RT–11 Installation Guide

Order Number AA–H376F–TC

August 1991

This guide tells you how to manually install the RT–11 operating system. It provides information necessary to bootstrap and preserve the distribution volume(s) and create, customize, preserve, and test the working system. Information in this guide is also useful to those who install their software distribution automatically.

Revision/Update Information: This manual supersedes the *RT–11 Installation Guide*, AA–H376E–TC.

Operating System: RT–11 Version 5.6

Digital Equipment Corporation
Maynard, Massachusetts

First Printing, March 1983
Updated, June 1983
Revised, July 1984
Revised, November 1985
Reprinted, August 1989
Revised, August 1991

S1523

This document was prepared using VAX DOCUMENT, Version 1.2.

Contents

Preface xi

Chapter 1 Introduction

1.1 Manual and Automatic Installation . . . 1–1
1.2 Identifying Your Needs . . . 1–1
1.2.1 Can You Automatically Install Your System? . . . 1–2
1.2.2 What Is Your Hardware Configuration? . . . 1–2
1.2.3 Which Monitor(s) Do You Need? . . . 1–2
1.2.3.1 Unmapped (SB and FB) Monitors . . . 1–2
1.2.3.2 Mapped (XB, XM, ZB, ZM) Monitors . . . 1–3
1.2.4 Which Features Do You Need? . . . 1–3
1.2.4.1 Is Installation All You Need to Do? . . . 1–3
1.2.4.2 Do You Need to Perform the System Generation Procedures? . . . 1–11
1.3 Choosing a Reading Path . . . 1–11

Chapter 2 Preparing for Installation

2.1 Deciding on Automatic or Manual Installation . . . 2–1
2.2 Surveying Installation Procedures . . . 2–1
2.3 Studying Software Kit Contents . . . 2–2
2.4 Selecting Components for Your Working System . . . 2–10
2.4.1 Components for Larger (Disk) System Devices . . . 2–10
2.4.2 Components for Smaller (Diskette) System Devices . . . 2–11
2.4.2.1 Monitor . . . 2–12
2.4.2.2 SWAP.SYS File . . . 2–12
2.4.2.3 System Device Handler . . . 2–12
2.4.2.4 Other Device Handlers . . . 2–13
2.4.2.5 Default System Library . . . 2–13
2.4.2.6 Help Package . . . 2–13
2.4.2.7 Printer Handlers . . . 2–13
2.4.2.8 MACRO–11 Assembler . . . 2–13
2.4.2.9 Printer Utilities . . . 2–14
2.4.2.10 Source Files . . . 2–14
2.4.2.11 Start-Up Command File . . . 2–14
2.4.2.12 Text Editors . . . 2–14
2.4.2.13 Utility Programs . . . 2–14
2.5 Planning the Arrangement of Components for Smaller System Devices . . . 2–15
2.5.1 Assigning the Default Device to the Data Device . . . 2–15

2.5.2 Creating a Separate Utilities Volume 2–15
2.5.3 Creating Several System Volumes 2–16
2.5.4 Creating Volumes for Use with *Introduction to RT–11* 2–16
2.5.4.1 Working System for Nonprogramming Part 2–17
2.5.4.2 Working System for Programming Part 2–17
2.6 Choosing Software Customizations 2–17
2.6.1 Changing Characters That Indicate Insertion/Deletion 2–18
2.6.2 Changing Default Output Device from Printer to Terminal 2–19
2.6.3 Changing the LD Handler Suffix Character 2–19
2.6.4 Changing the Number of Directory Columns 2–19
2.6.5 Changing the Default Order of Directory Listings 2–20
2.6.6 Changing the Number of /Q Program Sections LINK Allows 2–21
2.6.7 Changing the Size of LINK's Library Module List 2–21
2.6.8 Changing the Size of the QUEUE Work File 2–22
2.6.9 Modifying EDIT for Terminals with Nonstandard ESCAPE Codes 2–22
2.6.10 Installing Other Devices 2–23
2.6.10.1 CSR and Vector Addresses for Printers, RX01/02 Diskettes and MSCP Disks . . . 2–24
2.6.10.2 Hardware Magtape Support 2–25
2.6.10.3 Magtape Parity and Density 2–27
2.6.11 Modifying BATCH to Save Space 2–27
2.6.12 Modifying LINK to Change the Default SYSLIB Device 2–28
2.6.13 Modifying the Help Text 2–28
2.6.14 Preventing Fatal System Errors from Causing a Reset 2–30
2.6.15 Running RT–11 in Less Memory Than Is Available 2–31
2.6.16 Setting VTCOM Default Dial String 2–32
2.6.17 Setting Upper Limit on a File Size 2–33
2.6.18 Changing the Number of Bad Blocks That BUP Accepts 2–34
2.6.19 Changing the Default File Type Extension of IND Control Files 2–34
2.6.20 Setting Upper Limit on Memory Size 2–35
2.6.21 Suppressing the Bootstrap Message 2–35
2.6.22 Suppressing the Start-Up Indirect Command File 2–35
2.6.23 Suppressing the Start-Up Indirect Command File Echo 2–36
2.6.24 Changing the Bootstrap Message 2–36
2.6.25 Changing the Default Device for Indirect Command Files 2–36
2.6.26 Changing the Default File Type for Indirect Command Files 2–37
2.6.27 Changing the Default Device for the FRUN Command 2–37
2.6.28 Changing the Default File Type for the FRUN Command 2–38
2.6.29 Changing the Default Device for the EDIT Utility Command 2–38
2.6.30 Changing the Default File Name for the EDIT Command 2–39
2.6.31 Using Examine and Deposit Above the Background Job 2–39
2.6.32 Changing the Default Device for QUEMAN 2–39
2.6.33 Changing the Indirect Command File Nesting Depth 2–40
2.6.34 Changing the Threshold for Resuming Output-Stalled Jobs 2–40
2.6.35 Changing the Default Number of Directory Segments 2–41
2.6.36 Changing the Banner Page Width 2–42

2.6.37 Modifying Listing Page Length in LINK 2–43
2.6.38 Assigning the HELP File 2–43
2.6.39 Changing the Device from Which IND.SAV Is Run 2–44
2.6.40 Supporting Bad Block Replacement in User-Written Handlers 2–44
2.6.41 Supporting User-Written Magtape Handlers 2–45
2.6.42 Eliminating or Changing the IND Control File Terminating Message 2–46
2.6.43 Changing Default File Type of Logical Disk Files 2–47
2.6.44 Changing QUEUE to Allow First Form Feed 2–47
2.6.45 Changing Listing Page Length in MACRO and CREF 2–48
2.6.46 Changing Default Command File Processor to IND 2–48
2.6.47 Limiting Amount of Memory KEX Requests 2–49
2.6.48 Changing MACRO's Default .LIST/.NLIST Options 2–49
2.6.49 Changing MACRO's Default .ENABLE/.DISABLE Options 2–50
2.6.50 Modifying KED Default File Type 2–51
2.6.51 Changing SPOOL's Work File Size 2–51
2.6.52 Changing SPOOL's Output Device 2–52
2.6.53 Changing LINK for Default 132-Column Link Map and Global Cross-Reference Table 2–52
2.6.54 Changing Default Size of MACRO–11 Work File 2–52
2.6.55 Changing Name and Default Device for UCL.DAT 2–53
2.6.56 Suppressing Dynamic Allocation of a Region for IND 2–53
2.6.57 Changing Size of UCL.DAT 2–54
2.6.58 Changing Default VTCOM Send Speed 2–54
2.6.59 Modifying System Macro Library SYSMAC.SML 2–55
2.6.60 Modifying VTCOM Default Dial String Prefix 2–55
2.6.61 Modifying VTCOM Default Dial String Suffix 2–56
2.6.62 Automatically Creating KED/KEX Journal Files 2–57
2.6.63 Adjusting Default KED/KEX Journal File Size Allocation 2–58
2.6.64 Creating a Monitor-Independent ODT Debugger 2–58
2.6.65 Forcing an Explicit KED/KEX Journal File Device 2–59
2.6.66 Modifying the Storage Capacity for LEARN Macros 2–60
2.6.67 Modifying the Maximum Number of LEARN Macros 2–60
2.6.68 Restoring LOCAL Form Feed Processing 2–61
2.6.69 Removing the JSW Bit 6 Check for SL 2–62
2.6.70 Changing SL Character Edit Mode to REPLACE 2–63
2.6.71 Changing the VTCOM Command Mode Character 2–64
2.6.72 Forcing Obsolete SJ Monitor Value in CONFIG Bit $FBMON 2–64
2.6.73 Making LINK /G Option a Default 2–65
2.6.74 Changing the Default Magtape Label Size 2–66

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

3.1 Bootstrapping the Distribution Volume 3–1
3.2 Creating the Working System from Chosen Components 3–3
3.3 Installing the Bootstrap on Any Volumes That Need to Be Bootable 3–4
3.4 Customizing the System 3–4
3.5 Compressing Each Volume 3–5
3.6 Preserving the Working System 3–5
3.7 Testing the Working System 3–6
3.7.1 Preparing the Background Demonstration Program 3–7
3.7.1.1 Edit the Background Demonstration Program 3–7
3.7.1.2 Assemble the Background Demonstration Program 3–8
3.7.1.3 Link the Background Demonstration Program 3–8
3.7.1.4 Run the Background Demonstration Program 3–8
3.7.2 Preparing the Foreground Demonstration Program 3–9
3.7.2.1 Assemble the Foreground Demonstration Program 3–10
3.7.2.2 Link the Foreground Demonstration Program 3–10
3.7.2.3 Run the Foreground and Background Demonstration Programs 3–10
3.8 Performing the System Generation Process 3–11
3.9 Further Information 3–11

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

4.1 Bootstrapping the Distribution Volume 4–1
4.2 Copying the Distribution Volumes to the Disk 4–2
4.3 Preserving the Distribution Volumes 4–4
4.4 Creating the Working System from Chosen Components 4–4
4.5 Installing the Bootstrap on the Disk 4–5
4.6 Customizing the System 4–6
4.7 Compressing the Disk 4–6
4.8 Preserving the Working System 4–6
4.8.1 Backing Up the System on RX01 Diskettes 4–7
4.8.2 Backing Up the System on Another Disk 4–7
4.9 Testing the Working System 4–8
4.9.1 Preparing the Background Demonstration Program 4–9
4.9.1.1 Edit the Background Demonstration Program 4–9
4.9.1.2 Assemble the Background Demonstration Program 4–10
4.9.1.3 Link the Background Demonstration Program 4–12
4.9.1.4 Run the Background Demonstration Program 4–12
4.9.2 Preparing the Foreground Demonstration Program 4–12
4.9.2.1 Assemble the Foreground Demonstration Program 4–12
4.9.2.2 Link the Foreground Demonstration Program 4–13
4.9.2.3 Run the Foreground and Background Demonstration Programs 4–13
4.10 Performing the System Generation Process 4–14
4.11 Further Information 4–14

Chapter 5 Installing a System Distributed on RL02 to Run on Hard Disk

5.1 Bootstrapping the Distribution Disk 5–1
5.2 Preserving the Distribution Disk 5–2
5.3 Creating the Working System from Chosen Components 5–4
5.4 Installing the Bootstrap on the Disk 5–5
5.5 Customizing the System 5–5
5.6 Compressing the Disk 5–6
5.7 Preserving the Working System 5–6
5.8 Testing the Working System 5–7
5.8.1 Preparing the Background Demonstration Program 5–8
5.8.1.1 Edit the Background Demonstration Program 5–8
5.8.1.2 Assemble the Background Demonstration Program 5–8
5.8.1.3 Link the Background Demonstration Program 5–9
5.8.1.4 Run the Background Demonstration Program 5–10
5.8.2 Preparing the Foreground Demonstration Program 5–10
5.8.2.1 Assemble the Foreground Demonstration Program 5–10
5.8.2.2 Link the Foreground Demonstration Program 5–11
5.8.2.3 Run the Foreground and Background Demonstration Programs 5–11
5.9 Performing the System Generation Process 5–12
5.10 Further Information 5–12

Chapter 6 Installing a System Distributed on RL02 to Run on a Small Device

6.1 Bootstrapping the Distribution Disk 6–1
6.2 Preserving the Distribution Disk 6–2
6.3 Creating the Working System from Chosen Components 6–4
6.3.1 Initializing RX50 Diskettes 6–4
6.3.2 Formatting and Initializing RX01/RX02 Diskettes 6–5
6.3.3 Copying the Selected Files onto the Diskette 6–6
6.4 Installing the Bootstrap on Any Diskettes That Need to Be Bootable 6–6
6.5 Customizing the System 6–7
6.6 Compressing Each Diskette 6–7
6.7 Preserving the Working System 6–8
6.8 Testing the Working System 6–9
6.8.1 Preparing the Background Demonstration Program 6–10
6.8.1.1 Edit the Background Demonstration Program 6–10
6.8.1.2 Assemble the Background Demonstration Program 6–10
6.8.1.3 Link the Background Demonstration Program 6–11
6.8.1.4 Run the Background Demonstration Program 6–11
6.8.2 Preparing the Foreground Demonstration Program 6–12
6.8.2.1 Assemble the Foreground Demonstration Program 6–12
6.8.2.2 Link the Foreground Demonstration Program 6–13
6.8.2.3 Run the Foreground and Background Demonstration Programs 6–13
6.9 Performing the System Generation Process 6–14

6.10 Further Information 6–14

Chapter 7 Installing a System Distributed on RX02 to Run on RX02

7.1 Bootstrapping the Distribution Diskette 7–2
7.2 Creating the Working System from Chosen Components 7–3
7.3 Installing the Bootstrap on Any Volumes That Need to Be Bootable 7–5
7.4 Customizing the System 7–5
7.5 Compressing Each Diskette 7–5
7.6 Preserving the Working System 7–6
7.7 Testing the Working System 7–7
7.7.1 Preparing the Background Demonstration Program 7–8
7.7.1.1 Edit the Background Demonstration Program 7–8
7.7.1.2 Assemble the Background Demonstration Program 7–8
7.7.1.3 Link the Background Demonstration Program 7–10
7.7.1.4 Run the Background Demonstration Program 7–10
7.7.2 Preparing the Foreground Demonstration Program 7–10
7.7.2.1 Assemble the Foreground Demonstration Program 7–10
7.7.2.2 Link the Foreground Demonstration Program 7–11
7.7.2.3 Run the Foreground and Background Demonstration Programs 7–11
7.8 Performing the System Generation Process 7–12
7.9 Further Information 7–12

Chapter 8 Installing a System Distributed on Magtape to Run on Hard Disk

8.1 Bootstrapping the First Distribution Magtape 8–2
8.2 Preserving the Magtape Distribution 8–4
8.2.1 Using MTB.COM and MSB.COM 8–5
8.2.2 Using MUB.COM 8–6
8.3 Creating a Master (Distribution) Magtape 8–6
8.4 Creating the Working System from Chosen Components 8–6
8.5 Installing the Bootstrap on the Disk 8–7
8.6 Customizing the System 8–7
8.7 Compressing the Disk 8–7
8.8 Preserving the Working System 8–8
8.9 Testing the Working System 8–10
8.9.1 Preparing the Background Demonstration Program 8–11
8.9.1.1 Edit the Background Demonstration Program 8–11
8.9.1.2 Assemble the Background Demonstration Program 8–11
8.9.1.3 Link the Background Demonstration Program 8–13
8.9.1.4 Run the Background Demonstration Program 8–13
8.9.2 Preparing the Foreground Demonstration Program 8–13
8.9.2.1 Assemble the Foreground Demonstration Program 8–14
8.9.2.2 Link the Foreground Demonstration Program 8–14
8.9.2.3 Run the Foreground and Background Demonstration Programs 8–14

8.10 Performing the System Generation Process 8–15
8.11 Further Information 8–15

Chapter 9 Installing RT–11 on a Professional 300 Series Computer

9.1 Bootstrapping the System Diskette 9–2
9.2 Copying the System Diskette onto the System Disk 9–3
9.3 Installing the Bootstrap on the System Disk 9–3
9.4 Copying the Distribution Diskettes onto the System Disk 9–3
9.5 Creating the Working System from Chosen Components 9–4
9.6 Customizing the System 9–4
9.7 Compressing the System Disk 9–4
9.8 Preserving the Working System 9–5
9.9 Testing the Working System 9–6
9.9.1 Preparing the Background Demonstration Program 9–7
9.9.1.1 Edit the Background Demonstration Program 9–7
9.9.1.2 Assemble the Background Demonstration Program 9–8
9.9.1.3 Link the Background Demonstration Program 9–8
9.9.1.4 Run the Background Demonstration Program 9–8
9.9.2 Preparing the Foreground Demonstration Program 9–9
9.9.2.1 Assemble the Foreground Demonstration Program 9–10
9.9.2.2 Link the Foreground Demonstration Program 9–10
9.9.2.3 Run the Foreground and Background Demonstration Programs 9–10
9.10 Performing the System Generation Process 9–11
9.11 Further Information 9–11

Appendix A Modifying MDUP to Recognize a Device Handler

Appendix B Loading Software Bootstraps

B.1 Loading the TMSCP Bootstrap B–1
B.1.1 Loading the TMSCP Bootstrap, Using MICRO–ODT B–1
B.1.2 Loading the TMSCP Bootstrap, Using the Switch Register B–2
B.1.3 Addresses and Contents of the TMSCP Bootstrap Loader B–3
B.2 Loading the MSCP Bootstrap B–4
B.2.1 Loading the MSCP Bootstrap, Using MICRO–ODT B–4
B.2.2 Loading the MSCP Bootstrap, Using the Switch Register B–5
B.2.3 Addresses and Contents of the MSCP Bootstrap Loader B–6
B.3 Loading the RK11 (RK05) DECpack Bootstrap B–8
B.4 Loading the RX11 Bootstrap B–9
B.5 Loading the TJU16 or TM11 Magtape Bootstrap B–10
B.6 Loading the RK06/RK07 DECpack Bootstrap B–12
B.7 Loading the RL01/RL02 Disk Bootstrap B–14
B.8 Loading the RX211 Bootstrap B–15

B.9 Loading the TSV05 Bootstrap, Using MICRO–ODT B–17
B.10 Loading the TSV05 Bootstrap, Using the Switch Register B–18

Index

Figures

3–1 DEMOBG Assembly Listing 3–9
4–1 DEMOBG Assembly Listing 4–11
5–1 DEMOBG Assembly Listing 5–9
6–1 DEMOBG Assembly Listing 6–11
7–1 DEMOBG Assembly Listing 7–9
8–1 DEMOBG Assembly Listing 8–12
9–1 DEMOBG Assembly Listing 9–9

Tables

1–1 Mapped Monitor Description 1–3
1–2 Features Available Through Simple Software Customizations 1–4
1–3 Chapters You Should Read 1–11
2–1 RT–11 Software Components 2–2
2–2 Device Size 2–11
2–3 File-Structured and Hardware Magtape Handlers for Unmapped Monitors 2–26
2–4 File-Structured and Hardware Magtape Handlers for Mapped Monitors 2–27
B–1 TMSCP Bootstrap Loader B–3
B–2 MSCP Bootstrap Loader B–6
B–3 RK11 (RK05) Bootstrap Loader B–8
B–4 RX11 Bootstrap Loader B–9
B–5 TJU16 Bootstrap Loader B–10
B–6 TM11 Bootstrap Loader B–11
B–7 RK06/RK07 Bootstrap Loader B–13
B–8 RL01/RL02 Bootstrap Loader B–14
B–9 RX211 Bootstrap Loader B–15
B–10 TSV05 Bootstrap Loader B–18

Preface

Document Structure

This manual introduces you to the RT–11 Version 5 software kit and tells you how to manually install your operating system. For those who automatically install their software, Chapter 2 of this guide provides useful information on customizing the distributed software. For those who cannot install their software automatically, the modular structure of this manual allows you to read only the chapters that you need for your combination of hardware and software.

Chapter 1 of the manual introduces you to manual 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 procedures to modify the MDUP utility to recognize unsupported or user written device handlers 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 *Introduction to RT–11*, is particularly helpful when you perform the procedures in this manual.

If you are a FORTRAN, BASIC–PLUS, FMS–11, CTS–300, or other layered product user, you build your layered product 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.

Audience

This manual is written for the RT–11 users who are going to install their system manually and/or customize an automatically installed system.

Conventions

The following conventions are used in this guide.

