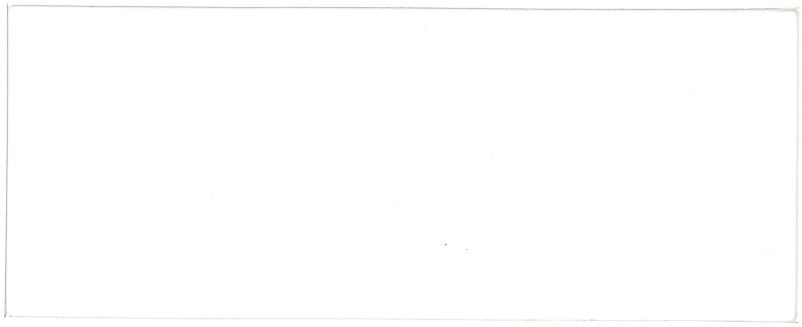
BASIC–11/RT–11 Installation Guide and Release Notes

AA–K724B–TC

digital software



BASIC–11/RT–11 Installation Guide and Release Notes

AA–K724B–TC

March 1983

This document contains detailed instructions for installing BASIC–11 on the RT–11 operating system. This manual also describes recommended practices for running and maintaining a BASIC–11 system.

This manual supersedes the *BASIC–11/RT–11 Installation Guide*, Order No. AA–K724A–TC.

Operating System: RT–11 Version 5.0

Software: BASIC–11/RT–11 Version 2.1

To order additional documents from within DIGITAL, contact the Software Distribution Center, Northboro, Massachusetts 01532.

To order additional documents from outside DIGITAL, refer to the instructions at the back of this document.

digital equipment corporation · maynard, massachusetts

First Printing, March 1981
Revised, March 1983

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

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

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by DIGITAL or its affiliated companies.

Printed in U.S.A.

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

The following are trademarks of Digital Equipment Corporation:

	digital™	
DEC	MASSBUS	UNIBUS
DECmate	PDP	VAX
DECsystem–10	P/OS	VMS
DECSYSTEM–20	Professional	VT
DECUS	Rainbow	Work Processor
DECwriter	RSTS	
DIBOL	RSX	

M22300

CONTENTS

FIGURES

TABLES

PREFACE

How to Use This Manual

This manual tells you how to install and run the BASIC-11 software on the RT-11 operating system. This manual assumes that you have an RT-11 working system volume. If you do not have an RT-11 working system volume, refer to the RT-11 Automatic Installation Booklet or the RT-11 Installation Guide for instructions on creating one.

To use this manual effectively, you should be familiar with the BASIC-11/RT-11 User's Guide. You should also be familiar with the RT-11 System User's Guide and the RT-11 System Utilities Manual, since BASIC-11 is a layered product of the RT-11 operating system.

Part I of this manual describes installation procedures, including both planning and implementation. You should read Chapter 1 to plan for installation, then choose the chapter (2, 3, or 4) that describes the installation procedure for the system most closely resembling your own hardware configuration. Chapter 5 describes the configuration process, which allows you to alter your BASIC-11 working system.

Part II of this manual describes recommended practices for running and maintaining a BASIC-11 system. Chapter 6 tells you about standard operating procedures.

Supplementary information is available in the appendixes. Appendix A describes feature differences between BASIC-11 Version 1B and Version 2.1. Appendix B supplies a sample program for converting from Version 1B to Version 2.1. Appendix C provides supplementary conversion information, and Appendix D contains the BASIC-11/RT-11 kit directory.

Documentation Conventions

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

1. Square brackets [ ] enclose optional items.
2. Braces { } enclose items from which you must choose only one.
3. Uppercase letters represent key words that you must type as shown.
4. Lowercase letters in a command that you must type are variables for which you must supply a value.

5. The lowercase letters yy, as they appear after RT11yy, identify the RT-11 monitor that your system uses. The monitors are SJ (single-job), FB (foreground/background), and XM (extended memory).

6. Ellipses (...) indicate that the immediately preceding item can be repeated.

7. Examples of computer input and output are presented in **bold type.** In these examples, user input appears in red where it must be differentiated from computer output.

8. The symbol <RET> represents a carriage return.

9. In examples, you need to distinguish between the letter O and the number Ø. Examples in this manual represent these characters as follows:

   letter O: O
   number Ø: Ø

Terms Used in This Manual

This manual refers to five types of volumes, defined below:

Distribution volume	The volume containing the software you receive as part of your software kit.
Backup volume	The volume created when you copy the distribution volume onto a blank volume.
Working system volume	The volume containing a working system. For BASIC-11/RT-11, the volume created when you copy both the BASIC-11 backup volume and the RT-11 working system onto one blank volume.
Small volumes	Diskettes: DIGITAL products include single-density diskettes and double-density diskettes. These volumes are discussed in Chapters 2 and 3.
Large volumes	Hard disks: DIGITAL products include RKØ5, RKØ6, RLØ1, RLØ2, RC25, and RD5Ø disks. These volumes are discussed in Chapters 3 and 4.

PART I

BASIC-11/RT-11 INSTALLATION GUIDE

CHAPTER 1

PREPARING FOR INSTALLATION

Before you begin to install your system, you should:

- Survey the general installation procedure.
- Study the contents of your distribution volume.
- Select the files you need on your working system volume.
- Acquire sufficient blank volumes.

1.1 GENERAL INSTALLATION PROCEDURE

Although the specific steps you must perform depend on your configuration, the BASIC-11/RT-11 installation procedure involves the following general steps:

1. Copying the distribution volume (if your distribution volume is a large volume).
2. Creating the working system volume.
3. Testing the working system volume.

When you have completed these steps, your working system volume is installed and ready to run. After installation, you can alter the working system by configuring and linking. This is necessary only if you choose to add new features to your working system or if you alter your hardware configuration following BASIC-11/RT-11 installation. If you alter your hardware, you will also need to perform the RT-11 system generation process, which is described in the RT-11 System Generation Guide.

Figure 1-1 provides a flowchart of BASIC-11/RT-11 installation.

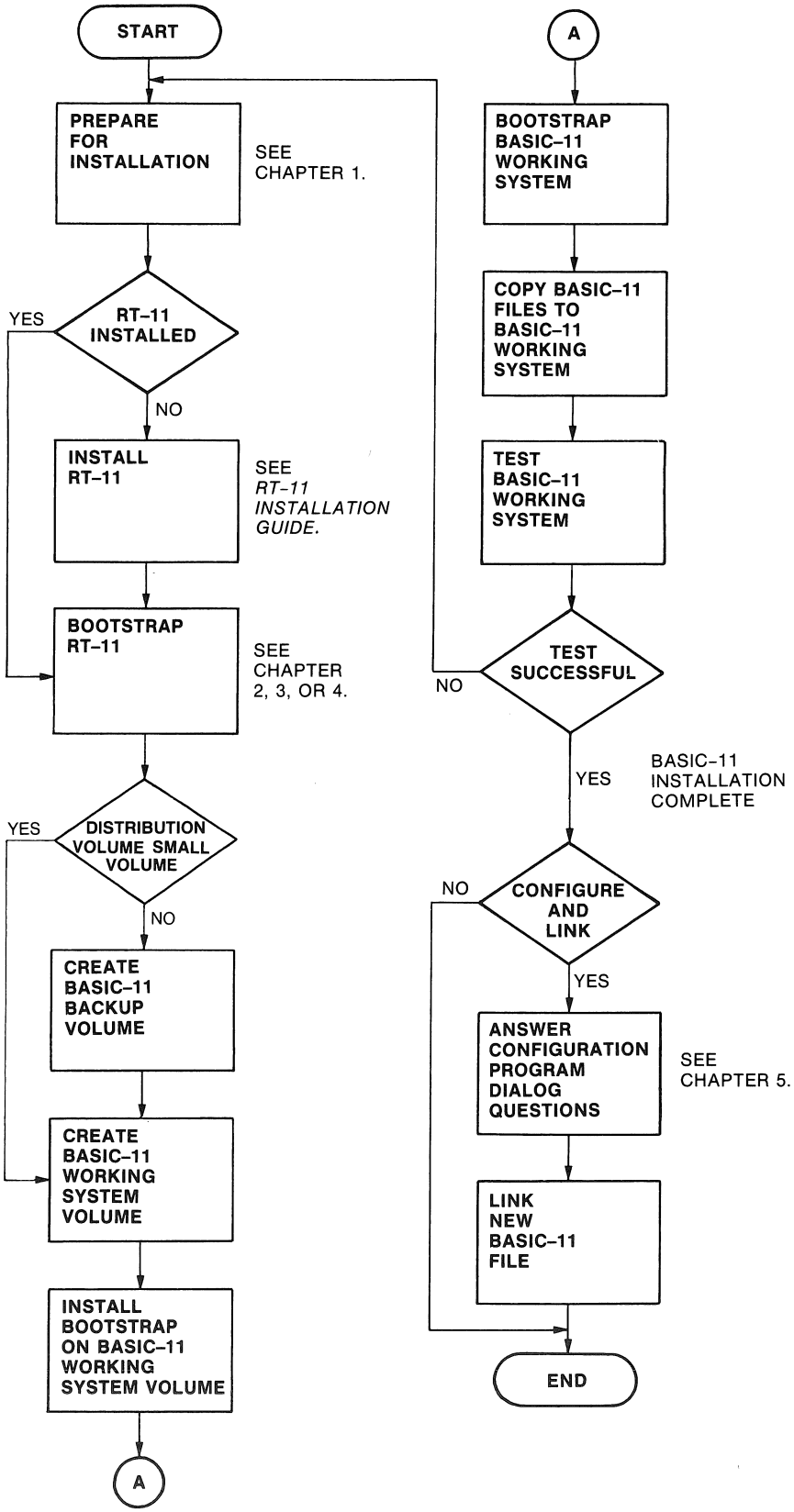


Figure 1-1 BASIC-11/RT-11 Installation Procedure

1.2 CONTENTS OF THE DISTRIBUTION VOLUME

You should become familiar with each file included on the BASIC-11 distribution volume. The files include libraries and the components necessary for building the BASIC-11/RT-11 system. Appendix D lists the files on the distribution volume.

The BASIC-11 files also include three running versions of BASIC:

1. BASIC.SAV -- Includes all optional features of the BASIC-11 language except the CALL statement. It is overlaid (overlay type 3) and does not include support for arithmetic hardware. The base address of BASIC.SAV is 1000.

2. BAS16K.SAV -- Can execute in 16K words of memory. It should be used only for 16K RT-11 systems. It does not include the following features of the BASIC-11 language:

   PRINT USING command
   SUB command
   CALL support
   RESEQ command
   Long error messages

   BAS16K.SAV is overlaid (overlay type 5) and does not include support for arithmetic hardware. Its base address is 700.

3. BASICD.SAV -- Has the same features of the BASIC-11 language as BASIC.SAV. It supports double-precision floating-point arithmetic and has a base address of 1200.

1.3 SELECTING FILES FOR THE WORKING SYSTEM VOLUME

Because space on volumes is limited, you should include only necessary files on your BASIC-11 working system volume. To decide which files these should be, first become familiar with your storage media's space limitations. Then determine how much space is occupied by RT-11 files, which are described in the RT-11 System Release Notes. Appendix D of this manual, the BASIC-11 Installation Guide and Release Notes, lists all BASIC-11 files and their space requirements. You can choose as many files as your volume's space limitations will allow.

