SQUEEZE xx1:RET
xx1:/Squeeze; Are you sure? YRET
.
```

Replace the diskette in Unit 1 with the next one you need to compress, and repeat this procedure for all the diskettes.

NOTE

When you compress a diskette with system files (.SYS), PIP warns you to reboot. Do reboot as advised. When you compress a system diskette, the system automatically reboots.

6.8 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files and preserve the system on backup diskettes.

Use the following command to protect all the files on the system diskette:

```
.PROTECT/SYSTEM *.*(RET)
 Files protected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

To protect files on other diskettes in the working system, insert each diskette in Unit 1 and use the following command:

```
.PROTECT/SYSTEM xx1:*.*(RET)
 Files protected:
xx1:aaaaaa.ttt
xx1:bbbbbb.ttt
xx1:cccccc.ttt
xx1:dddddd.ttt
      .
      .
      .
xx1:zzzzzz.ttt
.
```

Next, copy the working system to backup diskettes. Insert a blank diskette in Unit 1 with RT–11 still booted from Unit 0. Format (if you are using RX02 diskettes) and initialize the appropriate number of diskettes.

Copy all the files in your working system. You can use the SQUEEZE/OUTPUT:xx1: xx0: command to copy any bootable diskettes. Remember that you must use SQUEEZE/WAIT/OUTPUT:xx1: xx0: and change diskettes to copy the diskettes that are not bootable. (Refer to Section 3.4.) Also remember to copy the bootstrap to any diskettes that need to be bootable.

Store the backup working system diskettes. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

6.9 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, your working system must include at least the following components.

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Insert a blank diskette in Unit 1. If you are using RX02 diskettes, format the disk first.

```
.FORMAT DY1:(RET)
DY1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete
```

Initialize all diskettes, and check for bad blocks.

```
.INITIALIZE/BADBLOCKS xx1:(RET)
xx1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected xx1:

.ASSIGN xx1: DK:(RET)
```

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP  .SYS
RT11FB.SYS
LP    .SYS
XX    .SYS
EDIT  .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
     .
     .
     .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

> If you have shifted output to a scope and the directory scrolls by too quickly to read, type CTRL/S to stop the display and CTRL/Q to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

6.9.1 Preparing the Background Demonstration Program

6.9.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

6.9.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

> If your configuration does not include a line printer, use the console terminal.
>
> ```
> .ASSIGN TT: LST:(RET)
> .
> ```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOBG(RET)
```

(See Figure 6–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 6–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

                                          .TITLE  DEMOBG
                                          .IDENT  /V05.00/
                                  ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
                                  ; RING BELL IF FG JOB SENDS A MESSAGE.

                                          .MCALL  .RCVDC,.PRINT

     000000                       START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
     000034                               .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
     000042  000777                       BR      .                         ;AND LOOP

                                  ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE

     000044                       MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
     000052                               .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
     000106  000207                       RETURN                            ;AND RETURN FROM COMPLETION ROUTINE

                                  ;       ASCII MESSAGES
                                          .NLIST BEX
     000110     007     200       BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL

     000112     122     124   055 MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     000147     111     106   040         .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
                                  ;       .ASCII  /WELL DONE./
     000225     000                       .BYTE   0

     000226                       AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
     000242                       BUFFER:                                   ;RCVDC MESSAGE AREA
             000000'                      .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA     000226R        BUFFER  000242R       MSGIN    000044R       ...V1 = 000003       ...V2 = 000027
BELL     000110R        MSG     000112R       START    000000RG

. ABS.   000000    000   (RW,I,GBL,ABS,OVR)
         000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG
```

```
.
```

6.9.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

6.9.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME DEMOBG.BAK DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

6.9.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

6.9.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your data volume (DK).

6.9.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

6.9.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```
(The bell stops ringing.)
```
.DIRECTORY(RET)
 dd-mmm-yy
```
(The directory of the device DK prints on the terminal.)
```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
(CTRL/C)
(CTRL/C)
.
```
(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
(CTRL/F)
F>
(CTRL/C)
(CTRL/C)
B>
UNLOAD F(RET)
.
```

NOTE

If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

6.10 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Chapter 7
Installing a System Distributed on RX02 to Run on RX02

If RT–11 was distributed to you on RX02 double-density diskette, and you intend to build a system to run on double-density diskette, perform the procedures described in this chapter.

Your distribution kit contains two exact copies of the RT–11 operating system on RX02 diskettes. Store one copy of the RX02 diskettes in a safe place, and do not modify their contents; they provide a backup (master) copy of the distributed RT–11 operating system.

Store the second copy of diskettes in a safe place after installing software updates and creating your working system diskettes. Use these diskettes to create another working system. If these diskettes are ever damaged, use the master copy to reproduce them.

NOTE

> If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

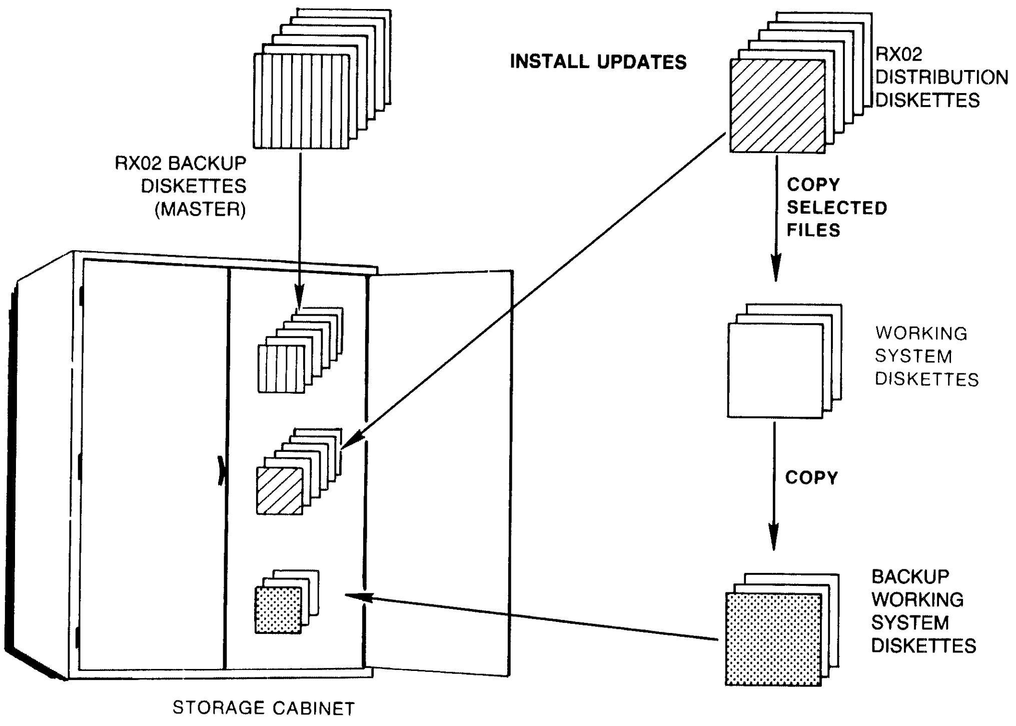
To install your system, perform the steps summarized in the following list. Sections 7.1 through 7.9 describe the procedures involved in each step. Figure 7–1 shows the diskettes you create when you install RT–11.

1. Bootstrap the distribution diskette.
2. Install software updates.
3. Create the working system from chosen components.
4. Install the bootstrap on diskettes that need to be bootable.
5. Customize the system.

6. Compress each diskette.
7. Preserve the working system.
8. Test the working system.
9. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 7–1: Sample Backup Diskettes


INSTALL UPDATES
RX02 DISTRIBUTION DISKETTES
RX02 BACKUP DISKETTES (MASTER)
COPY SELECTED FILES
WORKING SYSTEM DISKETTES
COPY
BACKUP WORKING SYSTEM DISKETTES
STORAGE CABINET

7.1 Bootstrapping the Distribution Diskette

The first procedure you perform when installing RT–11 is to bootstrap distribution diskette 1.

Begin by making sure that the processor is powered up but not running. Insert the distribution diskette labeled 0/5 in Unit 0 (which has the physical device name DY0:). Use the hardware bootstrap to boot the diskette. (If your configuration does not include a hardware bootstrap, see Appendix B for toggle-in software bootstraps.)

RT–11 should respond with the following message if you have successfully bootstrapped the diskette:

```
RT-11FB V05.00

.SET TT NO QUIET

.TYPE V5USER.TXT

Welcome to RT-11 Version 5. RT-11 V5 provides new hardware support and
some major enhancements over Version 4. Among the new features are:

      ● 22-bit addressing support for Q-bus
      ● Virtual Disk facility (VM)
      ● Logical Disk Subsetting handler (LD)
      ● Single line, keypad command editor (SL)
      ● MSCP disk handler for RD51, RC25 and RA80 disks (DU)
      ● New multi-volume Backup/Restore utility (BUP)
      ● New Control File processor (IND)
      ● Concise Command Language (CCL)
      ● User Command Linkage (UCL)
      ● Many new and enhanced keyboard commands and options
      ● Many enhancements to existing RT-11 file utilities

Please use the HELP command; it describes the new commands and options
and the proper syntax with which to invoke them.

.
```

After you boot the system, you can create a start-up file containing any commands or sequence of commands that you would normally want to execute at the start of each terminal session. For example, if your console device is a video terminal, you may wish to execute a start-up file containing the command

```
SET TT SCOPE
```

each time that you use the system. This command allows you to use the DELETE key to backspace and erase characters from the screen.

You can update your start-up file at any time to change, add, or delete commands. When you first create the file, or after you update it, you can execute it with the @ command, so that the commands it contains become effective.

Next, remove the protection from all the files on the distribution diskettes. The files on the distribution diskettes have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) (Note the P that prints next to the file size in the directory.) For the rest of these procedures, you need to remove the protection from these diskettes. Type the following command to remove it from the files on distribution diskette 1:

```
.UNPROTECT/SYSTEM *.*(RET)
 Files unprotected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
       .
       .
       .
DK:zzzzzz.ttt
.
```

Insert distribution diskette 2 in Unit 1 and type the following command:

```
.UNPROTECT/SYSTEM DY1:*.*(RET)
 Files unprotected:
DY1:aaaaaa.ttt
DY1:bbbbbb.ttt
DY1:cccccc.ttt
DY1:dddddd.ttt
      .
      .
      .
DY1:zzzzzz.ttt
.
```

Replace distribution diskette 2 with distribution diskette 3 in Unit 1 and repeat this command. In the same way, remove protection from the files on the remaining distribution diskettes.

7.2 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Updated modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide.* Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to problems, until the module containing the error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

7.3 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3) and have planned the best arrangement of them on diskettes (Section 2.4), you can create the working system by copying selected components to formatted and initialized blank diskettes.

Start by formatting and initializing a number of blank diskettes. Insert a blank diskette in Unit 1 so you can format and initialize it. Diskettes are available in single-density but not double-density format. Therefore, you must reformat all your blank diskettes as double-density diskettes. You can use the FORMAT utility program, which is included on distribution diskette 1, to format the blank diskettes and the INITIALIZE command to initialize them. Use the /BADBLOCKS option with INITIALIZE to cover any bad blocks that may be on your diskettes. If a diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal.

NOTE

> DIGITAL recommends that you use only diskettes that do not have bad blocks when you build a working system. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DYn:. You can use diskettes with bad blocks later for working or data volumes.

```
.FORMAT DY1:(RET)
DY1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DY1:(RET)
DY1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DY1:
```

The system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Repeat the process to create as many initialized blank diskettes as you need for the system that you have planned, leaving one initialized, blank diskette in Unit 1.

Then, use the COPY command with the /SYSTEM option to copy selected files from distribution diskette 1 to the diskette that becomes your working system diskette. The /SYSTEM option is required for copying .SYS files only if wildcards are used in input file specifications.

```
.COPY DY0:filnam.typ DY1:filnam.typ(RET)
.
```

You can use the following command to avoid typing numerous file specifications. RT–11 queries you about all the files on the diskette, and you choose the files it copies.

```
.COPY/SYSTEM/QUERY DY0: DY1:(RET)
  Files copied:
DY0:aaaaaa.ttt to DY1:aaaaaa.ttt? Y(RET)        (to include a specific file)
DY0:bbbbbb.ttt to DY1:bbbbbb.ttt? N(RET)        (to exclude a specific file)
(and so on)
```

To copy files from nonbootable diskettes, you have to alternate diskettes.

Use the SET command to set the USR to NOSWAP.

```
.SET USR NOSWAP(RET)
.
```

Type the following command, where filnam.typ is the name of the file you want to copy. In this case, you cannot use the /QUERY option; you must specify individual files.

```
.COPY/WAIT DY1:filnam.typ DY0:filnam.typ(RET)

Mount input volume in DY1:; Continue?
```

Place the diskette containing the file you want to copy in Unit 1.

```
Y(RET)
Mount output volume in DY0:; Continue?
```

Replace the system diskette in Unit 0 with the diskette to which you want to copy filnam.typ.

```
Y(RET)
Mount system volume in DY0:; Continue?
```

Replace the diskette in Unit 0 with backup diskette 1.

```
Y(RET)
.
```

Repeat this procedure to copy all the files you planned for the working system diskette. When you have copied all the files, label the diskette "RT–11 Working System V05 1/x" (where x is the number of diskettes in your working system). Repeat these procedures to create the other diskettes in the working system.

When you have created and labeled all the working system diskettes, you can permit the USR to swap again.