Convention	Meaning
Black print	In examples, black print indicates output lines or prompting characters that the system displays. For example: `.BACKUP/INITIALIZE DL0:F*.FOR DU1:WRK` `Mount output volume in DU1:; continue? Y`
Red print	In examples, red print indicates user input
Braces ({ })	In command syntax examples, braces enclose options that are mutually exclusive. You can choose only one option from the group of options that appear in braces.
Brackets ([ ])	Square brackets in a format line represent optional parameters, qualifiers, or values, unless specified otherwise.
lowercase characters	In command syntax examples, lowercase characters represent elements of a command for which you supply a value. For example: `DELETE filespec`
Question marks	In the customization patches, question marks (?) are used to represent current values in a location. Those values cannot be anticipated and therefore cannot be included in the patch illustration.
UPPERCASE characters	In command syntax examples, uppercase characters represent elements of a command that should be entered exactly as given.
RET	RET in examples represents the RETURN key. Unless the manual indicates otherwise, terminate all commands or command strings by pressing RET.
CTRL/*x*	CTRL/*x* indicates a control key sequence. While pressing the key labeled Ctrl, press another key. For example: CTRL/C

Associated Documents

Basic Books

- *Introduction to RT–11*
- *Guide to RT–11 Documentation*
- *PDP–11 Keypad Editor User's Guide*
- *PDP–11 Keypad Editor Reference Card*

- *RT–11 Commands Manual*
- *RT–11 Mini-Reference Manual*
- *RT–11 Master Index*
- *RT–11 System Message Manual*
- *RT–11 System Release Notes*

Installation Specific Books

- *RT–11 Automatic Installation Guide*
- *RT–11 System Generation Guide*

Programmer Oriented Books

- *RT–11 IND Control Files Manual*
- *RT–11 System Utilities Manual*
- *RT–11 System Macro Library Manual*
- *RT–11 System Subroutine Library Manual*
- *RT–11 System Internals Manual*
- *RT–11 Device Handlers Manual*
- *RT–11 Volume and File Formats Manual*
- *DBG–11 Symbolic Debugger User's Guide*

Chapter 1 Introduction

Your first step in using RT–11 is to install it on your system. You can install RT–11 automatically or manually, depending on your configuration.

1.1 Manual and Automatic Installation

You can install RT–11 automatically or manually. See the *RT–11 Automatic Installation Guide* to determine if you can automatically install RT–11. If your hardware does not support automatic installation, you must install RT–11 manually. Manual installation requires:

- Bootstrapping and preserving the distribution volumes.
- Creating the working system from selected components.
- Optionally customizing the working system.
- Preserving and testing the working system.

Usually, your system is ready to run once you have installed and tested it. However, if you need certain special features, such as multiterminal support, you must perform the system generation process. In this case, run the SYSGEN program, answer its questions, and assemble your own generated monitor(s) and device handlers. See the *RT–11 System Generation Guide* for information about generating specialized monitors.

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

1.2 Identifying Your Needs

To select the procedures you must perform and the reading path to follow, use the information in this chapter to answer the following questions:

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

1.2.1 Can You Automatically Install Your System?

Look at the *RT–11 Automatic Installation Guide* to determine if you can install RT–11 automatically. If possible, you should install RT–11 automatically unless you have a good reason for doing a manual installation. The automatic installation procedure is simple and it configures and tests your system automatically.

If, according to the *RT–11 Automatic Installation Guide*, your system is supported for automatic installation, go now to that manual for directions. After automatically installing and configuring your system, return to this chapter for more installation information.

If, according to the *RT–11 Automatic Installation Guide*, your system is not supported for automatic installation, continue reading this chapter and follow the appropriate reading path as defined in Table 1–3.

1.2.2 What Is Your Hardware Configuration?

Your hardware configuration is an important consideration for three reasons:

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

Chapters 3 through 9 contain detailed procedures for manually 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 1–3 to identify your reading path.

1.2.3 Which Monitor(s) Do You Need?

You have to decide which monitor or monitors you need to know what steps to follow in getting started. Your distribution kit includes the unmapped SB and FB monitors and the mapped XB, XM, ZB, and ZM monitors.

Sections 1.2.3.1 and 1.2.3.2 describe the RT–11 monitors. More monitor information is located in the *Introduction to RT–11*.

1.2.3.1 Unmapped (SB and FB) Monitors

The unmapped monitors (SB and FB) are appropriate for processors that contain only kernel (64K bytes) memory. The SB monitor supports one job while the FB monitor supports a foreground and a background job.

The SB monitor is the smallest RT–11 monitor. Use the SB monitor if your processor contains only 64K bytes of memory and your application is too large to run under the FB monitor.

Use the FB monitor if your processor contains only 64K bytes of memory and your application will run in the available address space. The FB monitor allows you to execute an 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 SB monitor; all the facilities available to you as an SB user are available to you as an FB user in the background.

1.2.3.2 Mapped (XB, XM, ZB, ZM) Monitors

The mapped monitors are for processors that contain extended or mapped memory. Table 1–1 lists the basic characteristics. For more information on mapped monitors, see the Software Product Description (SPD) and the *Introduction to RT–11*.

Table 1–1: Mapped Monitor Description

Monitor	Characteristics
XB	Single job. Does not support I & D (instruction and data) address space separation or Supervisor Mode
XM	Multiple jobs. Supports up to 8 concurrent jobs. Does not support I & D address space separation or Supervisor Mode
ZB	Single job. Supports I & D address space separation and Supervisor Mode
ZM	Multiple jobs. Supports up to 8 concurrent jobs, I & D address space separation and Supervisor Mode

1.2.4 Which Features Do You Need?

The characteristics of distributed software (monitors, utilities, and device handlers), are described in the Software Product Description (SPD), which is included with your documentation kit. You can change some characteristics of distributed software by using the customization patches listed in Table 1–2 and described in Chapter 2. If the distributed or customized software does not provide a feature you need, you must perform a system generation as described in the *RT–11 System Generation Guide*.

1.2.4.1 Is Installation All You Need to Do?

Decide whether installing the distributed software is all you need to do or whether you will also need to perform the system generation process. Table 1–2 summarizes the features you can add to your system by performing simple customizations during or after installation. Chapter 2 describes these software customizations in detail and provides instructions for implementing them. Many users will not need to make any of these customizations.

Table 1–2: Features Available Through Simple Software Customizations

Feature	Section	Description
Changing characters SRCCOM uses to indicate insertions and deletions	2.6.1	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	2.6.2	If your configuration does not include a printer, you can change the default output device (which certain monitor commands use) from printer to terminal. To make this change, you can edit the start-up command file.
Changing the LD handler suffix character	2.6.3	You can change the LD handler suffix character (LD*x*) to anything other than X.
Changing number of directory columns	2.6.4	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	2.6.5	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.
Changing number of /Q p-sects LINK allows	2.6.6	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	2.6.7	To change the default size of LINK's list of library modules, you can modify LINK. LINK normally creates a list of 252_8 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	2.6.8	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.

Table 1–2 (Cont.): Features Available Through Simple Software Customizations

Feature	Section	Description
Modifying the EDIT utility	2.6.9	If you have a terminal with nonstandard ESCAPE code (that is, the terminal generates 175_8 or 176_8 rather than 33_8), you must modify EDIT.SAV so that it operates correctly.
Installing other devices	2.6.10	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 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, 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 MM.SYS, MT.SYS, MS.SYS, or MU.SYS (or the mapped versions with an X suffix). *Set magtape parity and density*: If you need to set magtape parity or density to other than the standard 800 bits/in, odd parity, you can use the monitor SET command for TM11 or TJU16 magtape (but not TS11).
Modifying BATCH to save space	2.6.11	If (after performing the system generation procedure and including BATCH support) 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	2.6.12	By customizing the linker, you can change the device on which the default system library, SYSLIB.OBJ, resides.
Modifying help text	2.6.13	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.

Table 1–2 (Cont.): Features Available Through Simple Software Customizations

Feature	Section	Description
Preventing fatal system errors from causing a reset	2.6.14	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	2.6.15	You can customize your software if your application requires that RT–11 run in less memory than is available.
Setting VTCOM default dial string	2.6.16	You can modify VTCOM to set a default string for the DIAL command.
Setting upper limit on file size	2.6.17	If your application requires an upper limit to the size of a file, you can change the monitor to set the limit you require.
Changing the number of bad blocks that BUP accepts	2.6.18	If you want BUP to handle more than 25 bad blocks on the input device, change BUP to accept and process as many as you specify.
Changing the default file type extension of IND control files	2.6.19	You can change the default IND file type extension from .COM to whatever you specify.
Setting upper limit on memory size	2.6.20	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	2.6.21	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	2.6.22	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	2.6.23	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	2.6.24	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	2.6.25	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:.

Table 1–2 (Cont.): Features Available Through Simple Software Customizations

Feature	Section	Description
Changing default file type for indirect command files	2.6.26	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	2.6.27	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	2.6.28	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 utility command	2.6.29	If you want to change the default device for the EDIT command with EDIT.SAV, you can modify the monitor. Normally, the monitor looks for the hard-copy editor file EDIT.SAV on device SY.
Changing default file name for EDIT command	2.6.30	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	2.6.31	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	2.6.32	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	2.6.33	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.

Table 1–2 (Cont.): Features Available Through Simple Software Customizations

Feature	Section	Description
Changing threshold for resuming output-stalled jobs	2.6.34	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	2.6.35	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	2.6.36	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	2.6.37	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.
Assigning HELP file	2.6.38	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	2.6.39	You can modify the monitor to change the device from which IND is run.
Replacing bad blocks	2.6.40	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	2.6.41	You must customize DUP, PIP, and MDUP if you use a user-written magtape handler.
Eliminating or changing the IND control file terminating message	2.6.42	You can eliminate or change the terminating message of IND control files.
Changing default file type of logical disk files	2.6.43	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.
Changing QUEUE to allow first form feed	2.6.44	You can modify QUEUE to never suppress the initial form feed in a file.
Changing listing page length in MACRO and CREF	2.6.45	You can change the default (60 lines) page listing length for MACRO and CREF.
Changing default command file processor to IND	2.6.46	You can change the monitor command file processor from KMON to IND.

Table 1–2 (Cont.): Features Available Through Simple Software Customizations

Feature	Section	Description
Limiting amount of memory KEX requests	2.6.47	You can reduce memory requested by KEX from the default 32KW to your requirements by using the .SETTOP programmed request.
Changing MACRO's default .LIST /.NLIST options	2.6.48	You can change the default options for the .LIST/.NLIST macros.
Changing MACRO's default .ENABLE/.DISABLE options	2.6.49	You can change the default options for the .ENABLE/.DISABLE macros.
Modifying KED default file type	2.6.50	You can change the KED default file type.
Changing SPOOL's work file size	2.6.51	You can change the size of SPOOL's work file.
Changing SPOOL's output device	2.6.52	You can change SPOOL's default output device.
Changing LINK's default map and table output to 132 columns	2.6.53	You can change LINK's default link map and global cross-reference table output to 132 columns.
Changing default size of MACRO–11 work file	2.6.54	You can change the default size of MACRO–11's work file to a value from 128_{10} to a maximum of 256_{10} blocks.
Changing name and default device for UCL.DAT	2.6.55	You can change the name and default device for UCL.DAT.
Suppressing dynamic allocation of a region for IND	2.6.56	You can suppress IND's dynamic allocation of a region in extended memory for its symbol table.
Changing size of UCL.DAT	2.6.57	You can change the maximum number of user-defined UCL commands from 31_{10} to a maximum you specify.
Changing default VTCOM send speed	2.6.58	You can change the default VTCOM send speed to FAST if the host terminal service supports XON/XOFF.
Modifying system macro library SYSMAC.SML	2.6.59	You can modify the system macro library by running the SPLIT utility.
Modifying VTCOM default dial string prefix	2.6.60	You can modify the VTCOM default dial string prefix.
Modifying VTCOM default dial string suffix	2.6.61	You can modify the VTCOM default dial string suffix.
Automatically creating KED/KEX journal files	2.6.62	You can cause the keypad editor to automatically open and write a journal file at each editing session.
Adjusting default KED/KEX journal file size allocation	2.6.63	You can change the number of blocks allocated by the keypad editor on the default journal file device.

Table 1–2 (Cont.): Features Available Through Simple Software Customizations

Feature	Section	Description
Creating a monitor-independent ODT debugger	2.6.64	You can create an ODT variant that does not require that the operating system be loaded in memory.
Forcing an explicit KED/KEX journal file device	2.6.65	You can force the default journal file device to be other than DK.
Modifying the storage capacity for LEARN macros	2.6.66	You can modify the amount of storage capacity (in memory) that the keypad editor allocates for LEARN macro storage.
Modifying the maximum number of LEARN macros	2.6.67	By default, the keypad editor allows the definition of 8_{10} LEARN macros at one time. You can modify that limit.
Restoring LOCAL form feed processing	2.6.68	You can restore the recognition of form feeds in files being processed by the LOCAL keypad editor feature.
Removing the JSW bit 6 check for SL	2.6.69	The single-line editor (SL) checks bit 6 of the Job Status Word before editing a command line. You can remove that bit check.
Changing SL character edit mode to REPLACE	2.6.70	SL can edit single characters in two modes: REPLACE and INSERT. You can change the default edit mode from INSERT to REPLACE.
Changing the VTCOM command mode character	2.6.71	You can change the character you type to VTCOM to enter command mode from CTRL/P to one you specify.
Forcing obsolete SJ monitor value in CONFIG bit $FBMON	2.6.72	You can change the default value of the $FBMON bit in the CONFIG word from set to clear, if required, to load handlers that require a clear $FBMON bit.
Making LINK /G option a default	2.6.73	You can make the /G option a default whenever LINK is run. The /G option is not available under DCL; you can use this customization to include that option when issuing LINK commands under DCL.
Changing the default magtape label size	2.6.74	The current magtape label size is 80_{10} bytes. If you need to change that size to some other value, use this customization. (80 byte labels are required for any magtape that will be subsequently read by the VMS operating system.)

1.2.4.2 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 that are not included in the distributed software or listed in Table 1–2. 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* further information about these features.

1.3 Choosing a Reading Path

You only need to read the chapters that pertain to your configuration. Table 1–3 gives you the information you need to select these chapters.

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 Table 1–2, but you may want to delay making any customizations until you have gained more experience with RT–11. Further, be sure to read the *Introduction to RT–11* before using RT–11 for any critical applications.

Table 1–3: Chapters You Should Read

If Your Distribution Medium is:	And if Your System Device on Target System is:	Then Read Chapters:
Single-Density Diskette RX01	Single-Density Diskette	1, 2, 3
	Hard Disk RL01 RL02 RK05 RK06 RK07	1, 2, 4

Table 1–3 (Cont.): Chapters You Should Read

If Your Distribution Medium is:	And if Your System Device on Target System is:	Then Read Chapters:
Hard Disk	Hard Disk	1, 2, 5
RL02	RL01 RL02 RK05 RK06 RK07 RC25	
	Single-Density or Double-Density Diskette	1, 2, 6
Double-Density Diskette	Double-Density Diskette	1, 2, 7
RX02	RX02	
Magtape	Hard Disk	1, 2, 8
9-track, 1600 bpi TMSCP	RL01 RL02 RK05 RK06 RK07	
Diskette RX50		
	Professional 350/380 MSCP disk	1, 2, 9 Use Automatic Installation Procedure

Chapter 2
Preparing for Installation

You should take the following steps before you begin to install your system:

1. Decide on automatic or manual installation.
2. Survey the general installation procedures.
3. Study the contents of your software kit.
4. Select the components you need in your working system.
5. Plan the arrangement of components.
6. Choose the software customizations you need to make.

2.1 Deciding on Automatic or Manual Installation

If you are uncertain as to whether your hardware configuration supports automatic installation, see the *RT–11 Automatic Installation Guide* for information. In general, if your hardware configuration supports automatic installation, you should install your software distribution using that method and return to Section 2.6 of this chapter.

If your hardware configuration supports automatic installation, install your distribution using that method. The remainder of this chapter provides information you might find useful however you install your software. You can, however, safely ignore the information in this manual that follows this chapter.

2.2 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. You then automatically or manually 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 can 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.

2.3 Studying Software Kit Contents

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. RT–11 is distributed on an RL02 disk, on RX01, RX02, and RX50 diskettes, and on reel and cartridge 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
RT11MT.SYS	Magtape monitor used only to build MDUP.xx for user-supplied (not distributed) device handler
RT11SB.SYS	Unmapped single-job monitor
RT11FB.SYS	Unmapped foreground/background monitor
RT11XB.SYS	Mapped extended memory single-job monitor; not distributed—must be built by SYSGEN from distributed answer file
RT11XM.SYS	Mapped extended memory monitor
RT11ZB.SYS	Fully-mapped extended memory single-job monitor; not distributed—must be built by SYSGEN from distributed answer file
RT11ZM.SYS	Fully-mapped extended memory monitor
Device Handlers	
DL.SYS	RL11/RL01/RL02 handler for unmapped monitors
DLX.SYS	RL11/RL01/RL02 handler for mapped monitors
DM.SYS	RK611/RK06/RK07 handler for unmapped monitors
DMX.SYS	RK611/RK06/RK07 handler for mapped monitors
DU.SYS	MSCP device handler for unmapped monitors
DUX.SYS	MSCP device handler for mapped monitors
DW.SYS	Professional 300 series processor disk handler for unmapped monitors
DWX.SYS	Professional 300 series processor disk handler for mapped monitors
DX.SYS	RX11/RX01 single-density diskette handler for unmapped monitors
DXX.SYS	RX11/RX01 single-density diskette handler for mapped monitors
DY.SYS	RX211/RX02 handler for unmapped monitors

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

Type of Software	Description
DYX.SYS	RX211/RX02 handler for mapped monitors
DZ.SYS	Professional 300 series diskette handler for unmapped monitors
DZX.SYS	Professional 300 series diskette handler for mapped monitors
LD.SYS	Logical disk subsetting handler for unmapped monitors
LDX.SYS	Logical disk subsetting handler for mapped monitors
LP.SYS	Parallel interface printer handler for unmapped monitors
LPX.SYS	Parallel interface printer handler for mapped monitors
LS.SYS	Serial interface printer handler for unmapped monitors
LSX.SYS	Serial interface printer handler for mapped monitors
MM.SYS	File-structured TJU16 handler for unmapped monitors
MMX.SYS	File-structured TJU16 handler for mapped monitors
MS.SYS	File-structured TS11 handler for unmapped monitors
MSX.SYS	File-structured TS11 handler for mapped monitors
MT.SYS	File-structured TM11 handler for unmapped monitors
MTX.SYS	File-structured TM11 handler for mapped monitors
MU.SYS	File-structured TMSCP handler for unmapped monitors
MUX.SYS	File-structured TMSCP handler for mapped monitors
NCX.SYS	Ethernet handler for Professional 300 series processors
NL.SYS	Null handler for unmapped monitors
NLX.SYS	Null handler for mapped monitors
NQX.SYS	Ethernet handler for Q-Bus processors
NUX.SYS	Ethernet handler for UNIBUS processors
PI.SYS	Professional 300 series processor interface system support handler for unmapped monitors
PIX.SYS	Professional 300 series processor interface system support handler for mapped monitors
RK.SYS	RK11/RK05 handler for unmapped monitors
RKX.SYS	RK11/RK05 handler for mapped monitors
SP.SYS	Transparent print spooler handler for unmapped monitors
SPX.SYS	Transparent print spooler handler for mapped monitors
UBX.SYS	UNIBUS mapping register system support handler for mapped monitors

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

Type of Software	Description
VM.SYS	Virtual disk handler for unmapped monitors
VMX.SYS	Virtual disk handler for mapped monitors
XC.SYS	Professional 300 series processor communication port modem handler for unmapped monitors
XCX.SYS	Professional 300 series processor communication port modem handler for mapped monitors
XL.SYS	DL–11 communication port modem handler for unmapped monitors
XLX.SYS	DL–11 communication port modem handler for mapped monitors
Other System Files	
SL.SYS	Single-line editor pseudohandler for unmapped monitors
SLMIN.SYS	Minimal single-line editor pseudohandler
SLX.SYS	Single-line editor pseudohandler for mapped monitors
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
DUP.SAV	Disk utility
EDIT.SAV	Text editor
FILEX.SAV	Foreign file exchange utility
FORMAT.SAV	Disk formatting utility
GIDIS.SAV	Graphics utility for Professional 300 series processors
HELP.SAV	Help utility
IND.SAV	Indirect control file processor
INDEX.SAV	On-line index utility
INDEXX.SAV	Virtual memory on-line index utility
KED.SAV	Keypad editor
KEX.SAV	Virtual memory keypad editor
LIBR.SAV	Librarian

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

Type of Software	Description
LINK.SAV	Linker
MACRO.SAV	MACRO assembler
MDUP.AI	Magtape bootstrap utility for TMSCP auto installation
MDUP.MM	Magtape bootstrap utility for TJU16
MDUP.MS	Magtape bootstrap utility for TS11
MDUP.MT	Magtape bootstrap utility for TM11
MDUP.MU	Magtape bootstrap utility for TMSCP
MDUP.SAV	Magtape utility
MONMRG.SAV	System generation monitor linking program
MSCPCK.SAV	Q-bus revision level checking program for MSCP and TMSCP devices
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
SETUP.SAV	VT100 and Professional 300 series processor terminal setup utility
SIPP.SAV	Save image patch utility
SLP.SAV	Source language patch utility
SPOOL.REL	Transparent print spooler utility
SPOOL.SAV	Virtual memory tranparent print spooler utility
SRCCOM.SAV	Source compare utility
TRANSF.SAV	RT–11 host file transfer program for VTCOM
UCL.SAV	User command linkage executable image
VTCOM.REL	Virtual terminal communication and file transfer utility
VTCOM.SAV	Virtual version of VTCOM communication utility, for running under mapped monitors
Miscellaneous	
BATCH.SAV	BATCH processor
ERROUT.SAV	Error log display program
INDEX.IDX	Index pointer file for INDEX utility

