Documentation Conventions

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

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

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

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

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

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

   Letter O: O

   Number 0: 0

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

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

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

2.3.3 System Device Handler

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

2.3.4 Other Device Handlers

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

2.3.5 Default System Library

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

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

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

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

Unless you want to change the help text, HELP.SAV is the only file you need.

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

2.3.8 Line Printer Handlers

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

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

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

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

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

2.3.9 MACRO–11 Assembler

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

2.3.10 Line Printer Utilities

RT–11 contains two utilities, QUEUE and SPOOL, that are used to send output to the printer. Both utilities run with the FB and XM monitors only, and can be used as foreground or system jobs. You must perform the system generation process to generate support for system jobs under the FB monitor.

To use QUEUE, you need the files QUEUE.REL (which queues and prints the files you specify) and QUEMAN.SAV (which processes command lines and sends the information to QUEUE.REL). When you run QUEUE, it

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

```
.ASSIGN xx1: DK:RET
```

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

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

2.4.2 Creating a Separate Utilities Volume

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

NOTE

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

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

```
.RUN xx1:aaaaaaRET
*
```

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

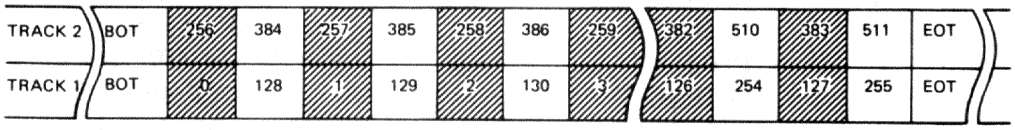
```
*CTRL/C
.
```

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

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

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

Figure 2–1: Block Locations on DECtape II



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

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

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

NOTE

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

2.4.4 Creating Several System Volumes

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

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

If you intend to perform the exercises in *Introduction to RT–11*, you need certain components on your working system. If your system device is one of the small devices (RX01, RX02 or RX50), you need to build four volumes, which are listed below with the files they contain. Be sure to copy the bootstrap to each volume.

your going through the system generation process. Others require less time-consuming procedures, such as modifying the distributed monitors. Compare Tables 1–3 and 1–4, and read the following sections to establish what you need to do. Identify any non-system-generation customizations you need to make, and record them on the worksheet at the end of this chapter (Figure 2–2). You can perform the procedure for each selected customization during the installation process (Chapters 3 through 11). System generation is necessary only if you need a customization that cannot be achieved with one of the procedures described in the following sections.

NOTE

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

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

```
.RUN SIPP
*filnam.COM=filnam.typ/L
```

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

In order to customize your software, you must:

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

NOTE

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

2.7.1 Changing Characters That Indicate Insertion/Deletion

You can modify SRCCOM to change the default characters that SRCCOM uses to indicate insertions and deletions on listings. Normally, when you

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

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

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

2.7.2 Changing Default Output Device from Line Printer to Terminal

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

2.7.3 Changing Location of VT11/VS60 Floating Vectors

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

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

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

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

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

2.7.4 Changing the Number of Directory Columns

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

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

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

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

2.7.5 Changing the Default Order of Directory Listings

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

In the following customization, SWS is the value of that symbol from the CUSTOM.TXT file. The variable nnnnnn represents the address of 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       000000      ;Rxxx(RET)
         000000        SWS+2     zzzzzz      (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.6 Changing the Number of /Q Program Sections LINK Allows

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

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

```
Load section: address?
```

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

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

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

2.7.7 Changing the Size of LINK's Library Module List

You can modify LINK to change the default size of LINK's list of library modules. LINK creates a list of 252 (octal) modules to be included from

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

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

NOTE

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

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

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

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

2.7.8 Changing the Size of the QUEUE Work File

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

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

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

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

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

Offset?  QCBLK(RET)

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

2.7.9 Modifying EDIT

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

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

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

```
.RUN SIPP<RET>
*EDIT.SAV<RET>
Base?      0<RET>
Offset?    DSARG<RET>

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

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

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

```
.RUN SIPP<RET>
*EDIT.SAV<RET>
Base?      0<RET>
Offset?    ALTMDE<RET>
```

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

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

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

2.7.10 Extracting the Overlay Handlers from SYSLIB

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.7.11 Installing Other Devices

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

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

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

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

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

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

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

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

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

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

2.7.11.1 CSR and Vector Addresses for Line Printers, DECtape II, and MSCP Disks — With the SET command, you can change the CSR and vector addresses of six devices: line printer, serial printer, DECtape II, RX01 and RX02 diskettes, and MSCP disks. You need to change the addresses if the

controller is installed at nonstandard addresses. When using the SET command, enter the following commands, where dd is the device mnemonic (LP, LS, DD, DX, DY, or DU), aaaaaa is the CSR address, and bbb is the vector address.

```
.SET dd: CSR2=aaaaaa<RET>
.SET dd: VEC2=bbb<RET>

.SET dd: CSR=aaaaaa<RET>
.SET dd: VECTOR=bbb<RET>
```

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

```
.SET DU: CSR3=aaaaaa<RET>
.SET DU: VEC3=bbb<RET>

.SET DU: CSR4=aaaaaa<RET>
.SET DU: VEC4=bbb<RET>
```

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

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

where n = unit number
x = partition number

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

where n = unit number
x = port number

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

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

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

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

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

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

1. Rename the distributed file-structured handler to save it, rename the corresponding hardware magtape handler to the original name of the file-structured handler, and reboot the system to let the bootstrap install the handler.

2. Use the INSTALL and REMOVE commands:

   - Make sure the hardware magtape handler for your monitor and device is on the system disk.
   - Rename the distributed file-structured handler to save it.
   - Remove the file-structured handler from the system volume.
   - Rename the corresponding hardware magtape handler to the name of the distributed file-structured handler.
   - Install the renamed hardware magtape handler.

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

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

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

2. Rename and install the hardware magtape handler.

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

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

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

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

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

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

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

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

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

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

2.7.13 Modifying LINK to Change the Default SYSLIB Device

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

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

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

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

2.7.14 Modifying the Help Text

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

The files HELP.TXT and HELP.EXE, which together make up the program HELP.SAV, are not provided on the distribution kit. Therefore, if you want 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
```

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 follow a specific format, as follows:

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

2.7.15 Preventing Fatal System Errors from Causing a Reset

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

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

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

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

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

2.7.16 Running RT–11 in Less Memory Than Is Available

If your application requires that RT–11 run in less memory than is available, you can make a customization that allows you to bootstrap the system to run in the lower 12K words or 8K words of a 16K word machine. The BL, SJ, and FB monitors have bootstraps that allow the system to run in less memory than is available. (This cannot be done for XM monitors, because the XM monitor requires 32K memory.) The distributed monitors automatically make use of all available memory, since most applications require that RT–11 do so. However, if your configuration includes a hardware switch register and your application requires less memory, you can make the following customization. In the customization, monitr.SYS is the name of the monitor file that you want to modify, 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     000405   0(RET)
       000000    BHALT+2   013704   (CTRL/Y)(RET)
*(CTRL/C)
.
```

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

2.7.17 Setting VTCOM Default Dial String

Apply the following customization 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, which is dictated by your modem. For example, the first character required in a dial string sent to a DF03 modem is ^B. The next 39(decimal) bytes are reserved for the remainder of the dial string, including other characters your modem may require. You can find the correct value for the symbol ..DIAL in the file CUSTOM.TXT on your distribution kit. Replace the symbols aaa, bbb, and so on, with the characters that make up your dial string. Use a null character in the last byte to terminate the dial string.

To modify VTCOM.REL:

```
.R SIPP(RET)
*VTCOM.REL(RET)
Base?     0(RET)
Offset? ..DIAL(RET)
      Base      Offset       Old    New?
       000000   ..DIAL    000002    \(RET)

      Base      Offset       Old    New?
       000000   ..DIAL       002    (RET)
       000000   ..DIAL+1     xxx    aaa(RET)
       000000   ..DIAL+2     xxx    bbb(RET)
          .         .          .       .
          .         .          .       .
          .         .          .       .
       000000   ..DIAL+51    xxx    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.18 Setting Upper Limit on a File Size

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

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

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

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

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

You can modify the monitor so that it causes the TIME command to base calculations on a 50-cycle clock rate rather than a 60-cycle rate, which is the standard rate in RT–11. The 50-cycle clock has specialized uses and is the common frequency in Europe. To alter the rate, you must modify the monitor so that bit 5 is set in the monitor configuration word. This customization has no effect on Professional 300 series processors.

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

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

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

2.7.20 Using CAPS–11 to Load Files

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

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

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

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

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

2.7.21 Setting Upper Limit on Memory Size

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

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

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

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

2.7.22 Suppressing the Bootstrap Message

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

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

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

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

2.7.23 Suppressing the Start-Up Indirect Command File

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

want to modify, and ..NIND is the value of that symbol from the monitor link map.

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

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

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

2.7.24 Suppressing the Start-Up Indirect Command File Echo

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

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

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

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

2.7.25 Changing the Bootstrap Message

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

2.7.26 Changing the Default Device for Indirect Command Files

If you want to change the default device for indirect command files, you can modify the monitor. Normally, when you invoke an indirect command file (by typing @filnam), the default device where the monitor looks for the

command file is DK:. If you have a special application, you can change this default to any three-character device name.

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

```
.RUN SIPP<RET>
*monitr.SYS<RET>
Base?       0<RET>
Offset?     ..ATDK<RET>

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

2.7.27 Changing the Default File Type for Indirect Command Files

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

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

```
.RUN SIPP<RET>
*monitr.SYS<RET>
Base?       0<RET>
Offset?     ..ATFX<RET>

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

2.7.28 Changing the Default Device for the FRUN Command

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

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

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

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

2.7.29 Changing the Default File Type for the FRUN Command

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

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

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

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

2.7.30 Changing the Default Device for the EDIT Command

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

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

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

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

2.7.31 Changing the Default File Name for the EDIT Command

If you want the monitor to run a program other than the default editor when you type the EDIT command, you can modify the monitor to change the default file name.