```
.SET USR SWAP(RET)
.
```

7.4 Installing the Bootstrap on Any Diskettes That Need to Be Bootable

Once you have created your system, you need to install the bootstrap on any diskettes that must be bootable (that is, that you can use as system diskettes). Generally, any diskette that includes a monitor file and system device handler should be bootable (but do not forget that the diskette would need SWAP.SYS and, for the SJ monitor, TT.SYS).

Insert in Unit 1 the diskette on which you need to install the bootstrap. In the command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT DY1:RT11aa.SYS DY1:(RET)
.
```

In this command, you identify the device on which the monitor that contains the bootstrap information resides, the name of that monitor file, and the device on which you need to install the bootstrap. This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same diskette.

Then, insert working system diskette 1 in Unit 0, and use the hardware bootstrap to boot your working system.

```
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

Store the updated distribution diskettes for future updates.

7.5 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 7.9.)

7.6 Compressing Each Diskette

DIGITAL recommends that you compress each working system diskette to make its free space contiguous. Consolidating free space allows you to use space on the diskette that would otherwise be too fragmented to be usable.

Continue to run RT–11 from Unit 0 and use the SQUEEZE command to compress free space. The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE DY0:(RET)
DY0:/Squeeze; Are you sure? Y(RET)
```

There is a delay while the squeeze operation takes place.

```
RT-11xx V05.00
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system diskette.

Then insert the next diskette that you need to compress in Unit 1.

```
.SQUEEZE DY1:(RET)
DY1:/Squeeze; Are you sure? Y(RET)
.
```

Replace the diskette in Unit 1 with the next one you need to compress, and repeat this procedure for all the diskettes.

NOTE

When you compress a diskette with system files (.SYS), PIP warns you to reboot. Do reboot as advised. When you compress a system diskette, the system automatically reboots.

7.7 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files and preserve the system on backup diskettes.

Use the following command to protect all the files on the system diskette:

```
.PROTECT/SYSTEM *.*(RET)
 Files protected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

To protect files on other diskettes in the working system, insert each diskette in Unit 1 and use the following command:

```
.PROTECT/SYSTEM DY1:*.*(RET)
 Files protected:
DY1:aaaaaa.ttt
DY1:bbbbbb.ttt
DY1:cccccc.ttt
DY1:dddddd.ttt
      .
      .
      .
DY1:zzzzzz.ttt
.
```

Next, copy the working system to backup diskettes. Insert each blank diskette in Unit 1 with RT–11 still booted from Unit 0. Format and initialize the appropriate number of diskettes.

Copy all the files in your working system. You can use the SQUEEZE/OUTPUT:DY1: DY0: command to copy any bootable diskettes. Remember that you must use SQUEEZE/WAIT/OUTPUT:DY1: DY0: and change diskettes to copy the diskettes that are not bootable. Also remember to copy the bootstrap to any diskettes that need to be bootable.

Store the backup diskettes. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

7.8 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, your working system must include at least the following components.

SWAP.SYS
RT11FB.SYS
DY.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Insert a blank diskette in Unit 1.

```
.FORMAT DY1:(RET)
DY1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DY1:(RET)
DY1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DY1:

.ASSIGN DY1: DK:(RET)
```

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP   .SYS
RT11FB.SYS
LP     .SYS
DY     .SYS
EDIT   .SAV
MACRO  .SAV
SYSMAC.SML
LINK   .SAV
PIP    .SAV
      .
      .
      .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

> If you have shifted output to a scope and the directory scrolls by too quickly to read, type CTRL/S to stop the display and CTRL/Q to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

7.8.1 Preparing the Background Demonstration Program

7.8.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

7.8.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

> If your configuration does not include a line printer, use the console terminal.
>
> ```
> .ASSIGN TT: LST:(RET)
> .
> ```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOBG(RET)
```

(See Figure 7–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 7–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

                                          .TITLE  DEMOBG
                                          .IDENT  /V05.00/
                                  ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
                                  ; RING BELL IF FG JOB SENDS A MESSAGE.

                                          .MCALL  .RCVDC,.PRINT

  000000                          START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
  000034                                  .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
  000042  000777                          BR      .                         ;AND LOOP

                                  ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE

  000044                          MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
  000052                                  .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
  000106  000207                          RETURN                            ;AND RETURN FROM COMPLETION ROUTINE

                                  ;       ASCII MESSAGES
                                          .NLIST BEX
  000110     007     200          BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL

  000112     122     124     055  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
  000147     111     106     040          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
                                  ;       .ASCII  /WELL DONE./
  000225     000                          .BYTE   0

  000226                          AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
  000242                          BUFFER:                                   ;RCVDC MESSAGE AREA
          000000'                         .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R        BUFFER  000242R        MSGIN   000044R        ...V1 = 000003        ...V2 = 000027
BELL    000110R        MSG     000112R        START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG
```

```
.
```

7.8.1.3 Link the Background Demonstration Program —

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

7.8.1.4 Run the Background Demonstration Program —

Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

7.8.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

7.8.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your data volume (DK).

7.8.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

7.8.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```
(The bell stops ringing.)
```
.DIRECTORY(RET)
 dd-mmm-yy
```
(The directory of the device DK prints on the terminal.)
```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
(CTRL/C)
(CTRL/C)
.
```
(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
(CTRL/F)
F>
(CTRL/C)
(CTRL/C)
B>
UNLOAD F(RET)
.
```

NOTE

If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

7.9 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Chapter 8
Installing a System Distributed on Magtape to Run on Hard Disk

If RT–11 was distributed to you on magtape, and you intend to build a system to run on hard disk, perform the procedures described in this chapter. These procedures assume that your configuration includes only one disk drive. With two disk drives, you still have to perform the logical steps summarized in the following list; however, you can simplify some procedures by using the additional disk drive. Sections 8.1 through 8.11 describe the procedures involved in each step. Figure 8–1 shows the various backup magtapes you create when you install RT–11.

NOTE

If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

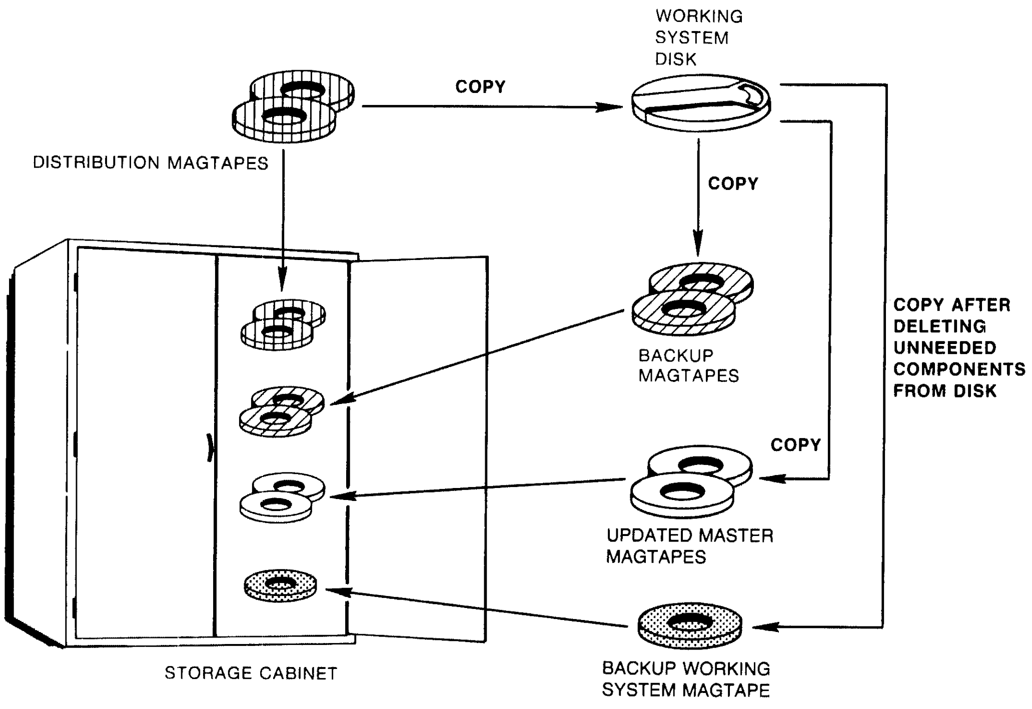
1. Bootstrap the first distribution magtape.
2. Preserve both distribution magtapes.
3. Install software updates.
4. Create updated master magtapes.
5. Create the working system on the disk from chosen components.
6. Install the bootstrap on the disk.
7. Customize the system.
8. Compress the disk.
9. Preserve the working system.

10. Test the working system.

11. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 8–1: Sample Backup Magtapes



8.1 Bootstrapping the First Distribution Magtape

The first procedure you perform when installing RT–11 is to bootstrap the first distribution magtape.

Begin by making sure that the processor is powered up but not running. Mount the distribution magtape labeled 1/2 (remove the write ring from the back of the tape reel). Manually position the magtape at the load point if the magtape is not in this position.

If your system has a hardware bootstrap capable of bootstrapping the magtape, boot the tape and proceed. If not, use the appropriate toggle-in software bootstrap in Appendix B.

The magtape moves as the primary bootstrap loads the secondary bootstrap file MSBOOT.BOT.

MSBOOT responds on the terminal.

```
MSBOOT V0x-yy
*
```

The next step depends on whether you have a TM11 or TJU16 as a magtape device. If you have a TM11, use the magtape build program named

MDUP.MT. If you have a TJU16, use the magtape build program named MDUP.MM.

```
MDUP.MT(RET)
```

or

```
MDUP.MM(RET)
```

The magtape moves as the specified MDUP program is loaded.

```
MDUP V0x.yy
*
```

The next procedure is to build a minimal system on the disk. Start by initializing the system disk. If your disk is an RK05, be sure to use a formatted disk.

You need to initialize the system disk and scan it for bad blocks before you can copy system files to it. You specify these operations to MDUP by entering the device name followed by combinations of the following options.

/Z to initialize the disk

/B to scan it for bad blocks

Mount a formatted disk (write-enabled).

Use the following command, where xx is the permanent device name (RK, DL, DM, DP, or DU) for your disk.

```
xx0:/Z/B(RET)
```

The system scans the disk for bad blocks and creates a directory.

```
*
```

Now you can build a minimal system on the disk. Depending on the type of tape drive you have, use one of the two following commands (where xx is the permanent device name for your disk).

If the magtape is TM11:

```
xx0:A=MT0:(RET)
```

If the magtape is TJU16:

```
xx0:A=MM0:(RET)
```

The tape moves while the system copies the monitor, swap file, system device handler, terminal handler, line printer handler, magtape handler, PIP, DUP, and DIR to the disk. When the files are copied, MDUP boots the minimal system from the disk.

```
RT-11FB V05.00
?KMON-F-Command file not found
.
```

You are now running from the minimal system on the disk. This minimal system supports enough file maintenance commands to allow you to complete the building process.

NOTE

> MDUP does not support automatic replacement of bad blocks for RK06, RK07, RL01, and RL02 disks. If your disk is an RK06, RK07, RL01, or RL02 and you want automatic bad block replacement, you must initialize a second disk and copy your files to it at a later time.

If you wish to set the date and time, use the DATE command to set the date (where dd–mmm–yy is the day, month, and year in the form 10–JAN–83); use the TIME command to set the time (where hh:mm:ss is the hour, minutes, and seconds).

```
.DATE dd-mmm-yy(RET)
.TIME hh:mm:ss(RET)
.
```

Now copy the rest of the files from distribution magtape 1 to the disk. Use the following command, where xx is MT or MM.

```
.COPY/SYSTEM/NOREPLACE xx0:*.* DK:(RET)
  Files copied:
xx0:aaaaaa.ttt to DK:aaaaaa.ttt
              .
              .
              .
xx0:zzzzzz.ttt to DK:zzzzzz.ttt
?PIP-W-Reboot
.
```

NOTE

> You must use the /NOREPLACE option in this command, so that the files you copied to the disk when you built the minimal system will not be copied again. The system prints a message to tell you which files it does not copy (for example, *SWAP.SYS not copied*).

Reboot as advised.

```
.BOOT SY:(RET)
```

8.2 Preserving Both Distribution Magtapes

Copy both distribution magtapes for backup, as a safety measure in case of machine failure or human error. To back up the magtapes, complete the following steps:

1. Replace distribution magtape 1 in the tape drive with a blank magtape.
2. Copy all the distribution files from the disk to the blank backup magtape.
3. Replace the backup magtape with distribution magtape 2.
4. Copy everything from distribution magtape 2 to disk.
5. Replace distribution magtape 2 with a blank magtape.
6. Copy those files that were on distribution magtape 2 from the disk to the blank backup magtape 2.

Step 1

Remove distribution magtape 1 and mount a blank magtape on the tape drive. Leave the write ring in the back of the reel, and make sure that the tape is positioned at the load point.

Step 2

Copy all files from the disk to the magtape by invoking the indirect command file DISMT1.COM (procedure follows). DISMT1.COM initializes the blank magtape, writes the primary bootstrap on it, and copies a duplicate of distribution magtape 1. Before you invoke this indirect command file, use the ASSIGN command to assign the logical name DIS: to your disk device and the logical name TAP: to your tape drive. In the commands, xx is MT or MM, and yy is your disk.