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

Type of Software	Description
INDEXA.IMG	First half of INDEX utility index file
INDEXB.IMG	Second half of INDEX utility index file
MSB.COM	MS type magtape distribution build file
MTB.COM	MM/MT type magtape distribution build file
MUB.COM	MU type magtape distribution build file
RTFB.MAP	Distributed FB monitor link map
RTSB.MAP	Distributed SB monitor link map
RTXB.MAP	Distributed XB monitor link map
RTXM.MAP	Distributed XM monitor link map
RTZB.MAP	Distributed ZB monitor link map
RTZM.MAP	Distributed ZM monitor link map
STRTFB.COM	FB start-up command file
STRTSB.COM	SB start-up command file
STRTXB.COM	XB start-up command file
STRTXM.COM	XM start-up command file
STRTZB.COM	ZB start-up command file
STRTZM.COM	ZM start-up command file
V5USER.TXT	Distribution kit message text file
System Generation Files	
SYSGEN.COM	SYSGEN command file
SBFB.ANS	Answer file to build distributed unmapped monitors
XB.ANS	Answer file to build distributed XB monitor
XM.ANS	Answer file to build distributed XM monitor
XMEL.ANS	Answer file to build distributed XM monitor with MSCP error logging support
ZB.ANS	Answer file to build distributed ZB monitor
ZM.ANS	Answer file to build distributed ZM monitor
Automatic Installation Files	
IVP.MAC	Automatic installation verification source file
STRTAI.COM	Start-up file for automatic installation
TERMID.SAV	Console terminal identification program

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

Type of Software	Description
VERIFY.COM	Automatic installation verification command file
Debuggers	
DBGSYM.SAV	DBG–11 symbol table utility
ODT.OBJ	Debugging aid
SDH.SYS	DBG–11 unmapped hardware I/O pseudohandler
SDHX.SYS	DBG–11 mapped hardware I/O pseudohandler
SDS.SYS	DBG–11 unmapped software I/O pseudohandler
SDSX.SYS	DBG–11 mapped software I/O pseudohandler
VDT.OBJ	Debugging aid for virtual and multiterminal jobs
Graphics Software	
ALPH00.FNT	PRO GIDIS font table
Libraries and Subroutines	
EDTL.MLB	Common monitor definitions
SYSLIB.OBJ	System-callable subroutines
SYSMAC.SML	System macro library
SYSTEM.MLB	System-callable data structure macros
Bootstraps	
MBOOT.BOT	Magtape primary bootstrap for TMSCP and 800 bits/in tape
MBOT16.BOT	Magtape primary bootstrap for 1600 bits/in tape
MSBOOT.BOT	Magtape secondary bootstrap
Demonstration Programs	
DEMOB1.BAS	Demonstration source—BASIC–PLUS program
DEMOBG.MAC	Demonstration source—SB/FB macro program
DEMOFG.MAC	Demonstration source—FB macro program
DEMOF1.FOR	Demonstration source—FORTRAN program
DEMOF2.FOR	Demonstration source—FORTRAN program
DEMOF3.FOR	Demonstration source—FORTRAN program
DEMOKD.TXT	Keypad editor demonstration source
DEMOM1.MAC	Debugging demonstration source

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

Type of Software	Description
Source Files	
BA.MAC	BATCH handler source file for system generation
BSTRAP.MAC	Bootstrap 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
DU.MAC	MSCP device handler source file for system generation
DW.MAC	Professional 300 series processor disk handler source file for system generation
DX.MAC	RX01 handler source file for system generation
DY.MAC	RX02 handler source file for system generation
DZ.MAC	Professional 300 series processor diskette handler source file for system generation
EDTG.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 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
LP.MAC	Parallel interface printer handler source file for system generation
LS.MAC	Serial interface 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
NC.MAC	Ethernet port handler source file for system generation for Professional 300 series processors (DECNA controller)
NI.MAC	Ethernet class handler source file for system generation

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

Type of Software	Description
NL.MAC	Null handler source file for system generation
NQ.MAC	Ethernet port handler source code file for system generation for Q-bus processors (DEQNA and DELQA controllers)
NU.MAC	Ethernet port handler source code file for system generation for UNIBUS processors (DEUNA and DELUA controllers)
RK.MAC	RK05 handler source file for system generation
RMON.MAC	Resident monitor source file for system generation
SP.MAC	Transparent print spooler handler 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
TU.MAC	TMSCP handler source file for system generation
UM.MAC	MSCP (and TMSCP) handler source file for system generation
USR.MAC	USR source file for system generation
VM.MAC	VM.SYS source file for system generation
XB.MAC	XB conditional file for system generation
XC.MAC	Professional 300 series processor communication port modem handler source file for system generation
XL.MAC	DL–11 communication port modem handler source file for system generation
XM.MAC	XM conditional source file for system generation
XMSUBS.MAC	Mapped monitor subroutines for system generation
ZB.MAC	ZB conditional file for system generation
ZM.MAC	ZM conditional file for system generation
Unsupported Software	
CONFIG.SAV	Automatic system software configuration services program
CONFIG.COM	Automatic system software configuration command file
CONSOL.MAC	Change boot-time console terminal
CUSTOM.TXT	Customization symbol location file
DATIME.MAC	Date and time source file for system generation

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

Type of Software	Description
DATIME.SAV	Date and time utility
NITEST.MAC	Ethernet test source file
RTMON.REL	System activity monitor
SPLIT.SAV	File split utility
TRANSF.EXE	VAX/VMS host file transfer program for VTCOM
TRANSF.TSK	RSX host file transfer program for VTCOM
V5NOTE.TXT	RT–11 release notes addenda
VBGEXE.SAV	Virtual RUN utility

2.4 Selecting Components for Your Working System

How you select components for your working system is largely determined by whether you have a larger (disk) or a smaller (diskette) system device.

2.4.1 Components for Larger (Disk) System Devices

If you have a disk system device, you can store all usable system files on one device and the single system device is your working system. Any of the disk devices contains enough storage capacity for all appropriate system files. RT–11 distributes a variety of system files and all are not needed on any one computer system. So, RT–11 provides a configuration program, CONFIG.COM, you can run to automatically delete all unneeded system files. Further, although certain system files could be used on your computer system, if you know you will not use them, you can manually delete them after running CONFIG. The CONFIG program is included in the automatic installation procedure. If you are installing your software distribution manually, after completing the installation, issue the following command and follow the prompts:

```
.$@CONFIG RET
```

Then examine your directory:

```
.DIRECTORY/FULL SY: RET
```

If ample space remains unused on your system volume, you are done. You have created your working system.

If the remaining storage space on your system device is limited, use the information in Table 2–1 to determine which files can be deleted manually. Delete those files. You have then created your working system.

2.4.2 Components for Smaller (Diskette) System Devices

If you have a diskette system device, you must be selective and perhaps only place system files for a particular application on each of several working systems; one for each application. To decide which components these should be, study the space limitations of your device, then consider the suggestions given for each software component (Sections 2.4.2.1 through 2.4.2.13).

Before selecting components, you should become familiar with your medium's space limitations. (See Table 2–2.) 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
RX50 Diskette	DU,DZ	786
Virtual Memory	VM	Variable, depending on physical memory available

Keep in mind the number of blocks 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. 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). Use the DIRECTORY command to examine a directory. The following is a sample directory.

```
 15-Mar-90
SWAP  .SYS    28P 16-Jan-90      RT11XM.SYS   123P 16-Jan-90
VMX   .SYS     3  14-Apr-89      PIX   .SYS    68P 16-Jan-90
DWX   .SYS     6P 16-Jan-90      DZX   .SYS     4P 16-Jan-90
LSX   .SYS     5P 16-Jan-90      SPX   .SYS    11P 16-Jan-90
XCX   .SYS     4P 16-Jan-90      LDX   .SYS    11P 16-Jan-90
SLX   .SYS    20P 16-Jan-90      DUP   .SAV    52P 16-Jan-90
DIR   .SAV    20P 16-Jan-90      PIP   .SAV    30P 16-Jan-90
KEX   .SAV    72P 16-Jan-90      VTCOM .SAV    24P 16-Jan-90
SPOOL .SAV    22P 16-Jan-90      DUMP  .SAV    10P 16-Jan-90
SETUP .SAV    42P 16-Jan-90      RESORC.SAV    33P 16-Jan-90
UCL   .SAV    16P 16-Jan-90      UCL   .DAT     9  09-Jan-89
VBGEXE.SAV    16P 16-Jan-90      SRCCOM.SAV    26P 16-Jan-90
STARTX.COM     2  13-Feb-90
 25 Files, 657 Blocks
 634 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 to help you make your selections. The following elements are described:

- Monitor
- SWAP.SYS File
- System Device Handler
- Other Device Handlers
- Default System Library
- Help Package
- Printer Handlers
- MACRO–11 Assembler
- Queue Package
- Source Files
- Start-Up Command File
- Text Editors
- Utility Programs

2.4.2.1 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.

2.4.2.2 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 to bootstrap your system successfully.

2.4.2.3 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.

2.4.2.4 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.

2.4.2.5 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. See the *RT–11 System Utilities Manual* for more information on LINK and overlays.

2.4.2.6 Help Package

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.

A customization patch describes how to tailor the help text to your specific needs.

2.4.2.7 Printer Handlers

The software kit includes the printer handlers for a parallel interface, LP.SYS (or LPX.SYS), and a serial interface, LS.SYS (or LSX.SYS). If your hardware configuration includes a serial interface printer instead of a parallel interface printer, you should include only LS.SYS or LSX.SYS in your working system. Further, if you are going to use the serial interface printer, you should include the following command line in your start-up command file:

```
ASSIGN LS LP
```

If your serial interface printer is installed at nonstandard vector and control status register (CSR) addresses, use the SET command to change the addresses. See the *RT–11 Commands Manual* for all LS handler SET commands.

2.4.2.8 MACRO–11 Assembler

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 *Introduction to RT–11* and the *RT–11 System Utilities Manual* for descriptions of the assembly process. If you are going to be assembling any system components, you should also have SYSTEM.MLB on your system device, as SYSTEM.MLB is a library of system data structures with the supported data structure names and elements. You should also have SYSTEM.MLB on your system device if you are going to call for any system data structures in your own programs. You will then be using the same label names as used by RT–11.

2.4.2.9 Printer Utilities

RT–11 contains two utilities, QUEUE and SPOOL, that are used to send output to the printer. Both utilities can be used as foreground or system jobs.

QUEUE is perhaps more appropriate for use on a small system as it does not use a permanent intermediate file, but rather a small temporary work file. SPOOL is very useful on larger systems, as output can be directed in multiple streams. See the *RT–11 System Utilities Manual* for a description of QUEUE and SPOOL.

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

2.4.2.11 Start-Up Command File

Each RT–11 monitor looks for a matching start-up command file (STRTxx.COM) whenever you bootstrap the system. If it finds one, the monitor executes the commands in the file (to print a message, assign devices, and so on). If a monitor does not find a start-up command file, it prints a message indicating that the start-up file has not been found. See the *Introduction to RT–11* for information on start-up command files.

2.4.2.12 Text Editors

RT–11 gives you a choice of two text editors. The Keypad Editor (KED or KEX) is an easy-to-use, cursor-oriented editor that you can use only on a VT100-compatible video terminal. KEX is the virtual version of KED; they function identically. KED is the default editor for unmapped monitors, and KEX is the default editor for the mapped monitors. The EDIT command automatically calls the correct version of the keypad editor for the monitor you are using, assuming the editor resides on your system device.

The EDIT utility is a character-oriented text editor that you use with a hard-copy terminal.

Refer to the *Introduction to RT–11* for an introduction to the keypad editor and the *PDP–11 Keypad Editor User's Guide* for complete information. Refer to the *RT–11 Commands Manual* for information on the EDIT utility.

2.4.2.13 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 to work. For example, PIP, DUP, and DIR are necessary for most of the keyboard commands to function. Appendix A of the *RT–11 Commands Manual* 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 utility program functions. If your system device is a large disk, you will 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.

You must include PIP, DUP, and DIR, and you probably want at least the correct version of the keypad editor and the HELP utility on your system volume. The utilities you use less frequently can occupy a second volume, as described in Section 2.5.2.

2.5 Planning the Arrangement of Components for Smaller System Devices

If your system volume is one of the smaller devices, the following is especially important. Consider the following suggestions before you build a working system; they can help you create an efficient system:

- Assign the default device to the data device.
- Create a separate utilities volume.
- Create several system volumes.
- Create volumes for use with *Introduction to RT–11*.

More details on each of these suggestions follow.

2.5.1 Assigning the Default Device to the Data Device

Once you have installed the system, you can assign the default device DK to the nonsystem device unit. 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 a nonsystem device is as follows, where *dev* is the nonsystem device unit:

```
.ASSIGN dev: DK: RET
```

Include this command in your start-up command file to assign the default device to *dev* whenever you boot the system. The procedures in this manual assume DK has not been assigned and is also the system device unless indicated otherwise. (See the *Introduction to RT–11*.)

2.5.2 Creating a Separate Utilities Volume

Create a separate volume for the utility programs you expect to use infrequently. This technique provides 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 the nonsystem device unit and copy the utility temporarily to the system volume. The utility can be run from the system volume by using appropriate DCL commands.

NOTE

The DIR, PIP, and DUP utilities should always reside on the system volume.

If a utility contains no overlays, you can run the utility directly from the nonsystem device unit which reads the entire utility into memory. You can then reinsert the data volume in that device unit, keeping the utility in memory. An overlaid utility must be copied to the system volume before the data volume is reinserted. You can use the SIPP utility to determine if a utility (or any program) is overlaid. If SIPP asks for a segment number, the program is overlaid. For example, the following illustrates an overlaid *prog.SAV*:

```
.RUN SIPP RET
*prog.SAV RET
Segment?
```

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

```
.RUN ddn:aaaaaa RET
*
```

Then, issue the appropriate utility (not DCL) commands. When you are done with the utility, exit from the program:

```
* CTRL/C
.
```

It is easier to simply copy a utility to the system device unit and run it from there. Not only do you not need to be concerned if a utility is overlaid, but you can use the simpler DCL command language (assuming the utility supports DCL commands, and most do). Try to keep some unused space on the system volume for temporarily storing and running seldom-used utilities.

2.5.3 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. When you have reached a logical stopping point, exchange the system volumes and reboot the processor. The chapter in the *Introduction to RT–11* that describes using VM as the system device contains examples of several system volumes.

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

To perform the exercises in *Introduction to RT–11*, you need certain components on your working system. Because your system device is one of the small devices (RX01, RX02, or RX50), you need to build one working system for all exercises other than the programming part, and one working system for each programming chapter you intend to complete.

2.5.4.1 Working System for Nonprogramming Part

For all exercises other than the programming part, place at least the following files on your working system:

SWAP.SYS
RT11FB.SYS
xx.SYS (system device handler)
LP.SYS or LS.SYS (if appropriate)
PIP.SAV
DUP.SAV
DIR.SAV
BUP.SAV
KED.SAV
SRCCOM.SAV
RESORC.SAV
DEMOF1.FOR
DEMOF2.FOR

2.5.4.2 Working System for Programming Part

The following files are required to build any working system:

SWAP.SYS
RT11FB.SYS (or a different monitor)
xx.SYS (system device handler)
DIR.SAV
DUP.SAV
PIP.SAV

The files required for each programming chapter are listed in the *Choosing a Programming Language* chapter. You will need the file, LIBR.SAV, to perform the exercises in *Constructing Library Files*. Additionally, you should place on the system volume the appropriate printer handler (if you have a printer) and an editor (KED or KEX). The demonstration programs are placed on the nonsystem (DK) device volume.

2.6 Choosing Software Customizations

In general, you use software customizations to modify an already supported feature. Also in general, you use the system generation procedure to add or remove support for a system feature.

Examine Table 1–2 and read the following sections to determine what you need to do. Identify any non-system-generation customizations you need to make. 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.

Use the SIPP utility to install software customizations. The *RT–11 System Utilities Manual* contains information on the SIPP utility; however, each customization patch contains the complete procedure.

To customize your software:

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 RTSB.MAP, RTFB.MAP, RTXB.MAP, RTXB.MAP, RTXM.MAP, RTZB.MAP, and RTZM.MAP. The values of symbols for utility customizations are 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, question marks (?) represent unknown characters. These characters vary according to the specific software component. All numeric input values are octal unless otherwise indicated.

2.6.1 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. You can modify SRCCOM to use characters other than the vertical bar and bullet characters,

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    ??????       ;A  RET
      000000   001000       <o>       ;Am RET
      000000   001001       <|>       ;An  RET
      000000   001002      <^C>       CTRL/Y   RET
* CTRL/C
.
```

2.6.2 Changing Default Output Device from Printer to Terminal

If your configuration does not have a 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.

2.6.3 Changing the LD Handler Suffix Character

Use the following customization patch to change the suffix character for the LD handler to anything other than X. (You can use the X suffix without performing a customization patch.) You must also use customization patch D.3, Changing the Handler File-Name Suffix, located in the *RT–11 System Generation Guide*. The following customization makes known to the monitor bootstrap the suffix character you specified in that customization; you must specify the same suffix character in both.

In this customization, *monitor.SYS* is the name of the monitor file you want to modify, and ..BLDS is the value of that symbol from the monitor link map. The field *y* represents the suffix character you want to add to the LD handler name.

```
.RUN SIPP RET
*monitor.SYS RET
Base?    0  RET
Offset?  ..BLDS  RET

     Base     Offset        Old       New?
      000000       ..BLDS       ??????       ;Ay  RET
      000000       ..BLDS+2     ??????       CTRL/Y   RET
* CTRL/C
.
```

2.6.4 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    ??????       ffffff  RET
      000000   001002    ??????       nnnnnn  RET
      000000   001004    ??????       CTRL/Y  RET
* CTRL/C
.
```

2.6.5 Changing the Default Order of Directory Listings

You can change the default order in which 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 SWS that SIPP returns in response to the search command. The variable *yyyyyy* represents the contents of nnnnnn+2. 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+4 RET
      000000   nnnnnn+2  yyyyyy       CTRL/Z  RET
Offset?   SWS RET

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

2.6.6 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_8; 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
.
```

2.6.7 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_8 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_8 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
.
```

2.6.8 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_{10} 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     ??????       nnn RET
      000000   QSIZE+2   ??????       CTRL/Z  RET

Offset?  QCBLK RET

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

2.6.9 Modifying EDIT for 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_8 or 176_8, rather than the standard 33_8, 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 Commands Manual*. In the change, ALTMDE is the value of that symbol from the CUSTOM.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    ??????        \ RET

     Base     Offset        Old        New?
      000000   ALTMDE       ???        nnn RET
      000000   ALTMDE+1     ???        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.

2.6.10 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 Commands Manual* for a list of available devices.) You can perform the system generation process to create monitors and handlers that support nonstandard devices. The *RT–11 Device Handlers 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 System Internals Manual* for a description of these priorities.

The SHOW command lists all installed device handlers and specifies when any empty device slots are available.

```
.SHOW RET
DU  (Resident)
    DU0 = SY , DK
LD
DX
SL
UB
LS
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_{10} device slots. If your application requires more than 16 device slots simultaneously, you must perform a system generation 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 the 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 unmapped monitors and *yyX.SYS* for mapped 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 parallel and serial interface printers, RX01 and RX02 diskettes, and MSCP disks, and how you can install hardware magtape support, and set magtape parity and density.