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 (octal) 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
$$K52	Command value for default editor K52

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

     Base      Offset        Old    New?
      000000   PROGDF     103225    \(RET)

     Base      Offset        Old    New?
      000000   PROGDF        225    EEE(RET)
      000000   PROGDF+1      206    (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.32 Using Examine and Deposit Above the Background Job

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

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

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

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

2.7.33 Changing the Default Device for QUEMAN

You can change the default device for the queue manager (QUEMAN). Normally, when you use the PRINT or DELETE/ENTRY command (in an

FB or XM system with QUEUE running as a foreground or system job) QUEMAN sends the file to the device LP: or deletes the entry from the LP: queue. If you frequently queue to another device (such as the serial printer, LS:), you can change the default device for these commands.

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

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

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

2.7.34 Changing the Indirect Command File Nesting Depth

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

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

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

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

2.7.35 Changing the Threshold for Resuming Output-Stalled Jobs

You can improve FB or XM system throughput by changing the threshold for resuming output-stalled jobs. In an RT–11 FB or XM system, a job is placed in a stalled state whenever it has terminal output to print but there is no room in its terminal output ring buffer. The system restarts the job when room becomes available in that ring buffer. The system's default mode of operation is to restart the job as soon as a single character of space

is available. If more than one job is running, this mode of operation can cause the system to spend much time swapping in the context of a job, simply to have it output a single character and then stall again.

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

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

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

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

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

2.7.36 Changing the Default Number of Directory Segments

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

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

```
.WORD     1000          ;Volumes with <= 512. blocks
.WORD     1             ;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 mmmmmm.

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

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

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

2.7.37 Changing the Banner Page Width

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

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

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

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

```
Offset?    ..NB2(RET)

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

Offset?    ..NB3(RET)

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

2.7.38 Modifying Listing Page Length in LINK

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

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

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

2.7.39 Assigning the HELP File

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

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

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

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

2.7.40 Changing the Device from Which IND.SAV Is Run

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

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

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

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

2.7.41 Supporting Bad Block Replacement in User-Written Handlers

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

Customization A

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

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

Customization B

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

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

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

2.7.42 Supporting User-Written Magtape Handlers

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

Customization 1

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

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

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

Customization 2

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

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

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

Customization 3

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

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

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

2.7.43 Replacing Magtape Bootstrap in DISMT1.COM

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

2.7.44 Changing Default File Type of Logical Disk Files

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

Customization 1

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

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

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

Customization 2

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

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

2.7.45 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 line 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      xxxxxx      240(RET)
         000000   ..DOFF+2    xxxxxx      240(RET)
         000000   ..DOFF+4    xxxxxx      (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.46 Changing Listing Page Length in MACRO and CREF

The default listing page size for MACRO and CREF, 60 lines long, is suitable for line printers that use standard size line printer paper (10.5 inches long). If you use line printer paper that is not standard size, or if your configuration does not include a line 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     000074      nnn(RET)
000000   000000     PGSIZE+2   003467      (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     000074      nnn(RET)
         000000   PGSIZE+2   003467      (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.47 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       000000       1(RET)
           000000    ..INDR+2     001403       (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.

2.7.48 Limiting Amount of Memory KEX Requests

KEX, by default, requests all available 16-bit memory up to 32KW. 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       xxxxxx       nnnnnn(RET)
           000000    ..MAXM+2     xxxxxx       (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.49 Changing MACRO's Default .LIST/.NLIST Options

MACRO–11/RT 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
LD = 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       103040      nnnnnn(RET)
           000000     LCBITS+2     xxxxxx      (CTRL/Y)(RET)

*(CTRL/C)
.
```

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

MACRO–11/RT 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  = 4
CRF  = 10
DBG  = 20
FPT  = 40
GBL  = 100
 LC  = 200
LCM  = 400
LSB  = 1000
MCL  = 2000
PNC  = 4000
REG  = 10000
```

Note that the flag must always be set.

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

```
          Base      Offset     Old        New?
           000000    EDBITS     003467     nnnnnn(RET)
           000000    EDBITS+2   xxxxxx     (CTRL/Y)(RET)
*(CTRL/C)
.
```

2.7.51 Modifying KED Default File Type

KED now 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:

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

          Base      Offset     Old        New?
           000000    bbbbbb     xxxxxx     ;R(RET)
           000000    bbbbbb     <xxx>      ;Rccc(RET)
           000000    bbbbbb+2   <xxx>      (CTRL/Y)(RET)
*(CTRL/C)
.
```

In this customization, substitute KED, K52, 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.

2.7.52 Changing SPOOL's Work File Size

SPOOL allocates by default 1024(decimal) blocks on 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. xxxxxx is a number that varies; it is not important for you to know this number.

```
.RUN SIPP(RET)
*SPOOL.REL(RET)
Base?      0(RET)
Offset?    ..SPSZ(RET)

         Base      Offset        Old       New?
          000000   ..SPSZ      xxxxxx      nnnnnn(RET)
          000000   ..SPSZ+2    xxxxxx      (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.

2.7.53 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. xxxxxx represents a number that varies but is not important for you to know.

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

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

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

The following customization patch causes 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      000003    000006(RET)
000001      000000   ..WDSZ+2    xxxxxx    (CTRL/Y)(RET)
*(CTRL/C)
.
```

can use the procedure in Section 2.4.3 when you build your working system but not when you back up the distribution volumes. The procedure in Section 2.4.3 creates dummy bad blocks on cartridges to improve response time.

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

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

```
.
```

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

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

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

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

NOTE

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

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

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

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

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

Then, type:

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

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

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

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

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

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

```
YRET
.
```

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

Again type the command:

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

Insert another initialized blank volume in Unit 1.

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

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

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

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

```
YRET
.
```

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

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

Use the hardware bootstrap to boot backup volume 1.

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

NOTE

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

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

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

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

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

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

3.3 Installing Software Updates

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

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

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

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

3.4 Creating the Working System from Chosen Components

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

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

NOTE

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

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

```
.COPY xx0:filnam.typ xx1:filnam.typ
.
```

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

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

To copy files from nonbootable volumes, alternate volumes.

Use the SET command to set the USR to NOSWAP.

```
.SET USR NOSWAPRET
.
```

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

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

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

```
YRET

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

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

```
YRET

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

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

```
YRET
.
```

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)
.
```

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

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

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

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

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

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

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

Store the updated backup volumes for future updating purposes.

3.6 Customizing the System

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

NOTE

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

3.7 Compressing Each Volume

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

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

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

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

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

The system automatically reboots when you compress a system volume.

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

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

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

NOTE

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

3.8 Preserving the Working System

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

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

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

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

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

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

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

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

3.9 Testing the Working System

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

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

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

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

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

NOTE

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

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

3.9.1 Preparing the Background Demonstration Program

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

```
.EDIT SY:DEMOBG.MACRET
*F;TAB,ASCIIESCESC
*OADESCESC
*EXESCESC
.
```

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

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

NOTE

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

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

Assemble DEMOBG.MAC as follows:

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

(See Figure 3–2.)

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

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

Figure 3–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1

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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

.
```

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

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

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

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

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

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

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

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

Then, repeat the editing procedure.

3.9.2 Preparing the Foreground Demonstration Program

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

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

3.9.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source

file; it must be assembled and linked before you can use it. Assemble DEMOFG.MAC as follows:

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

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

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

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

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

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

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

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

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

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

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

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

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

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

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

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

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

(The bell stops ringing.)

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

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

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

3.10 Performing the System Generation Process

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

```
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.

4.2 Copying the Distribution Volumes to the Disk

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

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

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

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

RK05 disk

```
.FORMAT RK0:RET
RK0:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete


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

Other type disk

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

or

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

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

```
.
```

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

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

NOTE

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

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

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

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

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

NOTE

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

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

```
.COPY/BOOT xx0:RT11FB.SYS xx0:RET
.SQUEEZE xx0:RET
xx0:/Squeeze? Are you sure? YRET
.
```

4.3 Preserving the Distribution Volumes

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

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

NOTE

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

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

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

4.4 Installing Software Updates

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

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

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

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

4.5 Creating the Working System from Chosen Components

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

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

RK05 disk

```
.FORMAT RK1:RET
RK1:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

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

Other type disk

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

or

```
.INITIALIZE/REPLACE xx1:RET
xx1:/Initialize; Are you sure? YRET
?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? YRET        (to include a specific file)
xx0:bbbbbb.ttt to xx1:bbbbbb.ttt? NRET        (to exclude a specific file)
(and so on)
```

4.6 Installing the Bootstrap on the Disk

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

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

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

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

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

4.7 Customizing the System

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

NOTE

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

4.8 Compressing the Disk

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

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

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

The system automatically reboots when you compress a system disk.

4.9 Preserving the Working System

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

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

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

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

4.9.1 Backing Up the System on RX01 Diskettes

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

Copy all the files in your working system, using the COPY/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
```
(Log of files copied)
```
Mount next output volume in DX1:;  Continue? Y
```
(Log of files copied)
```
Mount next output volume in DX1:;  Continue? Y
```
(Log of files copied)
```
Mount next output volume in DX1:;  Continue? Y
```
(Log of files copied)
```
Mount next output volume in DX1:;  Continue? Y
```
(and so on)

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

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

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

4.9.2 Backing Up the System on Another Disk

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

RK05 disk

```
.FORMAT RK1:RET
RK1:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

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

Other type disk

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

or

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

Copy all the files in your working system.

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

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

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

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

4.10 Testing the Working System

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

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

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

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

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:RET
  dd-mmm-yy
SWAP   .SYS
RT11FB.SYS
LP     .SYS
DX     .SYS
EDIT   .SAV
```

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

NOTE

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

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

4.10.1 Preparing the Background Demonstration Program

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

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

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

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

NOTE

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

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

Assemble DEMOBG.MAC as follows:

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

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MACRET
.
```

Figure 4–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1


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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

.
```

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

```
.LINK DEMOBGRET
.
```

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

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

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

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

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

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

Then, repeat the editing procedure.

4.10.2 Preparing the Foreground Demonstration Program

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

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

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

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

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

(The bell stops ringing.)

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

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

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

4.11 Performing the System Generation Process

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

5.1 Bootstrapping the Distribution Disk

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

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

RL02 and RC25 disks are normally installed manually. If you successfully bootstrap an RL02 or an RC25 disk, RT–11 responds with the dialog and installation verification message below. The dialog gives you the choice of installing RT–11 manually or by using the automatic installation procedure.

If you successfully bootstrap an RL01 or an RK05 disk, RT–11 responds with the installation verification message below.