1.4 ACQUIRING SUFFICIENT BLANK VOLUMES

BASIC-11 is distributed on the following types of media in the indicated quantities of volumes:

RX01 diskette	2 volumes
RX02 diskette	1 volume
RX50 diskette	1 volume
9-track magtape	1 volume
RC25 disk	1 volume
RK05 disk	1 volume
RK06 disk	1 volume
RL01 disk	1 volume
RL02 disk	1 volume

You will need at least twice as many blank volumes as distribution volumes to back up the BASIC-11 distribution volume and create the BASIC-11 working system volume.

NOTE

If your distribution volume(s) is a diskette, you need only enough blank volumes to create the BASIC-11 working system volume. A backup copy of the BASIC-11 distribution volume(s) is supplied by DIGITAL.

You may need additional volumes to meet your work space and data storage requirements.

DIGITAL recommends that you back up your BASIC-11 working system volume. This would require one additional blank volume. Backing up your working volume can save time and effort in case of damage caused by human error or machine failure.

Before you begin installation, become familiar with your distribution kit and acquire as many blank diskettes, disks, or magtapes as you need.

CHAPTER 2

SYSTEM INSTALLATION: FROM SMALL VOLUMES TO SMALL VOLUMES

Small volumes are diskettes. If BASIC-11 was distributed to you on a small volume and you intend to build a working system to run on a small volume, you should perform the procedures described in this chapter.

Installing your system requires the steps summarized in the following list:

1. Bootstrap the RT-11 working system.
2. Create the BASIC-11 working system volume.
3. Install the bootstrap on the BASIC-11 working system volume.
4. Copy the BASIC-11 files from the backup volume to the working system volume.
5. Bootstrap the BASIC-11 working system.
6. Test the BASIC-11 working system.
7. Protect and copy the BASIC-11 working system.

Sections 2.1 through 2.7 correspond to each of these steps and describe in detail the procedures you must perform.

Figure 2-1 shows the backup volumes you create when you install BASIC-11.

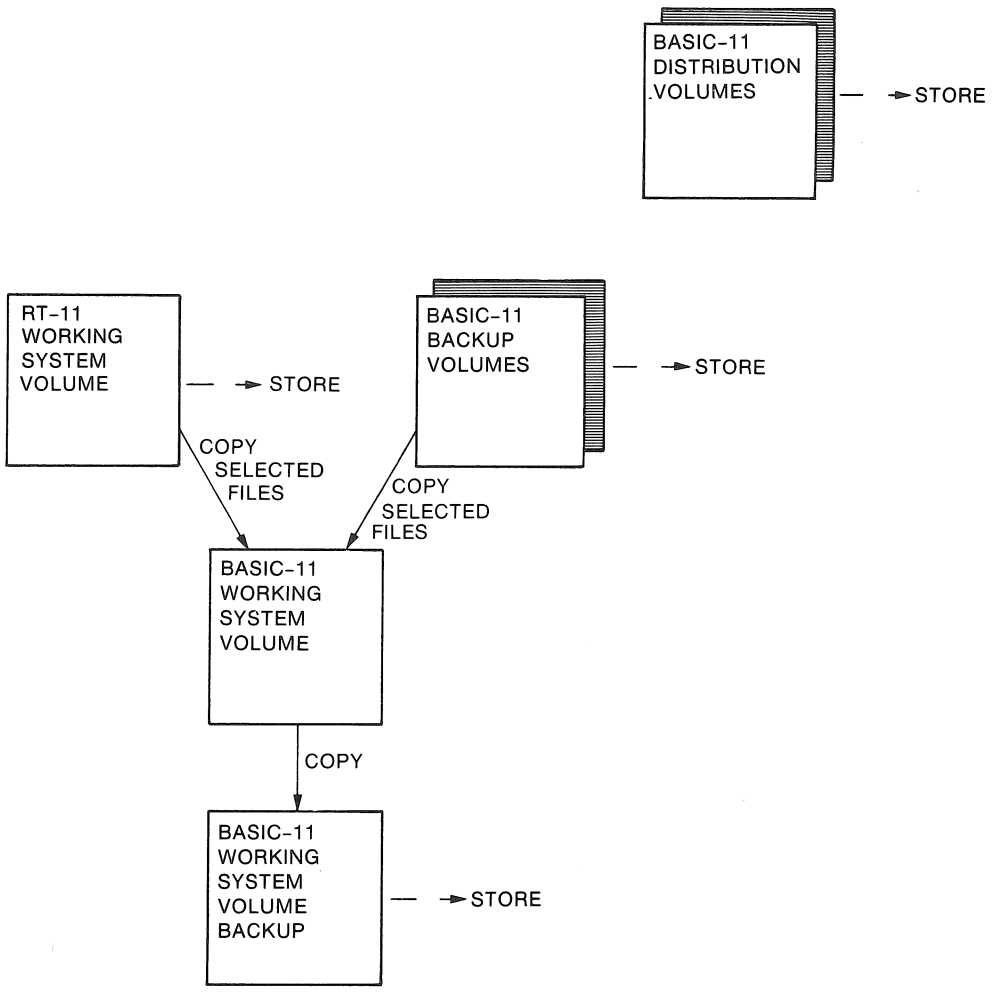


Figure 2-1 Volumes Created During Installation from Small Volumes to Small Volumes

2.1 BOOTSTRAPPING THE RT-11 WORKING SYSTEM

With the processor turned on, insert the RT-11 working system volume 1 or 1-T in drive unit Ø. If your hardware configuration is a PDT-11 Intelligent Terminal, insert volume 1-T and store volume 1. If your hardware configuration includes a PDP-11 processor, insert volume 1 and store volume 1-T.

Use the hardware bootstrap to boot the RT-11 working system. (If your configuration does not include a hardware bootstrap, see Appendix B of the RT-11 Installation Guide for instructions on using toggle switches to bootstrap software.) For some hardware configurations, you will need to supply your drive unit's device name to complete the bootstrapping procedure. If you are using a single-density diskette, the drive unit's device name is DX:; if you are using a double-density diskette, the drive unit's device name is DY:; if you are using an RX5Ø diskette, the device name is DU:; and for either PDT-11 device, the device name is PD:.

When you have successfully bootstrapped the system, RT-11 responds with the following message:

```
RT-11yy V05.00
(Followed by any startup file commands)
.
```

Use the DATE command to set the date (where dd-mmm-yy is the day, month, and year in the form 10-FEB-83).

```
.DATE dd-mmm-yy<RET>
.
```

2.2 CREATING THE BASIC-11 WORKING SYSTEM VOLUME

DIGITAL provided you with both a BASIC-11 distribution volume(s) and a backup copy of the distribution volume(s). Before you continue, store your BASIC-11 distribution volume(s) in a safe place. The backup copy of the distribution volume(s) is what you are going to use to create your BASIC-11 working system volume.

You can modify your backup volume, if necessary or desirable. The Software Dispatch describes software modifications. DIGITAL recommends that you review all articles when they are published in the Software Dispatch. A problem you are trying to solve may have been addressed by a published article.

Once you have created your working system, label and store the backup volumes. The backup volume will be a current version of your BASIC-11/RT-11 working system and provide a safeguard in case of damage to your working system volume.

Your next task is to copy the RT-11 files that you have selected onto the BASIC-11 working system volume.

Mount a write-enabled blank volume in drive unit 1. This volume will become your BASIC-11 working system volume. Then use the INITIALIZE/BADBLOCKS command to initialize the blank volume and to cover any bad blocks.

```
.INITIALIZE/BADBLOCKS xx1:<RET>
xx1:/Initialize; Are you sure? Y<RET>
```

The system then scans the volume for bad blocks and creates a new directory. This operation can take up to eight minutes, after which the following message appears if the volume does not contain bad blocks:

```
?DUP-I-No bad blocks detected xx1:
.
```

If the volume contains bad blocks, the following message gives you the number (nnnnn) of bad blocks:

```
?DUP-W-Bad blocks detected nnnnn
.
```

Copy the RT-11 files you will need from your RT-11 working system volume (mounted in drive unit 0) to the volume mounted in drive unit 1. Your selected files must include at least one monitor, the device handlers, and any utility programs that you need for your BASIC-11

system. You will also need the files DIR, DUP, PIP, and SWAP.SYS. You can use either of two commands to copy the files.

The following command copies the files that you identify by typing the file name:

```
.COPY/SYS  xxØ:filnam.typ xx1:filnam.typ<RET>
```

With the following command, RT-11 will prompt you to identify files you want copied. You will need to type only Y for yes or N for no. This command is practical if you are copying a large number of files.

```
.COPY/SYS/QUERY  xxØ: xx1:<RET>
   Files copied:
xxØ:aaaaaa.ttt to xx1:aaaaaa.ttt  Y<RET> (include file)
xxØ:bbbbbb.ttt to xx1:bbbbbb.ttt  N<RET> (exclude file)
            .
            .
            .
```

RT-11 returns the monitor prompt (.) when this operation is complete.

2.3 INSTALLING THE BOOTSTRAP ON THE BASIC-11 WORKING SYSTEM VOLUME

Your next step is to copy the bootstrap to the BASIC-11 working system volume. With the RT-11 working volume and the BASIC-11 working system volume still in drive units Ø and 1, respectively, type the following command:

```
.COPY/BOOT xx1:RT11yy.SYS xx1:<RET>
.
```

Remove the BASIC-11 working system volume from drive unit 1.

2.4 COPYING THE BASIC-11 FILES TO THE WORKING SYSTEM VOLUME

You are ready to complete the creation of your BASIC-11 working system volume by copying the files you have chosen from your BASIC-11 backup volume.

Mount your BASIC-11 backup volume in drive unit 1. Use either the .COPY/SYS command or the .COPY/SYS/QUERY command (described in Section 2.2) to add the selected files to your working system volume. When you have completed this operation, remove the volume from drive unit Ø and label it "BASIC-11 working system volume."

2.5 BOOTSTRAPPING THE BASIC-11 WORKING SYSTEM

Now halt the processor. Remove the RT-11 working system volume and mount the BASIC-11 working system volume in drive unit Ø. Use the hardware bootstrap to boot the BASIC-11 working system volume. RT-11 responds with the following message:

```
RT-11yy VØ5.ØØ
(Followed by any startup file commands.)
.
```

If the BASIC-11 working system does not boot, repeat the procedures described in Section 2.2 to create the BASIC-11 working system volume.

2.6 TESTING THE BASIC-11 WORKING SYSTEM

Once you have created the working system volume and backup volume, you can execute the following demonstration to test the system. This demonstration does not serve as a comprehensive system exercise.

```
.RUN BASIC<RET>
BASIC-11/RT-11 V2.1
OPTIONAL FUNCTIONS (ALL, NONE, OR INDIVIDUAL)? A<RET>

READY
NEW TST.BAS<RET>

READY
10 PRINT "CONGRATULATIONS.  YOU HAVE INSTALLED BASIC"<RET>
20 END<RET>
SAVE<RET>

READY
LISTNH<RET>
10 PRINT "CONGRATULATIONS.  YOU HAVE INSTALLED BASIC"
20 END

READY
RUNNH<RET>

CONGRATULATIONS.  YOU HAVE INSTALLED BASIC

READY
BYE<RET>
```

If your BASIC-11 working system does not complete this minimal test successfully, repeat all the procedures described in this chapter.

2.7 PROTECTING AND COPYING THE BASIC-11 WORKING SYSTEM

DIGITAL recommends that you protect all the files on your completed BASIC-11 working system volume and create a backup copy of the volume.

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

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

Next, copy the working system volume to a backup volume. Mount a blank volume in drive unit 1 with the BASIC-11 working system volume in drive unit Ø. Use the INITIALIZE/BADBLOCKS command to initialize the blank volume and check for bad blocks.