2.6.10.1 CSR and Vector Addresses for Printers, RX01/02 Diskettes and MSCP Disks

With the SET command, you can change the CSR and vector addresses of five devices: parallel and serial interface printers, 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, 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
```

Use the following command to assign RT–11 unit numbers to MSCP disks, where *n* is the RT–11 unit number and *x* is the MSCP unit number:

```
.SET DUn: UNIT=x RET
```

To partition MSCP disks, use the following command, where *n* is the RT–11 unit number and *x* is the partition number:

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

You can also assign unit numbers to the MSCP ports. To generate support for the following command, you must perform a system generation and respond to question 108 (How many ports are to be supported (1) ?) with 2, 3, or 4. The distributed monitors do not support this command. In the command, *n* is the RT–11 unit number and *x* is the port number:

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

See the description of the SET command in the *RT–11 Commands Manual* and the *RT–11 Device Handlers 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.

2.6.10.2 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. System generation produces 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, MSX.SYS, MU.SYS, and MUX.SYS are distributed file-structured handlers for TM11, TJU16, TS11, and TMSCP magtape devices. The files MTHD.SYS, MTHDX.SYS, MMHD.SYS, MMHDX.SYS, MSHD.SYS, MSHDX.SYS, MUHD.SYS, and MUHDX.SYS are the system-generated hardware magtape handlers for the same devices. There are two sets of handlers: one for unmapped monitors, the other for mapped monitors. The file-name suffix X indicates device support for mapped monitors.

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

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 TS11 hardware support in place of TS11 file-structured support, first make sure that the correct (unmapped or mapped) version of the hardware magtape handler (MSHD.SYS or MSHDX.SYS) is on the system volume, and then do the following. In the example, the mapped version of the handler is used:

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

   ```
   .RENAME MS.SYS MS.FIL RET
   .REMOVE MS: RET
   .
   ```

2. Rename and install the hardware magtape handler.

   ```
   .RENAME MSHD.SYS MS.SYS RET
   .INSTALL MS: RET
   .
   ```

The handler you have named MS.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 unmapped monitor. Make sure you save the distributed (file-structured) handler.

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

TM11	MT.SYS	MTHD.SYS	MT.SYS
TJU16	MM.SYS	MMHD.SYS	MM.SYS
TS11	MS.SYS	MSHD.SYS	MS.SYS
TMSCP	MU.SYS	MUHD.SYS	MU.SYS

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

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

TM11	MTX.SYS	MTHDX.SYS	MTX.SYS
TJU16	MMX.SYS	MMHDX.SYS	MMX.SYS
TS11	MSX.SYS	MSHDX.SYS	MSX.SYS
TMSCP	MUX.SYS	MUHDX.SYS	MUX.SYS

2.6.10.3 Magtape Parity and Density

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 Commands Manual*. 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.

2.6.11 Modifying BATCH to Save Space

BATCH support requires system generation. Once you have generated BATCH support, 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. In the patch, *nnnnnn* is the octal value for the symbol that corresponds to the program you want to affect. The values for the symbols can be found in the file CUSTOM.TXT on your distribution kit.

Program	**Symbol**
DIR	..RDIR
MACRO	..RMAC
FORTRAN	..RFOR
LINK	..RLIN
PIP	..RPIP
BASIC	..RBAS

```
.RUN SIPP  RET
*BATCH.SAV  RET
Base?    0  RET
Offset?  nnnnnn  RET

  Base       Offset      Old       New?
   BASE       nnnnnn  ??????       \ RET

  Base       Offset      Old       New?
   BASE       nnnnnn     ???       125  RET
   BASE       nnnnnn+1   ???       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
```

2.6.12 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
.
```

2.6.13 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 files HELP.TXT and HELP.EXE, which together make up the program HELP.SAV, are not provided on the distribution kit. Therefore, to change your HELP text, you must first recreate HELP.TXT and HELP.EXE from HELP.SAV using the unsupported utility SPLIT.

To recreate these files, type this command:

```
.SPLIT ddn:HELP.EXE,,ddn:HELP.TXT=ddn:HELP.SAV/B:..HLP1:..HLP2 RET
```

In the command, *ddn:* represents the device on which to create the files HELP.TXT and HELP.EXE, or the device on which HELP.SAV exists. The variables ..HLP1 and ..HLP2 represent the boundaries along which to split HELP.SAV. Refer to the file CUSTOM.TXT on your distribution kit for the values to substitute in the command line for ..HLP1 and ..HLP2.

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 use 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 in 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/CONCATENATE HELP.EXE,HELP.MLB 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) 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/CONCATENATE HELP.EXE,HELP.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.

2.6.14 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 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     ??????       240  RET
      000000   FATAL+2   ??????       CTRL/Y  RET
*  CTRL/C
.
```

2.6.15 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 SB and FB monitors have bootstraps that allow the system to run in less memory than is available. (This cannot be done for mapped monitors, because the mapped monitors require 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 unmapped monitor file that you want to modify, and BHALT is the value of that symbol from the monitor link map.

```
.RUN SIPP RET
*monitr.SYS RET
Base?     0 RET
Offset?   BHALT RET

     Base     Offset        Old       New?
      000000   BHALT     ??????       0 RET
      000000   BHALT+2   ??????       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
104000	17K
110000	18K
114000	19K
120000	20K
124000	21K

Value	Size in Words
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?     0 RET
Offset?   BHALT RET

     Base     Offset        Old        New?
      000000   BHALT     ??????        240 RET
      000000   BHALT     ??????      12704 RET
      000000   BHALT     ??????     nnnnnn RET
      000000   BHALT     ??????     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 2.6.20.

2.6.16 Setting VTCOM Default Dial String

Apply one of the following customizations to VTCOM.REL or VTCOM.SAV to set a default dial string for the DIAL command.

In the customization, the symbol ..DIAL represents the address of the first character in the dial string. You can find the correct value for the symbol ..DIAL in the file CUSTOM.TXT on your distribution kit. Replace the symbols *a, b,* and so on, with the ASCII characters that make up your dial string. Add a mandatory null byte (a zero) following the last character of your dial string. The dial string cannot be longer than 39 characters. The fortieth byte must be a null.

To modify VTCOM.REL:

```
.R SIPP RET
*VTCOM.REL RET
Base?     0 RET
Offset?   ..DIAL RET
```

```
     Base     Offset        Old        New?
      000000   ..DIAL    ??????        ;Aa  RET
      000000   ..DIAL+1     ???        ;Ab  RET
      000000   ..DIAL+2     ???        ;Ac  RET
       .          .           .         .
       .          .           .         .
       .          .           .         .
      000000   ..DIAL+47    ???         0  RET
      000000   ..DIAL+50    ???         CTRL/Y  RET
*  CTRL/C
.
```

To modify VTCOM.SAV:

```
.R SIPP  RET
*VTCOM.SAV  RET
Segment?  1  RET
Base?     0  RET
Offset?   ..DIAL  RET

Segment    Base     Offset        Old       New?
000001      000000   ..DIAL    ??????       ;Aa  RET
000001      000000   ..DIAL+1     ???       ;Ab  RET
 .             .        .          .         .
 .             .        .          .         .
 .             .        .          .         .
000001      000000   ..DIAL+47    ???       0  RET
000001      000000   ..DIAL+50    ???        CTRL/Y  RET
*  CTRL/C
.
```

2.6.17 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    ??????        nnnnnn  RET
      $RMON    000316    ??????         CTRL/Y  RET
*  CTRL/C
.
```

2.6.18 Changing the Number of Bad Blocks That BUP Accepts

By default, BUP successfully handles up to 25 bad blocks on the input device when backing up a disk to a tape or when restoring from a tape to a disk (copy back). If any bad blocks are encountered on the output device, BUP issues a fatal error message and stops the backup operation.

BUP issues a warning message each time it encounters a bad block on the input device, then continues to back up or restore. If BUP encounters more than 25 bad blocks on the input device, BUP issues a fatal error message and the operation is stopped.

Use the following customization to set your own value for the number of bad blocks on the input device that BUP will accept.

In the customization, ..MBAD is the symbol for the value found in the file CUSTOM.TXT, and *nnnnn* is the new value that you enter. The maximum value you can enter is 77777_8; that is, you can customize BUP to read 77777_8 bad blocks on the input device before the next bad block encountered causes a fatal error. The value you enter is also the number of verification mismatches that BUP reads on the input device before the subsequent mismatch causes a fatal error.

```
.RUN SIPP RET
*BUP.SAV RET
Segment? 0 RET
Base?    0 RET
Offset?  ..MBAD RET

     Base     Offset        Old       New?
      000000   ..MBAD        ??       nnnnn RET
      000000   ..MBAD+2    ????       CTRL/Y  RET
* CTRL/C
.
```

2.6.19 Changing the Default File Type Extension of IND Control Files

As distributed, the file type extension for IND control files is .COM. You can change that extension to distinguish IND control files from KMON command files (which also use the extension .COM).

In the customization, ..INDF is the symbol for a value located in the file CUSTOM.TXT, and *nnn* is the new default RAD50 3-character file type extension.

```
.RUN SIPP RET
*IND.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..INDF RET

Segment     Base     Offset       Old       New?
000000       000000   ..INDF    ??????      ;R RET
000000       000000   ..INDF    ??????      ;Rnnn CTRL/Y  RET
* CTRL/C
.
```

2.6.20 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 a mapped monitor.

In the customization, *monitr.SYS* is the name of the monitor file you want to modify (SB or FB), and ..28KW is the symbol for the value found in the monitor link map on your software distribution kit.

```
.RUN SIPP RET
*monitr.SYS/A RET
Base?    RET
Offset? ..28KW RET

     Base     Offset        Old       New?
      000000   ..28KW    ??????       160000 RET
      000000   ..28KW+2  ??????       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.

2.6.21 Suppressing the Bootstrap Message

You can modify a monitor to prevent the monitor identification message from printing when you bootstrap 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    ??????       1 RET
      000000   ..SLNT+2  ??????       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.

2.6.22 Suppressing the Start-Up Indirect Command File

You can suppress the monitor from calling a start-up command file when you bootstrap that monitor. The standard monitors include start-up 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    ??????       0 RET
      000000   ..NIND+2  ??????       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.

2.6.23 Suppressing the Start-Up Indirect Command File Echo

You can modify a monitor to execute the start-up indirect command file when you bootstrap a monitor but not echo (display) the command lines in the file. 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    ??????       1 RET
      000000   ..TTQU+2  ??????       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.

2.6.24 Changing the Bootstrap Message

You can modify a monitor to change the monitor identification message that appears when you bootstrap that monitor. 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_{10} 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.

2.6.25 Changing the Default Device for Indirect Command Files

You can change the default device for indirect command files. 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    ??????     ;R RET
      000000   ..ATDK     <DK >     ;Rnnn RET
      000000   ..ATDK+2   ?????     CTRL/Y  RET
* CTRL/C
.
```

2.6.26 Changing the Default File Type for Indirect Command Files

You can change the default file type for indirect command files. 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    ??????     ;R RET
      000000   ..ATFX     <COM>     ;Rnnn RET
      000000   ..ATFX+2   <???>     CTRL/Y  RET
* CTRL/C
.
```

2.6.27 Changing the Default Device for the FRUN Command

You can change the default device for the FRUN command. Normally, when you start a foreground program (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    ??????        ;R  RET
      000000   ..FRDK     <DK >        ;Rnnn  RET
      000000   ..FRDK+2   <???>        CTRL/Y  RET
*  CTRL/C
.
```

2.6.28 Changing the Default File Type for the FRUN Command

You can change the default file type for foreground programs from .REL (the default) 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    ??????        R  RET
      000000   ..FRUX     <REL>        ;Rnnn  RET
      000000   ..FRUX+2   ?????        CTRL/Y  RET
*  CTRL/C
.
```

2.6.29 Changing the Default Device for the EDIT Utility Command

The customization changes the default device for the EDIT hard-copy editor only; it does not change the default device for the KED or KEX keypad editors. Further, as EDIT.SAV is not the default editor under any monitor, you must include the command SET EDIT EDIT in your start-up command file or perform the customization patch in Section 2.6.30 before this patch has effect.

Assuming you have made EDIT the default editor, if you then 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 SY. 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, ..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    ??????        ;R  RET
      000000   ..EDDV     <SY >        ;Rnnn  RET
      000000   ..EDDV+2   ?????        CTRL/Y  RET
*  CTRL/C
.
```

2.6.30 Changing the Default File Name for the EDIT Command

You can change the default editor that is run when you type the EDIT command.

In the customization, *monitr.SYS* is the name of the monitor file that you want to modify, PROGDF is the value of that symbol from the monitor link map, and EEE is 200_8 plus the new byte value of one of the following symbols from the monitor link map:

$$EDIT	Command value for default editor EDIT
$$KED	Command value for default editor KED
$$KEX	Command value for default editor KEX

```
.RUN SIPP RET
*monitr.SYS RET
Base?     0 RET
Offset?   PROGDF RET

     Base     Offset        Old        New?
      000000   PROGDF    ??????        \ RET

     Base     Offset        Old        New?
      000000   PROGDF       ???        EEE RET
      000000   PROGDF+1     ???        CTRL/Y  RET
* CTRL/C
.
```

2.6.31 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    ??????        240 RET
      000000   ..EMON+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

2.6.32 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 interface 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    ??????       ;A RET
      000000   ..QULP       <L>       ;Ad RET
      000000   ..QULP+1     <P>       ;Av RET
      000000   ..QULP+2     <:>       CTRL/Y  RET
* CTRL/C
.
```

2.6.33 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       ???       nnn RET
      $RMON    000400       ???       CTRL/Y  RET
* CTRL/C
.
```

2.6.34 Changing the Threshold for Resuming Output-Stalled Jobs

You can improve system throughput by changing the threshold for resuming output-stalled jobs. In an RT–11 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 while 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_8, is the size of the ring; consequently, the monitor resumes a job when 47_8 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). Note that a value of 1 will cause the job to stay stalled until its ring buffer is empty.

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

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

2.6.35 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 Commands Manual*.) 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             ;get 1 segment directories
.WORD     4000          ;Volumes with <= 2048. blocks
.WORD     4             ;get 4 segment directories
.WORD     30000         ;Volumes with <= 12288. blocks
.WORD     20            ;get 16. segments
.WORD     177777        ;Volumes with <= 65535. blocks
.WORD     37            ;get 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 *mmmm*.

Size of Volume in 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    <??????>      mmmmmm RET
000010      SEGTBL    nnnnnn+2  <??????>      CTRL/Y  RET
* CTRL/C
.
```

2.6.36 Changing the Banner Page Width

You can modify QUEUE to change the width of the banner page on 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        ???        \ RET
     Base     Offset        Old       New?
      000000   ..NB1        ???       15  RET
      000000   ..NB1+1      ???       12  RET
      000000   ..NB1+2      ???        0  RET
      000000   ..NB1+3      ???         CTRL/Z  RET
Offset?   ..NB2  RET
     Base     Offset        Old       New?
      000000   ..NB2        ???       \  RET
     Base     Offset        Old       New?
      000000   ..NB2        ???       15  RET
      000000   ..NB2+1      ???       12  RET
      000000   ..NB2+2      ???        CTRL/Z  RET
Offset?   ..NB3  RET
     Base     Offset        Old       New?
      000000   ..NB3     ??????       240  RET
      000000   ..NB2+2   ??????        CTRL/Y  RET
*  CTRL/C
.
```

2.6.37 Modifying Listing Page Length in LINK

If you do not use printer paper of a standard size (10.5 inches long) or if your configuration does not include a printer, you can 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 printers that use paper 10.5 inches long. 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
.
```

2.6.38 Assigning the HELP File

You can assign your HELP file to a file and/or device other than SY:HELP.SAV.

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    ??????       ;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  <???>       CTRL/Y  RET
*  CTRL/C
.
```

2.6.39 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 @filnam. To change the device from which the IND.SAV file is run, 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    ??????       ;R  RET
      000000   ..INDN     <SY >       ;Rdev  RET
      000000   ..INDN+2   <IND>       CTRL/Y  RET
*  CTRL/C
.
```

2.6.40 Supporting Bad Block Replacement in User-Written Handlers

If a user-written handler supports bad block replacement, you must apply one of the following customizations. 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     ???       377  RET
000001     000030   ARDPS+1   ???        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   ??????       \ RET

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

2.6.41 Supporting User-Written Magtape Handlers

If a user-written magtape handler is used, you must apply the following three customizations. All three customizations assume that the device code for the device is 377_8.

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   ??????     \ RET

Segment    Base     Offset      Old     New?
000001      000000   MTDPS      ???     377 RET
000001      000000   MTDPS+1    ???     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     ???     \ RET

Segment    Base     Offset     Old     New?
000001      000000   PIPMT     ???     377 RET
000001      000000   PIPMT+1   ???     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       ???       \ RET

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

2.6.42 Eliminating or Changing the IND Control File Terminating Message

You can eliminate or change the terminating message of IND control files, using the following customization patch.

The terminating message in the distributed IND is '@ <EOF>'. That message appears whenever IND exits and returns control. You can eliminate that message by specifying only a null character or change that message by specifying up to 7 characters plus the null character.

In the customization, ..INEO is the symbol for a value located in the file CUSTOM.TXT, and each *n* is an ASCII character in the new message. The required null character is an octal 0.

```
.RUN SIPP RET
*IND.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..INEO RET

Segment     Base     Offset        Old       New?
000000       000000   ..INEO    ??????       ;A RET
000000       000000   ..INEO       <@>       ;An RET
000000       000000   ..INEO+1    < >        ;An RET
000000       000000   ..INEO+2    <<>        ;An RET
000000       000000   ..INEO+3    <E>        ;An RET
000000       000000   ..INEO+4    <O>        ;An RET
000000       000000   ..INEO+5    <F>        ;An RET
000000       000000   ..INEO+6    <>>        ;An RET
000000       000000   ..INEO+7    ???        ;O RET

Segment     Base     Offset        Old       New?
000000       000000   ..INEO+6  ??????       \ RET

Segment     Base     Offset        Old       New?
000000       000000   ..INEO+6     ???  RET
000000       000000   ..INEO+7     ???       0 CTRL/Y RET
* CTRL/C
.
```

2.6.43 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 where *typ* is the desired default file type:

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

Then, reassemble and relink LD.SYS by using the conditional file SYSGEN.CND.

Customization 2

Do the following customization patch:

```
.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
.
```

2.6.44 Changing QUEUE to Allow First Form Feed

QUEUE suppresses the first form feed in a file, because QUEUE assumes that the LP or LS handler is set to FORM0, which generates a form feed. If the printer handler is set to NOFORM0, no form feed is generated.

You can apply the following customization so that QUEUE will never suppress the initial form feed in a file. If you apply this customization and set your printer handler to FORM0, an extra blank page will be produced.

In the customization, ..DOFF is the value of that symbol from the file CUSTOM.TXT on your distribution kit.

```
.RUN SIPP  RET
*QUEUE.REL  RET
Base? 0  RET
Offset?  ..DOFF  RET

     Base     Offset        Old       New?
      000000   ..DOFF    ??????       240  RET
      000000   ..DOFF+2  ??????       240  RET
      000000   ..DOFF+4  ??????       CTRL/Y  RET
*  CTRL/C
.
```

2.6.45 Changing Listing Page Length in MACRO and CREF

The default listing page size for MACRO and CREF, 60 lines long, is suitable for printers that use standard size printer paper (10.5 inches long). If you use printer paper that is not standard size, or if your configuration does not include a printer, you may need to modify the listing page length in MACRO and CREF. In the customizations, *nnn* represents the desired listing page length (octal). Substitute for the symbol PGSIZE the value of that symbol given in the file CUSTOM.TXT on your distribution kit.

To modify MACRO:

```
.RUN SIPP RET
*MACRO.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?  PGSIZE RET

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

To modify CREF:

```
.RUN SIPP RET
*CREF.SAV RET
Base?     0 RET
Offset?  PGSIZE RET

     Base     Offset        Old       New?
      000000  PGSIZE     ??????       nnn RET
      000000  PGSIZE+2   ??????       CTRL/Y  RET
* CTRL/C
.
```

2.6.46 Changing Default Command File Processor to IND

In the distributed monitors, the default command file processor is KMON. Apply the following customization to change the default command file processor to IND. After you have applied this customization, you may use an IND control file as your start-up command file, providing the file IND.SAV resides on your system volume.

In the customization, *monitr.SYS* is the monitor file you want to modify, and ..INDR is the value of that symbol from the appropriate monitor link map. If the monitor you customize is hardware bootable, write a new system bootstrap with the COPY /BOOT command.

```
.RUN SIPP RET
*monitr.sys RET
Base?     0 RET
Offset?    ..INDR RET
```

```
     Base     Offset        Old       New?
      000000  ..INDR     ??????       1 RET
      000000  ..INDR+2   ??????        CTRL/Y  RET
* CTRL/C
.
```

If you do not wish to apply this customization, you can set the default command file processor to IND by setting the conditional IND$ON to 1 in the conditional file.

2.6.47 Limiting Amount of Memory KEX Requests

KEX, by default, requests all available 16-bit memory up to 32K words. This customization allows you to limit the amount of memory KEX requests with a .SETTOP programmed request.

In the customization, the symbol ..MAXM represents the highest address KEX will request. (In the distributed monitors, this value is set to 177776, the highest address possible.) If you set this value too low, KEX will fail and print the error message *?KEX-F-Insufficient memory*. You can find the correct value for the symbol ..MAXM in the file CUSTOM.TXT on your distribution kit. Replace the symbol *nnnnnn* with the new maximum address value you want KEX to request.