```
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): NORET

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 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.

5.2 Preserving the Distribution Disk

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

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

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

RK05 disk

```
.FORMAT RKn:RET
RKn:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

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

Other type disk

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

or

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

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

```
.
```

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

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

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

NOTE

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

Next, copy the bootstrap to the backup disk.

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

Now halt the processor, remove the distribution disk from Unit 0, and store it. For RC25 devices, use the hardware bootstrap to boot the backup disk on disk Unit 1; for all other devices, mount the backup disk on Unit 0, and use the hardware bootstrap to boot the backup disk.

```
RT-11FB V05.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 System User's Guide* for a description of file protection.) When you copied the files to the backup disk, RT–11

also copied the protection. (Note the P that prints next to the file size in the directory.) For the rest of these procedures, you need to remove the protection from the backup disk. Use the following command:

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

5.3 Installing Software Updates

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

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

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

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

5.4 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3) and have planned the best arrangement of them on disks (Section 2.4), you can create the working system.

If you are using an RC25 disk, you create your working system by deleting selected components from the backup disk. Use the following command to delete unnecessary components:

```
.DELETE/SYSTEM/QUERY DU1:RET
  Files deleted?
DU1:aaaaaa.ttt? YRET                (to delete a specific file)
DU1:bbbbbb:ttt? NRET                (to keep a specific file)
(and so on)
```

For all other disks, you create your working system by copying selected components to one or more disks.

Mount a blank disk in another disk unit and initialize it. If your disk is an RK05, format the disk before you initialize it.

RK05 disk

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

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

Other type disk

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

or

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

Copy the files you have selected from the backup disk to the disk that will become your working system disk. If you use the following command, RT–11 queries you about all the files on the backup disk, and you can choose the files it copies. In the command, yyn is the device name and unit number of your backup disk, and xxn is the device name and unit number of the disk to be your working system disk.

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

If you wish to create multiple working system disks, mount one blank disk for each working system you wish to create, format blank RK05 disks, and initialize the disks. Then, use the preceding COPY command to copy selected files to the appropriate disks.

5.5 Installing the Bootstrap on the Disk

Once you have created your working system disk(s), you need to install the bootstrap on all disks, except RC25. In the following command, aa is BL, SJ, FB, or XM.

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

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

Remove the backup disk from Unit 0 and store it for future updates. Remove the working system disk from Unit n and mount it on Unit 0.

Use the hardware bootstrap to boot the new working system disk.

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

5.6 Customizing the System

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

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 5.10.)

5.7 Compressing the Disk

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

Continue to run RT–11 from Unit 0 (Unit 1 for RC25 disk), and use the SQUEEZE command to compress free space. (The disk must be write-enabled.) The squeeze operation does not move files with the .BAD file type.

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

The system automatically reboots when you compress a system disk.

5.8 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you preserve it on the backup medium of your choice. The following sections describe backing up on disk. If you want to back up the system on another device, the procedure may appear elsewhere in this manual; refer to the table of contents.

Insert a blank disk (write-enabled) in another disk unit and initialize the blank disk. If your disk is an RK05, format the disk before initializing it. In the following commands, xxn is the device name and unit number for the backup working system disk.

RK05 disk

```
.FORMAT RKn:RET
RKn:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

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

Other type disk

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

or

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

Copy all the files in your working system. In the following command, yyn is the device name and unit number for your working system disk, and xxn is the device name and unit number for the backup working system disk.

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

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

```
.COPY/BOOT xxn:RT11aa.SYS xxn:RET
.
```

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

5.9 Testing the Working System

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

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

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

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

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

NOTE

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

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

5.9.1 Preparing the Background Demonstration Program

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

```
.EDIT SY:DEMOBG.MAC<RET>
*F;<TAB>.ASCII<ESC><ESC>
*OAD<ESC><ESC>
*EX<ESC><ESC>
.
```

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

```
.ASSIGN LP: LST:<RET>
.
```

NOTE

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

```
.ASSIGN TT: LST:<RET>
.
```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG<RET>
```
(See Figure 5–3.)

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC<RET>
.
```

Figure 5–3: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1


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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

.
```

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

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

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

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

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

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

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC<RET>
.
```

Then, repeat the editing procedure.

5.9.2 Preparing the Foreground Demonstration Program

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

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

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

```
.MACRO/LIST:LST: DEMOFG<RET>
.
```

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

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

```
.LINK/FOREGROUND DEMOFG<RET>
.
```

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

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

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

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

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

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

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

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

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

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

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

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

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

(The bell stops ringing.)

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

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

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

5.10 Performing the System Generation Process

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

```
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): NORET

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 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.

6.2 Preserving the Distribution Disk

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

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

If the disk is another type, initialize the disk and replace bad blocks (using the command INITIALIZE/REPLACE) or cover bad blocks (using the command INITIALIZE/BADBLOCKS). You have a choice of replacing or covering bad blocks if your disk is an RL01, RL02, RK06, or RK07. (Refer to the *RT–11 System User's Guide.*) If your disk is another type, for example, RC25, use INITIALIZE/BADBLOCKS to cover any bad blocks. If the disk contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal, but you can still use the disk. In the commands, xx is the permanent device name, and n is the unit number for your disk.

RK05 disk

```
.FORMAT RKn:RET
RKn:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

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

Other type disk

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

or

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

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

```
.
```

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

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

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

NOTE

Both the SQUEEZE command with the /OUTPUT:xxn: option and the COPY command transfer files from one device to another. SQUEEZE/OUTPUT:xxn: consolidates free space on the device at the same time. The procedures in this manual

use the command that is most appropriate and most efficient for an individual operation. For a better understanding of all RT–11 keyboard monitor commands, refer to the *RT–11 System User's Guide*.

Next, copy the bootstrap to the backup disk.

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

Now halt the processor, remove the distribution disk from Unit 0, and store it. For RC25 devices, use the hardware bootstrap to boot the backup disk on disk Unit 1. For all other devices, mount the backup disk on Unit 0, and use the hardware bootstrap to boot the backup disk.

```
RT-11FB V05.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 System User's Guide* for a description of file protection.) When you copied the files to the backup disk, RT–11 also copied the protection (note the P that prints next to the file size in the directory). For the rest of these procedures, you need to remove the protection from the backup disk. Use the following command:

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

6.3 Installing Software Updates

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

Updated software modules are distributed in update kits. Each kit contains complete software module replacements. Software corrections will already have been installed in the new modules and fully tested. Your only require-

ment is to copy the new modules from the distribution medium to your system, using the procedures described in the *RT–11 Update User's Guide*. Updates are distributed on the same medium on which you received the base system and contain modules for both base system and layered products.

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

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

6.4 Creating the Working System from Chosen Components

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

6.4.1 Initializing RX50 Diskettes

If you wish to create your working system on RX50 diskettes, insert a blank RX50 diskette in Unit 1 or 2 so you can initialize it. You can use the INITIALIZE command with the /BADBLOCKS option to initialize the diskette and to detect any bad blocks that may be on it. If the diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal. In the command, x is the device unit number.

NOTE

RX50 diskettes are not formattable. DIGITAL recommends that you use only diskettes that have no bad blocks. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DUn:. If bad blocks exist on a diskette, copy the contents of the diskette to an error-free diskette, and dispose of the diskette with bad blocks.

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

There may be a delay of up to 1 minute while the system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Now, remove the newly initialized diskette and initialize an adequate number of blank diskettes, leaving one initialized, blank, write-enabled (write-protect notch uncovered) diskette inserted in Unit 1 or 2.

6.4.2 Formatting and Initializing RX01/RX02 Diskettes

If you plan to create your working system on RX01 diskettes, initialize a number of blank RX01 diskettes. For RX02 diskettes, format and initialize the diskettes. Insert a blank diskette in Unit 1. Diskettes are available in single-density but not double-density format. Therefore, to create RX02 diskettes, you must format all your blank diskettes as double-density diskettes. You can use the FORMAT utility program to format the blank diskette and the INITIALIZE command to initialize it. Use the /BADBLOCKS option with INITIALIZE to cover any bad blocks that may be on your diskette. If the diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal.

NOTE

DIGITAL recommends that you use only diskettes that do not have bad blocks when you build a working system. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DYn:. You can use diskettes with bad blocks later, for working or data volumes.

For RX02 diskettes:

```
.FORMAT DY1:(RET)
DY1:/FORMAT-Are you sure? Y(RET)
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DY1:(RET)
DY1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DY1:
```

For RX01 diskettes, just initialize the diskette:

```
.INITIALIZE/BADBLOCKS DX1:(RET)
DX1:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DX1:
```

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

```
.
```

Now, remove the newly formatted and initialized diskette. Repeat the process to create as many initialized blank diskettes as you need for the system that you have planned. Leave one diskette in Unit 1.

6.4.3 Copying the Selected Files onto Diskette

Use the COPY command with the /SYSTEM option to copy selected files from the backup disk to the diskettes that will become your working system diskettes. The /SYSTEM option is required for copying .SYS files only if

wildcards are used in the input file specification. In the command, yyn is the device name and unit number of your backup disk, and xxn is the device name and unit numbers of the diskettes to be your working system diskettes.

```
.COPY yyn:filnam.typ xxn:filnam.typ(RET)
.
```

You can use the following command to avoid typing numerous file specifications. RT–11 queries you about all the files on the diskette, and you choose the files it copies.

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

When you have copied all the files you planned for the working system diskette, label the diskette "RT–11 Working System V05 x/y" (where x is the diskette number, and y is the number of diskettes in your working system).

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

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

Insert the diskette on which you need to install the bootstrap. In the command, xxn represents the device name and unit number of your diskette, and aa is BL, SJ, FB, or XM.

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

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

Then, insert working system diskette 1 in Unit 0, and boot your working system. Use the hardware bootstrap if possible. If not, type the following command; xxn represents the diskette to boot.

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

Store your updated backup distribution disk for future updates.

6.6 Customizing the System

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

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 6.10.)

6.7 Compressing Each Diskette

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

Continue to run RT–11 from Unit 0, and use the SQUEEZE command to compress free space. The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE xx0:RET
xx0:/Squeeze; Are you sure? YRET
```

There is a delay while the squeeze operation takes place.