Copy all the files, including the bootstrap, from your working system volume to the blank volume. You can use the COPY/BOOT command to copy the bootstrap and the COPY/SYS/QUERY command to copy the files.

Write protect the files on the backup volume. Reserve this volume as a safety precaution. If your working system volume is damaged, you can restore the system by copying the backup volume.

CHAPTER 3

SYSTEM INSTALLATION: FROM SMALL VOLUMES TO A LARGE VOLUME

Small volumes are diskettes. Large volumes are hard disks. If BASIC-11 was distributed to you on a small volume and you intend to build a working system to run on a large volume, you should perform the procedures described in this chapter.

To install your system, follow the steps summarized in the list below:

1. Bootstrap the RT-11 working system.
2. Create the BASIC-11 working system volume.
3. Install the bootstrap on the BASIC-11 working system volume.
4. Copy the BASIC-11 files from the backup volume to the working system volume.
5. Bootstrap the BASIC-11 working system.
6. Test the BASIC-11 working system.
7. Protect and copy the BASIC-11 working system.

Sections 3.1 through 3.7 correspond to each of these steps and describe in detail the procedures you must perform.

Figure 3-1 shows the various backup volumes you create when you install BASIC-11.

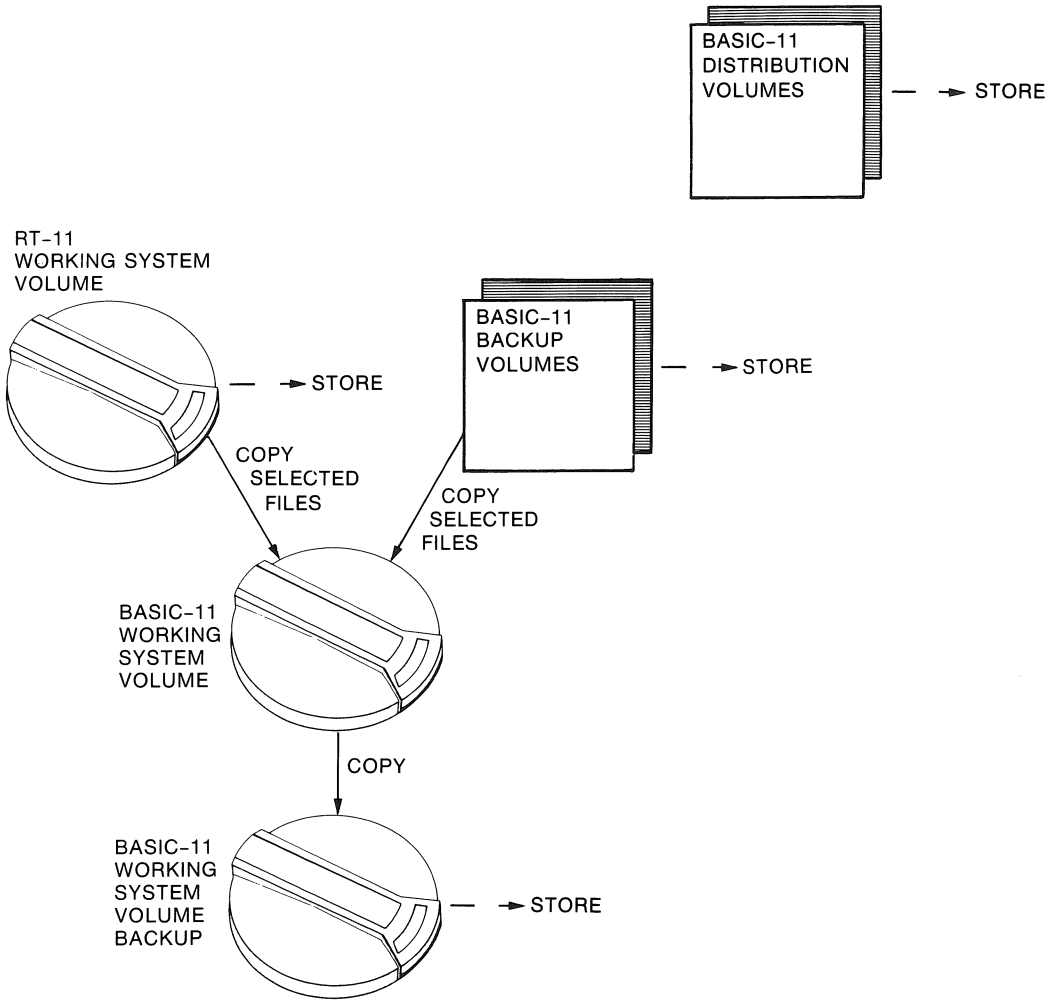


Figure 3-1 Volumes Created During Installation from Small Volumes to a Large Volume

3.1 BOOTSTRAPPING THE RT-11 WORKING SYSTEM

With the processor turned on, insert the RT-11 working system volume in hard disk Unit Ø.

Use the hardware bootstrap to boot the RT-11 working system. (If your configuration does not include a hardware bootstrap, see Appendix B of the RT-11 Installation Guide for instructions on using toggle switches to bootstrap software.)

When you have successfully bootstrapped the system, RT-11 responds with the following message:

```
RT-11yy V05.00
(Followed by any startup file commands)
.
```

Use the DATE command to set the date (where dd-mmm-yy is the day, month, and year in the form 10-FEB-83).

```
.DATE dd-mmm-yy<RET>
.
```

3.2 CREATING THE BASIC-11 WORKING SYSTEM VOLUME

DIGITAL provided you with both a BASIC-11 distribution volume(s) and a backup copy of the distribution volume(s). Before you continue, store your BASIC-11 distribution volume(s) in a safe place. The backup copy of the distribution volume(s) is what you are going to use to create your BASIC-11 working system volume.

You can now modify your backup volume, if necessary or desirable. The Software Dispatch describes software modifications. DIGITAL recommends that you review all articles when they are published in the Software Dispatch. A problem you are trying to solve may have been addressed by a published article.

Once you have created your working system, label and store the backup volumes. The backup volume will be a current version of your BASIC-11/RT-11 working system and provide a safeguard in case of damage to your working system volume.

Your next task is to copy the RT-11 files that you have selected onto the BASIC-11 working system volume.

Mount a blank volume in large volume drive unit 1, initialize this volume, and check for bad blocks. This volume will become your BASIC-11 working system volume.

```
.INITIALIZE/BADBLOCKS xx1:<RET>
xx1:/Initialize; Are you sure? Y<RET>
```

The system then scans the volume for bad blocks and creates a new directory. This operation can take up to eight minutes, after which the following message appears if the volume does not contain bad blocks:

```
?DUP-I-No bad blocks detected xx1:
.
```

If the volume contains bad blocks, the following message gives you the number (nnnnn) of bad blocks:

```
?DUP-W-Bad blocks detected nnnnn
.
```

Copy the RT-11 files you will need from your RT-11 working system volume (mounted in drive unit Ø) to the volume mounted in drive unit 1. Your selected files must include at least one monitor, the device handlers, and any utility programs that you need for your BASIC-11 system. You will also need the files DIR, DUP, PIP, and SWAP.SYS. You can use either of two commands to copy the files.

The following command copies the files that you identify by typing the file name:

```
.COPY/SYS  xxØ:filnam.typ xx1:filnam.typ<RET>
```

With the following command, RT-11 will prompt you to identify files you want copied. You will need to type only Y for yes or N for no. This command is practical if you are copying a large number of files.

```
.COPY/SYS/QUERY  xxØ: xx1:<RET>
   Files copied:
xxØ:aaaaaa.ttt to xx1:aaaaaa.ttt  Y<RET> (include file)
xxØ:bbbbbb.ttt to xx1:bbbbbb.ttt  N<RET> (exclude file)
            .
            .
            .
```

RT-11 returns the monitor prompt (.) when this operation is complete.

3.3 INSTALLING THE BOOTSTRAP ON THE BASIC-11 WORKING SYSTEM VOLUME

Your next step is to copy the bootstrap to the BASIC-11 working system volume. With the RT-11 working system volume and the BASIC-11 working system volume still in drive units Ø and 1, respectively, type the following command:

```
.COPY/BOOT xx1:RT11yy.SYS xx1:<RET>
.
```

3.4 COPYING THE BASIC-11 FILES TO THE WORKING SYSTEM VOLUME

You are ready to complete the creation of your BASIC-11 working system by copying the files you have chosen from your BASIC-11 backup volumes.

With your BASIC-11 working system volume mounted in large volume drive unit 1, mount your first BASIC-11 backup volume in small volume drive unit Ø. Then use the .COPY/QUERY command to begin the operation. RT-11 will list each BASIC-11 file contained on your first BASIC-11 backup volume. If you want the specified file, type Y<RET>, if you do not want the file, type N<RET>.

```
.COPY/QUERY  xxØ: xx1:<RET>
  Files copied:
xxØ:aaaaaa.ttt to xx1:aaaaaa.ttt  Y<RET> (include file)
xxØ:bbbbbb.ttt to xx1:bbbbbb.ttt  N<RET> (exclude file)
            .
            .
            .
```

Repeat the process with your second BASIC-11 backup volume mounted in small volume drive unit Ø.

3.5 BOOTSTRAPPING THE BASIC-11 WORKING SYSTEM

Use the following command to boot the BASIC-11 working system mounted in large volume drive unit 1.

```
.BOOT xx1:
```

RT-11 responds with the following:

```
RT-11yy VØ5.ØØ
(Followed by any startup file commands.)
.
```

If the BASIC-11 working system does not boot, repeat the procedures described in Section 3.2 to create the BASIC-11 working system.

3.6 TESTING THE BASIC-11 WORKING SYSTEM

Once you have created the working system volume and backup volume, you can execute the following demonstration to test the system. This demonstration does not serve as a comprehensive system exercise.

```
.RUN BASIC<RET>
BASIC-11/RT-11 V2.1
OPTIONAL FUNCTIONS (ALL, NONE, OR INDIVIDUAL)? A<RET>

READY
NEW TST.BAS<RET>

READY
10 PRINT "CONGRATULATIONS.  YOU HAVE INSTALLED BASIC"<RET>
20 END<RET>
SAVE<RET>

READY
LISTNH<RET>
10 PRINT "CONGRATULATIONS.  YOU HAVE INSTALLED BASIC"
20 END

READY
RUNNH<RET>

CONGRATULATIONS.  YOU HAVE INSTALLED BASIC

READY
BYE<RET>
```

If your BASIC-11 working system does not complete this minimal test successfully, repeat all the procedures described in this chapter.

3.7 PROTECTING AND COPYING THE BASIC-11 WORKING SYSTEM

DIGITAL recommends that you protect all the files on your completed BASIC-11 working system volume and create a backup copy of the volume.

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

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

Next, copy the working system volume to a backup volume. Mount a blank volume in drive unit Ø, leaving the BASIC-11 working system volume in drive unit 1. Use the INITIALIZE/BADBLOCKS command to initialize the blank volume and check for bad blocks.

Copy all the files, including the bootstrap, from your working system volume to the blank volume. You can use the COPY/BOOT command to copy the bootstrap and the COPY/SYS/QUERY command to copy the files.

Write protect the backup volume. Reserve this volume as a safety precaution. If your working system volume is damaged, you can restore the system by copying the backup volume.

CHAPTER 4

SYSTEM INSTALLATION: FROM A LARGE VOLUME TO A LARGE VOLUME

Large volumes are hard disks. If BASIC-11 was distributed to you on a large volume and you intend to build a working system to run on a large volume, you should perform the procedures described in this chapter. These procedures assume that your hardware configuration includes two disk drives.