```
.ASSIGN xxn: TAP:(RET)
.ASSIGN yyn: DIS:(RET)
.@DISMT1(RET)
```

(The commands in the indirect file print on the terminal.)

Step 3

Rewind the newly created backup magtape, remove it, and label it "Backup RT–11 V05 1/2".

Remove the write ring from the back of distribution magtape 2, if necessary, and mount the reel.

Step 4

Copy all files from distribution magtape 2 to the disk. Use the following command, where xx is MT or MM.

```
.COPY/SYSTEM/NOREPLACE xx0:*.* DK:(RET)
  Files copied:
xx0:aaaaaa.ttt to DK:aaaaaa.ttt
               .
               .
               .
xx0:zzzzzz.ttt to DK:zzzzzz.ttt
?PIP-W-Reboot
.
```

Reboot the disk.

Step 5

Rewind the distribution magtape and remove it. Mount another blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point.

Step 6

Use the indirect command file DISMT2.COM to initialize the magtape and then to duplicate the copies of files from distribution magtape 2 on the newly initialized blank magtape (procedure follows). Note that this tape is not a bootable magtape. Use the ASSIGN command to assign the logical name DIS: to your disk device and the logical name TAP: to your tape drive. In the commands, xx is MT or MM, and yy is your disk.

```
.ASSIGN xxn: TAP:(RET)
.ASSIGN yyn: DIS:(RET)
.@DISMT2(RET)
```

(The commands in the indirect file print on the terminal.)

Then, rewind the newly created backup magtape, remove it, and label it "Backup RT–11 V05 2/2".

Store the distribution magtapes.

8.3 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Updated modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide.* Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to problems, until the module containing the error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

8.4 Creating Updated Master Magtapes

Once you have installed updates in the components you copied to the disk, copy all the files to blank magtapes. These magtapes can serve as updated masters. As other mandatory updates are published, you can copy the affected component from the updated master magtape to the disk, install the update, and copy the file back to the updated master. In this way, you can make sure that you have up-to-date versions of all RT–11 components, even if your working system is destroyed.

Mount another blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point.

Use the following procedures to initialize the magtape and copy the same files to updated master magtape 1 as you copied to backup magtape 1 (including the bootstrap) and the same files to updated master magtape 2 as you copied to backup magtape 2. In the commands, xx is MT or MM, and yy is your disk.

```
.ASSIGN xxn: TAP:(RET)
.ASSIGN yyn: DIS:(RET)
.@DISMT1(RET)
```
(The commands in the indirect file print on the terminal.)

Rewind the newly created magtape, remove it, and label it "Updated Master RT–11 V05 1/2". Mount another blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point.

Use the following procedures to initialize the magtape and copy the same files (now updated) to this magtape that you copied to backup magtape 2. In the command, xx is MT or MM, and yyn is your disk.

```
.ASSIGN xxn: TAP:(RET)
.ASSIGN yyn: DIS:(RET)
.@DISMT2(RET)
```
(The commands in the indirect file print on the terminal.)

Then, rewind the newly created magtape, remove it, and label it "Updated Master RT–11 V05 2/2".

Store the two updated master magtapes, and use them when you install any future software updates.

8.5 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3), backed up the distribution magtapes, and created updated masters, you can create the working system by deleting unneeded components from the disk.

The following command queries you about all the files on the disk. Choose the files you want to delete.

```
.DELETE/SYSTEM/QUERY *.*(RET)
 Files deleted:
DK:aaaaaa.ttt? Y(RET)                    (to delete a specific file)
DK:bbbbbb.ttt? N(RET)                    (to retain a specific file)
```
(and so on)

8.6 Installing the Bootstrap on the Disk

Once you have created your working system, copy the bootstrap from the monitor file of your choice (RT11BL, RT11SJ, RT11FB, or RT11XM) to the disk. In the command, xx is the permanent device name for your disk, and aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xx0:RT11aa.SYS xx0:RET
.
```

Then, halt the processor, and use the hardware bootstrap to boot the working system disk.

```
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

NOTE

If the disk does not boot, repeat the procedures to this point.

8.7 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 8.11.)

8.8 Compressing the Disk

DIGITAL recommends that you compress the working system disk to make its free space contiguous. Consolidating free space allows you to use space on the disk that would otherwise be too fragmented to be usable.

Use the SQUEEZE command for this procedure. (The disk must be write-enabled.) The squeeze operation does not move files with the .BAD file type. In the command, xx is the device name for your disk.

```
.SQUEEZE xxn:RET
xxn:/Squeeze; Are you sure? YRET
RT-11aa V05.00
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system disk.

8.9 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files in it and preserve the system on the backup medium of your choice.

Use the following procedures to back up the disk on magtape. First, protect all the files on the disk.

```
.PROTECT/SYSTEM *.*(RET)
 Files protected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

Next, initialize another blank magtape and copy files to it. You may be able to fit the working system on one backup magtape. You can place approximately 3000 blocks on a 600 foot 800 bits/in magtape.

Mount a blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point. Use the following command, where xx is MT or MM:

```
.INITIALIZE/FILE:MBOOT.BOT xx0:(RET)
xx0:/Initialize; Are you sure? Y(RET)
.
```

Then, copy all the files from the disk to the magtape, in the order shown. Essentially, you should use the following procedure whenever you build a bootable magtape. If you do not copy the files in this order, bootstrapping the magtape will be a painfully long process. Keep track of all the files you copy, so that when you copy the rest of the files, you will know which files you have already copied. If you have deleted some device handlers from the system, the log that prints on the terminal will not include the deleted device handlers.

The following COPY procedure shows a log from a system that includes all the device handlers. In the commands, xx is MT, MM, or MS, and aa is BL, SJ, FB, or XM.

```
.COPY MSBOOT.BOT xx0:MSBOOT.BOT/POSITION:-1(RET)

.COPY/SYSTEM MDUP.* xx0:MDUP.*/POSITION:-1(RET)
 Files copied:
DK:MDUP.MM to xx0:MDUP.MM
DK:MDUP.MT to xx0:MDUP.MT
DK:MDUP.MS to xx0:MDUP.MS

.COPY SWAP.SYS xx0:SWAP.SYS/POSITION:-1(RET)

.COPY RT11aa.SYS xx0:RT11aa.SYS/POSITION:-1(RET)
```

```
.COPY TT.SYS xx0:TT.SYS/POSITION:-1(RET)

.COPY/SYSTEM D%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:DT.SYS to xx0:DT.SYS
DK:DP.SYS to xx0:DP.SYS
DK:DX.SYS to xx0:DX.SYS
      .
      .
      .

.COPY/SYSTEM R%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:RK.SYS to xx0:RK.SYS
DK:RF.SYS to xx0:RF.SYS
      .
      .
      .

.COPY/SYSTEM M%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:MM.SYS to xx0:MM.SYS
DK:MT.SYS to xx0:MT.SYS
DK:MS.SYS to xx0:MS.SYS
      .
      .
      .

.COPY/SYSTEM L%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:LS.SYS to xx0:LS.SYS
DK:LP.SYS to xx0:LP.SYS
      .
      .
      .

.COPY PIP.SAV xx0:PIP.SAV/POSITION:-1(RET)

.COPY DUP.SAV xx0:DUP.SAV/POSITION:-1(RET)

.COPY DIR.SAV xx0:DIR.SAV/POSITION:-1(RET)

.
```

Copy the rest of the files. The following command queries you about all the files on the disk so that you can choose the files it copies.

```
.COPY/SYSTEM/QUERY DK: xx0:/POSITION:-1(RET)
 Files copied:
DK:aaaaaa.ttt to xx0:aaaaaa.ttt? Y(RET)        (to include a specific file)
DK:zzzzzz.ttt to xx0:zzzzzz.ttt? N(RET)        (to exclude a specific file)
(and so on)
.
```

Rewind the newly created backup working system magtape, remove it, label it "Backup Working System RT–11 V05", and then store it.

8.10 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, you need a terminal with a bell, and your working system must include at least the following components.

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP  .SYS
RT11FB.SYS
LP    .SYS
DL    .SYS
MT    .SYS
EDIT  .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
      .
      .
      .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to a scope and the directory scrolls by too quickly to read, type CTRL/S to stop the display and CTRL/Q to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

8.10.1 Preparing the Background Demonstration Program

8.10.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

8.10.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

If your configuration does not include a line printer, use the console terminal.

```
.ASSIGN TT: LST:(RET)
.
```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG(RET)
```

(See Figure 8–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 8–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

                                       .TITLE  DEMOBG
                                       .IDENT  /V05.00/
                             ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
                             ; RING BELL IF FG JOB SENDS A MESSAGE.

                                       .MCALL  .RCVDC,.PRINT

   000000                    START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
   000034                            .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
   000042  000777                    BR      .                         ;AND LOOP

                             ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE

   000044                    MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
   000052                            .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
   000106  000207                    RETURN                            ;AND RETURN FROM COMPLETION ROUTINE

                             ;       ASCII MESSAGES
                                     .NLIST BEX
   000110     007     200    BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL

   000112     122     124     055    MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
   000147     111     106     040            .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
                             ;       .ASCII  /WELL DONE./
   000225     000                    .BYTE   0

   000226                    AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
   000242                    BUFFER:                                   ;RCVDC MESSAGE AREA
           000000'                   .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R       BUFFER  000242R       MSGIN   000044R       ...V1 = 000003       ...V2 = 000027
BELL    000110R       MSG     000112R       START   000000RG

. ABS.  000000    000    (RW,I,GBL,ABS,OVR)
        000242    001    (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG

.
```

8.10.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

8.10.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

8.10.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

8.10.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your system volume.

8.10.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

8.10.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```
(The bell stops ringing.)
```
.DIRECTORY(RET)
 dd-mmm-yy
```
(The directory of the device DK prints on the terminal.)
```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
(CTRL/C)
(CTRL/C)
.
```
(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
(CTRL/F)
F>
(CTRL/C)
(CTRL/C)
B>
UNLOAD F(RET)
.
```

NOTE

If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

8.11 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Chapter 9
Installing RT–11 on a MICRO/PDP–11 Processor

If you are installing RT–11 to run on a MICRO/PDP–11 processor, perform the procedures described in this chapter. You will have received the RT–11 operating system on RX50 diskettes, and will be installing it to run on a nonremovable RD51 disk.

Your distribution kit contains two exact copies of the RT–11 operating system on RX50 diskettes. Store one copy in a safe place, and do not modify the contents; this provides a backup (master) copy of the distributed RT–11 operating system.

Store the second copy in a safe place after copying the contents onto the RD51 disk. Use these diskettes if it is ever necessary to copy their contents onto the permanent disk, for example, to create another working system. If these diskettes are ever damaged, use the master copy to reproduce them.

NOTE

> If your hardware configuration includes a VT100 terminal, be sure to set AUTO XON/XOFF in SETUP mode B before attempting to bootstrap RT–11. Never set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

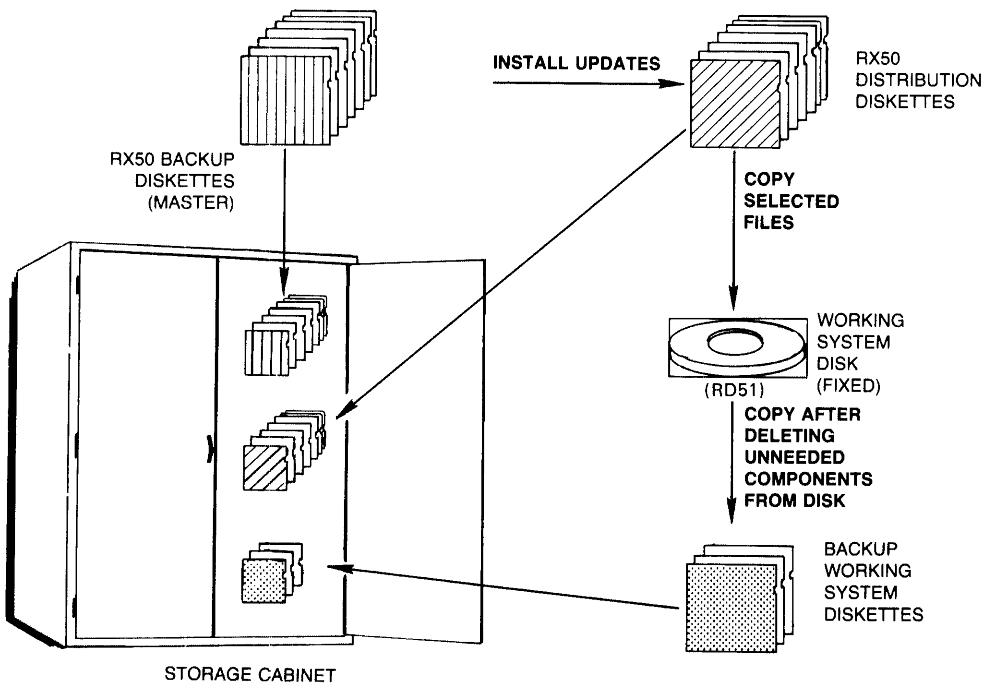
To install your system, perform the steps summarized in the following list. Sections 9.1 through 9.11 describe the procedures involved in each step. Figure 9–1 shows the backup working system diskettes you create when you install RT–11 on a MICRO/PDP–11 processor.

1. Bootstrap the system diskette.
2. Copy the system diskette onto the system disk.
3. Install the bootstrap on the system disk.
4. Copy the distribution diskettes onto the system disk.

5. Install software updates.
6. Create the working system from chosen components.
7. Customize the system.
8. Compress the system disk.
9. Preserve the working system.
10. Test the working system.
11. If appropriate, perform the system generation process.