```
RT-11aa V05.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? YRET
.
```

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

NOTE

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

6.8 Preserving the Working System

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

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

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

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

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

Next, copy the working system to backup diskettes. Insert a blank diskette in Unit 1 with RT–11 still booted from Unit 0. Format (if you are using RX02 diskettes) and initialize the appropriate number of diskettes.

Copy all the files in your working system. You can use the SQUEEZE/OUTPUT:xx1: xx0: command to copy any bootable diskettes. Remember that you must use SQUEEZE/WAIT/OUTPUT:xx1: xx0: and change diskettes to copy the diskettes that are not bootable. (Refer to Section 3.4.) Also remember to copy the bootstrap to any diskettes that need to be bootable.

Store the backup working system diskettes. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

6.9 Testing the Working System

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

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

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

Insert a blank diskette in Unit 1. If you are using RX02 diskettes, format the disk first.

```
.FORMAT DY1:RET
DY1:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete
```

Initialize all diskettes, and check for bad blocks.

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

.ASSIGN xx1: DK:RET
```

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

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

NOTE

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

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

6.9.1 Preparing the Background Demonstration Program

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

```
.EDIT SY:DEMOBG.MACRET
*F;TAB,ASCIIESCESC
*0ADESCESC
*EXESCESC
.
```

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

```
.ASSIGN LP: LST:RET
.
```

NOTE

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

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

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOBGRET
```

(See Figure 6–2.)

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MACRET
.
```

Figure 6–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1

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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

.
```

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

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

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

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

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

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

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

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

Then, repeat the editing procedure.

6.9.2 Preparing the Foreground Demonstration Program

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

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

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

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

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

(The bell stops ringing.)

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

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

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

6.10 Performing the System Generation Process

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

```
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.

Next, remove the protection from all the files on the distribution diskettes. The files on the distribution diskettes have been protected to prevent you from accidentally deleting them. (Refer to the *RT–11 System User's Guide* for a description of file protection.) (Note the P that prints next to the file size in the directory.) For the rest of these procedures, you need to remove the protection from these diskettes. Type the following command to remove it from the files on distribution diskette 1:

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

Insert distribution diskette 2 in Unit 1 and type the following command:

```
.UNPROTECT/SYSTEM DY1:*.*RET
 Files unprotected:
DY1:aaaaaa.ttt
DY1:bbbbbb.ttt
DY1:cccccc.ttt
DY1:dddddd.ttt
      .
      .
      .
DY1:zzzzzz.ttt
.
```

```
.FORMAT DY1:RET
DY1:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DY1:RET
DY1:/Initialize; Are you sure? YRET
?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.typRET
.
```

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? YRET          (to include a specific file)
DY0:bbbbbb.ttt to DY1:bbbbbb.ttt? NRET          (to exclude a specific file)
(and so on)
```

To copy files from nonbootable diskettes, you have to alternate diskettes.

Use the SET command to set the USR to NOSWAP.

```
.SET USR NOSWAPRET
.
```

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.typRET

Mount input volume in DY1:; Continue?
```

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

```
YRET
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)
.
```

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

Once you have created your system, you need to install the bootstrap on any diskettes that must be bootable (that is, that you can use as system diskettes). Generally, any diskette that includes a monitor file and system device handler should be bootable (but do not forget that the diskette would need SWAP.SYS and, for the SJ monitor, TT.SYS).

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

```
.COPY/BOOT DY1:RT11aa.SYS DY1:(RET)
.
```

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

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

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

Store the updated distribution diskettes for future updates.

7.5 Customizing the System

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

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 7.9.)

7.6 Compressing Each Diskette

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

Continue to run RT–11 from Unit 0 and use the SQUEEZE command to compress free space. The squeeze operation does not move files with the .BAD file type.

```
.SQUEEZE DY0:RET
DY0:/Squeeze; Are you sure? YRET
```

There is a delay while the squeeze operation takes place.

```
RT-11xx 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 DY1:RET
DY1:/Squeeze; Are you sure? YRET
.
```

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

NOTE

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

7.7 Preserving the Working System

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

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

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

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

```
.PROTECT/SYSTEM DY1:*.*(RET)
 Files protected:
DY1:aaaaaa.ttt
DY1:bbbbbb.ttt
DY1:cccccc.ttt
DY1:dddddd.ttt
      .
      .
      .
DY1:zzzzzz.ttt
.
```

Next, copy the working system to backup diskettes. Insert each blank diskette in Unit 1 with RT–11 still booted from Unit 0. Format and initialize the appropriate number of diskettes.

Copy all the files in your working system. You can use the SQUEEZE/OUTPUT:DY1: DY0: command to copy any bootable diskettes. Remember that you must use SQUEEZE/WAIT/OUTPUT:DY1: DY0: and change diskettes to copy the diskettes that are not bootable. Also remember to copy the bootstrap to any diskettes that need to be bootable.

Store the backup diskettes. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

7.8 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several

major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

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

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

Insert a blank diskette in Unit 1.

```
.FORMAT DY1:RET
DY1:/FORMAT-Are you sure? YRET
?FORMAT-I-Formatting complete

.INITIALIZE/BADBLOCKS DY1:RET
DY1:/Initialize; Are you sure? YRET
?DUP-I-No bad blocks detected DY1:

.ASSIGN DY1: DK:RET
```

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

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

NOTE

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

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

7.8.1 Preparing the Background Demonstration Program

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

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

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

```
.ASSIGN LP: LST:<RET>
.
```

NOTE

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

```
.ASSIGN TT: LST:<RET>
.
```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOBG<RET>
```

(See Figure 7–2.)

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC<RET>
.
```

Figure 7–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1

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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

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

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

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

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

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

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

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

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

Then, repeat the editing procedure.

7.8.2 Preparing the Foreground Demonstration Program

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

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

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

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

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

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

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

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

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

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.

MDUP.MT. If you have a TJU16, use the magtape build program named MDUP.MM.

```
MDUP.MT RET
```

or

```
MDUP.MM RET
```

The magtape moves as the specified MDUP program is loaded.

```
MDUP V0x.yy
*
```

The next procedure is to build a minimal system on the disk. Start by initializing the system disk. If your disk is an RK05, be sure to use a formatted disk.

You need to initialize the system disk and scan it for bad blocks before you can copy system files to it. You specify these operations to MDUP by entering the device name followed by combinations of the following options.

/Z to initialize the disk

/B to scan it for bad blocks

Mount a formatted disk (write-enabled).

Use the following command, where xx is the permanent device name (RK, DL, DM, DP, or DU) for your disk.

```
xx0:/Z/B RET
```

The system scans the disk for bad blocks and creates a directory.

```
*
```

Now you can build a minimal system on the disk. Depending on the type of tape drive you have, use one of the two following commands (where xx is the permanent device name for your disk).

If the magtape is TM11:

```
xx0:A=MT0: RET
```

If the magtape is TJU16:

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

The tape moves while the system copies the monitor, swap file, system device handler, terminal handler, line printer handler, magtape handler, PIP, DUP, and DIR to the disk. When the files are copied, MDUP boots the minimal system from the disk.

```
RT-11SJ V05.xx
?KMON-F-Command file not found
.
```

You are now running from the minimal system on the disk. This minimal system supports enough file maintenance commands to allow you to complete the building process.

NOTE

MDUP does not support automatic replacement of bad blocks for RK06, RK07, RL01, and RL02 disks. If your disk is an RK06, RK07, RL01, or RL02 and you want automatic bad block replacement, you must initialize a second disk and copy your files to it at a later time.

If you wish to set the date and time, use the DATE command to set the date (where dd–mmm–yy is the day, month, and year in the form 10–JAN–83); use the TIME command to set the time (where hh:mm:ss is the hour, minutes, and seconds).

```
.DATE dd-mmm-yy(RET)
.TIME hh:mm:ss(RET)
.
```

Now copy the rest of the files from distribution magtape 1 to the disk. Use the following command, where xx is MT or MM.

```
.COPY/SYSTEM/NOREPLACE xx0:*.* DK:(RET)
  Files copied:
xx0:aaaaaa.ttt to DK:aaaaaa.ttt
                .
                .
                .
xx0:zzzzzz.ttt to DK:zzzzzz.ttt
?PIP-W-Reboot
.
```

NOTE

You must use the /NOREPLACE option in this command, so that the files you copied to the disk when you built the minimal system will not be copied again. The system prints a message to tell you which files it does not copy (for example, *SWAP.SYS not copied*).

Reboot as advised.

```
.BOOT SY:(RET)
```

8.2 Preserving Both Distribution Magtapes

Copy both distribution magtapes for backup, as a safety measure in case of machine failure or human error. To back up the magtapes, complete the following steps:

1. Replace distribution magtape 1 in the tape drive with a blank magtape.
2. Copy all the distribution files from the disk to the blank backup magtape.
3. Replace the backup magtape with distribution magtape 2.
4. Copy everything from distribution magtape 2 to disk.
5. Replace distribution magtape 2 with a blank magtape.
6. Copy those files that were on distribution magtape 2 from the disk to the blank backup magtape 2.

Step 1

Remove distribution magtape 1 and mount a blank magtape on the tape drive. Leave the write ring in the back of the reel, and make sure that the tape is positioned at the load point.

Step 2

Copy all files from the disk to the magtape by invoking the indirect command file DISMT1.COM (procedure follows). DISMT1.COM initializes the blank magtape, writes the primary bootstrap on it, and copies a duplicate of distribution magtape 1. Before you invoke this indirect command file, use the ASSIGN command to assign the logical name DIS: to your disk device and the logical name TAP: to your tape drive. In the commands, xx is MT or MM, and yy is your disk.

```
.ASSIGN xxn: TAP:RET
.ASSIGN yyn: DIS:RET
.@DISMT1RET
```

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

Step 3

Rewind the newly created backup magtape, remove it, and label it "Backup RT–11 V05 1/2".

Remove the write ring from the back of distribution magtape 2, if necessary, and mount the reel.

Step 4

Copy all files from distribution magtape 2 to the disk. Use the following command, where xx is MT or MM.

```
.COPY/SYSTEM/NOREPLACE xx0:*.* DK:RET
  Files copied:
xx0:aaaaaa.ttt to DK:aaaaaa.ttt
                .
                .
                .
xx0:zzzzzz.ttt to DK:zzzzzz.ttt
?PIP-W-Reboot
.
```

Reboot the disk.