Installing your system requires the steps summarized in the following list:

1. Bootstrap the RT-11 working system.
2. Copy the BASIC-11 distribution volume.
3. Create the BASIC-11 working system volume.
4. Install the bootstrap on the BASIC-11 working system volume.
5. Copy the BASIC-11 files from the backup volume to the working system volume.
6. Bootstrap the BASIC-11 working system.
7. Test the BASIC-11 working system.
8. Protect and copy the BASIC-11 working system.

Sections 4.1 through 4.8 correspond to each of these steps and describe in detail the procedures you must perform.

Figure 4-1 shows the backup volumes you create when you install BASIC-11.

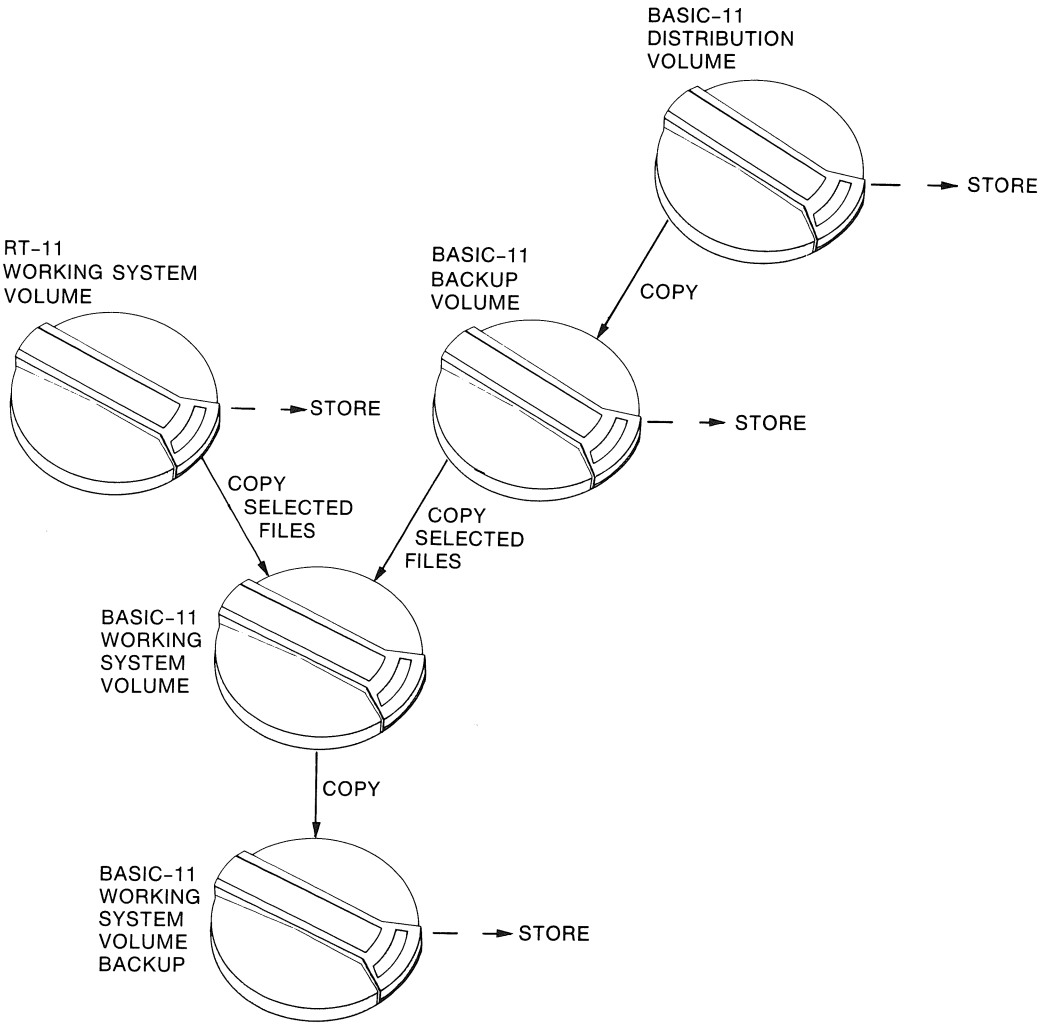


Figure 4-1 Volumes Created During Installation from a Large Volume to a Large Volume

4.1 BOOTSTRAPPING THE RT-11 WORKING SYSTEM

With the processor turned on, insert the RT-11 working system in drive unit Ø.

Use the hardware bootstrap to boot the RT-11 working system. (If your configuration does not include a hardware bootstrap, see Appendix B of the RT-11 Installation Guide for instructions on using toggle switches to bootstrap software.)

When you have successfully bootstrapped the system, RT-11 responds with the following message:

```
RT-11yy V05.ØØ
(Followed by any startup file commands)
.
```

Use the DATE command to set the date (where dd-mmm-yy is the day, month, and year in the form 10-FEB-83).

```
.DATE dd-mmm-yy<RET>
.
```

4.2 CREATING A COPY OF THE BASIC-11 DISTRIBUTION VOLUME

Your first operation using the running RT-11 working system is to create a backup copy of the BASIC-11 distribution volume. After you have copied the distribution volume, reserve the original and use the backup copy. If the volume you are using is damaged by machine failure or human error, you can create another copy from the original. If the original distribution volume is damaged, you will have to purchase another copy from DIGITAL.

Insert a write-enabled blank volume in drive unit 1. Then use the INITIALIZE/BADBLOCKS command to initialize the blank volume and to cover any bad blocks.

```
.INITIALIZE/BADBLOCKS xx1:<RET>
xx1:/INITIALIZE; Are you sure? Y<RET>
```

The system then scans the volume for bad blocks and creates a new directory. This operation can take up to eight minutes, after which the following message appears if the volume does not contain bad blocks:

```
?DUP-I-No bad blocks detected xx1:
.
```

If the volume contains bad blocks, the following message appears. This message gives you the number (nnnnn) of bad blocks on the volume.

```
?DUP-W-BAD blocks detected nnnnn
.
```

NOTE

DIGITAL recommends that you use only volumes that do not have bad blocks when you copy distribution volumes or working volumes. To discover whether an already initialized volume has bad blocks, use the command DIRECTORY/BAD xxn:.

Next, copy the BASIC-11 distribution volume to the BASIC-11 backup volume. With the RT-11 working system volume remaining in drive unit 0, type:

```
.COPY/SYS/WAIT xx0:  xx1:<RET>
Mount input volume in xx0:;  Continue?
```

Before you type Y<RET>, remove the RT-11 working system from drive unit 0 and replace it with the BASIC-11 distribution volume.

```
Mount input volume in xxØ:;  Continue?  Y<RET>
Mount output volume in xxl:;  Continue?  Y<RET>
```

At this point, RT-11 copies the distribution volume. You should not interact with the system while the file transfer operation takes place. When the operation is complete, the following prompt appears:

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

Before you type Y<RET>, remove the BASIC-11 distribution volume from drive unit Ø and replace it with the RT-11 working system volume. Store the distribution volume in a safe place.

The distribution volumes have been protected to prevent accidental deletion of files, and RT-11 copied this protection onto your backup volumes. You must now remove the protection from all the files on the backup volumes. With backup volume 1 mounted in drive unit 1, type the following command:

```
.UNPROTECT/SYSTEM xxl:*.* xxl:*.*<RET>
 Files unprotected:
xxl:aaaaaa.ttt    to xxl:aaaaaa.ttt
xxl:bbbbbb.ttt    to xxl:bbbbbb.ttt
xxl:cccccc.ttt    to xxl:cccccc.ttt
xxl:dddddd.ttt    to xxl:dddddd.ttt
                .
                .
                .
xxl:zzzzzz.ttt    to xxl:zzzzzz.ttt
.
```

You can now modify your backup volume, if necessary or desirable. The Software Dispatch describes software modifications. DIGITAL recommends that you review all articles when they are published in the Software Dispatch. A problem you are trying to solve may have been addressed by a published article.

Once you have created your working system, label and store the backup volume. The backup volume will be a current version of your BASIC-11/RT-11 working system and provide a safeguard in case of damage to your working system volume.

4.3 CREATING THE BASIC-11 WORKING SYSTEM VOLUME

Your next task is to copy the RT-11 files that you have selected onto the BASIC-11 working system volume.

Mount a blank volume in large volume drive unit 1, initialize this volume, and check for bad blocks. This volume will become your BASIC-11 working system volume. An RKØ5 disk must be formatted before you initialize it.

```
.INITIALIZE/BADBLOCKS xxl:<RET>
xxl:/Initialize; Are you sure?  Y<RET>
```

The system then scans the volume for bad blocks and creates a new directory. This operation can take up to eight minutes, after which the following message appears if the volume does not contain bad blocks:

```
?DUP-I-No bad blocks detected xxl:
.
```

If the volume contains bad blocks, the following message gives you the number (nnnnn) of bad blocks:

```
?DUP-W-Bad blocks detected nnnnn
.
```

Copy the RT-11 files you will need from your RT-11 working system volume (mounted in drive unit Ø) to the volume mounted in drive unit 1. Your selected files must include at least one monitor, the device handlers, and any utility programs that you need for your BASIC-11 system. You will also need the files DIR, DUP, PIP, and SWAP.SYS. You can use either of two commands to copy the files.

The following command copies the files that you identify by typing the file name:

```
.COPY/SYS  xxØ:filnam.typ xxl:filnam.typ<RET>
```

With the following command, RT-11 will prompt you to identify files you want copied. You will need to type only Y for yes or N for no. This command is practical if you are copying a large number of files.

```
.COPY/SYS/QUERY  xxØ: xxl:<RET>
   Files copied:
xxØ:aaaaaa.ttt to xxl:aaaaaa.ttt  Y<RET> (include file)
xxØ:bbbbbb.ttt to xxl:bbbbbb.ttt  N<RET> (exclude file)
            .
            .
            .
```

RT-11 returns the monitor prompt (.) when this operation is complete.

4.4 INSTALLING THE BOOTSTRAP ON THE BASIC-11 WORKING SYSTEM VOLUME

Your next step is to copy the bootstrap to the BASIC-11 working volume. With the RT-11 working system volume and the BASIC-11 working system volume still in drive units Ø and 1, respectively, type the following command:

```
.COPY/BOOT xxl:RT1lyy.SYS xxl:<RET>
.
```

4.5 COPYING THE BASIC-11 FILES TO THE WORKING SYSTEM VOLUME

You are ready to complete the creation of your BASIC-11 working system by copying the files you have chosen from your BASIC-11 backup volume.

Mount your BASIC-11 backup volume in drive unit 1. Use either the .COPY/SYS command or the .COPY/SYS/QUERY command (described in Section 4.3) to add the selected files to your working system volume. When you have completed this operation, remove the volume from drive unit Ø and label it "BASIC-11 working system."

4.6 BOOTSTRAPPING THE BASIC-11 WORKING SYSTEM

Now halt the processor. Remove the RT-11 working system volume and mount the BASIC-11 working system volume in drive unit Ø. Use the hardware bootstrap to boot the BASIC-11 working system. RT-11 responds with the following message:

```
RT-11yy V05.ØØ
(Followed by any startup file commands.)
.
```

If the BASIC-11 working system does not boot, repeat the procedures described in Section 4.3 to create the BASIC-11 working system volume.