```
.R SIPP RET
*KEX.SAV RET
Segment? RET
Base?    RET
Offset? ..MAXM RET

     Base     Offset        Old       New?
      000000   ..MAXM    ??????       ;nnnnnn RET
      000000   ..MAXM+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

2.6.48 Changing MACRO’s Default .LIST/.NLIST Options

MACRO–11 is distributed with the following .NLIST defaults:

.NLIST LD,ME,MEB,TTM

You may change the .NLIST defaults to any of the following, where the following bit significance applies (bit asserted implies ".NLIST"):

BEX =	2
BIN =	4
CND =	10
COM =	20
HEX =	40
LOC =	100
MC =	200
MD =	400
ME =	1000
MEB =	2000

SEQ =	4000
SRC =	10000
SYM =	20000
TOC =	40000
TTM =	100000

```
.RUN SIPP RET
*MACRO.SAV RET
Segment? RET
Base?     0 RET
Offset?    LCBITS RET

     Base     Offset        Old       New?
      000000   LCBITS    ??????       nnnnnn RET
      000000   LCBITS+2  ??????       CTRL/Y  RET
* CTRL/C
.
```

2.6.49 Changing MACRO's Default .ENABLE/.DISABLE Options

MACRO–11 is distributed with the following .DSABL defaults:

.DSABL ABS,AMA,CDR,DBG,FPT,LCM,LSB,MCL

You may change the .DSABL defaults to any of the following, where the following bit significance applies (bit asserted implies ".DSABL"):

ABS =	1
AMA =	2
CDR =	10
CRF =	20
DBG =	40
FPT =	100
GBL =	200
LC =	400
LCM =	1000
LSB =	2000
MCL =	4000
PNC =	10000
REG =	20000

```
.RUN SIPP RET
*MACRO.SAV RET
Segment? RET
Base?    0 RET
Offset?  EDBITS RET

     Base     Offset        Old       New?
      000000   EDBITS    ??????       nnnnnn RET
      000000   EDBITS+2  ??????       CTRL/Y  RET
* CTRL/C
.
```

2.6.50 Modifying KED Default File Type

KED supports default file types. When editing a file, the default input file type is .MAC; the default output file type is the same as the input file type. When inspecting a file, the default input file type is .LST. There is no default input file type when creating a file.

To specify a file with no file type, type only the file name and the period separating the file name and type (FILNAM.).

You can modify the default file types with the following software customization. In the customization, substitute KED or KEX for *aaa*. Substitute the value for the symbol ..EEXT (for the editing-file default) or ..IEXT (for the inspecting-file default) for *bbbbbb*. These values can be found in CUSTOM.TXT on your distribution kit (be sure to use the proper KED variant). Substitute the three-character file type default you want for *ccc*.

```
.R SIPP RET
*aaa.SAV/A RET
Base?   0 RET
Offset? bbbbbb RET

     Base     Offset        Old        New?
      000000   bbbbbb    ??????        ;R RET
      000000   bbbbbb     <???>        ;Rccc RET
      000000   bbbbbb     <???>        CTRL/Y  RET
* CTRL/C
.
```

2.6.51 Changing SPOOL's Work File Size

SPOOL allocates by default 1024_{10} blocks on logical device SFD or SY for its work file SPOOL.SYS. You can change the default size of SPOOL.SYS by applying the following software customization. In the customization, ..SPSZ is the offset for the current number of blocks SPOOL allocates for its work file. Substitute for ..SPSZ the value provided in the file CUSTOM.TXT. *nnnnnn* is the number (octal) of blocks you want SPOOL to allocate for its work file.

```
.RUN SIPP RET
*SPOOL.REL/A RET
Base?     0 RET
Offset?   ..SPSZ RET

     Base     Offset        Old        New?
      000000   ..SPSZ    ??????        nnnnnn RET
      000000   ..SPSZ+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

If there is not enough room on the volume for a work file of the default size, SPOOL.SYS occupies the largest empty area on the volume.

2.6.52 Changing SPOOL’s Output Device

You can change SPOOL’s default output device to any RT–11 non-file-structured device by installing the following software customization. In the customization, substitute for ..SPSO the value provided in the file CUSTOM.TXT. *nnn* is the new output device’s mnemonic.

```
.RUN SIPP RET
*SPOOL.REL/A RET
Base?     0 RET
Offset?   ..SPSO RET

     Base     Offset        Old       New?
      000000   ..SPSO    ??????       ;R RET
      000000   ..SPSO      <LP>       ;Rnnn RET
      000000   ..SPSO+2  ??????       CTRL/Y RET
* CTRL/C
.
```

2.6.53 Changing LINK for Default 132-Column Link Map and Global Cross-Reference Table

You can change the link map and global cross-reference table to default to 132 columns, rather than 80 columns. The value of ..WDSZ can be found in CUSTOM.TXT on your distribution kit.

```
.RUN SIPP RET
*LINK.SAV RET
Segment? 1 RET
Base? 0 RET
Offset? ..WDSZ RET

Segment   Base      Offset        Old      New?
000001     000000    ..WDSZ    ??????      000006 RET
000001     000000    ..WDSZ+2  ??????      CTRL/Y RET
* CTRL/C
.
```

2.6.54 Changing Default Size of MACRO–11 Work File

You can tailor the size of the MACRO–11 work file. If the work file is too small, a program that uses a great many symbols or macros can cause the work file to overflow and return the error message *?MACRO-F-Storage limit exceeded*. A work file that is too large may not fit on a small device such as VM.

In the customization, *nnn* represents the size of the work file and can be a value between 128 and 256_{10}. WRKSIZ is the symbol for the value given in the file CUSTOM.TXT on your distribution kit.

```
.R SIPP RET
*MACRO.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   WRKSIZ RET
```

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

2.6.55 Changing Name and Default Device for UCL.DAT

You can change the name of the UCL data file. With this patch, you can also change the default device on which the UCL data file resides.

The file specification for the UCL data file is stored in four RAD50 words. The first word is the device name without the colon (:). You must leave a trailing space if you use a 2-letter device name. The second word is the first three letters of the file name. The third word is the second three letters of the file name (if required). The fourth word is the file type. Do not include the period (.) between the file name and file type.

In the customization patch, ..UCLD is the symbol for the first RAD50 word. Substitute for ..UCLD the value for that word found in the file CUSTOM.TXT on your distribution kit. Substitute for *nnn* each part of the file specification that you changed when you created a new UCL data file.

```
.R SIPP RET
*UCL.SAV/A RET
Base?   0 RET
Offset?  ..UCLD RET

     Base     Offset       Old       New?
      0000     ..UCLD    ??????      ;R RET
      0000     ..UCLD    ??????      ;Rnnn RET
      0000     ..UCLD+2  ??????      ;Rnnn RET
      0000     ..UCLD+4  ??????      ;Rnnn RET
      0000     ..UCLD+6  ??????      ;Rnnn RET
      0000     ..UCLD+10 ??????      CTRL/Y  RET
* CTRL/C
.
```

2.6.56 Suppressing Dynamic Allocation of a Region for IND

IND dynamically allocates a region in extended memory for its symbol table when running RT–11 under a mapped monitor. You can suppress the dynamic allocation by using the following customization.

In the customization, ..ALRG is the symbol for the value given in the file CUSTOM.TXT on your distribution kit. Enter the value 0 (zero) for symbol *n* to suppress the dynamic allocation. Enter the value 1 for *n* to dynamically allocate a region.

```
.R SIPP RET
*IND.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..ALRG RET
```

```
Segment   Base      Offset       Old       New?
000000     000000    ..ALRG    ??????      n RET
000000     000000    ..ALRG+2  ??????       CTRL/Y  RET
* CTRL/C
.
```

2.6.57 Changing Size of UCL.DAT

You can change (increase or decrease) the maximum number of user-defined UCL commands in the UCL data file from 31_{10} to a maximum you specify. Perform the following customization patch to change the size of UCL.DAT. (Refer to the *Introduction to RT–11* for more information on UCL.)

In the customization patch, replace the symbol ..CMDS with the value given in the file CUSTOM.TXT on your distribution kit. Replace *nnnnn* with an octal value for the maximum number of defined (UCL) commands you want.

```
.R SIPP RET
*UCL.SAV/A RET
Base?    0 RET
Offset?  ..CMDS RET

     Base     Offset        Old       New?
      000000   ..CMDS        ??       nnnnn RET
      000000   ..CMDS+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

2.6.58 Changing Default VTCOM Send Speed

The VTCOM SEND command sends ASCII files at two speeds: SLOW or FAST. Use SLOW if the host terminal service does not support XON/XOFF and use FAST if it supports XON/XOFF. The default speed is SLOW.

The following customization patch lets you set the default speed at which VTCOM sends ASCII files. In the customization, ..FAST is the symbol for the value given in the file CUSTOM.TXT on your distribution kit. Set the value for *n* to 1 to specify FAST.

For VTCOM.REL:

```
.R SIPP RET
*VTCOM.REL RET
Base?     0 RET
Offset?   ..FAST RET

     Base     Offset        Old       New?
      000000   ..FAST    ??????       n RET
      000000   ..FAST+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

For VTCOM.SAV:

```
.R SIPP RET
*VTCOM.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..FAST RET

Segment   Base      Offset        Old      New?
000001     000000    ..FAST    ??????      n RET
000001     000000    ..FAST+2  ??????      CTRL/Y  RET
* CTRL/C
.
```

2.6.59 Modifying System Macro Library SYSMAC.SML

To modify the system macro library, recreate the file SYSMAC.MAC from the distributed file SYSMAC.SML by using the SPLIT utility.

The following command creates the file SYSMAC.MAC on your default device. The symbol ..SYSM represents the boundary along which to split SYSMAC.SML. Refer to the file CUSTOM.TXT on your distribution kit for the value to substitute for ..SYSM in the command line.

```
.SPLIT ,SYSMAC.MAC=SYSMAC.SML/B:..SYSM RET
```

Refer to the *RT–11 System Utilities Manual* for more information on using the SPLIT utility.

2.6.60 Modifying VTCOM Default Dial String Prefix

The default VTCOM dial string prefix is ^A^B. Perform the following customization patch if your modem requires a different dial string prefix. Consult your modem user guide for the correct prefix.

VTCOM first sends the dial string prefix and then sends the dial string. This patch does not affect the dial string.

In the customization, ..DPFX is the symbol for the value given in the file CUSTOM.TXT on your distribution kit, and nnn is the octal value for each ASCII character in the dial string prefix. Insert a null (000) in the location immediately following the last character in the dial string prefix. The prefix cannot be longer than 15 characters. The sixteenth byte must be a null.

For VTCOM.REL:

```
.R SIPP RET
*VTCOM.REL RET
Base?    0 RET
Offset?  ..DPFX RET
```

```
     Base     Offset        Old       New?
      000000   ..DPFX       ???       nnn  RET
      000000   ..DPFX+1     ???       nnn  RET
        .         .           .         .
        .         .           .         .
        .         .           .         .
      000000   ..DPFX+16    ???       nnn  RET
      000000   ..DPFX+17    ???        CTRL/Y   RET
*  CTRL/C
.
```

For VTCOM.SAV:

```
.R SIPP  RET
*VTCOM.SAV  RET
Segment? 1  RET
Base?    0  RET
Offset?  ..DPFX  RET

Segment   Base      Offset        Old      New?
000001     000000   ..DPFX        ???      nnn  RET
000001     000000   ..DPFX+1      ???      nnn  RET
  .          .         .            .        .
  .          .         .            .        .
  .          .         .            .        .
000001     000000   ..DPFX+16     ???      nnn  RET
000001     000000   ..DPFX+17     ???       CTRL/Y   RET
*  CTRL/C
.
```

2.6.61 Modifying VTCOM Default Dial String Suffix

The default dial string suffix is an exclamation (!) mark (041). Some modems require a different dial string suffix following the digits of the number to be dialed. Refer to your modem user guide for any required dial string suffix. Perform the following customization patch if your modem requires a different dial string suffix.

In the customization, ..DSFX is the symbol for the value given in the file CUSTOM.TXT on your distribution kit, and *nnn* is the octal value for each ASCII character in the dial string suffix. Insert a null (000) in the location immediately following the last character in the dial string suffix. The suffix cannot be longer than 15 characters. The sixteenth byte must be a null.

For VTCOM.REL:

```
.R SIPP  RET
*VTCOM.REL  RET
Base?    0  RET
Offset?  ..DSFX  RET
```

```
     Base     Offset        Old       New?
      000000   ..DSFX       ???       nnn RET
      000000   ..DSFX+1     ???       nnn RET
        .         .           .         .
        .         .           .         .
        .         .           .         .
      000000   ..DSFX+16    ???       nnn RET
      000000   ..DSFX+17    ???       CTRL/Y  RET
* CTRL/C
.
```

For VTCOM.SAV:

```
.R SIPP RET
*VTCOM.SAV RET
Segment? 1 RET
Base?    0 RET
Offset?  ..DSFX RET

Segment   Base      Offset        Old      New?
000001     000000    ..DSFX       ???      nnn RET
000001     000000    ..DSFX+1     ???      nnn RET
  .          .          .           .        .
  .          .          .           .        .
  .          .          .           .        .
000001     000000    ..DSFX+16    ???      nnn RET
000001     000000    ..DSFX+17    ???      CTRL/Y  RET
* CTRL/C
.
```

2.6.62 Automatically Creating KED/KEX Journal Files

The following customization patch automatically creates a journal file on the default data (DK) device for every work session.

In the customization, ..JOU is the symbol for a value located in the file CUSTOM.TXT.

For KED.SAV:

```
.RUN SIPP RET
*KED.SAV RET
Segment?  6 RET
Base?     0 RET
Offset?   ..JOU RET

Segment     Base     Offset        Old     New?
000006       000000   ..JOU     ??????     1 CTRL/Y  RET
* CTRL/C
.
```

For KEX.SAV:

```
.RUN SIPP RET
*KEX.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..JOU RET
```

```
Segment     Base     Offset        Old       New?
000001       000000   ..JOU     ??????       1 CTRL/Y  RET
* CTRL/C
.
```

2.6.63 Adjusting Default KED/KEX Journal File Size Allocation

By default, KED/KEX allocates 50_{10} blocks on the default or specified device for journal files. You can adjust the default number of allocated blocks, using the following customization patch.

In the customization, ..JSIZ is the symbol for a value located in the file CUSTOM.TXT. The symbol *nnn* represents the number of octal blocks you want to allocate as the default journal file size (62 octal blocks equal 50 decimal blocks).

For KED.SAV:

```
.RUN SIPP RET
*KED.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..JSIZ RET

Segment     Base     Offset        Old     New?
000000       000000   ..JSIZ    ??????     nnn CTRL/Y  RET
* CTRL/C
.
```

For KEX.SAV:

```
.RUN SIPP RET
*KEX.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..JSIZ RET

Segment     Base     Offset        Old       New?
000001       000000  ..JSIZ     ??????       nnn CTRL/Y  RET
* CTRL/C
.
```

2.6.64 Creating a Monitor-Independent ODT Debugger

The following procedure lets you create a monitor-independent ODT debugger; that is, an ODT that does not require that the operating system be loaded in memory.

The distributed debugger ODT.OBJ needs the operating system loaded in memory. You can modify ODT so it does not need the operating system loaded in memory, which could be useful, for example, in debugging the bootstrap.

Use the following procedure to create a debugger called ODTHWD.OBJ that functions independently of the operating system:

1. Use KED to create the following patch program. Name it ODTPAT.MAC. In the program, substitute for the symbol ..GVAL the value for that symbol located in the file CUSTOM.TXT.

```
.TITLE  ODT
.PSECT  $ODT$
BASE=..GVAL-1000
.=.+BASE
BR     .+34
.END
```

2. Assemble the created patch program:

```
.R MACRO RET
*ODTPAT=ODTPAT RET
* CTRL/C
```

3. Create the monitor-independent debugger ODTHWD.OBJ by modifying ODT.OBJ, which does not destroy the distributed ODT.OBJ but modifies a copy of it. Use the utility PAT.SAV in the following manner:

```
.R PAT RET
*ODTHWD=ODT,ODTPAT RET
* CTRL/C
```

4. Explicitly specify ODTHWD in the LINK/DEBUG command. If you do not specify ODTHWD, RT–11 by default links the distributed ODT.

```
.LINK/DEBUG:ODTHWD program RET
```

2.6.65 Forcing an Explicit KED/KEX Journal File Device

By default, KED/KEX creates a requested journal file on the DK device or a device you specify on the command line. You can force KED/KEX to create any requested journal file on a specified device, using the following customization patch. Note that if the customization is performed, KED/KEX creates a journal file on only the device specified in the customization; any device specified in the command line is ignored. Further, if the specified device cannot be found, a journal file is not created and KED/KEX returns an error message.

In the customization, ..JDEV is the symbol for a value located in the file CUSTOM.TXT. The symbol *rrr* represents the RAD50 value for the journal file device.

For KED.SAV:

```
.RUN SIPP RET
*KED.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..JDEV RET

Segment      Base     Offset        Old       New?
000000       000000  ..JDEV     ??????      ;R RET
000000       000000  ..JDEV     ??????      ;rrr CTRL/Y RET
* CTRL/C
.
```

For KEX.SAV:

```
.RUN SIPP RET
*KEX.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..JDEV RET

Segment     Base     Offset        Old       New?
000001       000000  ..JDEV     ??????       ;R RET
000001       000000  ..JDEV     ??????       ;rrr CTRL/Y  RET
* CTRL/C
.
```

2.6.66 Modifying the Storage Capacity for LEARN Macros

By default KED/KEX allocates 512_{10} locations for storing LEARN macros. That storage area is shared by up to 8 macro definitions. You can modify that storage capacity by performing the following customization. Increasing the storage capacity decreases the amount of memory available for paste buffer space and general file editing. Therefore, ask only the amount of storage capacity you require.

In the customization, ..MACS is the symbol for a value located in the file CUSTOM.TXT. The symbol *nnnnnnn* is the octal number of storage locations you want to support.

For KED.SAV:

```
.RUN SIPP RET
*KED.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..MACS RET

Segment     Base      Offset      Old     New?
000000       000000    ..MACS   ??????     nnnnnnn CTRL/Y  RET
*  CTRL/C
.
```

For KEX.SAV:

```
.RUN SIPP RET
*KEX.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..MACS RET
.b
Segment    Base     Offset        Old       New?
000001      000000   ..MACS    ??????       nnnnnnn CTRL/Y  RET
*  CTRL/C
.
```

2.6.67 Modifying the Maximum Number of LEARN Macros

By default KED/KEX allows 8_{10} LEARN macros to be defined at any one time. You can modify that limit by performing the following customization.

In the customization, ..MACN is the symbol for a value located in the file CUSTOM.TXT. The symbol *nn* is the octal number of macros you want to define.

The maximum is support for 27_{10} macro definitions, for which you would specify 33_8.

For KED.SAV:
```
.RUN SIPP RET
*KED.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..MACN RET

Segment     Base     Offset        Old       New?
000000       000000   ..MACN    ??????       nnn CTRL/Y  RET
* CTRL/C
.
```

For KEX.SAV:
```
.RUN SIPP RET
*KEX.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..MACN RET

Segment     Base     Offset        Old       New?
000001       000000   ..MACN    ??????       nnn CTRL/Y  RET
* CTRL/C
.
```

2.6.68 Restoring LOCAL Form Feed Processing

The LOCAL command ignores the presence of form feeds in files that are being processed. If for some reason you wish to restore the previous way the LOCAL command treated form feeds in files, you can perform the following customization.

In the customization, ..LCFF is the symbol for a value located in the file CUSTOM.TXT.

For KED.SAV:
```
.RUN SIPP RET
*KED.SAV RET
Segment?  0 RET
Base?     0 RET
Offset?   ..LCFF RET

Segment     Base     Offset        Old       New?
000000       000000   ..LCCF    ??????       000014 CTRL/Y  RET
* CTRL/C
.
```

For KEX.SAV:
```
.RUN SIPP RET
*KEX.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..LCFF RET
```

```
Segment     Base     Offset        Old       New?
000000       000000   ..LCCF    ??????       000014 CTRL/Y  RET
* CTRL/C
.
```

2.6.69 Removing the JSW Bit 6 Check for SL

Previously, SL checked only the state of bits 4 (EDIT$) and 12 (TTSPC$) in the JSW (job status word). SL refused to edit a command line if either was set. SL did not check if a program had set bit 6 (TCBIT$), the inhibit terminal wait bit. SL could hang in the rare event that a program had set bit 6 and not set bit 12 and was using the .TTINR request in a loop to clear the input buffer. (KED is such a program.)

SL checks the status of bits 4, 6, and 12 of the JSW. If a program sets any of those bits, SL does not edit command lines.

Possible incompatibilities remain with SL editing of command lines written and retrieved using requests other than .GTLIN. Digital continues to recommend you use .GTLIN to display prompts and retrieve terminal input.

If for some reason you require that SL ignore the status of bit 6 of the JSW, RT–11 provides the following customization patches to remove that functionality. Three patches are provided, one for each version of the SL command line editor. In the customization patches, _..SLCN, ..SLCX, and ..SLCM are the values for those symbols located in the file CUSTOM.TXT.

For SL.SYS:

```
.RUN SIPP RET
*SL.SYS/A RET
Base?    0 RET
Offset?   ..SLCN RET

     Base     Offset        Old       New?
      000000   ..SLCN    ??????       010020 RET
      000000   ..SLCN+2  ??????       CTRL/Y  RET
* CTRL/C
.
```

For SLX.SYS:

```
.RUN SIPP RET
*SLX.SYS/A RET
Base?    0 RET
Offset?   ..SLCX RET

     Base     Offset        Old       New?
      000000   ..SLCX    ??????       010020 RET
      000000   ..SLCX+2  ??????       CTRL/Y  RET
* CTRL/C
.
```

For SLMIN.SYS:

```
.RUN SIPP RET
*SLMIN.SYS/A RET
Base?    0 RET
Offset?   ..SLCM RET
```

```
     Base     Offset        Old        New?
      000000   ..SLCM    ??????       010020 RET
      000000   ..SLCM+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

2.6.70 Changing SL Character Edit Mode to REPLACE

SL can edit single characters in two modes: INSERT mode and REPLACE mode. INSERT mode lets you place a character in the command line by shifting to the right all characters beyond that character. REPLACE mode lets you place a character in the command line without shifting any characters. SL is distributed in INSERT mode. You can alternate between INSERT and REPLACE mode by pressing CTRL/A. SL returns to INSERT mode at each command line.