Step 5

Rewind the distribution magtape and remove it. Mount another blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point.

Step 6

Use the indirect command file DISMT2.COM to initialize the magtape and then to duplicate the copies of files from distribution magtape 2 on the newly initialized blank magtape (procedure follows). Note that this tape is not a bootable magtape. Use the ASSIGN command to assign the logical name DIS: to your disk device and the logical name TAP: to your tape drive. In the commands, xx is MT or MM, and yy is your disk.

```
.ASSIGN xxn: TAP:RET
.ASSIGN yyn: DIS:RET
.@DISMT2RET
```

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

Then, rewind the newly created backup magtape, remove it, and label it "Backup RT–11 V05 2/2".

Store the distribution magtapes.

8.3 Installing Software Updates

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

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

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

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

8.4 Creating Updated Master Magtapes

Once you have installed updates in the components you copied to the disk, copy all the files to blank magtapes. These magtapes can serve as updated masters. As other mandatory updates are published, you can copy the af-

fected component from the updated master magtape to the disk, install the update, and copy the file back to the updated master. In this way, you can make sure that you have up-to-date versions of all RT–11 components, even if your working system is destroyed.

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

Use the following procedures to initialize the magtape and copy the same files to updated master magtape 1 as you copied to backup magtape 1 (including the bootstrap) and the same files to updated master magtape 2 as you copied to backup magtape 2. In the commands, xx is MT or MM, and yy is your disk.

```
.ASSIGN xxn: TAP:RET
.ASSIGN yyn: DIS:RET
.@DISMT1RET
```
(The commands in the indirect file print on the terminal.)

Rewind the newly created magtape, remove it, and label it "Updated Master RT–11 V05 1/2". Mount another blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point.

Use the following procedures to initialize the magtape and copy the same files (now updated) to this magtape that you copied to backup magtape 2. In the command, xx is MT or MM, and yyn is your disk.

```
.ASSIGN xxn: TAP:RET
.ASSIGN yyn: DIS:RET
.@DISMT2RET
```
(The commands in the indirect file print on the terminal.)

Then, rewind the newly created magtape, remove it, and label it "Updated Master RT–11 V05 2/2".

Store the two updated master magtapes, and use them when you install any future software updates.

8.5 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3), backed up the distribution magtapes, and created updated masters, you can create the working system by deleting unneeded components from the disk.

The following command queries you about all the files on the disk. Choose the files you want to delete.

```
.DELETE/SYSTEM/QUERY *.*RET
 Files deleted:
DK:aaaaaa.ttt? YRET                    (to delete a specific file)
DK:bbbbbb.ttt? NRET                    (to retain a specific file)
```
(and so on)

8.6 Installing the Bootstrap on the Disk

Once you have created your working system, copy the bootstrap from the monitor file of your choice (RT11BL, RT11SJ, RT11FB, or RT11XM) to the disk. In the command, xx is the permanent device name for your disk, and aa is BL, SJ, FB, or XM.

```
.COPY/BOOT xx0:RT11aa.SYS xx0:RET
.
```

Then, halt the processor, and use the hardware bootstrap to boot the working system disk.

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

NOTE

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

8.7 Customizing the System

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

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 8.11.)

8.8 Compressing the Disk

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

Use the SQUEEZE command for this procedure. (The disk must be write-enabled.) The squeeze operation does not move files with the .BAD file type. In the command, xx is the device name for your disk.

```
.SQUEEZE xxn:RET
xxn:/Squeeze: Are you sure? YRET
RT-11aa V05.xx
```
(Followed by any start-up file commands.)
```
.
```

The system automatically reboots when you compress a system disk.

8.9 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files in it and preserve the system on the backup medium of your choice.

Use the following procedures to back up the disk on magtape. First, protect all the files on the disk.

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

Next, initialize another blank magtape and copy files to it. You may be able to fit the working system on one backup magtape. You can place approximately 3000 blocks on a 600 foot 800 bits/in magtape.

Mount a blank magtape, leaving the write ring in the back of the reel. Make sure that the tape is positioned at the load point. Use the following command, where xx is MT or MM:

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

Then, copy all the files from the disk to the magtape, in the order shown. Essentially, you should use the following procedure whenever you build a bootable magtape. If you do not copy the files in this order, bootstrapping the magtape will be a painfully long process. Keep track of all the files you copy, so that when you copy the rest of the files, you will know which files you have already copied. If you have deleted some device handlers from the system, the log that prints on the terminal will not include the deleted device handlers.

The following COPY procedure shows a log from a system that includes all the device handlers. In the commands, xx is MT, MM, or MS, and aa is BL, SJ, FB, or XM.

```
•COPY MSBOOT.BOT xx0:MSBOOT.BOT/POSITION:-1(RET)

•COPY/SYSTEM MDUP.* xx0:MDUP.*/POSITION:-1(RET)
 Files copied:
DK:MDUP.MM to xx0:MDUP.MM
DK:MDUP.MT to xx0:MDUP.MT
DK:MDUP.MS to xx0:MDUP.MS

•COPY SWAP.SYS xx0:SWAP.SYS/POSITION:-1(RET)

•COPY RT11aa.SYS xx0:RT11aa.SYS/POSITION:-1(RET)
```

```
.COPY TT.SYS xx0:TT.SYS/POSITION:-1(RET)

.COPY/SYSTEM D%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:DT.SYS to xx0:DT.SYS
DK:DP.SYS to xx0:DP.SYS
DK:DX.SYS to xx0:DX.SYS
      .
      .
      .

.COPY/SYSTEM R%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:RK.SYS to xx0:RK.SYS
DK:RF.SYS to xx0:RF.SYS
      .
      .
      .

.COPY/SYSTEM M%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:MM.SYS to xx0:MM.SYS
DK:MT.SYS to xx0:MT.SYS
DK:MS.SYS to xx0:MS.SYS
      .
      .
      .

.COPY/SYSTEM L%%.SYS xx0:*.SYS/POSITION:-1(RET)
 Files copied:
DK:LS.SYS to xx0:LS.SYS
DK:LP.SYS to xx0:LP.SYS
      .
      .
      .

.COPY PIP.SAV xx0:PIP.SAV/POSITION:-1(RET)

.COPY DUP.SAV xx0:DUP.SAV/POSITION:-1(RET)

.COPY DIR.SAV xx0:DIR.SAV/POSITION:-1(RET)

.
```

Copy the rest of the files. The following command queries you about all the files on the disk so that you can choose the files it copies.

```
.COPY/SYSTEM/QUERY DK: xx0:/POSITION:-1(RET)
 Files copied:
DK:aaaaaa.ttt to xx0:aaaaaa.ttt? Y(RET)      (to include a specific file)
DK:zzzzzz.ttt to xx0:zzzzzz.ttt? N(RET)      (to exclude a specific file)
(and so on)
.
```

Rewind the newly created backup working system magtape, remove it, label it "Backup Working System RT–11 V05", and then store it.

8.10 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several

major system components, it does serve as a minimal integrity check. Moreover, DIGITAL considers your system officially installed if the demonstration runs without error.

To execute this demonstration, you need a terminal with a bell, and your working system must include at least the following components.

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

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

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

NOTE

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

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

DEMOBG.MAC, and you must assemble and link the foreground program, DEMOFG.MAC.

8.10.1 Preparing the Background Demonstration Program

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

```
.EDIT SY:DEMOBG.MAC<RET>
*F;<TAB>.ASCII<ESC><ESC>
*OAD<ESC><ESC>
*EX<ESC><ESC>
.
```

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

```
.ASSIGN LP: LST:<RET>
.
```

NOTE

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

```
.ASSIGN TT: LST:<RET>
.
```

Assemble DEMOBG.MAC as follows:

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

(See Figure 8–2.)

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC<RET>
.
```


OLOL LBZ 7/19

Figure 8–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1

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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

.
```

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

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

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

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

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

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

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

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC<RET>
.
```

Then, repeat the editing procedure.

8.10.2 Preparing the Foreground Demonstration Program

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

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

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

```
.MACRO/LIST:LST: DEMOFG<RET>
.
```

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

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

```
.LINK/FOREGROUND DEMOFG<RET>
.
```

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

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

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

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

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

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

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

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

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

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

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

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

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

(The bell stops ringing.)

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

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

```
.INITIALIZE/BADBLOCKS DU0:(RET)
DU0:/Initialize; Are you sure? Y(RET)
?DUP-I-No bad blocks detected DU0:
```

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

```
.
```

Then, copy the contents of the system diskette to the disk.

```
.COPY/SYSTEM/NOQUERY DU1: DU0:(RET)
.
```

9.4 Installing the Bootstrap on the System Disk

Once you have copied the contents of the system diskette onto the system disk, you need to install the bootstrap on the system disk.

```
.COPY/BOOT DU0:RT11FB.SYS DU0:(RET)
.
```

In this command, you identify the device on which the monitor that contains the bootstrap information resides (DU0:), the name of the monitor file (RT11FB), and the device on which you need to install the bootstrap (DU0:). This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same volume. Remove the system diskette from Unit 1, and use the hardware bootstrap to boot the system disk.

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

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

9.5 Copying the Distribution Diskettes onto the System Disk

The next operation you perform with the running RT–11 system is to copy the remaining distribution diskettes onto the system disk, which will serve as your working system disk.

Insert distribution diskette 2 in Unit 1, and copy all its files to the system disk by typing the command:

```
.COPY/SYSTEM/NOQUERY DU1: DU0:(RET)
.
```

Remove the distribution diskette from Unit 1 and store it; mount distribution diskette 3 in Unit 1, and copy all its files to the system disk by typing the command:

```
.COPY/SYSTEM/NOQUERY DU1: DU0:(RET)
.
```

Repeat this procedure to copy the remaining distribution diskettes onto the system disk. When you have completed copying all distribution diskettes, store them in a safe place.

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

Type the following command to remove the protection from all files:

```
.UNPROTECT/SYSTEM DU0:*.*RET
 Files unprotected:
DU0:aaaaaa.ttt
DU0:bbbbbb.ttt
DU0:cccccc.ttt
DU0:dddddd.ttt
            .
            .
            .
DU0:zzzzzz.ttt
.
```

9.6 Creating the Working System from Chosen Components

Once you have chosen your system components (Section 2.3), you can create the working system by deleting selected components from your system disk.

You can use the following command to avoid typing numerous file specifications. RT–11 queries you about all the files on the disk, and you choose the files it deletes.