4.7 TESTING THE BASIC-11 WORKING SYSTEM

Once you have created and preserved the working system volume, you can execute the following demonstration to test the system. This demonstration does not serve as a comprehensive system exercise.

```
.RUN BASIC<RET>
BASIC-11/RT-11 V2.1
OPTIONAL FUNCTIONS (ALL, NONE, OR INDIVIDUAL)? A<RET>

READY
NEW TST.BAS<RET>

READY
1Ø PRINT "CONGRATULATIONS.  YOU HAVE INSTALLED BASIC"<RET>
2Ø END<RET>
SAVE<RET>

READY
LISTNH<RET>
1Ø PRINT "CONGRATULATIONS.  YOU HAVE INSTALLED BASIC"
2Ø END

READY
RUNNH<RET>

CONGRATULATIONS.  YOU HAVE INSTALLED BASIC

READY
BYE<RET>
```

If your BASIC-11 working system does not complete this minimal test successfully, repeat all the procedures described in this chapter.

4.8 PROTECTING AND COPYING THE BASIC-11 WORKING SYSTEM

DIGITAL recommends that you protect all the files on your completed BASIC-11 working system volume and create a backup copy of the volume.

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

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

Next, copy the working system volume to a backup volume. Mount a blank volume in drive unit 1 with the BASIC-11 working system volume in drive unit Ø. Use the INITIALIZE/BADBLOCKS command to initialize the blank volume and check for bad blocks.

Copy all the files, including the bootstrap, from your working system volume to the blank volume. You can use the COPY/BOOT command to copy the bootstrap and the COPY/SYS/QUERY command to copy the files.

Write protect the backup volume. Reserve this volume as a safety precaution. If your working system volume is damaged, you can restore the system by copying the backup volume.

CHAPTER 5

CREATING AND LINKING CONFIGURATIONS

Creating and linking a configuration is necessary if the distributed versions of BASIC -- BASIC.SAV, BASICD.SAV, and BAS16K.SAV -- are not suitable for your application, or if you decide to:

- Change the overlay type.
- Add support for arithmetic hardware.
- Include or exclude any BASIC-11 optional features.

Sections 5.1 through 5.3 describe your options for tailoring BASIC-11 to your needs. When you know what options you want, you can begin the configuration process, which is accomplished through a dialog (SUCNFG). Section 5.4 contains the text of this dialog as presented by BASIC-11. Section 5.5 explains how to begin the dialog; Section 5-6 provides an example of actual dialog interaction.

5.1 OVERLAY TYPES

BASIC-11 offers five overlay types, described in Table 5-1. BASIC-11 provides Overlay Type 1 unless you specify another type for your version of BASIC. The following descriptions of the overlay types help you decide on the one that meets the needs of your configuration.

As the overlay type number increases from 1 to 5, the memory space required by BASIC decreases and execution time increases. Type 2, however, has a very minimal execution time increase over Type 1, with a substantial saving in memory if long error messages are used. You should not choose overlay type 5 unless you are using an 16K system.

Table 5-1
BASIC-11 Overlay Types

Type	Memory Space Requirement* Bytes (Octal)	Bytes (Decimal)	Features Overlaid
1	63764	26612	None
2	60760	25072	Error messages and some of edit phase
3	42544	17764	Error messages, edit phase, and execute phase
4	36540	15712	Error messages; internal overlay of edit phase and execute phase
5	30320	12496	Error messages, math package, I/O routines; internal overlay of edit phase and execute phase

* Note that these figures are approximate and include all optional features (described in Section 5.3) except CALL support.

5.2 ARITHMETIC HARDWARE SUPPORT

The versions of BASIC provided in the distribution volume(s) do not support arithmetic hardware. If your hardware configuration includes any of the arithmetic hardware listed below, you should choose the supporting math package during the configuration. The following math packages are available:

Math Package	Supported Arithmetic Hardware
EAE	KE11-A or KE11-B Extended Arithmetic Element
EIS	KE11-E Extended Instruction Set; also supports a PDP-11/34, 11/45, 11/50, 11/55, 11/60, or 11/70 without the FP11 Floating Point Unit
FIS	KE11-F or KEV-11 Floating Instruction Set
FPU	FP11-A, FP11-B, FP11-C, and FP11-E Floating Point Unit; also supports the 11/34A unit on the PDP11/34.

Table 5-2 shows the potential combinations of central processing units, arithmetic hardware, and arithmetic software.

Table 5-2
Math Packages for Hardware Configurations

Central Processor	Arithmetic Hardware	Arithmetic Software
PDP-11V03, PDP-11/03	KEV-11	EIS or FIS
PDP-11/05, PDP-11/10, PDP-11/15, PDP-11/20	KE11-A or KE11-B	EAE
PDP-11/34	standard	EIS
	11/34-A	FPU
	FP11-A	FPU
	KE11-A	EAE
	KE11-B	EAE
PDP-11/35, PDP-11/40	KE11-A	EAE
	KE11-B	EAE
	KE11-E	EIS
	KE11-F	FIS
PDP-11/45, PDP-11/50	standard	EIS
	FP11-B	FPU
	KE11-A or KE11-B	EAE
PDP-11/55	standard	EIS
	FP11-C	FPU
	KE11-A or KE11-B	EAE
PDP-11/60	standard	EIS
	FP11-E	FPU
	KE11-A or KE11-B	EAE
PDP-11/70	standard	EIS
	FP11-B or FP11-C	FPU
	KE11-A or KE11-B	EAE

5.3 OPTIONAL FEATURES

The following optional BASIC-11 features are included on your working system volume if you do not configure or if you take default answers to configuration questions:

Feature	Memory Space Requirement* Bytes (Octal)	Bytes (Decimal)	Function
PRINT USING command	2444	1316	Allows you to control the appearance and location of data on the output line.
Transcendental functions	2174	1148	Include SQR, SIN, COS, ATN, LOG, LOG10, and EXP.
SUB command	610	392	Allows you to edit without retyping the entire BASIC string.
RESEQ command	570	376	Allows you to resequence the line numbers in a BASIC program.
Long error messages	1752	1002	Provide several word explanations of an error condition. Short error messages are three-character abbreviations of the long error messages.

* These values are approximate.

To exclude any of these features, type an N in response to the specific dialog question.

The following BASIC-11 feature is not included on your working volume unless you request the feature during configuration:

Feature	Function
CALL support	Allows BASIC to execute a routine written in another language.

When you include this feature by typing Y in response to the specific dialog question, you will be prompted to provide names of your assembly language routine modules.

5.4 CONFIGURATION DIALOG TEXT

Figure 5-1 lists all the questions and explanations contained in the dialog text.

```
.SAV and .COM file name (default is BASIC): <RET>

Enter the name in 6 or less characters for the name to be
given to the .SAV image of BASIC and the indirect file
to be used to build BASIC.

Do you want a map?  (Y or N; default is N): <RET>

Type Y for yes or N for no.  If the answer
is Y, the next question will ask for a name.

.MAP name (default is same as .SAV name): <RET>

Enter <ret> or any legal file name.

Background or Foreground ? (B or F; default is B): <RET>

B creates a .SAV image to be run in the background.  F creates
a .REL image to be run in the foreground.

Do you want PRINT USING? (Y or N; default is Y): <RET>

Y includes the PRINT USING option, N does not.

Do you want CALL support? (Y or N; default is N): <RET>

Y includes CALL support, N does not.  If CALL support is
included the next question will handle the modules to
be used with CALL in this BASIC.

Enter module names (one per line): <RET>

On each line enter the name of a module to be included.  Include your
modified BSCLI, your subroutines, and BSCLLB, if it is needed.
When all the modules have been entered, type <ret> alone.
For more information, please see the documentation on CALL.

Single or Double precision math? (S or D; default is S): <RET>

S creates a BASIC that supports the 2-word floating point
representation D creates a BASIC that supports the 4-word
floating point representation.

Arithmetic hardware (NONE, EAE, EIS, FIS, or FPU; default is NONE): <RET>
```

Figure 5-1 Configuration Program Dialog

```
Arithmetic Hardware support is defined as follows: <RET>
NONE - no arithmetic hardware is supported
EAE  - KE11A Extended Arithmetic Element is supported
EIS  - KE11E Extended Instruction Set is supported
FIS  - KE11F or KEV11F Floating Instruction Set is supported
FPU  - FPP11A, B, or C Floating Point Unit is supported

Do you want the transcendental functions?  (Y or N;  default is Y): <RET>

Y includes the transcendental functions, N does not.
The transcendental functions are SQR, SIN, COS, ATN,
LOG, LOG10, and EXP.

Do you want the SUB command? (Y or N, default is Y): <RET>

Y includes the SUB command, N does not, SUB allows editing
without retyping the entire BASIC statement.

Do you want the RESEQ command? (Y or N, default is Y): <RET>

Y includes the RESEQ command, N does not.  RESEQ allows the
resequencing of line numbers in a BASIC program.

Do you want long error messages? (Y or N; default is Y): <RET>

Y includes long error messages, N does not.
Long messages are several word explanations of an error
condition.  Short error messages are three-character
abbreviations of the long messages.

Overlay structure type (1, 2, 3, 4, or 5; default is 1): <RET>

The overlay structures are described as follows:
Type 1 - BASIC is not overlaid.
Type 2 - The error messages are overlaid with a segment of
         the edit phase.
Type 3 - The error message phase, edit phase, and the
         execute phase are overlaid.
Type 4 - Similar to type 3 except that the edit and execute
         phases are overlaid internally.
Type 5 - Similar to type 4 except that the math package, the
         I/O routines, and some other routines normally in the
         root segment are overlaid.

The higher the overlay structure type number, the less memory
is required by BASIC.  The saving in memory requirements is made
at the expense of performance.  Type 2, however, has very
minimal effect on the performance of BASIC, with a substantial
saving in memory if long error messages are used.  Please see
the documentation for more information.
```

Figure 5-1 (Cont.) Configuration Program Dialog

5.5 STARTING THE CONFIGURATION DIALOG

You must be running BASIC-11 in order to begin the configuration process. With your BASIC-11 working system volume mounted, type the following command:

```
.RUN BASIC<RET>
BASIC-11/RT-11 V2.1
OPTIONAL FUNCTIONS (ALL, NONE, OR INDIVIDUAL)? <RET>

READY
```

At this point, your system contains all the BASIC-11 defaults, including the BASIC.SAV file. Next, run the configuration program SUCNFG, which provides a dialog prompting you to choose your version of BASIC. This configuration program builds an indirect command file linking your newly configured version of BASIC-11 to your working system. Your new version of BASIC-11 is ready to run when the linking process is complete.

Begin the configuration program by using the following command:

```
RUN SUCNFG<RET>
BASIC-11/RT-11 V2.1 Configuration Program

Type "?<ret>" after any prompt for HELP

.SAV and .COM file name (default is BASIC):
```

NOTE

If you do not specify a file for the BASIC language during the configuration process, BASIC-11 defaults to a file named BASIC.SAV. While this file has the same name as the version of BASIC on the distribution volume, the files are different. If you choose your BASIC file by default during configuration, you lose the distributed version of BASIC.SAV. If you wish to retain the distributed version of BASIC, you must give the .SAV and .COM file a name other than BASIC. You can retrieve the distributed version from your backup volume.

When the dialog prompts you for a .SAV and .COM file name, enter the name of the BASIC version you want to create.

5.6 EXAMPLE OF CONFIGURATION

With the example of dialog interaction reproduced in Figure 5-2 you can:

(1) Link BASIC to run in the foreground.

(2) Add support for EIS arithmetic hardware.

(3) Execute the indirect command file.