The following sections correspond to each of these steps and describe in detail the procedures you must perform to complete each step.

Figure 9–1: Sample Backup Diskettes



INSTALL UPDATES
RX50 DISTRIBUTION DISKETTES
RX50 BACKUP DISKETTES (MASTER)
COPY SELECTED FILES
WORKING SYSTEM DISK (FIXED)
(RD51)
COPY AFTER DELETING UNNEEDED COMPONENTS FROM DISK
BACKUP WORKING SYSTEM DISKETTES
STORAGE CABINET

9.1 Bootstrapping the System Diskette

The first procedure you perform when installing RT–11 is to bootstrap the system diskette.

Begin by making sure that the processor is powered up but not running. Insert the system diskette, labeled 1/6, in RX50 diskette Unit 1 (which has the physical device name DU1:). Use the hardware bootstrap to boot the diskette. RT–11 should respond with the following message if you have successfully bootstrapped the system diskette.

```
RT-11FB V05.00

.SET TT NO QUIET

.TYPE V5USER.TXT

Welcome to RT-11 Version 5. RT-11 V5 provides new hardware support and
some major enhancements over Version 4. Among the new features are:

      ● 22-bit addressing support for Q-bus
      ● Virtual Disk facility (VM)
      ● Logical Disk Subsetting handler (LD)
      ● Single line, keypad command editor (SL)
      ● MSCP disk handler for RD51, RC25 and RA80 disks (DU)
      ● New multi-volume Backup/Restore utility (BUP)
      ● New Control File processor (IND)
      ● Concise Command Language (CCL)
      ● User Command Linkage (UCL)
      ● Many new and enhanced keyboard commands and options
      ● Many enhancements to existing RT-11 file utilities

Please use the HELP command; it describes the new commands and options
and the proper syntax with which to invoke them.

.
```

After you boot the system, you can create a start-up file containing any sequence of commands that you would normally want to execute at the start of each terminal session. For example, if your console device is a video terminal, you may wish to execute a start-up file containing the command

```
SET TT SCOPE
```

each time that you use the system. This command allows you to use the DELETE key to backspace and erase characters from the screen.

You can update your start-up file at any time to change, add, or delete commands. When you first create the file, or after you update it, you can execute it with the @ command, so that the commands it contains become effective.

9.2 Copying the System Diskette onto the System Disk

The first operation you perform with the running RT–11 system is to copy the system diskette onto the system disk (RD51).

You must first use the INITIALIZE command to initialize the RD51 system disk in Unit 0. Use the /BADBLOCKS option with INITIALIZE to cover any bad blocks that may be on your disk. If the disk contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal.

```
.INITIALIZE/BADBLOCKS DU0:(RET)
DU0:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DU0:
```

The system scans the disk for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Then, copy the contents of the system diskette to the disk.

```
.COPY/SYSTEM/NOQUERY DU1: DU0:(RET)
.
```

9.3 Installing the Bootstrap on the System Disk

Once you have copied the contents of the system diskette onto the system disk, you need to install the bootstrap on the system disk.

```
.COPY/BOOT DU0:RT11aa.SYS DU0:(RET)
.
```

In this command, you identify the device on which the monitor that contains the bootstrap information resides (DU0:), the name of the monitor file (RT11BL, RT11SJ, RT11FB, or RT11XM), and the device on which you need to install the bootstrap (DU0:). This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same volume. Remove the system diskette from Unit 1, and use the hardware bootstrap to boot the system disk.

RT–11 should respond with the following message if you have successfully bootstrapped the system disk:

```
RT-11aa V05.00
```
(Followed by any start-up file commands.)
```
.
```

9.4 Copying the Distribution Diskettes onto the System Disk

The next operation you perform with the running RT–11 system is to copy all the distribution diskettes onto the system disk, which will serve as your working system disk.

Insert distribution diskette 1/6 in Unit 1, and copy all its files to the system disk by typing the command:

```
.COPY/NOQUERY DU1: DU0:(RET)
.
```

Remove the distribution diskette from Unit 1 and store it; mount distribution diskette 2/6 in Unit 1, and copy all its files to the system disk by typing the command:

```
.COPY/NOQUERY DU1: DU0:(RET)
.
```

Repeat this procedure to copy the remaining distribution diskettes onto the system disk. When you have completed copying all distribution diskettes, store them in a safe place.

Next, remove the protection from all the files on the system disk. The files on the distribution diskettes have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) When you copied the files to the system disk, RT–11 also copied the protection. (Note the P that prints next to the file size in the directory.) For the rest of these procedures, you must remove the protection from the files on the system disk.

Type the following command to remove the protection from all files:

```
.UNPROTECT/SYSTEM DU0:*.*(RET)
 Files unprotected:
DU0:aaaaaa.ttt
DU0:bbbbbb.ttt
DU0:cccccc.ttt
DU0:dddddd.ttt
           .
           .
           .
DU0:zzzzzz.ttt
.
```

9.5 Installing Software Updates

To make sure that RT–11 operates correctly, you must install updated software modules at this point. Update modules are critical to system or component operation. They correct software errors discovered since the last release, and may add new functions to the operating system.

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only requirement is to copy the new modules from the update kit to your system, using the procedures described in the *RT–11 Update User's Guide*. Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

When you submit an SPR, the *RT–11 Software Dispatch Review* will report serious problems and may contain suggested solutions to the problems until the module containing the error is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide*.

9.6 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3), you can create the working system by deleting selected components from your system disk.

You can use the following command to avoid typing numerous file specifications. RT–11 queries you about all the files on the disk, and you choose the files it deletes.

```
.DELETE/SYSTEM/QUERY DU0:(RET)
  Files deleted:
DU0:aaaaaa.ttt  Y(RET)                    (to delete a specific file)
DU0:bbbbbb.ttt  N(RET)                    (to retain a specific file)
(and so on)
```

Repeat this procedure to retain only the files you planned for the working system disk. When you have deleted all undesired files, label the disk "RT–11 Working System V05".

9.7 Customizing the System

You may want to make certain customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 9.11.)

9.8 Compressing the System Disk

DIGITAL recommends that you compress the working system disk to make its free space contiguous. Consolidating free space allows you to use space on the disk that would otherwise be too fragmented to be usable.

Use the SQUEEZE command to compress free space. (The volume must be write-enabled.)

```
.SQUEEZE DU0:(RET)
DU0:/Squeeze; Are you sure? Y(RET)

RT-11xx V05.00
```

The system automatically reboots when you compress a system disk.

9.9 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files and preserve the system on backup diskettes.

Use the following command to protect all the files on the system disk:

```
.PROTECT/SYSTEM DU0:*.*(RET)
 Files protected:
DU0:aaaaaa.ttt
DU0:bbbbbb.ttt
DU0:cccccc.ttt
       .
       .
       .
DU0:zzzzzz.ttt
.
```

Insert a blank RX50 diskette in Unit 1 or 2 so you can initialize it. You can use the INITIALIZE command with the /BADBLOCKS option to initialize the diskette and to detect any bad blocks that may be on it. If the diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal. In the command, x is the device unit number.

NOTE

> RX50 diskettes are not formattable. DIGITAL recommends that you use only diskettes that have no bad blocks. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DUn:, where n is the drive number. If bad blocks exist on a diskette, copy the contents of the diskette to an error-free diskette, and dispose of the diskette with bad blocks.

```
.INITIALIZE/BADBLOCKS DUx:(RET)
DUx:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DUx:
```

There will be a delay while the system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Now, remove the newly initialized diskette and initialize an adequate number of blank diskettes, leaving one initialized, blank, write-enabled (write-protect notch uncovered) diskette inserted in Unit 1 or 2.

Copy all the files in your working system, using the COPY/SYSTEM/QUERY command. In the command, x is the device unit number.

```
.COPY/SYSTEM/QUERY DU0: DUx:(RET)
 Files copied:
DU0:aaaaaa.ttt to DUx:aaaaaa.ttt Y(RET)        (to include a specific file)
DU0:bbbbbb.ttt to DUx:bbbbbb.ttt N(RET)        (to exclude a specific file)
(and so on)
```

If not enough space exists on the diskette to contain all the files, remove the diskette when it is full, insert a blank, initialized diskette in the drive, and issue the COPY command again. Answer N to all files that have already been copied to a backup diskette.

Then, install the bootstrap on any backup working system diskettes that need to be bootable. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT DUx:RT11aa DUx:(RET)
.
```

Remove the newly created working system diskette from Unit 1 or 2, and label it "Backup RT–11 V05 x/y" (where x is the diskette number, and y is the number of diskettes in your working system). (Use a soft-tipped pen when you label diskettes.) Then, store it in a safe place. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

As long as the diskette you intend to copy is bootable and contains certain system utility programs, you can boot RT–11 from that diskette and copy the diskette. Distribution diskette 0/6 is bootable, but the rest of the diskettes in your kit are not bootable, because they lack the necessary components. Remember that a bootable diskette needs an appropriate monitor file, a bootstrap, a system device handler, the SWAP.SYS file, and for the SJ monitor, the TT.SYS file.

9.10 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, your working system must include at least the following components.

SWAP.SYS
RT11FB.SYS
DU.SYS (system device handler)
LP.SYS
EDIT.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

NOTE

If your configuration includes a VT11 or VS60 display processor and scope, shift system output to the display scope. Type GT ON. Verify that the scope is on by turning the BRIGHTNESS knob to an adequate level. You can still enter commands at the keyboard, but the echo is on the screen.

Display the directory of the system volume on the terminal. The directory varies according to your particular working system. As long as a directory prints, you need not worry if it does not match the one in the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:(RET)
  dd-mmm-yy
SWAP   .SYS
RT11FB.SYS
LP     .SYS
DU     .SYS
EDIT   .SAV
MACRO  .SAV
SYSMAC.SML
LINK   .SAV
PIP    .SAV
     .
     .
     .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

> If you have shifted output to a scope and the directory scrolls by too quickly to read, type (CTRL/S) to stop the display and (CTRL/Q) to restart it.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

9.10.1 Preparing the Background Demonstration Program

9.10.1.1 Edit the Background Demonstration Program — Use a text editor, for example, EDIT.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the output lines in the program is preceded by a semicolon, which makes the line a comment field. The semicolon prevents the line from being printed; thus, it must be deleted from that line. If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

```
.EDIT SY:DEMOBG.MAC(RET)
*F;(TAB).ASCII(ESC)(ESC)
*0AD(ESC)(ESC)
*EX(ESC)(ESC)
.
```

9.10.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it. To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LP: LST:(RET)
.
```

NOTE

> If your configuration does not include a line printer, use the console terminal.
>
> ```
> .ASSIGN TT: LST:(RET)
> .
> ```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG(RET)
```

(See Figure 9–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 9–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

      1                                              .TITLE  DEMOBG
      2                                              .IDENT  /V05.00/
      3                                      ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
      4                                      ; RING BELL IF FG JOB SENDS A MESSAGE.
      5
      6                                              .MCALL  .RCVDC,.PRINT
      7
      8 000000                               START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
      9 000034                                       .PRINT  #MSG                    ;PRINT DEMONSTRATION MESSAGE
     10 000042  000777                               BR      .                       ;AND LOOP
     11
     12                                      ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE
     13
     14 000044                               MSGIN:  .PRINT  #BELL                   ;RING BELL IN RESPONSE TO MESSAGE
     15 000052                                       .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
     16 000106  000207                               RETURN                          ;AND RETURN FROM COMPLETION ROUTINE
     17
     18                                      ;       ASCII MESSAGES
     19                                              .NLIST BEX
     20 000110     007      200              BELL:   .BYTE   7,200                   ;MESSAGE THAT RINGS BELL
     21
     22 000112     122      124      055     MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111      106      040             .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24                                      ;       .ASCII  /WELL DONE./
     25 000225     000                               .BYTE   0
     26
     27 000226                               AREA:   .BLKW 6                         ;EMT ARGUMENT AREA
     28 000242                               BUFFER:                                 ;RCVDC MESSAGE AREA
     29         000000'                              .END    START

DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R        BUFFER  000242R        MSGIN   000044R        ...V1 = 000003        ...V2 = 000027
BELL    000110R        MSG     000112R        START   000000RG

. ABS.  000000    000    (RW,I,GBL,ABS,OVR)
        000242    001    (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG

.
```

9.10.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

9.10.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C)
(CTRL/C)
.
```

If you incorrectly edited the file, you can repeat this exercise, although you can continue without correcting the file. However, if you want to repeat the exercise, begin by using the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Then, repeat the editing procedure.

9.10.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV. DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

9.10.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ. This file resides on your system volume.

9.10.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type signifies to the system that the file is a foreground program and is to be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

9.10.2.3 Run the Foreground and Background Demonstration Programs — Type the following command to load and start DEMOFG.REL as the foreground job.

```
.FRUN DEMOFG(RET)
F>
FOREGROUND DEMONSTRATION PROGRAM
SENDS A MESSAGE TO THE BACKGROUND PROGRAM DEMOBG
EVERY 2 SECONDS, TELLING IT TO RING THE BELL.
(CTRL/B)
B>
```

DEMOFG.REL is now running and queuing the message for DEMOBG every 2 seconds. Now execute DEMOBG.SAV in the background and receive the messages.

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```
(The bell stops ringing.)
```
.DIRECTORY(RET)
 dd-mmm-yy
```
(The directory of the device DK prints on the terminal.)
```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
CTRL/C
CTRL/C
.
```

(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
CTRL/F
F>
CTRL/C
CTRL/C
B>
UNLOAD F RET
.
```

NOTE

If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

9.11 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Appendix A
Building System Programs

If your application requires changes to RT–11 system programs other than the monitors and device handlers, you will need the information in this appendix to assemble and link the components.