```
.DELETE/SYSTEM/QUERY DU0:RET
  Files deleted:
DU0:aaaaaa.ttt   YRET          (to delete a specific file)
DU0:bbbbbb.ttt   NRET          (to retain a specific file)
(and so on)
```

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

9.7 Customizing the System

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

NOTE

Later, you can perform the system generation process to implement additional customizations. (See Section 9.11.)

9.8 Compressing the System Disk

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

Use the SQUEEZE command to compress free space. (The volume must be write-enabled.)

```
.SQUEEZE DU0:RET
DU0:/Squeeze; Are you sure? YRET

RT-11xx V05.xx
```

The system automatically reboots when you compress a system disk.

9.9 Preserving the Working System

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

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

```
.PROTECT/SYSTEM DU0:*.*RET
 Files protected:
DU0:aaaaaa.ttt
DU0:bbbbbb.ttt
DU0:cccccc.ttt
      .
      .
      .
DU0:zzzzzz.ttt
.
```

Insert a blank RX50 diskette in Unit 1 or 2 so you can initialize it. You can use the INITIALIZE command with the /BADBLOCKS option to initialize the diskette and to detect any bad blocks that may be on it. If the diskette contains bad blocks, the *?DUP–W–Bad blocks detected nnnnnn* message appears on the terminal. In the command, x is the device unit number.

NOTE

RX50 diskettes are not formattable. DIGITAL recommends that you use only diskettes that have no bad blocks. To ascertain whether an already initialized diskette has bad blocks, use the command DIRECTORY/BAD DUn:, where n is the drive number. If bad blocks exist on a diskette, copy the contents of the diskette to an error-free diskette, and dispose of the diskette with bad blocks.

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

There will be a delay while the system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Now, remove the newly initialized diskette and initialize an adequate number of blank diskettes, leaving one initialized, blank, write-enabled (write-protect notch uncovered) diskette inserted in Unit 1 or 2.

Copy all the files in your working system, using the COPY/MULTI-VOLUME command. In the command, x is the device unit number. 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 DU0: DUx: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
(Log of files copied)
Mount next output volume in DX1:;  Continue? Y
(Log of files copied)
Mount next output volume in DX1:;  Continue? Y
(Log of files copied)
Mount next output volume in DX1:;  Continue? Y
(Log of files copied)
Mount next output volume in DX1:;  Continue? Y
(and so on)
```

Then, install the bootstrap on any backup working system diskettes that need to be bootable. In the following command, aa is BL, SJ, FB, or XM.

```
.COPY/BOOT DUx:RT11aa DUx:RET
.
```

Remove the newly created working system diskette from Unit 1 or 2, and label it "Backup RT–11 V05 x" (where x is the diskette number). (Use a soft-tipped pen when you label diskettes.) Then, store it in a safe place. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

As long as the diskette you intend to copy is bootable and contains certain system utility programs, you can boot RT–11 from that diskette and copy

the diskette. Distribution diskette 0 is bootable, but the rest of the diskettes in your kit are not bootable, because they lack the necessary components. Remember that a bootable diskette needs an appropriate monitor file, a bootstrap, a system device handler, the SWAP.SYS file, and for the SJ monitor, the TT.SYS file.

9.10 Testing the Working System

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

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

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

NOTE

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

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

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:RET
  dd-mmm-yy
SWAP   .SYS
RT11FB.SYS
LP     .SYS
DU     .SYS
EDIT   .SAV
MACRO  .SAV
```

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

NOTE

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

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

9.10.1 Preparing the Background Demonstration Program

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

```
.EDIT SY:DEMOBG.MAC RET
*F;TAB.ASCII ESC ESC
*OAD ESC ESC
*EX ESC ESC
.
```

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

```
.ASSIGN        LP: LST: RET
.
```

NOTE

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

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

Assemble DEMOBG.MAC as follows:

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

(See Figure 9–2.)

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

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

Figure 9–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1

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


DEMOBG  MACRO V05.01b Friday 09-Dec-83 10:16  Page 1-1
Symbol table

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

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

*** Assembler statistics

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

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

.
```

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

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

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

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

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

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

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

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

Then, repeat the editing procedure.

9.10.2 Preparing the Foreground Demonstration Program

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

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

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

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

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

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

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

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

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

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

```
.RUN DEMOBG(RET)
```

(The bell rings quickly several times, then once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

Execute a DIRECTORY command in the background to obtain a directory listing.

```
(CTRL/C)
(CTRL/C)
```

(The bell stops ringing.)

```
.DIRECTORY(RET)
 dd-mmm-yy
```

(The directory of the device DK prints on the terminal.)

```
.
```

Rerun DEMOBG to collect all the foreground messages queued while the directory was printing.

```
.RUN DEMOBG(RET)
```

(The bell rings several times in rapid succession, then rings once every 2 seconds.)

```
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
CTRL/C
CTRL/C
.
```

(The bell stops ringing.)

Now, stop the foreground program and remove it from memory.

```
CTRL/F
F>
CTRL/C
CTRL/C
B>
UNLOAD F RET
.
```

NOTE

If your configuration includes a graphics display, turn it off at this point. Type GT OFF.

If you completed these exercises without error, your system has passed this minimal test and you can consider it successfully installed.

9.11 Performing the System Generation Process

If you have decided that you need RT–11 features that are available only if you generate your own monitor(s) and handlers, perform the system generation process at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation process on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

(this prevents the system from trying to use them). 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 initialized diskette has bad blocks, use the command DIRECTORY/BAD DZn:. You can use diskettes with bad blocks later for temporary storage or as work volumes.

```
.INITIALIZE/BADBLOCKS DZ1:RET
DZ1:/Initialize; Are you sure? YRET
?DUP-I-No bad blocks detected DZ1:
```

The system scans the diskette for bad blocks and creates a new directory. The monitor dot appears when this process is complete.

```
.
```

Repeat this step to create as many initialized blank diskettes as you need for the system you have planned, leaving one initialized, blank diskette in Unit 1.

Then, use the COPY command to copy selected files from distribution diskette 1 to the diskette that will become your working system diskette. The /SYSTEM option is required for copying .SYS files only if wildcards are used in input file specifications.

```
.COPY DZ0:filnam.typ DZ1:filnam.typRET
.
```

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 DZ0: DZ1:RET
  Files copied:
DZ0:aaaaaa.ttt to DZ1:aaaaaa.ttt? YRET   (to include a specific file)
DZ0:bbbbbb.ttt to DZ1:bbbbbb.ttt? NRET   (to exclude a specific file)
(and so on)
```

To copy files from nonbootable diskettes, you have to alternate diskettes.

Type the following command, where filnam.typ is the name of the file you want to copy.

```
.COPY/QUERY/WAIT DZ1:filnam.typ DZ0:filnam.typRET

Mount input volume in DZ1:; Continue?
```

Place the diskette containing the file you want to copy in Unit 1.

```
Y(RET)

  Files copied:
DZ1:aaaaaa.ttt to DZ0:aaaaaa.ttt?
```

Type Y to include or N to exclude the file.

```
Mount output volume in DZ0:; 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 DZ0:; Continue?
```

Replace the diskette in Unit 0 with the system diskette.

```
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.n 1" (where n is the release number of RT–11 V5).

Repeat those procedures to create the other diskettes in the working system.

10.4 Installing the Bootstrap on Diskettes that Must be Bootable

Once you have created your working system, you need to install the bootstrap on any diskettes that must be bootable (that you can use as system diskettes). Generally, any diskette that includes a monitor file and system device handler should be bootable. The diskette also needs SWAP.SYS, and PI(X).SYS to run on a Professional 325.

Insert in Unit 1 the diskette on which you need to install the bootstrap. In the following command, aa is FB or XM.

```
.COPY/BOOT DZ1:RT11aa.SYS DZ1:(RET)
.
```

This command copies bootstrap information from the monitor file to blocks 0 and 2 through 5 of the same diskette. In the command, you identify the device on which the monitor that contains the bootstrap information resides, the name of the monitor file (RT11FB.SYS or RT11XM.SYS), and the device on which you need to install the bootstrap.

Then, insert working system diskette 1 in Unit 0, and use the hardware bootstrap (turn the system off, then on) to boot your working system.

```
RT-11aa V05.xx
```
(Followed by any start-up file commands.)
```
.
```

Store the updated distribution diskettes for future updates.

10.5 Customizing the System

You may want to make customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

The files on the distribution diskettes have been protected to prevent you from accidentally deleting them. (Note the P that prints next to the file size in the directory. Refer to the *RT–11 System User's Guide* for a description of file protection.) You need to remove the protection from the files you want to customize. Type the following command to remove protection from the files on distribution diskette 1 (your system diskette):

```
.UNPROTECT/SYSTEM *.*RET
 Files unprotected:
DK:aaaaaa.ttt
DK:bbbbbb.ttt
DK:cccccc.ttt
DK:dddddd.ttt
      .
      .
      .
DK:zzzzzz.ttt
.
```

To remove protection from files on other diskettes, insert each diskette in Unit 1. Then type the following command:

```
.UNPROTECT/SYSTEM DZ1:*.*RET
 Files unprotected:
DZ1:aaaaaa.ttt
DZ1:bbbbbb.ttt
DZ1:cccccc.ttt
DZ1:dddddd.ttt
      .
      .
      .
DZ1:zzzzzz.ttt
.
```

NOTE

Later, you can perform the system generation procedures to implement additional customizations.

10.6 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you preserve the system on backup diskettes.