(4) Run the new version of BASIC in the foreground.

```
RUN SUCNFG
BASIC-11/RT-11 V2.1 Configuration Program

Type "?<ret>" after any prompt for HELP.

.SAV and .COM file name (default is BASIC): MYBAS <RET>

Do you want a map? (Y or N; default is N): Y <RET>

.MAP name (default is same as .SAV name):    <RET>

(1) Background or Foreground ? (B or F; default is B): F <RET>

Do you want PRINT USING? (Y or N; default is Y): Y <RET>

Do you want CALL support? (Y or N; default is N): N <RET>

Single or Double precision math? (S or D; default is S): D <RET>

(2) Arithmetic hardware (NONE, EAE, EIS, FIS, or FPU; default is NONE): EIS <RET>

Do you want the transcendental functions? (Y or N; default is Y): Y <RET>

Do you want the SUB command? (Y or N, default is Y): Y <RET>

Do you want the RESEQ command? (Y or N, default is Y): Y <RET>

Do you want long error messages? (Y or N; default is Y): Y <RET>

Overlay structure type (1, 2, 3, 4, or 5; default is 1): 3 <RET>

The following GLOBALS will be undefined.  These
errors may be ignored.
..UAC$, ..NRC$, AND ..MSP$ (from no CALL).
```

Figure 5-2 Example of Configuration Dialog Interaction

```
To create the BASIC that you have just defined,
type "BYE<ret>" after BASIC prints "READY".  When the system
prompts for input, type "@MYBAS<ret>" to link the new BASIC.
See the User's Guide for instructions on running
BASIC in the Foreground.

READY
BYE <RET>

(3)  .@MYBAS <RET>

.R LINK
*MYBAS,/W=/R:1200//
*BSPTRD,BSPAT,BSRØD
*BSR1D,BSPRØ
*BSCLIN
*SUO1ID,BSOTØD.EIS,BSOT1D.EIS
*SUIOPR,SUIOPO
*SUXID,BSXØA,BSXØB,BSX1A,BSX1B/O:1
*BSPR1,BSCLSN
*SUX2ID,BSX2/O:1
*SUE1ID,BSEØ,BSE1,BSKEYS/O:1
*BSCMP,BSSUB,BSRSQ
*SUDTCM/O:1
*BSERR,BSERML/O:1
*SUIMP,BSFUNC,SUOPT/O:2
*//
?LINK-W-Undefined globals:
..UAC$
..NRC$
..MSP$
*^C

(4)  .FRUN MYBAS/BUF:3000 <RET>

.
F>
BASIC-11/RT-11 V2.1
OPTIONAL FUNCTIONS (ALL, NONE, OR INDIVIDUAL)? CTRL/F A <RET>

READY
1Ø PRINT "WE ARE RUNNING IN FG NOW" <RET>
RUNNH <RET>

WE ARE RUNNING IN FG NOW

READY
BYE <RET>

B>
UNLOAD FG <RET>
```

Figure 5-2 (Cont.) Example of Configuration Dialog Interaction

PART II

BASIC-11/RT-11 RELEASE NOTES

CHAPTER 6

STANDARD OPERATING PROCEDURES

This chapter provides recommendations for and restrictions on system operation. This chapter is divided into three sections.

1. Configurations and Operations Under RT-11
2. Programming Techniques
3. Compiling Operations

6.1 CONFIGURATIONS

This section describes the functions of and restrictions on BASIC-11 configurations and operations under RT-11.

6.1.1 Features Required to Run the Configuration Program

The following features, listed as optional in Chapter 1 of the BASIC-11/RT-11 User's Guide, are required in order to run SUCNFG.BAS:

CHR$

INT

POS

STR$

VAL

6.1.2 Omitting Trigonometric Functions

When you configure your BASIC-11 system, you can specify no transcendental functions (SUCNFG dialog question 8). However, when you specify no transcendental functions, you must be sure that your copy of SYSLIB.OBJ does not include FORLIB.OBJ. If FORLIB.OBJ has been incorporated as part of SYSLIB.OBJ, some of the FORTRAN math functions will be used to resolve global references in your BASIC configuration, which then will not work.

To configure a BASIC with Overlay Type 2, 3, 4, or 5, the linker must reference SYSLIB.OBJ to get the overlay handler module. Because of the potential problem with FORLIB.OBJ, DIGITAL recommends that only an original, distributed copy of SYSLIB.OBJ be used for a configuration of BASIC-11.

6.1.3 Non-Resident Device Handlers

If a BASIC-11 program uses a virtual array that resides on a device whose handler is not resident, references you make to that device (with the OPEN statement, or the CHAIN command, for example) can cause the system to crash. Be sure that the device handlers required for all arrays are loaded before you run the program.

6.1.4 Changing Volumes

Physical volumes should never be changed while a program is running. This action can destroy the working system. Changing volumes is safe only when a prompt character appears (KMON's dot, or PIP's or DUP's asterisk).

NOTE

If you use a single-disk drive system or a dual-disk drive system and the system volume is neither the input nor the output volume, you can use the /WAIT option to change volumes. See the RT-11 System User's Guide for more information.

6.2 PROGRAMMING TECHNIQUES

Some aspects of programming in BASIC-11 are unique. This section describes techniques you should use on a BASIC-11 system.

6.2.1 File Types

When you call a file (with an OLD, RUN, or CHAIN command), BASIC-11 looks first for a compiled file (with a .BAC extension), and then for a source file (with a .BAS extension). BASIC-11 looks for a .BAC extension first, even if you specify .BAS as part of the file name. If you specify OLD X.BAC and X.BAC does not exist, BASIC-11 continues to search for the .BAS file.

6.2.2 GOSUB and GOTO Statements

DIGITAL recommends that you do not use GOSUB and GOTO statements for operations in immediate mode. An immediate mode GOSUB to a routine that has an INPUT or LINPUT statement can cause your BASIC-11 system to crash. This occurs because BASIC-11 uses the same area of memory to process immediate mode statements and terminal input.

While an immediate GOTO usually can execute the subroutine and print a warning message, in many cases an immediate GOTO will also return control to the last statement before the subroutine and overwrite that statement with the terminal input. No warning message appears. This problem will almost always occur when you are using a compiled program.

6.2.3 Integers and Floating-Point Numbers

If not explicitly defined as integers, all numbers in BASIC-11 are stored in floating-point representation. Calculations using floating-point representation are slower than calculations using integers. To maximize execution speed, use integers by placing a percent sign after a number whenever possible.

6.2.4 Maximum Line Length

Each line of a BASIC-11 program can contain 132 characters, which is the size of the input buffer, unless the RT-11 input ring buffer has fewer characters. The smaller of the buffers limits the maximum characters per line.

The commands SET TT:WIDTH=n and SET TT:CRLF limit the visible terminal output, but do not change the number of characters entered to your program. If CRLF is set, the output is limited to the size of BASIC-11's TTYSET value or RT-11's TT:WIDTH, whichever is smaller. If NOCRLF is set, only the current value of BASIC-11's TTYSET function limits the size of line that can be output.

6.2.5 Overlaying While in a Subroutine

Do not use either an OVERLAY command or a CHAIN command after a GOSUB and before RETURN. Either of these commands interrupting a subroutine could cause your BASIC-11 system to crash.

6.2.6 Reading a String

If your last item in a DATA statement or an input line is a null string, use two adjacent quotation marks rather than two adjacent commas. The following example shows two null strings: one is indicated by two adjacent commas; the other, by two adjacent quotation marks.

```
10 DATA 5,ABC,6,,DEF,""
```

6.2.7 REM Statements Containing Left Parenthesis

A left parenthesis in a REM statement causes deletion of subsequent spaces and periods.

To retain the spaces and periods, place two consecutive periods after the parentheses. For example:

```
1Ø REM A$(16)..ONE TWO 3
```

is printed and stored as is, while the statement

```
2Ø REM B$(16) ONE TWO 3
```

becomes:

```
2Ø REM B$(16)ONETWO3
```

6.2.8 REM Statements Terminated by a Backslash

A REM statement terminated by a backslash (\) becomes longer by one space each time a program containing it is saved by the SAVE command. If a program containing such a line is called and saved (with the OLD and SAVE commands) several times, the line can become too long for BASIC-11 to handle properly. Use the SUB command to delete the extra spaces before the line becomes too long.

6.2.9 Subscripts

BASIC-11 calculates subscripts by performing an integer (INT) function rather than by using true truncation. The integer method of calculating causes no problems for subscripts with values greater than or equal to Ø. Integer calculations are also no problem for subscripts with values lesser than or equal to -1, since BASIC-11 does not support negative subscripts. However, BASIC-11 gives the error message ?SUBSCRIPT OUT OF BOUNDS for subscripts with values between Ø and -1.

6.2.1Ø CTRLC and RCTRLC Functions

The CTRLC function allows you to use the CTRL/C command to stop execution of a BASIC-11 program without returning to the RT-11 monitor. If you type one CTRL/C, BASIC-11 interrupts your program at the next request for input. If you type two consecutive CTRL/Cs, BASIC-11 interrupts your program immediately. After interrupting your program, BASIC-11 prints:

```
^C
STOP AT LINE nnnnn

READY
```

The number nnnnn is the number of the line that was interrupted when the CTRL/C command halted the program.

If the program was already stopped, BASIC-11 prints:

```
^C
STOP

READY
```

You may need to override the effect of the CTRL/C command within your program. To disable the CTRL/C command, invoke the RCTRLC function with the following command:

```
[LET] variable = RCTRLC
```

where variable is the target variable.

When the RCTRLC function is set, typing any number of CTRL/C commands does not interrupt your program.

If you have disabled CTRL/C with the RCTRLC function and want to see if CTRL/C has been typed, use the system function SYS(6). SYS(6) initially returns zero in program execution. SYS(6) returns a one only if program execution would have been interrupted under the CTRLC function by typing one or more CTRL/C commands.

The CTRLC function is set unless you specify the RCTRLC function. You can reset the CTRLC function with the command:

```
[LET] variable = CTRLC
```

Setting the CTRLC function resets the value of SYS(6) to zero.