Apply the following customization to change the default mode for single-character editing from INSERT to REPLACE. You can still continue to alternate between REPLACE and INSERT mode by pressing CTRL/A. SL returns to REPLACE mode at each command line.

Three patches are provided, one for each version of the SL command line editor. In the customization patches, _..IRDF, ..IRDX, and ..IRDM are the values for those symbols located in the file CUSTOM.TXT.

For SL.SYS:

```
.RUN SIPP RET
*SL.SYS/A RET
Base?    0 RET
Offset?  ..IRDF RET

     Base     Offset        Old        New?
      000000   ..IRDF    ??????       000000 RET
      000000   ..IRDF+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

For SLX.SYS:

```
.RUN SIPP RET
*SLX.SYS/A RET
Base?    0 RET
Offset?  ..IRDX RET

     Base     Offset        Old        New?
      000000   ..IRDX    ??????       000000 RET
      000000   ..IRDX+2  ??????        CTRL/Y  RET
* CTRL/C
.
```

For SLMIN.SYS:

```
.RUN SIPP RET
*SLMIN.SYS/A RET
Base?    0 RET
Offset?  ..IRDM RET
```

```
     Base     Offset        Old        New?
      000000   ..IRDM    ??????        000000 RET
      000000   ..IRDM+2  ??????        CTRL/Y  RET
*  CTRL/C
.
```

2.6.71 Changing the VTCOM Command Mode Character

You can change the character you type to VTCOM to enter command mode. The default command mode character for VTCOM is CTRL/P. You can change the character CTRL/P to any you choose by using the following customization patch.

In the customization, ..LCCH is the symbol for a value located in the file CUSTOM.TXT, *nnn* is the octal value for the new default command mode character. That new character cannot be printable and cannot be a character that RT–11 intercepts and processes, such as CTRL/F, CTRL/B, CTRL/X, CTRL/O, and so on.

For VTCOM.REL:

```
.R SIPP  RET
*VTCOM.REL  RET
Base?    0  RET
Offset?  ..LCCH  RET

     Base     Offset        Old        New?
      000000   ..LCCH    ??????        \  RET

     Base     Offset        Old        New?
      000000   ..LCCH       ???        nnn  RET
      000000   ..LCCH+1     ???        CTRL/Y  RET
*  CTRL/C
.
```

For VTCOM.SAV:

```
.R SIPP   RET
*VTCOM.SAV   RET
Segment? 1   RET
Base?    0   RET
Offset?  ..LCCH   RET

Segment     Base     Offset        Old        New?
000001       000000   ..LCCH    ??????        \  RET

Segment     Base     Offset        Old        New?
000001       000000   ..LCCH       ???        nnn  RET
000001       000000   ..LCCH+1     ???        CTRL/Y  RET
*  CTRL/C
.
```

2.6.72 Forcing Obsolete SJ Monitor Value in CONFIG Bit $FBMON

Previously, a handler or program could check the state of the $FBMON bit in the CONFIG word (bit 0 in RMON fixed offset 300) to determine under which monitor it was running. In particular, $FBMON being clear indicated the SJ monitor. Further, some programs may have used the state of $FBMON to determine other monitor characteristics. The removal of the SJ monitor and the advent of new monitors have decoupled those characteristics with any particular distributed monitor, and

$FBMON is set with all distributed monitors. All distributed RT–11 software is also decoupled from $FBMON.

This customization patch is provided because a user-written device handler or program might require $FBMON be clear as a condition for installation. In that case, the following procedure is recommended:

1. Perform the customization patch, which clears $FBMON in the file image, allowing the device handler to install once the customized monitor is booted.
2. Reset $FBMON early in your start-up command file, using the command SET MODE NOSJ.

   The SET command conditions SET MODE SJ and SET MODE NOSJ clear or set bit zero (the $FBMON bit) in CONFIG. Those SET commands affect only the memory resident monitor image. Therefore, $FBMON is clear at installation time but correctly set before the system is used.

Digital recommends you do not run RT–11 with $FBMON clear as that setting does not accurately describe any monitor distributed with the operating system.

In the customization, *monitr.sys* is the monitor you are running, and *CONFIG* is the address of that symbol in the monitor link map.

To clear $FBMON and thereby simulate the obsolete SJ monitor setting, clear the low bit in CONFIG. To set $FBMON (the distributed condition), set the low bit in CONFIG.

```
.RUN SIPP RET
*monitr.SYS RET
Base?     0 RET
Offset?   CONFIG RET

Base     Offset        Old       New?
000000    CONFIG    ??????       \ RET

Base     Offset        Old       New?
000000    CONFIG       ???          value with bit zero
                                        set or clear RET
000000    CONFIG+1     ???       CTRL/Y  RET
* CTRL/C
.
```

2.6.73 Making LINK /G Option a Default

The LINK /G option (increasing the directory buffer size) is available only from CSI command syntax; it is not available as a DCL option. The following customization patch enables the /G option whenever LINK is run by any command syntax. You should read the LINK /G option section in the *RT–11 System Utilities Manual* before performing this patch as there are possible negative effects.

In the customization, ..GON is the symbol for a value located in the file CUSTOM.TXT.

```
.RUN SIPP RET
*LINK.SAV RET
Segment?  1 RET
Base?     0 RET
Offset?   ..GON RET

Segment     Base     Offset        Old       New?
000001       000000   ..GON     ??????          240 RET
000001       000000   ..GON+2   ??????          240 RET
000000       000000   ..GON+4   ??????              CTRL/Y  RET
* CTRL/C
.
```

2.6.74 Changing the Default Magtape Label Size

You can change the default magtape label size from 40_{10} words to any value between 40_{10} and 512_{10}. (The value is entered in the customization in octal radix.)

In the following customization, ..MTLS is the symbol for the value located in CUSTOM.TXT, and *nnnn* is the octal value for the default magtape label size you want.

```
.R SIPP RET
*DUP.SAV RET
Segment?  7 RET
Base?     0 RET
Offset?  ..MTLS RET

Segment   Base     Offset     Old       New?
000007     000000   ..MTLS     ???       nnnn RET
000001     000000   ..MTLS+2   ???        CTRL/Y  RET
* CTRL/C
.
```

Chapter 3

Installing a System Distributed on RX01 to Run on RX01

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

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. Do not 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 in the following list. Sections 3.1 through 3.8 describe the procedures for each step.

1. Bootstrap the distribution volume.
2. Create the working system from chosen components.
3. Install the bootstrap on volumes that need to be bootable.
4. Customize the system.
5. Compress each volume.
6. Preserve the working system.
7. Test the working system.
8. If appropriate, perform the system generation process.
9. Read the *Introduction to RT–11* for more information on using RT–11.

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

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. The device name for the RX01 drive is DX, so unit 0 is DX0.

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.xx

.TYPE V5USER.TXT

                    RT-11 V5.xx

Installation of RT-11....(followed by V5USER.TXT message)....

.
```

After you boot the system, you can edit the start-up file to turn on or enter any commands or sequence of commands that you would normally want to execute when you boot your processor. Editing start-up command files is described in the *Introduction to RT–11*.

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 Commands Manual* 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 DX1:*.* 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.

3.2 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.4) and have planned the best arrangement of them on volumes (Section 2.5), you can create the working system by copying selected components to initialized, blank volumes.

Start by initializing a number of blank volumes. Insert a write-enabled, blank volume in Unit 1 (with the system booted from Unit 0), and type INITIALIZE /BADBLOCKS DX1:. Repeat the process to create as many initialized blank volumes as you need for the system that you have planned.

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 DX0:filnam.typ DX1:filnam.typ  RET
.
```

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

```
.COPY/QUERY/SYSTEM DX0: DX1:  RET
  Files copied:
DX0:aaaaaa.ttt to DX1:aaaaaa.ttt? Y  RET  (includes aaaaaa.ttt)
DX0:bbbbbb.ttt to DX1:bbbbbb.ttt? N  RET  (excludes bbbbbb.ttt)
(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 DX1:filnam.typ DX0:filnam.typ  RET
Mount input volume in DX1:; Continue?
```

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

```
Y  RET

Mount output volume in DX0:; 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 DX0:; 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". 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
.
```

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

Once you have created your system, 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, SWAP.SYS, and system device handler should be bootable.

Insert in Unit 1 the volume on which you need to install the bootstrap. In the following command, *aa* is SB, FB, or one of the mapped monitors.

```
.COPY/BOOT DX1:RT11aa.SYS DX1:  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.xx
(Followed by any start-up file commands.)
.
```

Store the updated backup volumes for future updating purposes.

3.4 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–2 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 3.8.)

3.5 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.

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 DX0: RET
DX0:/Squeeze; Are you sure? Y RET
```

There is a delay while the system compresses the volume.

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

The system automatically reboots when you compress a system volume.

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

```
.SQUEEZE DX1: RET
DX1:/Squeeze; Are you sure? Y RET
.
```

Replace the volume in Unit 1 with the next one you need to compress. 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.

3.6 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 DX1:*.* RET
 Files protected:
DX1:aaaaaa.ttt
DX1:bbbbbb.ttt
DX1:cccccc.ttt
DX1:dddddd.ttt
      .
      .
      .
DX1: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:DXn: command to copy any bootable volumes. Remember to use SQUEEZE/WAIT /OUTPUT:DXn: and change volumes to copy the volumes that are not bootable. Also, remember to copy the bootstrap to any volumes that need to be bootable.

Write-protect the backup working system volumes and store them. To restore the working system, make copies of the backup working system volumes.

3.7 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 (monitor)
DX.SYS (system device handler)
LP.SYS (or LS.SYS)
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

Display the directory of the system volume on the terminal. The directory varies according to your 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
KED   .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press CTRL/S or HOLD SCREEN to stop the display and CTRL/Q or HOLD SCREEN again 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.

3.7.1 Preparing the Background Demonstration Program

3.7.1.1 Edit the Background Demonstration Program

Use the KED text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC RET
PF1  PF3
Model? ; tab .ASCII 4  ,
PF1  7
Command? EXIT ENTER
.
```

3.7.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP RET
.ASSIGN LP LST RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 3–1.)

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
.
```

Then, repeat the editing procedure.

3.7.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG RET
.
```

3.7.1.4 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 twice pressing CTRL/C:

```
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
.
```

Figure 3–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200     ;MESSAGE THAT RINGS BELL
     21
     22 000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24 000225     127          .ASCII  /WELL DONE./
     25 000237     000          .BYTE   0
     26
     27 000240          AREA:   .BLKW 6           ;EMT ARGUMENT AREA
     28 000254          BUFFER:                   ;RCVDC MESSAGE AREA
     29         000000'         .END    START
DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

3.7.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

3.7.2.1 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.

3.7.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

3.7.2.3 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
.
```

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

3.8 Performing the System Generation Process

If you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation on a computer system containing more suitable storage media and then use the generated modules on your RX01 system.

3.9 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Chapter 4

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

If your distribution media is single-density diskettes, 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. Do not 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 in the following list. Sections 4.1 through 4.10 describe the procedures for each step.

1. Bootstrap the distribution volume.
2. Copy the distribution volumes to the disk.
3. Preserve the distribution volumes.
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.
11. Read the *Introduction to RT–11* for more information on using RT–11.

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 (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.xx

.TYPE V5USER.TXT

RT-11 V5.xx

Installation of RT-11....(followed by V5USER.TXT message)....

.
```

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.

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. See the *Introduction to RT–11* for information on start-up command files.

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 MSCP or 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.

RL02 disk

```
.FORMAT DL0: RET
DL0:/FORMAT-Are you sure? Y RET
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DL0: RET
DL0:/Initialize; Are you sure? Y RET
?DUP-I-No bad blocks detected DL0:
```

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.

```
.SQUEEZE/OUTPUT:xx0: DX0: 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 Commands Manual*.

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 DX1: xx0: RET
 Files copied:
DX1:aaaaaa.ttt  to xx0:aaaaaa.ttt
DX1:bbbbbb.ttt  to xx0:bbbbbb.ttt
            .
            .
            .
DX1: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.xx
(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 Commands Manual* 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, 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 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.)

RL02 disk

```
.INITIALIZE/BADBLOCKS DL1: RET
DL1:/Initialize; Are you sure? Y RET
?DUP-I-No bad blocks detected DL1:
```

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  (includes aaaaaa.ttt)
xx0:bbbbbb.ttt to xx1:bbbbbb.ttt? N RET  (excludes bbbbbb.ttt)
(and so on)
```

4.5 Installing the Bootstrap on the Disk

Once you have created your system, install the bootstrap on the system disk. In the following command, *aa* is SB, FB, or one of the mapped monitors.

```
.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, 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.xx
(Followed by any start-up file commands.)
.
```

4.6 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–2 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 4.10.)

4.7 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.

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.xx
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system disk.

4.8 Preserving the Working System

Once you build a satisfactory working system, Digital recommends that you protect all the files in the working system and 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.

4.8.1 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/MULTIVOLUME command. In the command, yy is the name for your disk. The system copies all the files from the input volume that will fit on the output volume. When no more files will fit on the output volume, the system prompts you to mount the next output volume and prints the *Continue?* message.

Mount another blank, initialized diskette, and type Y to continue. Continue to mount diskettes in this fashion until all the files are copied.

```
.COPY/MULTIVOLUME/SYSTEM yy0: DX1: RET
  Files copied:
yy0:aaaaaa.ttt to DX1:aaaaaa.ttt
yy0:bbbbbb.ttt to DX1:bbbbbb.ttt
(and so on)

Mount next output volume in DX1:; Continue? Y RET
(Log of files copied)
Mount next output volume in DX1:; Continue? Y RET
(Log of files copied)
Mount next output volume in DX1:; Continue? Y RET
(Log of files copied)
Mount next output volume in DX1:; Continue? Y RET
(Log of files copied)
Mount next output volume in DX1:; Continue? Y RET
.
.
```

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.

Store the backup diskettes. To restore the working system, you can copy the backup diskettes to the disk.

4.8.2 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.

RL02 disk

```
.INITIALIZE/BADBLOCKS DL1: RET
DL1:/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 SB, FB, or one of the mapped monitors.

```
.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.

4.9 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 (or LS.SYS)
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

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
KED   .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press CTRL/S or HOLD SCREEN to stop the display and CTRL/Q or HOLD SCREEN again 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.

4.9.1 Preparing the Background Demonstration Program

4.9.1.1 Edit the Background Demonstration Program

Use the KED text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC RET
PF1  PF3
Model? ; tab .ASCII 4  ,
PF1  7
Command? EXIT ENTER
.
```

4.9.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP RET
.ASSIGN LP LST RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 4–1.)

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 4–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200     ;MESSAGE THAT RINGS BELL

000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
000225     127          .ASCII  /WELL DONE./
000237     000          .BYTE   0

000240          AREA:   .BLKW 6           ;EMT ARGUMENT AREA
000254          BUFFER:                   ;RCVDC MESSAGE AREA
        000000'         .END    START
DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

4.9.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG RET
.
```

4.9.1.4 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 twice pressing CTRL/C:

```
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.

4.9.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

4.9.2.1 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.

4.9.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

4.9.2.3 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
.
```

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

4.10 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 hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

4.11 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Chapter 5

Installing a System Distributed on RL02 to Run on Hard Disk

If your distribution media is an RL02 hard disk 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. Do not 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 in the following list. Sections 5.1 through 5.9 describe the procedures for each step.

1. Bootstrap the distribution disk.
2. Preserve the distribution disk.
3. Create the working system from chosen components.
4. Install the bootstrap on the disk.
5. Customize the system.
6. Compress the disk.
7. Preserve the working system.
8. Test the working system.
9. If appropriate, perform the system generation process.
10. Read the *Introduction to RT–11* for more information on using RT–11.

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. 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.)

RL02 disks are normally installed manually. If you successfully bootstrap an RL02 disk, RT–11 responds with the following dialog and installation verification message. The dialog gives you the choice of installing RT–11 manually or by using the automatic installation procedure.

```
Welcome to RT-11 Version 5.

You have bootstrapped the RT-11 Distribution Disk. Use this disk to
install your RT-11 system, then store it in a safe place.

RT-11 V5 provides an automatic installation procedure which will
back up your distribution disk and build a working system disk which
should be used for your work with RT-11. This working system disk
will only contain the RT-11 operating system. After the RT-11
installation is complete, follow the installation instructions
packaged with any optional languages or utility software which you
will be using.

Press the "RETURN" key when ready to continue.  RET
You can choose to install RT-11 manually. This procedure is
described in the RT-11 Installation Guide.

If you are a new user of RT-11, Digital highly recommends that you
use the automatic installation procedure.

Do you want to use the automatic installation procedure?
(Type YES or NO and press the "RETURN" key): NO RET

The standard RT-11 monitor bootstrap (RT11FB) will now be
copied to this disk and this system disk will be rebooted.
Please refer to the RT-11 Installation Guide for assistance
while manually installing RT-11 Version 5.

Press the "RETURN" key when ready to continue. RET

RT-11FB V05.xx

.TYPE V5USER.TXT

RT-11 V5.xx

Installation of RT-11....(followed by V5USER.TXT message)....

.
```

Now you can edit the start-up file to turn on or create any commands or sequence of commands that you would normally want to execute at the start of each terminal session. Editing start-up command files is described in the *Introduction to RT–11*.

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. If the disk is an RK05 disk, format it, 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 MSCP or is an RK06, RK07, RL01, or RL02. (Refer to the *RT–11 Commands Manual*.) 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.

RL02 disk

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

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 Commands Manual*.

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. Mount the backup disk on Unit 0, and use the hardware bootstrap to boot the backup disk.

```
RT-11FB V05.xx
(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 Commands Manual* 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, 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 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.4) and have planned the best arrangement of them on a disk (Section 2.5), 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.)

RL02 disk

```
.INITIALIZE/BADBLOCKS DL1: RET
DL1:/Initialize; Are you sure? Y RET
?DUP-I-No bad blocks detected DL1:
```

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  (includes aaaaaa.ttt)
xx0:bbbbbb.ttt to xx1:bbbbbb.ttt? N RET  (excludes bbbbbb.ttt)
(and so on)
```

5.4 Installing the Bootstrap on the Disk

Once you have created your system, install the bootstrap on the system disk. In the following command, aa is SB, FB, or one of the mapped monitors.

```
.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, 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.xx
(Followed by any start-up file commands.)
.
```

5.5 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–2 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 5.9.)

5.6 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.xx
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system disk.

5.7 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 backup 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.

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, *xxn* is the device name and unit number for the backup working system disk.

RL02 disk

```
.INITIALIZE/BADBLOCKS DL1: RET
DL1:/Initialize; Are you sure? Y RET
?DUP-I-No bad blocks detected DL1:
```

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. In the following command, *yyn* is the device name and unit number for your working system, 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 SB, FB, or one of the mapped monitors.

```
.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.

5.8 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
xx.SYS (system device handler)
LP.SYS (or LS.SYS)
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

Display the directory of the system volume on the terminal. The directory varies according to your 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
KED   .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press CTRL/S or HOLD SCREEN to stop the display and CTRL/Q or HOLD SCREEN again 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.

5.8.1 Preparing the Background Demonstration Program

5.8.1.1 Edit the Background Demonstration Program

Use the KED text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC  RET
 PF1   PF3
Model? ; tab .ASCII  4   ,
 PF1   7
Command? EXIT  ENTER
.
```

5.8.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP  RET
.ASSIGN LP LST  RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST  RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 5–1.)

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 5–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200     ;MESSAGE THAT RINGS BELL
     21
     22 000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24 000225     127          .ASCII  /WELL DONE./
     25 000237     000          .BYTE   0
     26
     27 000240          AREA:   .BLKW 6           ;EMT ARGUMENT AREA
     28 000254          BUFFER:                   ;RCVDC MESSAGE AREA
     29         000000'         .END    START
DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

5.8.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG  RET
.
```

5.8.1.4 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 twice pressing CTRL/C:

```
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.

5.8.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

5.8.2.1 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: DEMOFG RET
.
```

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

5.8.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

5.8.2.3 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
.
```

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

5.9 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 hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

5.10 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Chapter 6

Installing a System Distributed on RL02 to Run on a Small Device

If your distribution media is an RL02 hard disk 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. Do not 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 in the following list. Sections 6.1 through 6.9 describe the procedures for each step.

1. Bootstrap the distribution disk.
2. Preserve the distribution disk.
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.
10. Read the *Introduction to RT–11* for more information on using RT–11.

6.1 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. 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.)

RL02 disks are normally installed manually. If you successfully bootstrap an RL02 disk, RT–11 responds with the following dialog and installation verification message. The dialog gives you the choice of installing RT–11 manually or by using the automatic installation procedure.