Use RT–11 MACRO as the assembler and RT–11 LINK as the linker for all components. Make sure that all assemblies (except ODT and VDT) and all links (except where otherwise noted) are error free.

NOTE

The default system library, SYSLIB.OBJ, is required to link many of these components.

The sections in this appendix contain IND streams that generate the following components:

- BATCH
- BINCOM
- BUP
- CREF
- DIR
- DUMP
- DUP
- EDIT
- ERLOG
- FILEX
- FORMAT
- HELP
- IND
- LIBR
- LINK
- MDUP, MBOOT, MSBOOT, and MBOT16
- ODT and VDT
- PAT
- PIP
- QUEUE and QUEMAN
- RESORC
- SIPP
- SLP
- SRCCOM
- SWAP
- SYSLIB
- SYSMAC
- ULBLIB
- VTLIB

You can use BATCH control files, indirect command files, or direct keyboard monitor commands if you do not want to use IND, as long as you use equivalent commands. However, you should be familiar with the following conventions used in these listings.

1. Default file types are not explicitly specified. The default file type for source files is .MAC; for assembler output, .OBJ; and for linker output, .SAV.
2. The system macro library, SYSMAC.SML, must be on the system device during all these assemblies.
3. Sources are assumed to reside on logical device SRC:, binary output to go to BIN:, object files to go to OBJ:, and listing and map files to go to LST:. You can use any appropriate device for these files.
4. SYSLIB refers to a system library that contains only the FORTRAN OTS routines.

You can alter the command strings in these listings to take full advantage of all RT–11 MACRO and LINK commands. In addition, you can substitute equivalent monitor commands for the CSI-level commands shown here.

The indirect command files that SYSGEN creates contain assembly and link instructions for the monitor and handlers. Examine SYSGEN.MON and SYSGEN.DEV after a SYSGEN run for assembly and link instructions.

A.1 BATCH

```
! BUILD BATCH.SAV
!
! ASSEMBLE BATCH
R MACRO
OBJ:BATCH,LST:BATCH/L:MEB/C=SRC:BATCH
^C
! LINK BATCH
R LINK
BIN:BATCH,MAP:BATCH/W=OBJ:BATCH
^C
```

A.2 BINCOM

```
! BUILD BINCOM
!
! ASSEMBLE BINCOM
R MACRO
OBJ:BINCOM,LST:BINCOM/C=SRC:ULBMAC,BINCOM
^C
! LINK BINCOM
R LINK
BIN:BINCOM,MAP:BINCOM/W=OBJ:BINCOM,ULBLIB
^C
```

A.3 BUP

```
! BUILD BUP.SAV
! ASSEMBLE BUP
R MACRO
OBJ:BUP,LST:BUP/C=SRC:ULBMAC,BUP
OBJ:BUPBAC,LST:BUPBAC/C=SRC:ULBMAC,BUPINI,BUPIMA
OBJ:BUPRES,LST:BUPRES/C=SRC:ULBMAC,BUPRES
OBJ:BUPDIR,LST:BUPDIR/C=SRC:ULBMAC,BUPDIR
OBJ:BUPSUB,LST:BUPSUB/C=SRC:ULBMAC,BUPSUB
OBJ:BUPMT1,LST:BUPMT1/C=SRC:ULBMAC,BUPMT1
OBJ:BUPMT2,LST:BUPMT2/C=SRC:ULBMAC,BUPMT2
OBJ:BUPMT3,LST:BUPMT3/C=SRC:ULBMAC,BUPMT3
R LINK
BIN:BUP,MAP:BUP/W=OBJ:BUP,ULBLIB//
OBJ:BUPBAC/O:1
OBJ:BUPRES/O:1
OBJ:BUPDIR/O:1
OBJ:BUPMT1/O:1
OBJ:BUPMT2/O:1
OBJ:BUPMT3/O:1
//
```

A.4 CREF

```
! BUILD CREF.SAV
!
! ASSEMBLE CREF
R MACRO
OBJ:CREF,LST:CREF/C=SRC:CREF
^C
! LINK CREF
R LINK
BIN:CREF,MAP:CREF/W=OBJ:CREF
^C
```

A.5 DIR

```
! BUILD DIR.SAV
!
! ASSEMBLE DIR
R MACRO
OBJ:DIR1ST,LST:DIR1ST/C=SRC:DIR1ST
OBJ:DIRDAT,LST:DIRDAT/C=SRC:DIRPRE,DIRDAT
OBJ:DIRMAN,LST:DIRMAN/C=SRC:DIRPRE,DIRMAN
OBJ:DIRT11,LST:DIRT11/C=SRC:DIRPRE,DIRT11
OBJ:DIRSWT,LST:DIRSWT/C=SRC:DIRPRE,DIRSWT
OBJ:DIRMAT,LST:DIRMAT/C=SRC:DIRPRE,DIRMAT
OBJ:DIRDK,LST:DIRDK/C=SRC:DIRPRE,DIRDK
OBJ:DIRMT,LST:DIRMT/C=SRC:DIRPRE,DIRMT
OBJ:DIRCT,LST:DIRCT/C=SRC:DIRPRE,DIRCT
OBJ:DIRSUP,LST:DIRSUP/C=SRC:DIRPRE,DIRSUP
OBJ:DIRSRT,LST:DIRSRT/C=SRC:DIRPRE,DIRSRT
OBJ:DIROUT,LST:DIROUT/C=SRC:DIRPRE,DIROUT
OBJ:DIRERR,LST:DIRERR/C=SRC:DIRPRE,DIRERR
^C
! LINK DIR
R LINK
BIN:DIR,MAP:DIR/W=OBJ:DIR1ST,DIRDAT,DIRMAN,DIRT11,DIRSWT,DIRMAT//
OBJ:DIRDK,DIRMT,DIRCT,DIRSUP,DIRSRT,DIROUT
OBJ:DIRERR,ULBLIB
//
^C
```

A.6 DUMP

```
! BUILD DUMP.SAV
!
! ASSEMBLE DUMP
R MACRO
OBJ:DUMP,LST:DUMP/C=SRC:DUMP
^C
! LINK DUMP
R LINK
BIN:DUMP,MAP:DUMP/W=OBJ:DUMP
^C
```

A.7 DUP

```
! BUILD DUP.SAV
!
! ASSEMBLE DUP
R MACRO
OBJ:DUPROT,LST:DUPROT/C=SRC:DUPROT
OBJ:DUPOV1,LST:DUPOV1/C=SRC:DUPPRE,DUPOV1
OBJ:DUPIN1,LST:DUPIN1/C=SRC:DUPPRE,DUPIN1
OBJ:DUPIN2,LST:DUPIN2/C=SRC:DUPPRE,DUPIN2
OBJ:DUPOV2,LST:DUPOV2/C=SRC:DUPPRE,DUPOV2
OBJ:DUPCRE,LST:DUPCRE/C=SRC:DUPPRE,DUPCRE
OBJ:DUPUNI,LST:DUPUNI/C=SRC:DUPPRE,DUPUNI
OBJ:DUPIMA,LST:DUPIMA/C=SRC:DUPPRE,DUPIMA
OBJ:DUPSCN,LST:DUPSCN/C=SRC:DUPPRE,DUPSCN,DUPMRG
OBJ:DUPOV5,LST:DUPOV5/C=SRC:DUPPRE,DUPOV5
OBJ:DUPBOT,LST:DUPBOT/C=SRC:DUPPRE,DUPBOT
OBJ:DUPWBT,LST:DUPWBT/C=SRC:DUPPRE,DUPWBT
OBJ:DUPSQU,LST:DUPSQU/C=SRC:DUPPRE,DUPSQU
OBJ:DUPOV7,LST:DUPOV7/C=SRC:DUPPRE,DUPOV7
OBJ:DUPVOL,LST:DUPVOL/C=SRC:DUPPRE,DUPVOL
OBJ:DUPZMC,LST:DUPZMC/C=SRC:DUPPRE,DUPZMC
OBJ:DUPZRO,LST:DUPZRO/C=SRC:DUPPRE,DUPZRO,DUPMRG
^C
! LINK DUP
R LINK
BIN:DUP,MAP:DUP/W=OBJ:DUPROT,ULBLIB//
OBJ:DUPOV1,DUPIN1,DUPIN2/O:1
OBJ:DUPOV2,DUPCRE,DUPUNI/O:1
OBJ:DUPIMA/O:1
OBJ:DUPSCN/O:1
OBJ:DUPOV5,DUPBOT,DUPWBT/O:1
OBJ:DUPSQU/O:1
OBJ:DUPOV7,DUPVOL,DUPZMC/O:1
OBJ:DUPZRO/O:1
//
^C
```

A.8 EDIT

```
! BUILD EDIT
!
! ASSEMBLE EDIT
R MACRO
OBJ:VTCED1,LST:VTCED1/C=SRC:EDITDF,VTCAL1
OBJ:VTCED4,LST:VTCED4/C=SRC:EDITDF/P:1,VTCAL4
OBJ:VTBEDT,LST:VTBEDT/C=SRC:EDITDF/P:1,VTBASE
OBJ:EDIT,LST:EDIT/C=SRC:VTMAC,EDIT
^C
! LINK EDIT
R LINK
BIN:EDIT,MAP:EDIT/W=OBJ:VTCED1,VTCED4,VTBEDT,EDIT
^C
```

A.9 ERLOG

```
! BUILD ERROR LOGGING UTILITIES
!
! ASSEMBLE FB AND XM ERROR LOGGER
R MACRO
OBJ:ELCOPY,LST:ELCOPY/C=SRC:SYSGEN.CND,ELCOPY
OBJ:ELINIT,LST:ELINIT/C=SRC:SYSGEN.CND,ELINIT
OBJ:ELTASK,LST:ELTASK/C=SRC:ELTASK
OBJ:ERROUT,LST:ERROUT/C=SRC:ERROUT
OBJ:ERRTXT,LST:ERRTXT/C=SRC:ERRTXT
^C
! LINK FB AND XM ERROR LOGGER
R LINK
BIN:ERRLOG,MAP:ERRLOG/W=OBJ:ELCOPY,ELTASK/R
BIN:ELINIT,MAP:ELINIT/W=OBJ:ELINIT
BIN:ERROUT,MAP:ERROUT/W=OBJ:ERROUT,ERRTXT
^C
!
! ASSEMBLE SJ ERROR LOGGER
R MACRO
OBJ:EL,LST:EL/C=SRC:SYSGEN.CND,EL
^C
! LINK SJ ERROR LOGGER
R LINK
BIN:EL.SYS,MAP:EL/W=EL
^C
```

A.10 FILEX

```
! BUILD FILEX.SAV
!
! ASSEMBLE FILEX
R MACRO
OBJ:FILEX,LST:FILEX/L:MEB/C=SRC:FILEX
^C
! LINK FILEX
R LINK
BIN:FILEX,MAP:FILEX/W=OBJ:FILEX
^C
```

A.11 FORMAT

```
! BUILD FORMAT
!
! ASSEMBLE FORMAT
R MACRO
OBJ:FORMAT,LST:FORMAT/C=SRC:FORMAT
OBJ:FMTDEV,LST:FMTDEV/C=SRC:FMTDEV
OBJ:FMTDM,LST:FMTDM/C=SRC:FMTDM
OBJ:FMTRK,LST:FMTRK/C=SRC:FMTRK
OBJ:FMTDY,LST:FMTDY/C=SRC:FMTDY
OBJ:FMTDP,LST:FMTDP/C=SRC:FMTDP
OBJ:FMTDL,LST:FMTDL/C=SRC:FMTDL
OBJ:FMTDX,LST:FMTDX/C=SRC:FMTDX
OBJ:FMTDT,LST:FMTDT/C=SRC:FMTDT
OBJ:FMTDD,LST:FMTDD/C=SRC:FMTDD
^C
! LINK FORMAT
R LINK
BIN:FORMAT,MAP:FORMAT/W=OBJ:FORMAT,FMTDEV//
OBJ:FMTDM/O:1
OBJ:FMTRK/O:1
OBJ:FMTDY/O:1
OBJ:FMTDP/O:1
OBJ:FMTDL/O:1
OBJ:FMTDX/O:1
OBJ:FMTDT/O:1
OBJ:FMTDD/O:1
//
^C
```

A.12 HELP

```
! BUILD HELP
!
R MACRO
OBJ:HELP,LST:HELP/C=SRC:HELP
^C
! LINK HELP
R LINK
BIN:HELP.EXE,MAP:HELP/W=OBJ:HELP
^C
! BUILD HELP LIBRARY
R LIBR
OBJ:HELP.MLB=SRC:HELP.TXT/M
^C
```