6.2.11 String Manipulation in Assembly Language Routines

The BASIC-11 string manipulation routines for assembly languages include:

Command	Function
$FIND	returns the location and length of the string in arg1
$ALC	allocates a temporary string
$STORE	moves the string from one place in memory to another
$DEALC	deallocates the temporary string and resets the stack

The following routine is formatted like a direct string assignment statement in BASIC-11. For example:

```
CALL CAF1('BOSTON',B$)
```

is equivalent to:

```
B$='BOSTON'
```

The following example helps you see how to set up the parameters required for an assembly language program. This example accepts two string arguments. The first argument is a string ("BOSTON" or A$) that will be stored in the variable named as the second argument (B$). If the first argument is null ("") the original contents of the named variable are preserved.

```
            .TITLE  CAF1

            .PSECT  SUBRS,RO,I

            .GLOBL  CAFTAB
CAFTAB:     .BYTE   4
            .ASCII  'CAF1'
            .EVEN
            .WORD   CAF1


            .GLOBL  $ARGER,$BOMB
            .GLOBL  $ALC,$DEALC,$FIND,$STORE

;
;           This routine takes two string arguments, the first is read,
;           the second must be writable:
;           eg.
;                   CALL CAF1('BOSTON',B$) OR
;                   CALL CAF1(A$,B$)

;
;           FUNCTION: The routine will replace the contents of the string
;                   named in the second argument with the string specified
;                   in the first argument unless the first argument is null.
;
;
CAF1:       CMPB    (R5)+,#2        ;Are there 2 arguments?
            BEQ     20$             ;Yes
10$:        JMP     $ARGER          ;No
20$:        CMPB    (R5)+,#202      ;Are we being called by B-11
                                    ;with arg descriptors?
            BNE     99$             ;No

30$:        MOV     -4(R5),R4       ;Get ptr to 1st element in adl
            JSR     PC,GETDSC       ;Get descriptor word of 1st argument
            BIC     #160200,R3      ;Is it a string?
            CMP     #1100,R3
            BNE     10$             ;No
            JSR     PC,GETDSC       ;Yes - get dw of 2nd arg
            BIC     #160000,R3      ;Is it a writable string?
            CMP     #1100,R3
            BNE     10$             ;No

60$:        MOV     -4(R5),R0       ;Move ptr to 1st adl element to R0
            MOV     (R0),R0         ;Move 1st arg's descr. ptr to R0
            ADD     #2,R0           ;Make R0 ptr to 1st arg's str ref ptr
            MOV     (R0),R0         ;Move str ref ptr to R0
            MOV     #100,R1         ;Move 100 to R1
            SUB     #2,R5           ;Restore R5
            JSR     PC,$FIND        ;Find string
            BCS     70$             ;Branch if error in $FIND

61$:        TST     R1              ;Test for null string
            BEQ     80$             ;Branch if null
            MOV     R1,R0           ;String length
            MOV     #100,R1
            JSR     PC,$ALC
            BCS     71$             ;Branch if error in $ALC
```

```
62$:    MOV     SP,R1           ;Move temp str's str ref ptr to R1:
        MOV     -2(R5),RØ       ;Move ptr to 1st adl element to RØ
        MOV     (RØ),RØ         ;Move 1st arg's descr. ptr to RØ
        ADD     #2,RØ           ;Make RØ ptr to 1st arg's str ref ptr
        MOV     (RØ),RØ         ;Move str ref ptr to RØ
        MOV     #1ØØ,R2         ;Move 1ØØ to R2
        JSR     PC,$STORE       ;Store string into temp string
        BCS     73$             ;Branch if error in $STORE

63$:    MOV     R1,RØ           ;Make RØ str ref ptr of temp str
        MOV     -2(R5),R1       ;Move ptr to 1st adl element to R1
        ADD     #2,R1           ;Make R1 point to 2nd adl element
        MOV     (R1),R1         ;Move 2nd arg's descr. ptr to R1
        ADD     #2,R1           ;Make R1 ptr to 2nd arg's str ref ptr
        MOV     (R1),R1         ;Move 2nd arg's str ref ptr to R1
        MOV     #1ØØ,R2         ;Move 1ØØ to R2
        JSR     PC,$STORE       ;Store temp string into 2nd arg
        BCS     73$             ;Branch if error in $STORE

64$:    JSR     PC,$DEALC
        BCS     72$             ;Branch if error in $DEALC
        RTS     PC              ;Return

7Ø$:    JSR     R1,$BOMB
        .ASCIZ  'ERROR IN $FIND'
        .EVEN

71$:    JSR     R1,$BOMB
        .ASCIZ  'ERROR IN $ALC'
        .EVEN

72$:    JSR     R1,$BOMB
        .ASCIZ  'ERROR IN $DEALC'
        .EVEN

73$:    JSR     R1,$BOMB
        .ASCIZ  'ERROR IN $STORE'
        .EVEN

8Ø$:    RTS     PC              ;Return

99$:    JSR     R1,$BOMB
        .ASCIZ  'ERROR IN CALLING ID'
        .EVEN


;
;GETDSC returns next argument's descriptor word (dw)

;Inputs:
;       R4 points to the word in descriptor list

;Outputs:
;       R3 contains the dw for current arg
;       R4 is updated to point to next element in adl

GETDSC: MOV     (R4)+,R3        ;Get descriptor
        BIT     #1,R3           ;Is it a pointer?
        BNE     1Ø$             ;No
        MOV     (R3),R3         ;Yes - get actual descriptor
1Ø$:    RTS     PC
        .END
```

6.2.12 String Variables

The length of a line in a virtual string array is determined by the number of characters in that line. Virtual string arrays recognize only embedded null characters. Therefore, any trailing null characters are not included in line length determination.

6.2.13 Creating and Accessing Virtual Array Files

When BASIC-11 creates an integer virtual array file, and the subscript you specify in the DIM statement is 1 or 2 less than a multiple of 256, BASIC-11 will include an extra block in the file.

The size of an array file that you can create is limited by the larger of the file size implied by the DIM statement and the file size specified by the FILESIZE option in the OPEN statement. However, you cannot access the elements of a file that exceed the boundaries of the smaller file size specification. For example, if the FILESIZE option in an OPEN FOR OUTPUT command specifies a larger file than that implied by the DIM statement, the file created is as large as the FILESIZE option specification. However, you will not be able to access all the file's elements. If you attempt to access elements beyond the smaller file boundary, BASIC-11 returns the SUBSCRIPT OUT OF BOUNDS error message.

6.2.14 In-Memory Array and Virtual Array Dimensions

In-memory arrays have a maximum subscript value of 32766 and start with element Ø, not element 1. Thus, the maximum number of elements per dimension is 32767. A one-dimensional array in the statement:

```
DIM X(4Ø)
```

will contain 41 elements. A two-dimensional array in the statement:

```
DIM Y(24,3)
```

will contain 1ØØ elements, or 25*4.

Virtual arrays have a maximum subscript value of 32766 and start with the element Ø, not element 1. Thus, as for in-memory arrays, the maximum number of elements per dimension is 32767.

The maximum subscript values for a two-dimensional virtual array must not exceed 32767. That is, (b+1)*(c+1) must be less than or equal to 32767, where b and c are the subscripts in the following statement:

```
DIM 1,Y(b,c)
```

6.3 COMPILING OPERATIONS

When you compile programs in BASIC-11, you should be aware of the restrictions described in this section. The restrictions apply to generating and preserving files.

6.3.1 Generating a File with COMPILE Command

The COMPILE command generates a file with a .BAC file type under BASIC-11 with single-precision arithmetic. Similarly, the COMPILE command generates a file with a .BAX file type under BASIC-11 with double-precision arithmetic. If you use an OLD, RUN, or CHAIN command to call a .BAC file under double-precision BASIC-11 or a .BAX file under single-precision BASIC-11, BASIC-11 can crash. This problem occurs only if you have specified the wrong file type. BASIC-11 executes properly when you rely on default file types.

6.3.2 Issuing Commands Before Compiling

Before you issue a COMPILE command or a CHAIN command, you should preserve the current program with the SAVE/REPLACE and OLD command. If variable string pointers are in order, then variable tables will be compiled in the minimum memory space.

APPENDIX A

FEATURE CHANGES FROM BASIC-11/RT-11 VERSION 1B

This appendix summarizes the differences between BASIC-11/RT-11 Version 1B and BASIC-11/RT-11 Version 2.1.

Table A-1 lists the features of BASIC-11/RT-11 Version 2.1.

Table A-1
Features of BASIC-11/RT-11 Version 2.1

Category	Name
Integer Data Type	-
Double-Precision Data Type	-
Statements	COMMON DIM# KILL LINPUT NAME ON GOTO ON GOSUB ON THEN PRINT USING RESET
Functions	ABORT CTRLC LOG1Ø PI RCTRLC RCTRLO SYS TTYSET
Commands	APPEND BYE COMPILE DEL LENGTH RESEQ SUB UNSAVE CTRL/Q CTRL/S

FEATURE CHANGES FROM BASIC-11/RT-11 VERSION 1B

Some statements, functions, and commands are not entirely new in BASIC-11/RT-11 Version 2.1, but have syntactical or functional differences from their counterparts in BASIC-11/RT-11 Version 1B. Tables A-2, A-3, and A-4 list both the old and new statements, functions, and commands. The tables also provide a brief explanation of the changes.

Table A-2
Changes in Statements from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Version	Statement
1B	CALL "routine name" [[(argument list)]]
2.1	CALL [["]]routine name[["]] [[(argument list)]] In Version 2.1, the quotation marks delimiting the routine names are optional.
1B	CHAIN "string" LINE number
2.1	CHAIN string [[LINE expression]] In Version 2.1, the line number specification can be an expression.
1B	CLOSE [[{VFn / #n}]]
2.1	CLOSE [[[#]]expr1,[[#]]expr2,[[#]]expr3,...]] In Version 2.1, the pound sign (#) is optional. You can specify more than one file, and the file specification can be an expression. The VFn syntax is no longer permitted.
1B	DATA list
2.1	DATA list In Version 2.1, unquoted strings are permitted. The DATA statement must be the last statement on a multi-statement line.

(continued on next page)

Table A-2 (Cont.)
Changes in Statements from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Version	Statement
1B	DEF FNletter (var1[,var2,...var5])=expression
2.1	DEF FNletter [{$ / %}] (var1[,var2,...,var5])=expression
	In Version 2.1, if the function returns a string, the function name (FNletter) must end with a dollar sign ($). If the function returns an integer, the function name must end with a percent sign (%).
1B	IF relational expression {THEN / GO TO} line number
	IF END #n {THEN / GO TO} line number
2.1	IF relational expression {THEN statement / THEN line number / GO TO line number}
	IF END #expr {THEN statement / THEN line number / GO TO line number}
	In Version 2.1, the THEN keyword can be followed by a statement as well as by a line number.
1B	INPUT variable1[,variable2,...]
2.1	INPUT [#expr,]variable1[,variable2,...]
	In Version 2.1, you can type more than one string on a line.
1B	INPUT #expr: variable1[,variable2,...]
2.1	INPUT [#expr,]variable1[,variable2,...]
	DIGITAL recommends that you follow the expression by a comma (,) rather than by a colon (:), even though the colon is still permitted.
1B	[LET] variable=expression
	[LET] VFn(i)=expression
2.1	[LET] variable=expression
	In Version 2.1, the VF format is not permitted.

(continued on next page)

Table A-2 (Cont.)
Changes in Statements from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Version	Statement
1B	OPEN file [[FOR {INPUT / OUTPUT}]] [[(b)]] AS FILE #digit DOUBLE BUF
	OPEN file [[FOR {INPUT / OUTPUT}]] [[(b)]] AS FILE VFdigit (dimension)=string length
2.1	OPEN string [[{FOR INPUT / FOR OUTPUT}]] AS FILE # expr DOUBLE BUF [[,RECORDSIZE expr]] [[,MODE expr]] [[,FILESIZE expr]] In Version 2.1, the file specification can be any string expression. The pound sign (#) is optional. Specify a file size by using the FILESIZE keyword. The VF format is not permitted.
1B	OVERLAY "file descriptor"
2.1	OVERLAY string [[LINE expression]] In Version 2.1, the line number specification can be an expression.
1B	PRINT #expr: list
2.1	PRINT [[#expr,]] [[list]] DIGITAL recommends that you follow the expression by a comma (,) rather than by a colon (:), even though the colon is still permitted.
1B	RANDOMIZE
2.1	RANDOMIZE In Version 2.1, the random number generator starts at a new number each time the statement is executed.
1B	REM comment
2.1	REM comment In Version 2.1, you can terminate a comment by a backslash (\).

Table A-3
Changes in Functions from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Version	Function
1B	ASC(string)
2.1	ASC(string) In Version 2.1, this function returns an 8-bit (rather than 7-bit) value.
1B	BIN[[(]]string[[)]]
2.1	BIN(string) In Version 2.1, the parentheses delimiting the argument are required.
1B	CHR$(expr)
2.1	CHR$(expr) In Version 2.1, this function accepts an 8-bit (rather than 7-bit) value.
1B	OCT[[(]]string[[)]]
2.1	OCT(string) In Version 2.1, the parentheses delimiting the argument are required.
1B	STR$(expr)
2.1	STR$(expr) In Version 2.1, this function can return more than six digits (with double precision arithmetic).