```
Welcome to RT-11 Version 5.

You have bootstrapped the RT-11 Distribution Disk. Use this disk to
install your RT-11 system, then store it in a safe place.

RT-11 V5 provides an automatic installation procedure which will
back up your distribution disk and build a working system disk which
should be used for your work with RT-11. This working system disk
will only contain the RT-11 operating system. After the RT-11
installation is complete, follow the installation instructions
packaged with any optional languages or utility software which you
will be using.

Press the "RETURN" key when ready to continue.  RET
You can choose to install RT-11 manually. This procedure is
described in the RT-11 Installation Guide.

If you are a new user of RT-11, Digital highly recommends that you
use the automatic installation procedure.

Do you want to use the automatic installation procedure?
(Type YES or NO and press the "RETURN" key): NO RET

The standard RT-11 monitor bootstrap (RT11FB) will now be
copied to this disk and this system disk will be rebooted.
Please refer to the RT-11 Installation Guide for assistance
while manually installing RT-11 Version 5.

Press the "RETURN" key when ready to continue. RET

RT-11FB V05.xx

.TYPE V5USER.TXT

RT-11 V5.xx

Installation of RT-11....(followed by V5USER.TXT message)....

.
```

Now you can edit the start-up file to turn on or create any commands or sequence of commands that you would normally want to execute when you boot the processor. Editing start-up command files is described in the *Introduction to RT–11*.

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 disk to build your working system.

Mount a blank disk in the other disk drive, if necessary. 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 MSCP or is an RK06, RK07, RL01, or RL02. (Refer to the *RT–11 Commands Manual*.) 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.

RL02 disk

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

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 Commands Manual*.

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. Mount the backup disk on Unit 0, and use the hardware bootstrap to boot the backup disk.

```
RT-11FB V05.xx
(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 Commands Manual* 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, 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 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.4) and have planned the best arrangement of them on diskettes (Section 2.5), you can create the working system by copying selected components to blank diskettes.

6.3.1 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.

6.3.2 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.

6.3.3 Copying the Selected Files onto the 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 number of 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   (includes aaaaaa.ttt)
yyn:bbbbbb.ttt to xxn:bbbbbb.ttt? N RET   (excludes bbbbbb.ttt)
(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).

6.4 Installing the Bootstrap on Any Diskettes That Need to Be Bootable

Once you have created your working system diskettes, 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, SWAP.SYS, and system device handler should be bootable.

Insert the diskette on which you need to install the bootstrap. In the following command, *xxn* represents the device name and unit number of your diskette, and *aa* is SB, FB, or one of the mapped monitors.

```
.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, where *xxn* represents the diskette to boot.

```
.BOOT xxn: RET

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

Store your updated backup distribution disk for future updates.

6.5 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–2 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 6.9.)

6.6 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 xx0: RET
xx0:/Squeeze; Are you sure? Y RET
```

There is a delay while the squeeze operation takes place.

```
RT-11aa V05.xx
(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? Y  RET
.
```

Replace the diskette in Unit 1 with the next one you need to compress. 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.

6.7 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 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 to use SQUEEZE/WAIT /OUTPUT:xx1: xx0: 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 working system diskettes. To restore the working system, make copies of the backup working system diskettes.

6.8 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
xx.SYS (system device handler)
LP.SYS (or LS.SYS)
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

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
KED   .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press CTRL/S or HOLD SCREEN to stop the display and CTRL/Q or HOLD SCREEN again 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.

6.8.1 Preparing the Background Demonstration Program

6.8.1.1 Edit the Background Demonstration Program

Use the KED text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC RET
PF1 PF3
Model? ; tab .ASCII 4 ,
PF1 7
Command? EXIT ENTER
.
```

6.8.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP RET
.ASSIGN LP LST RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 6–1.)

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 6–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200                  ;MESSAGE THAT RINGS BELL
     21
     22 000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24 000225     127          .ASCII  /WELL DONE./
     25 000237     000          .BYTE   0
     26
     27 000240          AREA:   .BLKW 6                        ;EMT ARGUMENT AREA
     28 000254          BUFFER:                                ;RCVDC MESSAGE AREA
     29         000000'         .END    START

DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

6.8.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG RET
.
```

6.8.1.4 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 twice pressing CTRL/C:

```
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.

6.8.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

6.8.2.1 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: DEMOFG RET
.
```

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

6.8.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

6.8.2.3 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
.
```

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

6.9 Performing the System Generation Process

If 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.

6.10 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Chapter 7

Installing a System Distributed on RX02 to Run on RX02

If your distribution is on RX02 double-density diskettes, and you intend to build a system to run on double-density diskettes, 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. Do not 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 in the following list. Sections 7.1 through 7.8 describe the procedures for each step.

1. Bootstrap the distribution diskette.
2. Create the working system from chosen components.
3. Install the bootstrap on diskettes that need to be bootable.
4. Customize the system.
5. Compress each diskette.
6. Preserve the working system.
7. Test the working system.
8. If appropriate, perform the system generation process.
9. Read the *Introduction to RT–11* for more information on using RT–11.

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 1 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.xx

.TYPE V5USER.TXT
RT-11 V5.xx

Installation of RT-11....(followed by V5USER.TXT message)....

.
```

After you boot the system, you can edit the start-up file to turn on or enter any commands or sequence of commands that you would normally want to execute when you boot the processor. Editing start-up command files is described in the *Introduction to RT–11*.

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 Commands Manual* 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 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.4) and have planned the best arrangement of them on diskettes (Section 2.5), 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   (includes aaaaaa.ttt)
DY0:bbbbbb.ttt to DY1:bbbbbb.ttt? N  RET   (excludes bbbbbb.ttt)
(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”. 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
.
```

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

Once you have created your system, 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, SWAP.SYS, and system device handler should be bootable.

Insert in Unit 1 the volume on which you need to install the bootstrap. In the following command, *aa* is SB, FB, or one of the mapped monitors.

```
.COPY/BOOT DY1:RT11aa.SYS DY1: 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.xx
(Followed by any start-up file commands.)
.
```

Store the updated backup volumes for future updating.

7.4 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–2 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 7.8.)

7.5 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 volume 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 volume must be write enabled.) 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 system compresses the volume.

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

The system automatically reboots when you compress a system volume.

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

```
.SQUEEZE DY1: RET
DY1:/Squeeze; Are you sure? Y RET
.
```

Replace the volume in Unit 1 with the next one you need to compress. 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.

7.6 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 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 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:DYn: command to copy any bootable volumes. Remember to use SQUEEZE/WAIT /OUTPUT:DYn: and change volumes to copy the volumes that are not bootable. 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, make copies of the backup working system volumes.

7.7 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 (or LS.SYS)
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

Display the directory of the system volume on the terminal. The directory varies according to your 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
KED   .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
```

```
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press CTRL/S or HOLD SCREEN to stop the display and CTRL/Q or HOLD SCREEN again 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.

7.7.1 Preparing the Background Demonstration Program

7.7.1.1 Edit the Background Demonstration Program

Use the KED text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC  RET
 PF1    PF3
Model? ; tab .ASCII  4   ,
 PF1    7
Command? EXIT  ENTER
.
```

7.7.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP  RET
.ASSIGN LP LST  RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST  RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 7–1.)

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 7–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200                    ;MESSAGE THAT RINGS BELL
     21
     22 000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24 000225     127          .ASCII  /WELL DONE./
     25 000237     000          .BYTE   0
     26
     27 000240          AREA:   .BLKW 6                          ;EMT ARGUMENT AREA
     28 000254          BUFFER:                                  ;RCVDC MESSAGE AREA
     29         000000'         .END    START

DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

7.7.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG RET
.
```

7.7.1.4 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 twice pressing CTRL/C:

```
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.

7.7.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

7.7.2.1 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.

7.7.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

7.7.2.3 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
.
```

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

7.8 Performing the System Generation Process

If you need RT–11 features that are available only if you generate your own monitor(s) and handlers, you have at least two possible courses of action.

You can attempt a system generation on the RX02 system. Note that the *RT–11 System Generation Guide* does not recommend attempting a system generation with an RX02 configuration. Although perhaps possible, the procedure with this configuration is difficult, complicated, and very time consuming.

If possible, perform the system generation on a computer system containing more suitable storage media and then use the generated modules on your RX02 system.

7.9 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Chapter 8

Installing a System Distributed on Magtape to Run on Hard Disk

If your distribution is on magtape, and you intend to build a system to run on hard disk, you can install the system automatically or manually. If you decide to install the system manually, 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.

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. Do not set TT NOPAGE when you use this terminal. Refer to your hardware manuals for more information about these settings.

Sections 8.1 through 8.10 describe the procedures for each of the following steps.

1. Bootstrap the first distribution magtape.
2. Preserve both distribution magtapes.
3. Create bootable master magtapes.
4. Create the working system on the disk 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.
11. Read the *Introduction to RT–11* for more information on using RT–11.

8.1 Bootstrapping the First Distribution Magtape

The first procedure you perform when installing RT–11 is to bootstrap the first distribution magtape. (If your distribution is a TMSCP magtape cartridge, you have only one magtape and this procedure is the same except where indicated otherwise.)

Begin by making sure that the processor is powered up but not running. If your distribution magtape is the 9-track reel, mount the 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 distribution is a TK50 cartridge, insert the magtape as descibed in your system documentation.

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, TS11, TJU16 or TMSCP as a magtape device. If you have a TM11, use the magtape build program named MDUP.MT. If you have a TS11, use the magtape build program named MDUP.MS. If you have a TJU16, use the magtape build program named MDUP.MM. If you have a TMSCP, use the magtape build program named MDUP.MU. Issue one of the following commands:

```
.MDUP.MT  RET

.MDUP.MS  RET

.MDUP.MM  RET

or

.MDUP.MU  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, 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 TS11:
```

```
*xx0:A=MS0: RET
```

```
If the magtape is TJU16:
```

```
*xx0:A=MM0: RET
```

```
If the magtape is TMSCP:
```

```
*xx0:A=MU0: 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-11SB V05.xx
?KMON-F-File not found DK:STRTSB.COM
.
```

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 MSCP, RK06, RK07, RL01, and RL02 disks. If your disk is an MSCP, 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-91); 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 MM, MS, MT or MU:

```
.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*).

If the distribution magtape was TK50, reboot as shown below. If the distribution magtape was the 2-volume reel-type, momentarily ignore the message to reboot your system. Remove distribution magtape 1 from your drive and install volume 2. Repeat the command:

```
.COPY/SYSTEM/NOREPLACE xx0:*.* DK: RET
```

Reboot as advised.

```
.BOOT SY: RET
```

8.2 Preserving the Magtape Distribution

Use the indirect command files MTB.COM, MSB.COM, or MUB.COM to preserve the distribution magtapes.

NOTE

Although RT–11 is not distributed on 800 bpi magtapes, the procedure to back up the distribution to 800 bpi magtapes and create bootable magtapes is also described.

- Use MTB.COM if your magtape is written at 800 bpi.
- Use MSB.COM if your magtape is written at 1600 bpi.
- Use MUB.COM if your magtape is a TK50.

If your magtape device is 800 bpi or 1600 bpi, proceed to Section 8.2.1. If your magtape device is a TK50, proceed to Section 8.2.2.

8.2.1 Using MTB.COM and MSB.COM

The following procedure describes using MTB.COM and MSB.COM:

1. Mount a blank magtape in the tape drive. Ensure that the write ring is in the back of the reel and that the tape is positioned at the load point.

2. Copy from the disk to your blank magtape those files that were on distribution magtape 1. To copy those files, invoke the indirect command file MTB.COM if your magtape is 9-track, 800 bpi. If your magtape is 9-track, 1600 bpi, invoke MSB.COM. (The commands you enter are shown below.)

   MTB.COM or MSB.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 BIN: to your disk device and the logical name KIT: to your tape drive. In the commands, *xx* is MT, MS, or MM, and yy is your disk.

   If your magtape is written at 800 bpi, issue:

   ```
   .ASSIGN xxn: KIT: RET
   .ASSIGN yyn: BIN: RET
   .$@MTB RET
   ```

   If your magtape is written at 1600 bpi, issue:

   ```
   .ASSIGN xxn: KIT: RET
   .ASSIGN yyn: BIN: RET
   .$@MSB RET
   ```

   (The commands in the indirect file appear on the terminal.)

3. Replace the backup magtape with a blank magtape. When MTB.COM or MSB.COM finishes copying files to the first magtape, it prints the following question on the terminal:

   ```
   xxn:/Initialize; Are you sure?
   ```

   Before you type a response, rewind the newly created backup magtape, remove it, write protect it by removing the ring, and label it “RT–11 Bootable V05.6 1/2”.

   Then, mount another blank magtape, ensuring that the the write ring is in the back of the reel. Make sure that the tape is positioned at the load point.

4. Copy from the disk to the blank magtape those files that were on distribution magtape 2.

   Type a Y RET in response to the “Initialize; Are you sure?” question on the terminal.

   MTB.COM or MSB.COM initializes the blank magtape and copies a duplicate of distribution magtape 2. When the indirect command file finishes executing, the monitor prompt (.) appears on your terminal. Rewind the newly created backup magtape, remove it, write protect it by removing the ring, and label it “RT–11 Bootable V05.6 2/2”.

Store the magtapes.

8.2.2 Using MUB.COM

To use MUB.COM:

1. Slide the write-protect switch on the front of a blank TK50 cartridge to the right to enable writing on the tape. Insert and load the blank TK50 magtape cartridge in the tape drive.
2. When the TK50 magtape cartridge is loaded, invoke the indirect command file MUB.COM to copy the files from the disk to your blank TK50 magtape. MUB.COM initializes the blank TK50 magtape, writes the primary bootstrap on it, and copies files in their correct order from the disk to the TK50 magtape. Before you invoke MUB.COM, use the ASSIGN command to assign the logical name BIN: to your disk device and the logical name KIT: to your tape drive. In the following commands, *yy* is your disk.

   ```
   .ASSIGN MUn: KIT: RET
   .ASSIGN yyn: BIN: RET
   .$@MUB RET
   ```

   (The commands in the indirect file appear on the terminal.)
3. When MUB.COM finishes executing, the monitor prompt (.) appears on your terminal. Rewind the newly created backup TK50 magtape, remove it, write protect it, and label it “RT–11 Bootable V05.6 1/1”.

   Store the TK50 magtape.

8.3 Creating a Master (Distribution) Magtape

MTB.COM, MSB.COM, and MUB.COM are procedures that let you create a bootable master magtape while you run RT–11 from a disk.

Use MTB.COM if your magtape is to be written at 800 bpi. Use MSB.COM if your magtape is to be written at 1600 bpi. Use MUB.COM if your magtape is a TK50.

Section 8.2.1 describes using MTB.COM and MSB.COM. Section 8.2.2 describes using MUB.COM. Proceed to the appropriate section to create a bootable master magtape.

8.4 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.4), 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   (deletes aaaaaa.ttt)
DK:bbbbbb.ttt? N RET   (retains bbbbbb.ttt)
(and so on)
```

8.5 Installing the Bootstrap on the Disk

Once you have created your working system, copy the bootstrap from the monitor file of your choice to the disk. In the command, *xx* is the permanent device name for your disk, and *aa* is SB, FB, or one of the mapped monitors.

```
.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.xx
(Followed by any start-up file commands.)
.
```

NOTE

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

8.6 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any of these software customizations. Table 1–2 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 8.10.)

8.7 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.

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? Y RET
RT-11aa V05.xx
(Followed by any start-up file commands.)
.
```

The system automatically reboots when you compress a system disk.

8.8 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.

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. If your device name is MT or MM, use the following command (where *xx* is MT or MM):

```
.INITIALIZE/FILE:MBOOT.BOT xx0: RET
xx0:/Initialize; Are you sure?  Y RET
.
```

If your device name is MS, use the following command:

```
.INITIALIZE/FILE:MBOT16.BOT MS0: RET
MS0:/Initialize; Are you sure?  Y RET
.
```

If your device name is MU, use the following command:

```
.INITIALIZE/FILE:MBOOT.BOT MU0: RET
MU0:/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, MS, or MU and *aa* is SB, FB, or one of the mapped monitors:

```
.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
DK:MDUP.MU to xx0:MDUP.MU

.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:DL.SYS to xx0:DL.SYS
DK:DM.SYS to xx0:DM.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
.
.
.
.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
DK:MU.SYS to xx0:MU.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   (includes aaaaaa.ttt)
DK:zzzzzz.ttt to xx0:zzzzzz.ttt? N RET   (excludes zzzzzz.ttt)
(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.

8.9 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 (or LS.SYS)
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

Display the directory of the system volume on the terminal. The directory varies according to your 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
MS    .SYS
KED   .SAV
MACRO .SAV
SYSMAC.SML
```

```
LINK  .SAV
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press CTRL/S or HOLD SCREEN to stop the display and CTRL/Q or HOLD SCREEN again 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.

8.9.1 Preparing the Background Demonstration Program

8.9.1.1 Edit the Background Demonstration Program

Use the KED text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC RET
PF1  PF3
Model? ; tab .ASCII 4  ,
PF1  7
Command? EXIT ENTER
.
```

8.9.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP RET
.ASSIGN LP LST RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 8–1.)

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 8–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200                   ;MESSAGE THAT RINGS BELL
     21
     22 000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24 000225     127          .ASCII  /WELL DONE./
     25 000237     000          .BYTE   0
     26
     27 000240          AREA:   .BLKW 6                         ;EMT ARGUMENT AREA
     28 000254          BUFFER:                                 ;RCVDC MESSAGE AREA
     29         000000'         .END    START

DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

8.9.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG RET
.
```

8.9.1.4 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 twice pressing CTRL/C:

```
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.

8.9.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

8.9.2.1 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.

8.9.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

8.9.2.3 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
.
```

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

8.10 Performing the System Generation Process

If 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.

8.11 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Chapter 9

Installing RT–11 on a Professional 300 Series Computer

If you are installing RT–11 to run on a Professional 300 series computer, 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 disk.

Your distribution kit contains two copies of the RT–11 operating system on RX50 diskettes. Store one set of the RX50 diskettes in a safe place and do not modify their contents. They provide a master copy of the distributed RT–11 operating system.

Store the second set of diskettes in a safe place after installing software updates and copying their contents onto the nonremoveable disk. The second set provides a backup copy of your updated RT–11 system. Use those diskettes, if it is ever necessary, to create another working system. If those diskettes are ever damaged, use the master copy of the distributed RT–11 operating system to reproduce them.

To install your system, perform the steps listed below and described in Sections 9.1 through 9.10.

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. Create the working system from chosen components.
6. Customize the system.
7. Compress the system disk.
8. Preserve the working system.
9. Test the working system.
10. If appropriate, perform the system generation procedures.
11. Read the *Introduction to RT–11* for more information on using RT–11.

9.1 Bootstrapping the System Diskette

The first procedure you perform when installing RT–11 bootstraps the system diskette.

First, turn off the processor (press the power switch to 0).

In the software distribution, the highest numbered diskette is bootable on Professional 300 series computers. Remove that diskette from the kit, make sure it is write protected (with a write-protect tape over the notch), and insert it in diskette Unit 0 (which has the physical device name DZ0:). Unit 0 is the top slot if your computer sits horizontally on a table top, or the left slot if it sits vertically in a floor stand.

Turn on the processor (perform a hard boot) by pressing the power switch to 1 (on). Wait while the system runs diagnostic tests. If there are no problems, the Digital logo appears on the screen in about 30 seconds.

If a problem occurs, a diagram of the computer appears on your screen. If the problem is in the hardware, the part of the computer that has a problem is highlighted in the diagram. Write down the numbers you see on the screen.

If the top number is 051124, refer to Chapter 7 of the *RT–11 System Message Manual*.

If the top number is 010030 followed underneath by 000401, you have an unformatted hard disk. Software cannot be installed on the hard disk. After displaying the diagnostic picture, the RT–11 system boots from the diskette. Execute the following sequence of commands to format and install the hard disk:

```
.FORMAT DW: RET
.INSTALL DW: RET
.LOAD DW: RET
.FORMAT/VERIFY:ONLY DW: RET
```

The command FORMAT DW: generates the informational message *?FORMAT-I-Install DW*. Attempting to verify a hard disk with FORMAT/VERIFY or FORMAT /VERIFY:ONLY before formatting an unformatted hard disk generates the error message *?FORMAT-F-Invalid device for /VERIFY, or not formatted*.

If your screen shows a number other than those discussed in the preceding paragraphs, refer to your computer Owner's Manual.

If nothing happens when you turn the power on, refer to the computer Owner's Manual for help.

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

```
RT-11XM V05.xx

.TYPE V5USER.TXT

                               RT-11 V5.xx

Installation of RT-11....(followed by V5USER.TXT message)....