If you have unprotected files to make software customizations (Section 10.5), 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 DZ1:*.*RET
 Files protected:
DZ1:aaaaaa.ttt
DZ1:bbbbbb.ttt
DZ1:cccccc.ttt
DZ1:dddddd.ttt
      .
      .
      .
DZ1:zzzzzz.ttt
.
```

Next, copy the working system to backup diskettes. Insert blank diskettes in Unit 1 with RT–11 still booted from Unit 0, and initialize the appropriate number of diskettes.

Then, copy all the files in your working system. To copy the system diskette, type:

```
.SQUEEZE/OUTPUT:DZ1:   DZ0:RET
```

To copy other diskettes, type this command and follow the instructions to change diskettes:

```
.SQUEEZE/WAIT/OUTPUT:DZ1:   DZ0:RET
```

Remember to copy the bootstrap to any diskettes that need to be bootable.

Store the backup diskettes. If you ever need to restore the working system, you can make copies of the backup working system diskettes.

10.7 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 or RT11XM.SYS
DZ.SYS or DZX.SYS (system device handler)
PI.SYS or PIX.SYS
LS.SYS or LSX.SYS
KED.SAV
MACRO.SAV
SYSMAC.SML
LINK.SAV
PIP.SAV
DUP.SAV
DIR.SAV
DEMOBG.MAC
DEMOFG.MAC

Insert a blank diskette in Unit 1.

```
.INITIALIZE/BADBLOCKS DZ1:RET
DZ1:/Initialize; Are you sure? YRET
?DUP-I-No bad blocks detected DZ1:

.ASSIGN DZ1: DK:RET
```

Display the system volume's directory 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 inn the following example.

```
.DIRECTORY/BRIEF/COLUMNS:1 SY:RET
  dd-mmm-yy
SWAP   .SYS
RT11xx.SYS
PI     .SYS
LS     .SYS
DZ     .SYS
KED    .SAV
MACRO  .SAV
SYSMAC.SML
LINK   .SAV
PIP    .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If the directory scrolls by too quickly, press the HOLD SCREEN key to stop the display. Press HOLD SCREEN again to continue scrolling.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and then you must assemble and link the foreground program, DEMOFG.MAC.

10.7.1 Preparing the Background Demonstration Program

To prepare the background demonstration program, DEMOBG.MAC, follow these instructions to edit, assemble, link, and run the program.

10.7.1.1 Edit the Background Demonstration Program — Use a text editor, for example KED, to modify the background demonstration program, DEMOBG.MAC. One of the lines in the program contains a message to be printed, but it is preceded by a semicolon (the symbol for a comment field). You must delete the semicolon so the message will print.

If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

To start KED, type:

```
.KED DEMOBG.MAC
```

Once the file is displayed, press the down-arrow key until the cursor rests on top of the semicolon (;) at the beginning of the line:

```
;       .ASCII     /WELL DONE./
```

Delete the semicolon by pressing the comma key (,) on the keypad.

Then, exit the file: Press the PF1 key, then the 7 key on the keypad. When KED prompts *Command:*, type EXIT and press the ENTER key:

```
Command:  EXIT
```

10.7.1.2 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 listing, make sure your configuration has a line printer that is on-line and ready.

```
.ASSIGN LS: LST:
.
```

NOTE

If your configuration does not include a line printer, use the console terminal.

```
.ASSIGN TT: LST:RET
.
```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOBGRET
```
(See Figure 10–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits as described in Section 10.7.1.1. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MACRET
.
```

Figure 10–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1

                                        .TITLE  DEMOBG
                                        .IDENT  /V05.00/
                                ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
                                ; RING BELL IF FG JOB SENDS A MESSAGE.

                                        .MCALL  .RCVDC,.PRINT

000000                          START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
000034                                  .PRINT  #MSG                    ;PRINT DEMONSTRATION MESSAGE
000042  000777                          BR      .                       ;AND LOOP

                                ;       COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE

000044                          MSGIN:  .PRINT  #BELL                   ;RING BELL IN RESPONSE TO MESSAGE
000052                                  .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
000106  000207                          RETURN                          ;AND RETURN FROM COMPLETION ROUTINE

                                ;       ASCII MESSAGES
                                        .NLIST BEX
000110      007     200         BELL:   .BYTE   7,200                   ;MESSAGE THAT RINGS BELL

000112      122     124     055 MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
000147      111     106     040         .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
                                ;       .ASCII  /WELL DONE./
000225      000                         .BYTE   0

000226                          AREA:   .BLKW 6                         ;EMT ARGUMENT AREA
000242                          BUFFER:                                 ;RCVDC MESSAGE AREA
        000000'                         .END    START


DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R         BUFFER  000242R         MSGIN   000044R         ...V1 = 000003          ...V2 = 000027
BELL    000110R         MSG     000112R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG

.
```

10.7.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

10.7.1.4 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 typing two CTRL/Cs.

```
(CTRL/C) (CTRL/C)
.
```

If you incorrectly edited the file, you can repeat the exercise, although you can continue without correcting the file. 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 (see Section 10.7.1.1).

10.7.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and then execute it in conjunction with DEMOBG.SAV.

DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes those messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

10.7.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source

file which you must assemble and link before running. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: SY:DEMOFG(RET)
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ, which resides on your data volume (DK).

10.7.2.2 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 tells the system that the file is a foreground program and should be run as a priority job.

```
.LINK/FOREGROUND DEMOFG(RET)
.
```

10.7.3 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 the minimal test and you can consider it successfully installed.

10.8 Performing the System Generation Procedures

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 procedures at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation procedures on your particular hardware configuration.

DIGITAL does not recommend that you perform a system generation on a Professional 325. The process is lengthy and is not automated. However, should you decide to perform a system generation, read the *RT–11 System Generation Guide* for guidance.

The *RT–11 Software Dispatch Review* reports serious problems submitted in SPRs and may provide temporary solutions to use until the module containing the errors is corrected by an update kit.

For a complete description of how to update your software, refer to the *RT–11 Update User's Guide.*

11.3 Copying the System Diskette onto the System Disk

Once you have installed software updates, copy the system diskette onto the nonremovable system disk (RD50 or RD51).

You must first use the INITIALIZE command to initialize the nonremovable system 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? YRET
?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/NOPROTECT DZ0: DW0:RET
.
```

11.4 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. In the command, you identify the device (DW0:) on which resides the monitor that contains the bootstrap information, the name of the monitor file (RT11XM), and the device on which you need to install the bootstrap (DW0:).

Remove the system diskette from Unit 0, and use the hardware bootstrap to boot the system disk. RT–11 should respond with the following message if you have successfully bootstrapped the system disk:

```
RT-11XM V05.xx
(Followed by any start-up file commands.)
.
```

11.5 Copying the Distribution Diskettes onto the System Disk

The next operation you perform with the running RT–11 system copies the remaining distribution diskettes onto the system disk that will serve as your working system disk.

Insert distribution diskette 2 in Unit 0, and copy all its files to the system disk by typing the command:

```
.COPY/SYSTEM/NOPROTECT DZ0: DW0:RET
.
```

Remove the 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.

11.6 Creating the Working System from Chosen Components

Once you have chosen your system components (see Section 2.3), you can create the working system by deleting selected components from your system disk.

You can use the following command to avoid typing numerous file specifications. RT–11 queries you about all the files on the disk, and you choose the files it deletes.

```
.DELETE/SYSTEM/QUERY DW0:RET
  Files deleted:
DW0:aaaaaa.ttt   YRET          (to delete a specific file)
DW0:bbbbbb.ttt   NRET          (to retain a specific file)
(and so on)
```

Repeat this procedure to retain only the files you planned for the working system disk.

11.7 Customizing the System

You may want to make customizations (described in Section 2.7) to the distributed RT–11 components. At this point, perform the procedures to implement any software customizations. Table 1–3 summarizes the available customizations and directs you to the section in Chapter 2 that describes a particular customization and the procedure for implementing it.

NOTE

Later, you can perform the system generation procedures to implement additional customizations.

11.8 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.

Use the SQUEEZE command to compress free space. (The volume must be write-enabled.)

```
.SQUEEZE DW0:RET
DW0:/Squeeze; Are you sure? YRET

RT-11xx V05.xx
```

The system automatically reboots when you compress a system disk.

11.9 Preserving the Working System

Once you build a satisfactory working system, DIGITAL recommends that you protect all the files and preserve the system on backup diskettes.

Use the following command to protect all the files on the system disk:

```
.PROTECT/SYSTEM DW0:*.*RET
 Files protected:
DW0:aaaaaa.ttt
DW0:bbbbbb.ttt
DW0:cccccc.ttt
      .
      .
      .
DW0:zzzzzz.ttt
.
```

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.

NOTE

> 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. If bad blocks exist on a diskette, copy the contents of the diskette to an error-free diskette. You can use diskettes with bad blocks later for temporary storage or as work volumes.

```
.INITIALIZE/BADBLOCKS DZn:RET
DZn:/Initialize; Are you sure? YRET
?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/MULTI-VOLUME 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? YRET
(Log of files copied)
Mount next output volume in DZn:;  Continue? YRET
(Log of files copied)
(and so on)
```

Then, install the bootstrap on any backup working system diskettes that need to be bootable. In the following command, aa is FB or XM.

```
.COPY/BOOT DZn:RT11aa DZn:RET
.
```

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.

If you ever need to restore the working system, you can copy the backup working system diskettes to DW:.

As long as the diskette you intend to copy is bootable and contains certain system utility programs, you can boot RT–11 from that diskette and copy the diskette. Distribution diskette 1 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 Professional interface PI(X).SYS.