A.13 IND

```
! BUILD IND.SAV
!
! CREATE MACRO LIBRARY
R LIBR
SRC:INDMLB=SRC:INDMLB
^C
! ASSEMBLE LIBRARY FILES
R MACRO
OBJ:CATB=SRC:INDMLB/M,SRC:CATB
OBJ:CAT5=SRC:INDMLB/M,SRC:CAT5
OBJ:CKFS=SRC:INDMLB/M,SRC:CKFS
OBJ:CKOPN=SRC:INDMLB/M,SRC:CKOPN
OBJ:CLOSE=SRC:INDMLB/M,SRC:CLOSE
```

```
OBJ:DELETE=SRC:INDMLB/M,SRC:DELETE
OBJ:OPENA=SRC:INDMLB/M,SRC:OPENA
OBJ:OPENR=SRC:INDMLB/M,SRC:OPENR
OBJ:OPENW=SRC:INDMLB/M,SRC:OPENW
OBJ:RDBLK=SRC:INDMLB/M,SRC:RDBLK
OBJ:RDREC=SRC:INDMLB/M,SRC:RDREC
OBJ:WRBLK=SRC:INDMLB/M,SRC:WRBLK
OBJ:WRREC=SRC:INDMLB/M,SRC:WRREC
^C
! CREATE OBJECT LIBRARY
R LIBR
OBJ:NV20LB=OBJ:DELETE//
OBJ:CATB
OBJ:CAT5
OBJ:CKFS
OBJ:CKOPN
OBJ:CLOSE
OBJ:OPENA
OBJ:OPENR
OBJ:OPENW
OBJ:RDBLK
OBJ:RDREC
OBJ:WRBLK
OBJ:WRREC
//
^C
! ASSEMBLE IND
R MACRO
OBJ:IND=OBJ:INDMLB/M,SRC:IND
OBJ:INDASK=SRC:INDMLB/M,SRC:INDASK
OBJ:INDAS1=SRC:INDMLB/M,SRC:INDAS1
OBJ:INDAS2=SRC:INDMLB/M,SRC:INDAS2
OBJ:INDCTL=SRC:INDMLB/M,SRC:INDCTL
OBJ:INDERA=SRC:INDMLB/M,SRC:INDERA
OBJ:INDERR=SRC:INDMLB/M,SRC:INDERR
OBJ:INDDCL=SRC:INDMLB/M,SRC:INDDCL
OBJ:INDDMP=SRC:INDMLB/M,SRC:INDDMP
OBJ:INDFDC=SRC:INDMLB/M,SRC:INDFDC
OBJ:INDFEX=SRC:INDMLB/M,SRC:INDFEX
OBJ:INDGCM=SRC:INDMLB/M,SRC:INDGCM
OBJ:INDGNB=SRC:INDMLB/M,SRC:INDGNB
OBJ:INDIF1=SRC:INDMLB/M,SRC:INDIF1
OBJ:INDIMP=SRC:INDMLB/M,SRC:INDIMP
OBJ:INDINO=SRC:INDMLB/M,SRC:INDINO
OBJ:INDIN1=SRC:INDMLB/M,SRC:INDIN1
OBJ:INDOPN=SRC:INDMLB/M,SRC:INDOPN
OBJ:INDPAR=SRC:INDMLB/M,SRC:INDPAR
OBJ:INDPRC=SRC:INDMLB/M,SRC:INDPRC
OBJ:INDSET=SRC:INDMLB/M,SRC:INDSET
OBJ:INDSU1=SRC:INDMLB/M,SRC:INDSU1
OBJ:INDSU2=SRC:INDMLB/M,SRC:INDSU2
OBJ:INDSU3=SRC:INDMLB/M,SRC:INDSU3
OBJ:INDTES=SRC:INDMLB/M,SRC:INDTES
OBJ:INDTIM=SRC:INDMLB/M,SRC:INDTIM
OBJ:INDUTL=SRC:INDMLB/M,SRC:INDUTL
OBJ:INDSYM=SRC:INDMLB/M,SRC:INDSYM
^C
! LINK IND
R LINK
BIN:IND,MAP:IND=OBJ:ULBLIB,OBJ:NV20LB/Y:1000//
OBJ:INDIMP
OBJ:IND
OBJ:INDDCL
OBJ:INDASK
OBJ:INDAS1
```

```
OBJ:INDAS2
OBJ:INDERR
OBJ:INDFDC
OBJ:INDFEX
OBJ:INDGCM
OBJ:INDGNB
OBJ:INDIF1
OBJ:INDINO
OBJ:INDOPN
OBJ:INDPRC
OBJ:INDSET
OBJ:INDSU1
OBJ:INDSU2
OBJ:INDSU3
OBJ:INDCTL/O:1
OBJ:INDDMP/O:1
OBJ:INDERA/O:1
OBJ:INDIN1/O:1
OBJ:INDPAR/O:1
OBJ:INDTES/O:1
OBJ:INDTIM/O:1
OBJ:INDUTL/O:1
OBJ:INDSYM/O:2
//
INDIMP
```

A.14 LIBR

```
! BUILD LIBR.SAV
!
! ASSEMBLE LIBR
R MACRO
OBJ:LIBR0,LST:LIBR0/C=SRC:LIBR0
OBJ:LIBR1,LST:LIBR1/C=SRC:LIBR1
OBJ:LIBR2,LST:LIBR2/C=SRC:LIBR2
OBJ:LIBR3,LST:LIBR3/C=SRC:LIBR3
OBJ:LIBR4,LST:LIBR4/C=SRC:LIBR4
OBJ:LIBR5,LST:LIBR5/C=SRC:LIBR5
OBJ:LIBR6,LST:LIBR6/C=SRC:LIBR6
OBJ:LBREM,LST:LBREM/C=SRC:LBREM
^C
! LINK LIBR
R LINK
BIN:LIBR,MAP:LIBR/W=OBJ:LIBR0//
OBJ:LIBR1/O:1
OBJ:LIBR2/O:1
OBJ:LIBR3/O:1
OBJ:LIBR4/O:1
OBJ:LIBR5/O:1
OBJ:LIBR6/O:1
OBJ:LBREM/O:1
//
^C
```

A.15 LINK

```
! BUILD LINK.SAV
!
! ASSEMBLE LINK
R MACRO
OBJ:LINK0,LST:LINK0/C=SRC:LINK0
OBJ:LINK1,LST:LINK1/C=SRC:LINK1
OBJ:LINK2,LST:LINK2/C=SRC:LINK2
```

```
OBJ:LINK3,LST:LINK3/C=SRC:LINK3
OBJ:LINK4,LST:LINK4/C=SRC:LINK4
OBJ:LINK5,LST:LINK5/C=SRC:LINK5
OBJ:LINK6,LST:LINK6/C=SRC:LINK6
OBJ:LINK7,LST:LINK7/C=SRC:LINK7
OBJ:LINK8,LST:LINK8/C=SRC:LINK8
OBJ:LNKEM,LST:LNKEM/C=SRC:LNKEM
OBJ:LNKLB1,LST:LNKLB1/C=SRC:LNKLB1
^C
! BUILD LINK LIBRARY
R LIBR
OBJ:LNKLB1=OBJ:LNKLB1
^C
! LINK LINK
R LINK
BIN:LINK,MAP:LINK/W=OBJ:LINK0,OBJ:LNKLB1/D//
OBJ:LINK1/O:1
OBJ:LINK2/O:1
OBJ:LINK3/O:1
OBJ:LINK4/O:1
OBJ:LINK5/O:1
OBJ:LINK6/O:1
OBJ:LINK7/O:1
OBJ:LINK8/O:1
OBJ:LNKEM/O:1
//
BITST
GETBUF
WRITO
WRTLRU
ZSWFIL
^C
```

A.16 MDUP, MBOOT, MSBOOT, and MBOT16

```
! BUILD MDUP
!
! ASSEMBLE MDUP
R MACRO
OBJ:MDPROT,LST:MDPROT/C=SRC:DUPPRE,MDUP,DUPROT
OBJ:MDPIN1,LST:MDPIN1/C=SRC:DUPPRE,MDUP,DUPIN1
OBJ:MDPIN2,LST:MDPIN2/C=SRC:DUPPRE,MDUP,DUPIN2
OBJ:MDPCRE,LST:MDPCRE/C=SRC:DUPPRE,MDUP,DUPCRE
OBJ:MDPSCN,LST:MDPSCN/C=SRC:DUPPRE,MDUP,DUPSCN,DUPMRG
OBJ:MDPBOT,LST:MDPBOT/C=SRC:DUPPRE,MDUP,DUPBOT
OBJ:MDPWBT,LST:MDPWBT/C=SRC:DUPPRE,MDUP,DUPWBT
OBJ:MDPZRO,LST:MDPZRO/C=SRC:DUPPRE,MDUP,DUPZRO
^C
! LINK MDUP
R LINK
BIN:MDUP,MAP:MDUP/W=OBJ:MDPROT//
OBJ:MDPIN1
OBJ:MDPIN2
OBJ:MDPCRE
OBJ:MDPSCN
OBJ:MDPBOT
OBJ:MDPWBT
OBJ:MDPZRO
OBJ:ULBLIB
//
^C
```

This procedure builds the program MDUP.SAV. MDUP.SAV builds the magtape build programs, MDUP.MM for TJU16 magtape, MDUP.MS for

TS11 magtape, and MDUP.MT for TM11 magtape. The magtape build program runs from the booted magtape, initializes the system disk, and copies a minimal system from the magtape to the disk.

The Version 5 MDUP.MM, MDUP.MS, and MDUP.MT programs support the following devices:

RK05	RP02
RK06	RP03
RK07	RS03
RL01	RS04
RL02	RF11

If you need to create MDUP for a disk or magtape not supported by RT–11 (for which you have written your own handler), use MDUP.SAV as follows.

Use the hardware version of the magtape handler and the standard (distributed) single-job monitor. Your configuration must include at least 28K words of memory. Apply the following customization to the monitor. In the customization, xx is the name of the device on which the monitor resides.

```
.R SIPP(RET)
*xxn:RT11SJ.SYS(RET)
Base?       0(RET)
Offset?     1030(RET)

      Base          Offset       Old          New?
      000000        001030       000407       240(RET)
      000000        001032       013704       12704(RET)
      000000        001034       177570       60000(RET)
      000000        001036       042704       (CTRL/Y)(RET)
*(CTRL/C)
.
```

Then copy the bootstrap:

```
.COPY/BOOT xxn:RT11SJ.SYS xxn:(RET)
.
```

This procedure causes the system to use only 12K words of memory when booted.

Next apply the following customization to every handler to be supported by the version of MDUP you are building. In the customization, xx is the name of the device on which the handler resides, and yy is the name of the handler.

```
.R SIPP(RET)
*xxn:yy.SYS(RET)
Base?       0(RET)
Offset?     176(RET)

      Base          Offset       Old          New?
      000000        000176       ??????       0(RET)
      000000        000200       ??????       0(RET)
      000000        000202       ??????       (CTRL/Y)(RET)
*(CTRL/C)
.
```

Now boot the monitor you have modified. Set the USR NOSWAP. Load all the handlers to be supported. Run MDUP.SAV and issue the following command. In the command, xx is the physical device name, and filnam.typ is your version of MDUP (MDUP.??).

```
.R MDUP.SAV(RET)
*xx:filnam.typ=/H(RET)
```

MDUP.SAV creates the file filnam.typ and exits.

If you make a typing error in the command line, use CTRL/C to abort MDUP.SAV and run MDUP.SAV again. If you get an insufficient memory error message, you cannot build MDUP with all the handlers loaded. Unload one of the handlers, and run MDUP.SAV again.

```
! BUILD MBOOT
!
! ASSEMBLE MBOOT
R MACRO
OBJ:MBOOT,LST:MBOOT/C=SRC:MBOOT
^C
! LINK MBOOT
R LINK
BIN:MBOOT.BOT,MAP:MBOOT/W=OBJ:MBOOT
^C

! BUILD MSBOOT
!
! ASSEMBLE MSBOOT
R MACRO
OBJ:MSBOOT,LST:MSBOOT/C=SRC:MSBOOT
^C
! LINK MSBOOT
R LINK
BIN:MSBOOT.BOT,MAP:MSBOOT/W=OBJ:MSBOOT
^C

! BUILD MBOT16
!
! ASSEMBLE MBOT16
R MACRO
OBJ:MBOT16,LST:MBOT16/C=SRC:MBT16,MBOOT
^C
! LINK MBOT16
R LINK
BIN:MBOT16.BOT,MAP:MBOT16/W=OBJ:MBOT16
^C
```

A.17 ODT and VDT

```
! BUILD ODT.OBJ AND VDT.OBJ
!
! ASSEMBLE ODT AND VDT
R MACRO
BIN:ODT,LST:ODT/C=SRC:ODT
BIN:VDT,LST:VDT/C=SRC:VDT,ODT
^C
```

NOTE

When you assemble ODT, an assembly error occurs (a Z error). Ignore this error.