.
```

9.2 Copying the System Diskette onto the System Disk

Once you have installed software updates, copy the system diskette onto the nonremovable disk.

You must first use the INITIALIZE command to initialize the nonremovable disk. Use the /BADBLOCKS option with INITIALIZE to cover any bad blocks that may be on your disk (this prevents the system from trying to use them). If the disk contains bad blocks, the *?DUP-W-Bad blocks detected nnnnnn* message appears on the terminal.

```
.INITIALIZE/BADBLOCKS DW0: RET
DW0:/Initialize; Are you sure? Y RET
?DUP-I-No bad blocks detected DW0:
```

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 onto the disk.

```
.COPY/SYSTEM DZ0: DW0: 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 DW0:RT11XM.SYS DW0: RET
.
```

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 0. Turn the computer off (0) and then on (1) to hard boot the system disk. RT–11 responds with the following message if you have successfully bootstrapped the system disk:

```
RT-11XM V05.xx
(Followed by any start-up file commands.)
.
```

9.4 Copying the Distribution Diskettes onto the System Disk

The next operation you perform copies the remaining distribution diskettes onto the system disk.

Insert distribution diskette 1 in Unit 0. Distribution diskette 1 contains some files that are duplicates of files on the diskette you first copied to the hard disk. Because the original files are protected, the copy operation does not copy the duplicates. Now, copy all nonduplicate files to the system disk by typing the command:

```
.COPY/SYSTEM DZ0: DW0: RET
.
```

Remove that distribution diskette from Unit 0 and store it. Then, 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.

9.5 Creating the Working System from Chosen Components

Once you have chosen your system components (see Section 2.4), you can create the working system by deleting selected files from your system disk. You can delete unwanted files automatically or manually.

To delete unwanted files automatically, run the CONFIG file as follows and obey the prompts:

```
.$@ CONFIG RET
```

You can manually delete unwanted files by using the following command. RT–11 queries you about all the files on the disk, and you choose the files it deletes.

```
.DELETE/SYSTEM/QUERY DW0: RET
  Files deleted:
DW0:aaaaaa.ttt Y RET   (deletes aaaaaa.ttt)
DW0:bbbbbb.ttt N RET   (retains bbbbbb.ttt)
(and so on)
```

Repeat this procedure to retain only the files you planned for the working system disk.

9.6 Customizing the System

You may want to make customizations (described in Section 2.6) to the distributed RT–11 components. At this point, perform the procedures to implement any software customizations. Table 1–2 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.

9.7 Compressing the System Disk

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

If you used CONFIG to remove unwanted files, your system disk was squeezed by CONFIG. If you removed unwanted files manually, use the SQUEEZE command to compress free space. (The volume must be write enabled.)

```
.SQUEEZE DW0: RET
DW0:/Squeeze; Are you sure? Y RET

RT-11xx V05.xx
```

The system automatically reboots when you compress a system disk.

9.8 Preserving the Working System

Once you build a satisfactory working system, Digital recommends that you preserve all the system files on backup diskettes.

Insert a blank RX50 diskette in diskette Unit 0 or 1 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, *n* is the device unit number.

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 DZn: where n is the drive number. You can use diskettes with bad blocks later for temporary storage or as work volumes.

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

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 0 or 1.

Copy all the files in your working system, using the COPY/MULTIVOLUME command below. In the command, *n* is the device unit number. The system copies all the files from the disk (the input volume) that will fit on the diskette (the output volume). When no more files will fit on the diskette, the system prompts you to mount another output volume and prints the *Continue?* message.

Mount another blank, initialized diskette, and type Y to continue. Continue to mount diskettes in this fashion until all the files are copied.

```
.COPY/MULTIVOLUME/SYSTEM DW0: DZn: RET
  Files copied:
DW0:aaaaaa.ttt to DZn:aaaaaa.ttt
DW0:bbbbbb.ttt to DZn:bbbbbb.ttt
(and so on)

Mount next output volume in DZn:; Continue? Y RET
(Log of files copied)
Mount next output volume in DZn:; Continue? Y RET
(Log of files copied)
(and so on)
```

Remove each diskette from Unit 0 or 1 and label it “Backup RT–11 V05 x” (where *x* is the diskette number). Use a soft-tipped pen when you label diskettes. Then, store them in a safe place.

To restore the working system, you can copy the backup working system diskettes to DW.

Any backed-up working system diskette you intend to be bootable must contain certain system utility programs. For example, the highest numbered distribution diskette is bootable, but the rest of the diskettes in your kit are not bootable because they lack the necessary components. A bootable diskette needs an appropriate monitor file, a bootstrap, a system device handler, the SWAP.SYS file, and the interface device handler PIX.SYS.

Then, install the bootstrap on any backup working system diskettes that contain the necessary programs and need to be bootable.

```
.COPY/BOOT DZn:RT11XM DZn: RET
.
```

9.9 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
RT11XM.SYS
PIX.SYS
DWX.SYS (system device handler)
DZX.SYS
LSX.SYS
KEX.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

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
RT11XM.SYS
PIX   .SYS
LSX   .SYS
DZX   .SYS
DWX   .SYS
KEX   .SAV
MACRO .SAV
SYSMAC.SML
LINK  .SAV
PIP   .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If you have shifted output to your terminal and the directory scrolls by too quickly to read, press [CTRL/S] or [HOLD SCREEN] to stop the display and [CTRL/Q] or [HOLD SCREEN] again 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.

9.9.1 Preparing the Background Demonstration Program

9.9.1.1 Edit the Background Demonstration Program

Use the keypad text editor 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, you must remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC). The following commands and keypad strokes open DEMOBG.MAC for editing and remove the semicolon from the file:

```
.EDIT SY:DEMOBG.MAC [RET]
[PF1] [PF3]
Model? ; [tab] .ASCII [4] [,]
[PF1] [7]
Command? EXIT [ENTER]
.
```

9.9.1.2 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 printed listing, first make sure that your configuration has an on-line printer that is ready and issue the following commands:

```
.ASSIGN LS LP RET
.ASSIGN LP LST RET
.
```

If your configuration does not include a printer, use the console terminal:

```
.ASSIGN TT LST RET
.
```

Next, assemble DEMOBG.MAC as follows:

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

(See Figure 9–1.)

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
.
```

9.9.1.3 Link the Background Demonstration Program

Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG RET
.
```

9.9.1.4 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 twice pressing CTRL/C:

```
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
.
```

Figure 9–1: DEMOBG Assembly Listing

```
DEMOBG MACRO V05.05  Sunday 13-Jan-91 13:33  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  BELL:   .BYTE   7,200                    ;MESSAGE THAT RINGS BELL
     21
     22 000112     122  MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
     23 000147     111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
     24 000225     127          .ASCII  /WELL DONE./
     25 000237     000          .BYTE   0
     26
     27 000240          AREA:   .BLKW 6                          ;EMT ARGUMENT AREA
     28 000254          BUFFER:                                  ;RCVDC MESSAGE AREA
     29         000000'         .END    START
DEMOBG  MACRO V05.05  Sunday 13-Jan-91 13:33  Page 1-1
Symbol table

AREA    000240R         MSG     000112R         ...V1 = 000003
BELL    000110R         MSGIN   000044R         ...V2 = 000027
BUFFER  000254R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000254    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


Work  file  reads: 0
Work  file writes: 0
Size of work file: 10912 Words  ( 43 Pages)
Size of core pool: 21760 Words  ( 85 Pages)
Operating  system: RT-11

Elapsed time: 00:00:06.16
DEMOBG,DEMOBG/L:TTM=DEMOBG
```

Then, repeat the editing procedure.

9.9.2 Preparing the Foreground Demonstration Program

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, 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.

To execute the foreground program, assemble the program DEMOFG.MAC, link it for the foreground, and execute it in conjunction with DEMOBG.SAV.

9.9.2.1 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.

9.9.2.2 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 the priority job.

```
.LINK/FOREGROUND DEMOFG RET
.
```

9.9.2.3 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
.
```

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

9.10 Performing the System Generation Process

If 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.

9.11 Further Information

After installing RT–11, read the *Introduction to RT–11*. The *Introduction to RT–11* provides a tutorial interactive explanation of RT–11.

Appendix A

Modifying MDUP to Recognize a Device Handler

The Version 5 MDUP.MM, MDUP.MS, MDUP.MT, and MDUP.MU programs support the following devices:

All MSCP (DU) devices
RK05/RK06
RL01/RL02
Virtual Memory (VM) device

If you need to create MDUP for a disk or magtape not supported by RT–11 (or for which you have written your own handler), use MDUP.SAV as follows.

Use the SYSGEN procedure to create a hardware version of the magtape handler—that is, a version of the handler that does not include the FSM.

Your configuration must include at least 28K words of memory. Apply the following customization to the distributed magtape (RT11MT.SYS) monitor. In the customization, *xx* is the name of the device on which the monitor resides.

```
.R SIPP RET
*xxn:RT11MT.SYS RET
Base?    0 RET
Offset?  1122 RET

    Base      Offset      Old     New?
     000000   001122   000405   240 RET
     000000   001124   013704   12704 RET
     000000   001126   177570   74000 RET
     000000   001130   xxxxxx    CTRL/Y  RET
* CTRL/C
.
```

Then, copy the bootstrap:

```
.COPY/BOOT xxn:RT11MT.SYS xxn: RET
.
```

This procedure causes the system to use only 15K 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    xxxxxx     0  RET
     000000    000200    xxxxxx     0  RET
     000000    000202    xxxxxx      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.

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–10).

B.1 Loading the TMSCP Bootstrap

If your processor does not contain the correct MU boot ROM and you want to boot your processor from a bootable TK50 magtape, you must manually load the TMSCP software bootstrap. You can load the TMSCP software bootstrap, using MICRO–ODT, or the console switch register if your processor has a console switch register. You can boot RT–11 only from TK50 controller unit 0, and you must install that unit at CSR address 774500 and vector address 260.

B.1.1 Loading the TMSCP Bootstrap, Using MICRO–ODT

Deposit the TMSCP bootstrap loader in memory as follows:

1. Place the bootable TK50 magtape in drive unit 0 and power up the drive.
2. Turn on your processor. If it is already on, halt it.
3. At the console, type the first address in Table B–1 (76026) followed by a slash (/):

   ```
   76026/
   ```

   The system responds by printing the contents of address 76026 (represented below by xxxxxx) on the console:

   ```
   76026/ xxxxxx
   ```

4. On the same line, type the first contents value in Table B–1 (5000) followed by a line feed:

   ```
   76026/ xxxxxx 5000 LF
   ```

   The system deposits the value you just typed (5000) and displays the next memory location:

   ```
   76030/
   ```

5. Type the contents of the next memory location followed by a line feed.
6. Repeat step 5 until you have deposited all the contents. Then press RETURN.

7. Finally, enable the processor (remove the halt condition) and type:

   ```
   76026G
   ```

   The processor reads the software bootstrap from the magtape into memory. The system prints the following prompt when it is finished:

   ```
   MSBOOT V05.0x
   *
   ```

   If the system does not print the MSBOOT prompt after you type 76026G, you probably made an error in entering the bootstrap loader. Open and examine each location, using MICRO–ODT, and correct the error.

8. Promptly respond to the asterisk (*) by typing MDUP.MU:

   ```
   *MDUP.MU  RET
   ```

9. Follow the instructions in Chapter 8.

B.1.2 Loading the TMSCP Bootstrap, Using the Switch Register

Deposit the TMSCP bootstrap loader in memory as follows:

1. Place the bootable TK50 magtape in drive unit 0 and power up the drive.
2. Set the ENABLE/HALT switch on the processor to HALT.
3. Set the first address in Table B–1 (076026) in the switch register.
4. Press the LOAD ADDR switch.
5. Set the contents for the first address in Table B–1 (5000) in the switch register.
6. Lift the DEP switch. The processor automatically advances to the next address.
7. Set the contents for the next address in Table B–1 in the switch register.
8. Lift the DEP switch.
9. Repeat steps 7 and 8 until you have deposited all the instructions.

Now, verify that you properly deposited the bootstrap loader.

1. Set the starting address, 076026, 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 contents 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, correct the contents value and lift the DEP switch. Press EXAM again to display the contents. Verify the contents of all addresses in this way.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 076026, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

   The processor reads the software bootstrap from the magtape into memory. The system prints the following prompt when it is finished:

   ```
   MSBOOT V05.0x
   *
   ```

   If the system does not print the MSBOOT prompt after you press the START switch, you probably made an error in entering the bootstrap loader. Repeat the procedure.

5. Promptly respond to the asterisk (*) by typing MDUP.MU:

   ```
   *MDUP.MU  RET
   ```

6. Follow the instructions in Chapter 8.

B.1.3 Addresses and Contents of the TMSCP Bootstrap Loader

The following are the addresses and contents of the TMSCP bootstrap loader:

Table B–1: TMSCP Bootstrap Loader

Address	Contents
076026	005000
076030	012701
076032	174500
076034	012704
076036	076160
076040	012705
076042	004000
076044	010102
076046	012422
076050	005712
076052	100770
076054	031205
076056	001774
076060	012412
076062	006305
076064	100371
076066	010502
076070	005022
076072	105702

Table B–1 (Cont.): TMSCP Bootstrap Loader

Address	Contents
076074	100375
076076	105237
076100	100077
076102	105237
076104	100075
076106	010037
076110	100104
076112	112437
076114	100115
076116	112437
076120	100110
076122	010522
076124	010522
076126	012722
076130	100100
076132	010512
076134	024242
076136	005711
076140	005712
076142	100776
076144	005737
076146	100012
076150	001331
076152	105704
076154	100344
076156	005007
076160	046525
076162	100000
076164	100200
076166	000000
076170	000001
076172	004400
076174	020476
076176	020402

B.2 Loading the MSCP Bootstrap

You can load the MSCP software bootstrap, using MICRO–ODT or the console switch register if your processor has a console switch register.

B.2.1 Loading the MSCP Bootstrap, Using MICRO–ODT

Deposit the MSCP bootstrap loader in memory as follows:

1. Place the bootable MSCP volume in physical unit DU0: (the unit installed at DU CSR0) and power up the drive.
2. Turn on your processor. If it is already on, halt it.

3. At the console, type the first address in Table B–2 (76000) followed by a slash (/):

   ```
   76000/
   ```

   The system responds by printing the contents of address 76000 (represented below by xxxxxx) on the console:

   ```
   76000/ xxxxxx
   ```

4. On the same line, type the first contents value in Table B–2 (5000) followed by a line feed:

   ```
   76000/ xxxxxx 5000 LF
   ```

   The system deposits the value you just typed (5000) and displays the next memory location:

   ```
   76002/
   ```

5. Type the contents of the next memory location followed by a line feed.
6. Repeat step 5 until you have deposited all the contents. Then, press RETURN.
7. Finally, enable the processor (remove the halt condition) and type:

   ```
   76000G
   ```

   The processor reads the software bootstrap from the MSCP volume into memory. When it is finished, the system prints an RT–11 monitor identification message such as the following:

   ```
   RT11FB V05.0x
   ```

   If the system does not print an RT–11 monitor identification after you type 76000G, you probably made an error in entering the bootstrap loader. Open and examine each location, using MICRO–ODT and correct the error.

B.2.2 Loading the MSCP Bootstrap, Using the Switch Register

Deposit the MSCP bootstrap loader in memory as follows:

1. Place the bootable MSCP volume in physical unit DU0: (the unit installed at DU CSR0) and power up the drive.
2. Set the ENABLE/HALT switch on the processor to HALT.
3. Set the first address from Table B–2 (076000) in the switch register.
4. Press the LOAD ADDR switch.
5. Set the contents for the first address in Table B–2 (005000) in the switch register.
6. Lift the DEP switch. The processor automatically advances to the next address.
7. Set the contents for the next address in Table B–2 in the switch register.
8. Lift the DEP switch.
9. Repeat steps 7 and 8 until you have deposited all the instructions.

Now, verify that you properly deposited the bootstrap loader.

1. Set the starting address, 076000, 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 contents 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, correct the contents value and lift the DEP switch. Press EXAM again to display the contents. Verify the contents of all addresses in this way.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 076000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

   The processor reads the software bootstrap from the MSCP volume into memory. When it is finished, the system prints an RT–11 monitor identification message such as the following:

   ```
   RT11FB V05.0x
   ```

   If the system does not print an RT–11 monitor identification after you press the START switch, you probably made an error in entering the bootstrap loader. Repeat the procedure.

B.2.3 Addresses and Contents of the MSCP Bootstrap Loader

The following are the addresses and contents of the MSCP bootstrap loader:

Table B–2: MSCP Bootstrap Loader

Address	Contents
76000	005000
76002	012701
76004	172150
76006	012704
76010	076156
76012	012705
76014	004000
76016	010102
76020	010022

Table B–2 (Cont.): MSCP Bootstrap Loader

Address	Contents
76022	005712
76024	100770
76026	031205
76030	000401
76032	042125
76034	001772
76036	014412
76040	006305
76042	100367
76044	105744
76046	001002
76050	010704
76052	005007
76054	005002
76056	005022
76060	020227
76062	017204
76064	103774
76066	105237
76070	017101
76072	010037
76074	017110
76076	111437
76100	017114
76104	017121
76106	012722
76110	017004
76112	010522
76114	012722
76116	017104
76120	010512
76122	024242
76124	005711
76126	005712
76130	100776
76132	005737
76134	017016
76136	001742
76140	000000
76142	020402
76144	004400
76146	000001
76150	000000
76152	017204
76154	100000

B.3 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–3).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (in 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 (in 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: 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–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 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–4).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–4) 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) 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: 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–4.
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.5 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–5 or B–6).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–5 for TJU16 magtape or from Table B–6 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–5 or 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–5: TJU16 Bootstrap Loader

Address	Contents
001000	012700
001002	172440
001004	012710
001006	000021

Table B–5 (Cont.): TJU16 Bootstrap Loader

Address	Contents
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

Table B–6: 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

Table B–6 (Cont.): TM11 Bootstrap Loader

Address	Contents
010034	100777
010036	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–5 or 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. 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.6 Loading the RK06/RK07 DECpack Bootstrap

Deposit the basic RK06 or RK07 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: RK06/RK07 Bootstrap Loader

Address	Contents RK06	Contents RK07
001000	012701	012701
001002	177440	177440
001004	012711	012711
001006	000003	002003
001010	032711	032711
001012	100200	100200
001014	001775	001775
001016	012761	012761
001020	177400	177400
001022	000002	000002
001024	012711	012711
001026	000021	002021
001030	032711	032711
001032	100200	100200
001034	001775	001775
001036	005007	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 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–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: 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–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, 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.8 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–9).
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (from Table B–9) 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–9) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Table B–9: 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

Table B–9 (Cont.): RX211 Bootstrap Loader

Address	Contents
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–9.
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.

B.9 Loading the TSV05 Bootstrap, Using MICRO–ODT

Deposit the TSV05 bootstrap loader in memory as follows:

1. Turn on your processor. If it is already on, halt it.
2. At the console, type the first address in Table B–10 (7776) followed by a slash (/):

   ```
   7776/
   ```

   The system responds by printing the contents of address 7776 (represented below by xxxxxx) on the console:

   ```
   7776/ xxxxxx
   ```

3. On the same line, type the first contents value in Table B–10 (46523) followed by a line feed.

   ```
   7776/ xxxxxx 46523 LF
   ```

   The system deposits the value you type and displays the next memory location.

   ```
   10000/
   ```

4. Type the contents for the next memory location (Table B–10) followed by a line feed.
5. Repeat step 4 until you have deposited all the instructions. Then, press RETURN.

6. Finally, enable the processor (remove the halt condition) and type:

   ```
   10000G
   ```

   The processor reads the software bootstrap from the magtape into memory. The system prints the following prompt when it is finished:

   ```
   MSBOOT V05.xx
   *
   ```

7. Respond to the asterisk (*) as described in Chapter 8.

Table B–10: TSV05 Bootstrap Loader

Address	Contents
007776	046523
010000	012701
010002	172522
010004	010102
010006	005000
010010	105711
010012	100376
010014	010704
010016	112737
010020	000200
010022	172523
010024	005242
010026	105711
010030	100376
010032	005711
010034	100761
010036	005007

B.10 Loading the TSV05 Bootstrap, Using the Switch Register

Deposit the TSV05 bootstrap loader in memory as follows:

1. Set the ENABLE/HALT switch to HALT.
2. Set the first address in Table B–10, 7776, in the switch register.
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (046523 from Table B–10) 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–10) in the switch register.
7. Lift the DEP switch.
8. Repeat steps 6 and 7 until you have deposited all the instructions.

Now, verify that you deposited the bootstrap loader properly.

1. Set the starting address, 007776, 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 contents value for that address in Table B–10.
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 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, 010000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.