11.10 Testing the Working System

Once you have built and preserved the working system, you can execute the following demonstration to test that system. This demonstration does not serve as a comprehensive system exercise; however, because it uses several major system components, it does serve as a minimal integrity check. More-

over, 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 or RT11XM.SYS
DW.SYS or DWX.SYS (system device handler)
PI.SYS or PIX.SYS
LS.SYS or LSX.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
LS     .SYS
DW     .SYS
PI     .SYS
KED    .SAV
MACRO  .SAV
SYSMAC.SML
LINK   .SAV
PIP    .SAV
    .
    .
    .
xxx Files, bbb Blocks
fff Free blocks
.
```

NOTE

If the directory scrolls by too quickly, press the HOLD SCREEN key to stop the display. Press HOLD SCREEN again to continue scrolling.

Before you can execute the background and foreground demonstration programs, you must first edit, assemble, and link the background program, DEMOBG.MAC, and then you must assemble and link the foreground program, DEMOFG.MAC.

11.10.1 Preparing the Background Demonstration Program

To prepare the background demonstration program, DEMOBG.MAC, follow these instructions to edit, assemble, link, and run the program.

11.10.1.1 Edit the Background Demonstration Program — Use a text editor, for example, KED.SAV, to modify the background demonstration program, DEMOBG.MAC. One of the lines in the program contains a message to be printed, but it is preceded by a semicolon (the symbol for a comment field). You must delete the semicolon so the message will print.

If DEMOBG.MAC is a protected file, remove the protection before making the edits (UNPROTECT SY:DEMOBG.MAC).

To start KED, type:

```
.KED DEMOBG.MAC<RET>
```

Once the file is displayed, press the down-arrow key until the cursor rests on top of the semicolon (;) at the beginning of the line:

```
;       .ASCII     /WELL DONE./
```

Delete the semicolon by pressing the comma key (,) on the keypad.

Then, exit the file: press the PF1 key, then the 7 key on the keypad. When KED prompts *Command:*, type EXIT and press the ENTER key:

```
Command:  EXIT <ENTER>
```

11.10.1.2 Assemble the Background Demonstration Program — The background program, DEMOBG.MAC, is an assembly language source file; it must be assembled and linked before you can execute it.

To assemble DEMOBG.MAC and obtain a listing, make sure that your configuration has a line printer that is on-line and ready.

```
.ASSIGN LS: LST:<RET>
.
```

NOTE

If your configuration does not include a line printer, use the console terminal.

```
.ASSIGN TT: LST:<RET>
.
```

Assemble DEMOBG.MAC as follows:

```
.MACRO/LIST:LST: DEMOBG<RET>
```

(See Figure 11–2.)

If any errors occur when you assemble DEMOBG.MAC, you have incorrectly edited the file and should repeat the edits as described in Section 11.10.1.1. Use the backup demonstration program.

```
.RENAME SY:DEMOBG.BAK SY:DEMOBG.MAC(RET)
.
```

Figure 11–2: DEMOBG Assembly Listing

```
DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1


     1                                          .TITLE  DEMOBG
     2                                          .IDENT  /V05.00/
     3                                  ; DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
     4                                  ; RING BELL IF FG JOB SENDS A MESSAGE.
     5
     6                                          .MCALL  .RCVDC,.PRINT
     7
     8 000000                           START:: .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESSAGE
     9 000034                                   .PRINT  #MSG                      ;PRINT DEMONSTRATION MESSAGE
    10 000042  000777                           BR      .                         ;AND LOOP
    11
    12                                  ;      COMPLETION ROUTINE ENTERED WHEN FG SENDS MESSAGE
    13
    14 000044                           MSGIN:  .PRINT  #BELL                     ;RING BELL IN RESPONSE TO MESSAGE
    15 000052                                   .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE REQUEST
    16 000106  000207                           RETURN                            ;AND RETURN FROM COMPLETION ROUTINE
    17
    18                                  ;      ASCII MESSAGES
    19                                          .NLIST BEX
    20 000110     007     200           BELL:   .BYTE   7,200                     ;MESSAGE THAT RINGS BELL
    21
    22 000112     122     124     055   MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/<15><12>
    23 000147     111     106     040           .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./<15><12>
    24                                  ;      .ASCII  /WELL DONE./
    25 000225     000                          .BYTE   0
    26
    27 000226                          AREA:   .BLKW 6                           ;EMT ARGUMENT AREA
    28 000242                          BUFFER:                                   ;RCVDC MESSAGE AREA
    29          000000'                        .END    START


DEMOBG  MACRO V05.00 Saturday 08-Jan-83 11:44  Page 1-1
Symbol table

AREA    000226R         BUFFER  000242R         MSGIN   000044R         ...V1 = 000003          ...V2 = 000027
BELL    000110R         MSG     000112R         START   000000RG

. ABS.  000000    000   (RW,I,GBL,ABS,OVR)
        000242    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics

Work  file  reads: 0
Work  file writes: 0
Size of work file: 9188 Words  ( 36 Pages)
Size of core pool: 16128 Words  ( 63 Pages)
Operating  system: RT-11

Elapsed time: 00:00:05.45
DK:DEMOBG,DK:DEMOBG=DK:DEMOBG

.
```

11.10.1.3 Link the Background Demonstration Program — Link the program DEMOBG.OBJ to produce an executable background program, DEMOBG.SAV.

```
.LINK DEMOBG(RET)
.
```

11.10.1.4 Run the Background Demonstration Program — Run the program DEMOBG to check the results of the first exercise.

```
.RUN DEMOBG(RET)
RT-11 DEMONSTRATION PROGRAM
IF INCORRECTLY EDITED, THIS IS THE LAST LINE.
WELL DONE.
```

If you did not delete the semicolon character, the last line will not print. Return to the monitor by typing two CTRL/Cs.

```
(CTRL/C) (CTRL/C)
.
```

If you incorrectly edited the file, you can repeat the exercise, although you can continue without correcting the file. 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 (see Section 11.10.1.1).

11.10.2 Preparing the Foreground Demonstration Program

To execute the foreground program, you must assemble the program DEMOFG.MAC, link it for the foreground, and then execute it in conjunction with DEMOBG.SAV.

DEMOFG.MAC is a small foreground program that sends a message every 2 seconds to DEMOBG (running in the background), telling it to ring the terminal bell. DEMOBG recognizes those messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every 2 seconds, this exercise can execute other programs in the background besides DEMOBG. The circuit is complete and messages are successfully received and honored only when DEMOBG is active. During those periods when DEMOBG is not running, DEMOFG enters the messages in the monitor message queue. Once you restart DEMOBG in the background, the system immediately releases all the messages queued since the last forced exit, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell rings every 2 seconds as each current message is sent and honored.

11.10.2.1 Assemble the Foreground Demonstration Program — The foreground demonstration program, DEMOFG.MAC, is an assembly language source file which you must assemble and link before running. Assemble DEMOFG.MAC as follows:

```
.MACRO/LIST:LST: DEMOFG<RET>
.
```

The output resulting from this MACRO command is an object file called DEMOFG.OBJ which resides on your system volume.

11.10.2.2 Link the Foreground Demonstration Program — You must link the DEMOFG.OBJ file to produce an executable program. Use the /FOREGROUND option to produce the load module DEMOFG.REL. The .REL file type tells the system that the file is a foreground program and should be run as a priority job.

```
.LINK/FOREGROUND DEMOFG<RET>
.
```

11.10.3 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 the minimal test and you can consider it successfully installed.

11.11 Performing the System Generation Procedures

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 procedures at this point. You should have thoroughly studied Chapter 1 to make this decision and to establish that you can perform the system generation procedures on your particular hardware configuration. Read the *RT–11 System Generation Guide* for guidance in planning and performing system generation.

Appendix A
Modifying MDUP to Recognize a Device Handler

The Version 5 MDUP.MM, MDUP.MS, and MDUP.MT programs support the following devices:

RK05	RP02
RK06	RP03
RK07	RS03
RL01	RS04
RL02	RF11

If you need to create MDUP for a disk or magtape not supported by RT–11 (for which you have written your own handler), use MDUP.SAV as follows.

Use the hardware version of the magtape handler and the standard (distributed) single-job monitor. Your configuration must include at least 28K words of memory. Apply the following customization to the monitor. In the customization, xx is the name of the device on which the monitor resides.

```
.R SIPP(RET)
*xxn:RT11SJ.SYS(RET)
Base?      0(RET)
Offset?    1116(RET)

      Base          Offset     Old        New?
      000000        001116     000407     240(RET)
      000000        001120     013704     12704(RET)
      000000        001122     177570     60000(RET)
      000000        001124     042704     (CTRL/Y)(RET)
*(CTRL/C)
.
```

Then copy the bootstrap:

```
.COPY/BOOT xxn:RT11SJ.SYS xxn:(RET)
.
```

This procedure causes the system to use only 12K words of memory when booted.

Next apply the following customization to every handler to be supported by the version of MDUP you are building. In the customization, xx is the name of the device on which the handler resides, and yy is the name of the handler.

```
.R SIPP(RET)
*xxn:yy.SYS(RET)
Base?      0(RET)
Offset?    176(RET)

     Base          Offset     Old        New?
     000000        000176     ??????     0(RET)
     000000        000200     ??????     0(RET)
     000000        000202     ??????     (CTRL/Y)(RET)
*(CTRL/C)
.
```

Now boot the monitor you have modified. Set the USR NOSWAP. Load all the handlers to be supported. Run MDUP.SAV and issue the following command. In the command, xx is the physical device name, and filnam.typ is your version of MDUP (MDUP.??).

```
.R MDUP.SAV(RET)
*xx:filnam.typ=/H(RET)
```

MDUP.SAV creates the file filnam.typ and exits.

If you make a typing error in the command line, use (CTRL/C) to abort MDUP.SAV and run MDUP.SAV again. If you get an insufficient memory error message, you cannot build MDUP with all the handlers loaded. Unload one of the handlers, and run MDUP.SAV again.

Now verify that you deposited the bootstrap loader properly.

1. Set the first address, 002000, in the switch register.
2. Press the LOAD ADDR switch.
3. Press the EXAM switch to display the contents of that address in the data register.
4. Compare the value in the data register with the value for that address in Table B–8.
5. If the values are the same, press EXAM again to display the contents of the next address. If the values are not the same, repeat the entire procedure for depositing the bootstrap. Verify the contents of all the addresses in this way. If any instruction is incorrect, repeat the entire deposit procedure.

Once you have correctly deposited the bootstrap in memory, start the computer as follows:

1. Set the starting address, 002000, in the switch register.
2. Press the LOAD ADDR switch.
3. Set the ENABLE/HALT switch to ENABLE.
4. Press the START switch.

B.8 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–9 (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 from Table B–9 (46523) followed by a line feed.

   ```
   7776/ xxxxxx 46523LF
   ```

   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–9) followed by a line feed.
5. Repeat step 4 until you have deposited all the instructions. Then, type a carriage return.
6. Finally, 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. Wherever MM or MT appears in the instructions, substitute MS.

Table B–9: TSV05 Bootstrap Loader

Address	Contents
007776	046523
100000	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.9 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–9, 7776, in the switch register.
3. Press the LOAD ADDR switch.
4. Set the contents for the first address (46523 from Table B–9) in the switch register.