A.18 PAT

```
! BUILD PAT.SAV
!
! ASSEMBLE PAT
R MACRO
OBJ:PAT,LST:PAT/C=SRC:PAT
^C
! LINK PAT
R LINK
BIN:PAT,MAP:PAT/W=OBJ:PAT
^C
```

A.19 PIP

```
! BUILD PIP.SAV
!
! ASSEMBLE PIP
R MACRO
OBJ:PIPROT,LST:PIPROT/C=SRC:PIPROT
OBJ:PIPINI,LST:PIPINI/C=SRC:PIPPRE,PIPINI
OBJ:PIPEXE,LST:PIPEXE/C=SRC:PIPPRE,PIPEXE
OBJ:PIPDK,LST:PIPDK/C=SRC:PIPPRE,PIPDK
OBJ:PIPMT,LST:PIPMT/C=SRC:PIPPRE,PIPMT
OBJ:PIPCT,LST:PIPCT/C=SRC:PIPPRE,PIPCT
^C
! LINK PIP
R LINK
BIN:PIP,MAP:PIP/W=OBJ:PIPROT,ULBLIB//
OBJ:PIPINI/O:1
OBJ:PIPEXE/O:1
OBJ:PIPDK/O:2
OBJ:PIPMT/O:2
OBJ:PIPCT/O:2
//
^C
```

A.20 QUEUE and QUEMAN

```
! BUILD QUEUE PACKAGE
!
! ASSEMBLE QUEMAN AND QUEUE
R MACRO
OBJ:QUEMAN,LST:QUEMAN/C=SRC:ULBMAC,QUEMAN
OBJ:QUEUE,LST:QUEUE/C=SRC:QUEUE
^C
! LINK QUEMAN AND QUEUE
R LINK
BIN:QUEMAN,MAP:QUEMAN/W=OBJ:QUEMAN,ULBLIB
BIN:QUEUE/R,MAP:QUEUE/W=OBJ:QUEUE
^C
```

A.21 RESORC

```
! BUILD RESORC.SAV
!
! ASSEMBLE RESORC
R MACRO
OBJ:RESORC,LST:RESORC/C=SRC:ULBMAC,RESORC
^C
! LINK RESORC
R LINK
BIN:RESORC,MAP:RESORC/W=OBJ:RESORC,ULBLIB
^C
```

A.22 SIPP

```
! BUILD SIPP
!
! ASSEMBLE SIPP
R MACRO
OBJ:SIPP,LST:SIPP/L:MEB/C=SRC:ULBMAC,SIPP
^C
! LINK SIPP
R LINK
BIN:SIPP,MAP:SIPP/W=OBJ:SIPP,ULBLIB
^C
```

A.23 SLP

```
! BUILD SLP
!
! ASSEMBLE SLP
R MACRO
OBJ:SLP,LST:SLP/C=SRC:SLPPRE,SLP
OBJ:SLPERR,LST:SLPERR/C=SRC:SLPPRE,SLPERR
OBJ:SLPIO,LST:SLPIO/C=SRC:SLPPRE,SLPIO
OBJ:SLPSUB,LST:SLPSUB/C=SRC:SLPPRE,SLPSUB
OBJ:POSIT,LST:POSIT/C=SRC:SLPPRE,TPMAC,POSIT
OBJ:TPARS,LST:TPARS/C=SRC:TPARS
^C
! LINK SLP
R LINK
BIN:SLP,MAP:SLP/W=OBJ:SLP//
OBJ:SLPERR
OBJ:SLPIO
OBJ:SLPSUB
OBJ:POSIT
OBJ:TPARS
OBJ:ULBLIB
//
^C
```

A.24 SRCCOM

```
! BUILD SRCCOM
!
! ASSEMBLE SRCCOM
R MACRO
OBJ:SRCCOM,LST:SRCCOM/C=SRC:ULBMAC,SRCCOM
OBJ:SRCWCD,LST:SRCWCD/C=SRC:ULBMAC,SRCWCD
^C
! LINK SRCCOM
R LINK
BIN:SRCCOM,MAP:SRCCOM/W=OBJ:SRCCOM,SRCWCD,ULBLIB
^C
```

A.25 SWAP

```
! BUILD SWAP.SYS
!
! ASSEMBLE SWAP
R MACRO
OBJ:SWAP,LST:SWAP/C=SRC:SWAP
^C
! LINK SWAP
R LINK
BIN:SWAP.SYS=OBJ:SWAP
^C
```

A.26 SYSLIB

```
! BUILD SYSLIB.OBJ
!
! ASSEMBLE SYSLIB
R MACRO
OBJ:OHANDL,LST:OHANDL/C=SRC:OHANDL
OBJ:VHANDL,LST:VHANDL/C=SRC:VHANDL
OBJ:CHAIN,LST:CHAIN/C=SRC:CHAIN
OBJ:CLOSEC,LST:CLOSEC/C=SRC:CLOSEC
OBJ:CMPLT,LST:CMPLT/C=SRC:CMPLT
OBJ:CONCAT,LST:CONCAT/C=SRC:CONCAT
OBJ:CVTTIM,LST:CVTTIM/C=SRC:CVTTIM
OBJ:DEVICE,LST:DEVICE/C=SRC:DEVICE
OBJ:DIVGO,LST:DIVGO/C=SRC:DIVGO
OBJ:GTIM,LST:GTIM/C=SRC:GTIM
OBJ:GTJB,LST:GTJB/C=SRC:GTJB
OBJ:GTLIN,LST:GTLIN/C=SRC:GTLIN
OBJ:IABTIO,LST:IABTIO/C=SRC:IABTIO
OBJ:IADDR,LST:IADDR/C=SRC:IADDR
OBJ:IASIGN,LST:IASIGN/C=SRC:IASIGN
OBJ:ICDFN,LST:ICDFN/C=SRC:ICDFN
OBJ:ICHCPY,LST:ICHCPY/C=SRC:ICHCPY
OBJ:ICMKT,LST:ICMKT/C=SRC:ICMKT
OBJ:ICSI,LST:ICSI/C=SRC:ICSI
OBJ:ICSTAT,LST:ICSTAT/C=SRC:ICSTAT
OBJ:IDELET,LST:IDELET/C=SRC:IDELET
OBJ:IDSTAT,LST:IDSTAT/C=SRC:IDSTAT
OBJ:IENTER,LST:IENTER/C=SRC:IENTER
OBJ:IFETCH,LST:IFETCH/C=SRC:IFETCH
OBJ:IFREEC,LST:IFREEC/C=SRC:IFREEC
OBJ:IGETC,LST:IGETC/C=SRC:IGETC
OBJ:IGETSP,LST:IGETSP/C=SRC:IGETSP
OBJ:IFPROT,LST:IFPROT/C=SRC:IFPROT
OBJ:IJCVT,LST:IJCVT/C=SRC:IJCVT
OBJ:ILUN,LST:ILUN/C=SRC:ILUN
OBJ:INDEX,LST:INDEX/C=SRC:INDEX
OBJ:INSERT,LST:INSERT/C=SRC:INSERT
OBJ:INTSET,LST:INTSET/C=SRC:INTSET
OBJ:IPEEK,LST:IPEEK/C=SRC:IPEEK
OBJ:IPEEKB,LST:IPEEKB/C=SRC:IPEEKB
OBJ:IPUT,LST:IPUT/C=SRC:IPUT
OBJ:IQSET,LST:IQSET/C=SRC:IQSET
OBJ:IRAD50,LST:IRAD50/C=SRC:IRAD50
OBJ:IRCVD,LST:IRCVD/C=SRC:IRCVD
OBJ:IRCVDF,LST:IRCVDF/C=SRC:IRCVDF
OBJ:IREAD,LST:IREAD/C=SRC:IREAD
OBJ:IREADF,LST:IREADF/C=SRC:IREADF
OBJ:IRENAM,LST:IRENAM/C=SRC:IRENAM
OBJ:IREOPN,LST:IREOPN/C=SRC:IREOPN
OBJ:ISAVES,LST:ISAVES/C=SRC:ISAVES
OBJ:ISCHED,LST:ISCHED/C=SRC:ISCHED
OBJ:ISDAT,LST:ISDAT/C=SRC:ISDAT
OBJ:ISDATF,LST:ISDATF/C=SRC:ISDATF
OBJ:ISLEEP,LST:ISLEEP/C=SRC:ISLEEP
OBJ:ISFDAT,LST:ISFDAT/C=SRC:ISFDAT
OBJ:ISPFN,LST:ISPFN/C=SRC:ISPFN
OBJ:ISPFNF,LST:ISPFNF/C=SRC:ISPFNF
OBJ:ISPY,LST:ISPY/C=SRC:ISPY
OBJ:ITLOCK,LST:ITLOCK/C=SRC:ITLOCK
OBJ:ITTINR,LST:ITTINR/C=SRC:ITTINR
OBJ:ITTOUR,LST:ITTOUR/C=SRC:ITTOUR
OBJ:ITWAIT,LST:ITWAIT/C=SRC:ITWAIT
OBJ:IUNTIL,LST:IUNTIL/C=SRC:IUNTIL
OBJ:IWAIT,LST:IWAIT/C=SRC:IWAIT
```

```
OBJ:IWRITE,LST:IWRITE/C=SRC:IWRITE
OBJ:IWRITF,LST:IWRITF/C=SRC:IWRITF
OBJ:JADD,LST:JADD/C=SRC:JADD
OBJ:JCMP,LST:JCMP/C=SRC:JCMP
OBJ:JDIV,LST:JDIV/C=SRC:JDIV
OBJ:JFIX,LST:JFIX/C=SRC:JFIX
OBJ:JFLT,LST:JFLT/C=SRC:JFLT
OBJ:JICVT,LST:JICVT/C=SRC:JICVT
OBJ:JJCVT,LST:JJCVT/C=SRC:JJCVT
OBJ:JMOV,LST:JMOV/C=SRC:JMOV
OBJ:JMUL,LST:JMUL/C=SRC:JMUL
OBJ:JSUB,LST:JSUB/C=SRC:JSUB
OBJ:JTIME,LST:JTIME/C=SRC:JTIME
OBJ:LEN,LST:LEN/C=SRC:LEN
OBJ:LOCK,LST:LOCK/C=SRC:LOCK
OBJ:LOOKUP,LST:LOOKUP/C=SRC:LOOKUP
OBJ:MRKT,LST:MRKT/C=SRC:MRKT
OBJ:MTGET,LST:MTGET/C=SRC:MTGET
OBJ:MTSET,LST:MTSET/C=SRC:MTSET
OBJ:MWAIT,LST:MWAIT/C=SRC:MWAIT
OBJ:PRINT,LST:PRINT/C=SRC:PRINT
OBJ:PURGE,LST:PURGE/C=SRC:PURGE
OBJ:R50ASC,LST:R50ASC/C=SRC:R50ASC
OBJ:RCHAIN,LST:RCHAIN/C=SRC:RCHAIN
OBJ:RCTRLO,LST:RCTRLO/C=SRC:RCTRLO
OBJ:REPEAT,LST:REPEAT/C=SRC:REPEAT
OBJ:RESUME,LST:RESUME/C=SRC:RESUME
OBJ:SCCA,LST:SCCA/C=SRC:SCCA
OBJ:SCOMP,LST:SCOMP/C=SRC:SCOMP
OBJ:SCOPY,LST:SCOPY/C=SRC:SCOPY
OBJ:SECNDS,LST:SECNDS/C=SRC:SECNDS
OBJ:SETCMD,LST:SETCMD/C=SRC:SETCMD
OBJ:STRPAD,LST:STRPAD/C=SRC:STRPAD
OBJ:SUBSTR,LST:SUBSTR/C=SRC:SUBSTR
OBJ:SUSPND,LST:SUSPND/C=SRC:SUSPND
OBJ:SYSLBV,LST:SYSLBV/C=SRC:SYSLBV
OBJ:TIMASC,LST:TIMASC/C=SRC:TIMASC
OBJ:TIME,LST:TIME/C=SRC:TIME
OBJ:TIME1,LST:TIME1/C=SRC:TIME1
OBJ:TIMSUB,LST:TIMSUB/C=SRC:TIMSUB
OBJ:TRANSL,LST:TRANSL/C=SRC:TRANSL
OBJ:TRIM,LST:TRIM/C=SRC:TRIM
OBJ:UNLOCK,LST:UNLOCK/C=SRC:UNLOCK
OBJ:VERIFY,LST:VERIFY/C=SRC:VERIFY
^C
! COMPILE SYSLIB
R FORTRA
OBJ:PUTSTR,LST:PUTSTR=SRC:PUTSTR/I:THR/S
OBJ:GETSTR,LST:GETSTR=SRC:GETSTR/I:THR/S
^C
! BUILD SYSLIB
R LIBR
BIN:SYSLIB,MAP:SYSLIB/W=OBJ:VHANDL,OHANDL/G//
OBJ:CHAIN,CLOSEC,CMPLT,CONCAT,CVTTIM
OBJ:DEVICE,DIVGO,GTIM,GTJB,GTLIN
OBJ:IABTIO,IADDR,IASIGN,ICDFN,ICHCPY,ICMKT
OBJ:ICSI,ICSTAT,IDELET,IDSTAT,IENTER
OBJ:IFETCH,IFPROT,IFREEC,IGETC,IGETSP
OBJ:IJCVT,ILUN,INDEX,INSERT,INTSET
OBJ:IPEEK,IPEEKB,IPUT,IQSET,IRAD50
OBJ:IRCVD,IRCVDF,IREAD,IREADF,IRENAM
OBJ:IREOPN,ISAVES,ISCHED,ISDAT,ISDATF
OBJ:ISFDAT,ISLEEP,ISPFN,ISPFNF,ISPY
OBJ:ITLOCK,ITTINR,ITTOUR,ITWAIT,IUNTIL
OBJ:IWAIT,IWRITE,IWRITF
OBJ:JADD,JCMP,JDIV,JFIX,JFLT
```

```
OBJ:JICVT,JJCVT,JMOV,JMUL,JSUB
OBJ:JTIME,LEN,LOCK,LOOKUP,MRKT
OBJ:MTGET,MTSET,MWAIT,PRINT,PURGE,R50ASC
OBJ:RCHAIN,RCTRLO,REPEAT,RESUME,SCCA
OBJ:SCOMP,SCOPY,SECNDS,SETCMD,STRPAD
OBJ:SUBSTR,SUSPND,SYSLBV,TIMASC,TIME
OBJ:TIME1,TIMSUB,TRANSL,TRIM,UNLOCK
OBJ:VERIFY,PUTSTR,GETSTR,ULB002,CBOMG
OBJ:ULB012,FNASC
//
$OVRH
^C
```

A.27 SYSMAC

```
! BUILD SYSMAC.SML
!
! BUILD SYSMAC
R LIBR
SYSMAC.SML=SYSMAC.MAC/M
^C
```

A.28 ULBLIB

```
! BUILD ULBLIB.OBJ
!
! ASSEMBLE ULBLIB
R MACRO
OBJ:ULB001,LST:ULB001/C=SRC:ULB001
OBJ:ULB002,LST:ULB002/C=SRC:ULB002
OBJ:ULB003,LST:ULB003/C=SRC:ULB003
OBJ:ULB004,LST:ULB004/C=SRC:ULB004
OBJ:ULB005,LST:ULB005/C=SRC:ULB005
OBJ:ULB006,LST:ULB006/C=SRC:ULB006
OBJ:ULB007,LST:ULB007/C=SRC:ULB007
OBJ:ULB008,LST:ULB008/C=SRC:ULB008
OBJ:ULB009,LST:ULB009/C=SRC:ULB009
OBJ:ULB010,LST:ULB010/C=SRC:ULB010
OBJ:ULB011,LST:ULB011/C=SRC:ULB011
OBJ:ULB012,LST:ULB012/C=SRC:ULB012
OBJ:ULB013,LST:ULB013/C=SRC:ULBMAC,ULB013
OBJ:ULB014,LST:ULB014/C=SRC:ULB014
OBJ:ULB015,LST:ULB015/C=SRC:ULB015
OBJ:ULB016,LST:ULB016/C=SRC:ULB016
OBJ:ULB017,LST:ULB017/C=SRC:ULB017
OBJ:ULB018,LST:ULB018/C=SRC:ULB018
OBJ:ULB019,LST:ULB019/C=SRC:ULB019
OBJ:ULB020,LST:ULB020/C=SRC:ULB020
OBJ:ULB021,LST:ULB021/C=SRC:ULB021
OBJ:ULB022,LST:ULB022/C=SRC:ULB022
OBJ:ULB023,LST:ULB023/C=SRC:ULB023
OBJ:ULB024,LST:ULB024/C=SRC:ULB024
OBJ:ULB025,LST:ULB025/C=SRC:ULB025
OBJ:ULB026,LST:ULB026/C=SRC:ULB026
OBJ:ULB027,LST:ULB027/C=SRC:ULB027
OBJ:ULB028,LST:ULB028/C=SRC:ULBMAC,ULB028
OBJ:ULB029,LST:ULB029/C=SRC:ULB029
OBJ:ULB030,LST:ULB030/C=SRC:ULBMAC,ULB030
OBJ:ULB031,LST:ULB031/C=SRC:ULBMAC,ULB031
OBJ:ULB032,LST:ULB032/C=SRC:ULBMAC,ULB032
OBJ:ULB033,LST:ULB033/C=SRC:ULBMAC,ULB033
OBJ:ULB034,LST:ULB034/C=SRC:ULB034
```

```
OBJ:ULB035,LST:ULB035/C=SRC:ULB035
OBJ:ULB036,LST:ULB036/C=SRC:ULB036
OBJ:ULB037,LST:ULB037/C=SRC:ULB037
OBJ:ULB038,LST:ULB038/C=SRC:ULB038
OBJ:ULB039,LST:ULB039/C=SRC:ULB039
OBJ:ULB040,LST:ULB040/C=SRC:ULB040
OBJ:ULB041,LST:ULB041/C=SRC:ULB041
OBJ:ULB042,LST:ULB042/C=SRC:ULB042
OBJ:ULB043,LST:ULB043/C=SRC:ULB043
OBJ:ULB044,LST:ULB044/C=SRC:ULB044
^C
! BUILD ULBLIB
R LIBR
BIN:ULBLIB/A,MAP:ULBLIB/W=//
OBJ:ULB001,ULB002,ULB003,ULB004,ULB005,ULB006
OBJ:ULB007,ULB008,ULB009,ULB010,ULB011,ULB012
OBJ:ULB013,ULB014,ULB015,ULB016,ULB017,ULB018
OBJ:ULB019,ULB020,ULB021,ULB022,ULB023,ULB024
OBJ:ULB025,ULB026,ULB027,ULB028,ULB029,ULB030
OBJ:ULB031,ULB032,ULB033,ULB034,ULB035,ULB036
OBJ:ULB037,ULB038,ULB039,ULB040,ULB041,ULB042
OBJ:ULB043,ULB044
//
^C
```

A.29 VTLIB

```
$ASSEMBLE VTLIB
.R MACRO
*OBJ:VTCAL1,LST:VTCAL1/C=SRC:VTCAL1
*OBJ:VTCAL2,LST:VTCAL2/C=SRC:VTCAL2
*OBJ:VTCAL3,LST:VTCAL3/C=SRC:VTCAL3
*OBJ:VTCAL4,LST:VTCAL4/C=SRC:VTCAL4
*OBJ:VTBASE,LST:VTBASE/C=SRC:VTBASE
^C

$BUILD VTLIB
.R LIBR
*BIN:VTHDLR,MAP:VTHDLR=OBJ:VTCAL1.OBJ,VTCAL2.OBJ,VTCAL3.OBJ,VTCAL4.OBJ/C
*OBJ:VTBASE.OBJ
^C
```

Appendix B
Loading Software Bootstraps

If your hardware configuration procedure does not include a hardware bootstrap, use one of the following procedures for loading a software bootstrap. Find the section for bootstrapping your device, and use the switch register to deposit the bootstrap in memory. If your hardware configuration includes a pushbutton console emulator instead of a switch register, follow the instructions in your hardware manual to load the appropriate bootstrap loader (listed in Tables B–1 through B–8).

B.1 Loading the RK11 (RK05) DECpack Bootstrap

Deposit the basic RK11 (RK05) disk bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 001000, in the switch register (see Table B–1).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–1) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.
6. Set the contents for the next address (from Table B–1) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–1: RK11 (RK05) Bootstrap Loader

Address	Contents
001000	012700
001002	177406
001004	012710
001006	177400
001010	012740
001012	000005
001014	105710
001016	100376
001020	005007

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–1.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

B.2 Loading the TC11 DECtape Bootstrap

Deposit the basic TC11 DECtape bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 001000, in the switch register (see Table B–2).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–2) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.

6. Set the contents for the next address (from Table B–2) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–2: TC11 Bootstrap Loader

Address	Contents
001000	012700
001002	177344
001004	012710
001006	177400
001010	012740
001012	004002
001014	005710
001016	100376
001020	012710
001022	000003
001024	105710
001026	100376
001030	012710
001032	000005
001034	105710
001036	100376
001040	005007

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–2.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

B.3 Loading the RX11 Bootstrap

The procedure to deposit the RX11 disk bootstrap loader in memory depends on the type of processor you have. If your computer is a PDP–11V/03, PDP–11/03, or LSI–11, see the *PDP–11/03 User's Manual* for instructions. Otherwise deposit the RX11 disk bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 001000, in the switch register (see Table B–3).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–3) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.
6. Set the contents for the next address (from Table B–3) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–3: RX11 Bootstrap Loader

Address	Contents
001000	005000
001002	012701
001004	177170
001006	105711
001010	001776
001012	012711
001014	000003
001016	005711
001020	001776
001022	100405
001024	105711
001026	100004
001030	116120
001032	000002
001034	000770
001036	000000
001040	005000
001042	000110

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–3.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

B.4 Loading the TJU16 or TM11 Magtape Bootstrap

Deposit the basic TJU16 or TM11 magtape bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 001000 or 010000, in the switch register (see Table B–4 or B–5).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–4 for TJU16 magtape or from Table B–5 for TM11 magtape) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.
6. Set the contents for the next address (from Table B–4 or B–5) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–4: **TJU16 Bootstrap Loader**

Address	Contents
001000	012700
001002	172440
001004	012710
001006	000021
001010	012760
001012	001300
001014	000032
001016	012760
001020	177777
001022	000006
001024	012720
001026	000031
001030	105760
001032	000010
001034	100375
001036	012710
001040	177000
001042	012740
001044	000071
001046	032710
001050	100200
001052	001775
001054	100007
001056	022760
001060	001000
001062	000014
001064	001403
001066	000005
001070	000167
001072	177704
001074	005007

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 001000 or 010000, in the switch register.

2. Press the LOAD ADDR switch.

3. Press the EXAM switch to display the contents of that address in the data register.

4. Compare the value in the data register with the value for that address in Table B–4 or B–5.

5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Table B–5: TM11 Bootstrap Loader

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

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. If the magtape is not positioned at the load point, rewind the magtape manually.
2. Set the starting address, 001000 or 010000, in the switch register.
3. Press the LOAD ADDR switch.
4. Set the ENABLE/HALT switch to ENABLE.
5. Press the START switch.

B.5 Loading the RK06 DECpack Bootstrap

Deposit the basic RK06 disk bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 001000, in the switch register (see Table B–6).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–6) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.
6. Set the contents for the next address (from Table B–6) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–6: RK06 Bootstrap Loader

Address	Contents
001000	012701
001002	177440
001004	012711
001006	000003
001010	032711
001012	100200
001014	001775
001016	012761
001020	177400
001022	000002
001024	012711
001026	000021
001030	032711
001032	100200
001034	001775
001036	005007

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–6.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

B.6 Loading the RL01/RL02 Disk Bootstrap

Deposit the basic RL01/RL02 disk bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 001000, in the switch register (see Table B–7).

3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–7) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.
6. Set the contents for the next address (from Table B–7) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–7: RL01/RL02 Bootstrap Loader

Address	Contents
001000	012701
001002	174400
001004	012761
001006	000013
001010	000004
001012	012711
001014	000004
001016	105711
001020	100376
001022	005061
001024	000002
001026	005061
001030	000004
001032	012761
001034	177400
001036	000006
001040	012711
001042	000014
001044	105711
001046	100376
001050	005007

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–7.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 001000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

B.7 Loading the RX211 Bootstrap

Deposit the basic RX211 bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address, 002000, in the switch register (see Table B–8).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–8) in the switch register.
5. Lift the DEP switch. The computer automatically advances to the next address.
6. Set the contents for the next address (from Table B–8) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–8: **RX211 Bootstrap Loader**

Address	Contents
002000	012701
002002	177170
002004	012700
002006	100240
002010	005002
002012	012705
002014	000200
002016	012704
002020	000401
002022	012703
002024	177172
002026	030011
002030	001776
002032	100440
002034	012711
002036	000407
002040	030011
002042	001776
002044	100433
002046	110413
002050	000304
002052	030011
002054	001776
002056	110413
002060	000304
002062	030011
002064	001776
002066	100422
002070	012711
002072	000403
002074	030011
002076	001776
002100	100415
002102	010513
002104	030011
002106	001776
002110	100411
002112	010213
002114	060502
002116	060502
002120	122424
002122	120427
002124	000007
002126	003737
002130	005000
002132	005007
002134	000000

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 002000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–8.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 002000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

INDEX

HOW TO ORDER ADDITIONAL DOCUMENTATION

From	Call	Write
Chicago	312–640–5612 8:15 A.M. to 5:00 P.M. CT	Digital Equipment Corporation Accessories & Supplies Center 1050 East Remington Road Schaumburg, IL 60195
San Francisco	408–734–4915 8:15 A.M. to 5:00 P.M. PT	Digital Equipment Corporation Accessories & Supplies Center 632 Caribbean Drive Sunnyvale, CA 94086
Alaska, Hawaii	603–884–6660 8:30 A.M. to 6:00 P.M. ET *or* 408–734–4915 8:15 A.M. to 5:00 P.M. PT	
New Hampshire	603–884–6660 8:30 A.M. to 6:00 P.M. ET	Digital Equipment Corporation Accessories & Supplies Center P.O. Box CS2008 Nashua, NH 03061
Rest of U.S.A., Puerto Rico*	1–800–258–1710 8:30 A.M. to 6:00 P.M. ET	
Canada		
British Columbia	1–800–267–6146 8:00 A.M. to 5:00 P.M. ET	Digital Equipment of Canada Ltd 940 Belfast Road Ottawa, Ontario K1G 4C2 Attn: A&SG Business Manager
Ottawa–Hull	613–234–7726 8:00 A.M. to 5:00 P.M. ET	
Elsewhere	112–800–267–6146 8:00 A.M. to 5:00 P.M. ET	
Elsewhere		Digital Equipment Corporation A&SG Business Manager*

*Prepaid orders from Puerto Rico must be placed with the local DIGITAL subsidiary (call 809–754–7575)

*c/o DIGITAL's local subsidiary or approved distributor

RT–11
Installation Guide
AA–H376B–TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- __ Assembly language programmer
- __ Higher-level language programmer
- __ Occasional programmer (experienced)
- __ User with little programming experience
- __ Student programmer
- __ Other (please specify) ______

Name ______ Date ______

Organization ______ Telephone ______

Street ______

City ______ State ______ Zip Code or Country ______

— — Do Not Tear — Fold Here and Tape — — — — — — — — — — — — — — — — — —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG/ML PUBLICATIONS, ML05–5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MA 01754

— — Do Not Tear — Fold Here —

Cut Along Dotted Line