Table A-4
Changes in Commands from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Version	Command
1B	LIST[[NH]] LIST[[NH]]line number LIST[[NH]]-line number LIST[[NH]]line number - [[END]] LIST[[NH]]line number - line number
2.1	LIST[[NH]] line specification1,line specification2,... In Version 2.1, you can list more than one line range specification. The END specification is no longer allowed. The other Version 1B forms of the command are still valid in Version 2.1.
1B	NEW[["program name"]] OLD[["file specification"]] RENAME "program name" REPLACE "file specification" SAVE[["file specification"]]
2.1	NEW program name OLD file specification RENAME program name REPLACE file specification SAVE file specification In Version 2.1, these commands must not contain quotation marks that delimit the file specifications.
1B	RUN RUNNH
2.1	RUN[[NH]] RUN[[NH]] file specification In Version 2.1, you can include a file specification in the RUN and RUNNH commands.
1B	CTRL/C
2.1	CTRL/C In Version 2.1, CTRL/C returns control to BASIC (in Version 1B, it returned control to the RT-11 monitor). CTRL/C also prints the line number of the line that was executing when it interrupted the program.

FEATURE CHANGES FROM BASIC-11/RT-11 VERSION 1B

Table A-5 lists changes in miscellaneous features from BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1.

Table A-5
Changes in Miscellaneous Features from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Feature	Change
Maximum line number	In Version 2.1, the maximum line number is 32767 (in Version 1B, it was 65532).
String concatenation operator	In Version 2.1, the string concatenation operator can be either the ampersand (&) or the plus sign (+) (in Version 1B it was the ampersand).
Virtual arrays	In Version 2.1, virtual arrays are like arrays in memory (except that they are not zeroed at run time); they do not have special variable names. Virtual arrays can have one or two subscripts (Version 1B allowed just one subscript). The string length can have any value between 1 and 255; it does not have to be a power of 2. The default string length is 16 (in Version 1B, it was 32).
Terminal width	In Version 2.1, the terminal width can be set from 1 to 255 characters. See the BASIC-11/RT-11 User's Guide for a description of the TTYSET function.
Closing files	In Version 2.1, files are closed by the CLOSE or END statement (or by execution of the last line of the program) but are left open by the STOP statement. Any open output files are deleted by the RUN, SCR, OLD, and NEW commands. (In Version 1B, STOP, RUN, SCR, OLD, and NEW closed files.)
Deleting a character	In Version 2.1, the DELETE (or RUBOUT) key deletes a character; the backarrow (or underscore) key does not. (Both DELETE and backarrow were valid in Version 1B.)
Deleting a line	In Version 2.1, the CTRL/U command deletes a line; the ESCAPE (or ALTMODE) key does not. (Both CTRL/U and ESCAPE were valid in Version 1B.)

(continued on next page)

Table A-5 (Cont.)
Changes in Miscellaneous Features from
BASIC-11/RT-11 Version 1B to BASIC-11/RT-11 Version 2.1

Feature	Change
Lower case	In Version 2.1, lower case keywords and variable names are always converted to upper case. Lower case string constants and remarks are permitted.
Arithmetic hardware support	BASIC-11/RT-11 Version 2.1 supports the following hardware (Version 1B also supported this hardware): EAE KE11-A and KE11-B Extended Arithmetic Element EIS KE11-E Extended Instruction Set FPU FP11-A, FP11-B, and FP11-C Floating Point Unit Support of the following hardware is new in Version 2.1: FIS KE11-F Floating Point Instruction Set KEV-11 Extended Arithmetic Element
CALL interface	In Version 2.1, the CALL interface is similar to the FORTRAN IV CALL interface. Note that arrays are stored with the second subscript varying faster; they are stored independently of the scalar with the same name. Many internal table formats and offsets have changed. See the BASIC-11/RT-11 User's Guide for a complete description of the CALL interface.

APPENDIX B

THE CONVERSION PROGRAM

The BASIC-11 software kit contains a conversion program, SUCVT, that helps you modify programs written under BASIC-11/RT-11 Version 1B so that they execute properly under BASIC-11/RT-11 Version 2.1. Use this program to modify all Version 1B programs that contain any of the following items:

- Virtual arrays
- User-defined functions that return strings
- REM statements that include a backslash (\)
- BIN or OCT functions that do not contain parentheses
- Line numbers greater than 32767
- DATA statements on multi-statement lines
- CHR$ or STR$ functions
- RANDOMIZE statements
- CALL statements

See Appendix A for a summary of how the items listed above have changed in BASIC-11/RT-11 Version 2.1.

B.1 STARTING THE CONVERSION PROGRAM

To use the conversion program, first make sure that the following two programs, which are provided in the software kit, reside on your working volume:

1. SUCVT.BAS
2. SUCVT1.BAS

Then start BASIC-11. Respond to the READY prompt by typing:

```
RUN SUCVT
```

The conversion program first requests the name of the file you need to convert. Enter the input file specification in the standard RT-11 format (dev:filnam.typ). The program then requests the name to be

assigned to the resulting converted program. Enter the output file specification. If you type only the RETURN key for the output file name, the conversion program examines the input file but does not create a converted output file. If you omit the file type in the input or output file specification, the conversion program uses .BAS. The following is an example of the initial dialog with the conversion program:

```
RUN SUCVT

INPUT FILE NAME? CVTEST
OUTPUT FILE NAME? CVOUT
```

Next, the conversion program requests the items you need to check for Version 2.1 BASIC-11 compatibility (a complete list of the possible items is reproduced at the beginning of this Appendix):

```
WHICH FEATURES WOULD YOU LIKE TO CHECK (ALL, INDIVIDUAL SELECTION)?
```

To check all possible items for Version 2.1 compatibility, type an A. Typing only the RETURN key is equivalent to typing an A. If you type an A (or a RETURN), the conversion program begins to examine the input file.

If you type an I, the conversion program lists each feature separately. If you need to examine a particular item, type a Y (or a RETURN). If you do not need to check a feature, type an N. The following is a sample dialog:

```
WHICH FEATURES WOULD YOU LIKE TO CHECK (ALL, INDIVIDUAL SELECTION)? I
VIRTUAL ARRAYS(Y OR N) ? Y
DEF STATEMENTS(Y OR N) ? Y
REM STATEMENTS(Y OR N) ? N
BIN AND OCT USAGE(Y OR N) ? Y
STR$ AND CHR$ USAGE(Y OR N) ? Y
DATA STATEMENTS(Y OR N) ? Y
RANDOMIZE STATEMENTS(Y OR N) ? N
CALL STATEMENTS(Y OR N) ? Y
```

As a result of the dialog shown above, the conversion program would check all items except REM and RANDOMIZE statements. When you respond to the last line of the dialog, the conversion program begins to examine the input file.

B.2 INTERACTING WITH THE CONVERSION PROGRAM

As the conversion program examines the input file, it searches for the items that you requested in the dialog. When it finds an occurrence of a requested item, it prints the item (in most cases) and gives you an opportunity to modify a program line or add new program lines.

When the conversion program finds a virtual array **OPEN** statement, it prints the existing array name and requests a new array name. If you type just the RETURN key, the conversion program repeats the request. If you type LIST, the conversion program prints the program line that contains the OPEN statement, then repeats the request for a new array name. When you type a new virtual array name, the conversion program replaces all occurrences of the existing name with the new name. Make sure that the new name you type is not used already as a variable in

the program. After you enter a new virtual array name, the conversion program requests a channel number for that array. If you type only the RETURN key, the conversion program repeats the request. Make sure the channel number you enter is in the range 1 to 12 and is not used in any OPEN statement already in the program.

When the conversion program finds a **DEF** statement, it prints the existing function name and requests a new function name. If you type just the RETURN key, the conversion program retains the existing function name and continues processing the input file. If you type LIST, the conversion program prints the program line that contains the DEF statement, then repeats the request. When you type a new function name, the conversion program replaces all occurrences of the existing name with the new name. Again, make sure that the new name you type is not used in any other DEF statement in the program.

The conversion program automatically corrects program lines that contain **REM** statements with a backslash (\) and those that use the **BIN** or **OCT** functions without parentheses. For REM statements, the conversion program substitutes the words "BACK-SLASH" for the actual backslash character (\). For BIN and OCT functions, the conversion program supplies the missing parentheses. The conversion program does not request any information from you when it examines REM statements and BIN and OCT functions.

The conversion program always checks the line numbers in your program to see if there are line numbers greater than 32767. If the conversion program detects such a line number, it prints the line identified by that line number and requests a new line number for it. Respond by typing a new line number; do not retype the entire statement on that line. Note that if you do not enter a new line number that is less than or equal to 32767, the program cannot execute under Version 2.1.

When the conversion program finds a **CHR$** or **STR$** function, a **CALL** or a **RANDOMIZE** statement, or a **DATA** statement that is on a multi-statement line and is not the last item, it first prints the appropriate warning message from the following list:

WARNING! FUNCTION CHR$ HAS CHANGED

WARNING! FUNCTION STR$ HAS CHANGED

CALL INTERFACE HAS CHANGED

FUNCTION OF RANDOMIZE STATEMENT HAS CHANGED

DATA STATEMENT NOT LAST

Then the conversion program prints the line that contains the function or statement and requests a new line. If you type only the RETURN key, the conversion program leaves the existing program line unchanged. If you type a new program line, the conversion program deletes the original line and inserts the new line. The conversion program continues to request new lines so that you can properly modify your Version 1B program. Make sure that the new lines you type do not have the same line numbers as existing program lines, except for the line number of the deleted line. When you are finished, respond to the new line request by typing just the RETURN key.

When the conversion program finishes examining the input file, it asks if you need to convert another program:

```
CONVERSION COMPLETE.
ANOTHER CONVERSION?  (Y OR N) ?
```

Type a Y (or the RETURN key) to restart the conversion program. Type an N to terminate the program and return control to BASIC-11.

B.3 CONVERTING A SAMPLE PROGRAM

This section contains:

1. A sample BASIC-11/RT-11 Version 1B program,
2. A reproduction of the conversion program dialog, and
3. A listing of the resulting converted file. Read the dialog carefully. Then compare the Version 1B program with the Version 2.1 program.

Figure B-1 lists Part I of the BASIC-11/RT-11 Version 1B program called CVTEST.BAS.

```
100 DEF FNV(S)=CHR$(S)
200 DEF FNX(T)=T+T^2
300 OPEN 'TEST1.DAT' AS FILE #6
400 OPEN 'TEST2.DAT' FOR OUTPUT(4) AS FILE #4
500 OPEN 'TEST3.DAT' FOR OUTPUT AS FILE VF1(25)
600 OPEN 'TEST4.DAT' FOR OUTPUT(10) AS FILE VF2%(25)
700 OPEN 'TEST5.DAT' FOR OUTPUT AS FILE VF5$(64)
750 OPEN 'TEST6.DAT' AS FILE VF6$(6)=16
800 REM TEST PROGRAM FOR V1B
900 REM CONTAINS A \ FOLLOWEDBYMORETEXT
1000 DATA 'NOT','AT','END','OF','A','LINE'\GO TO 1100
1100 K=OCT('707')
1200 PRINT K
1250 PRINT K+VF1(4)
1300 K=OCT'707'
1400 PRINT K
1450 PRINT K+VF2(3)
1500 K=BIN('101010')
1600 PRINT K
1700 K=BIN'101010'
1800 PRINT K
1900 K=BIN'10101'+12
2000 PRINT K
2100 P$='GTY'&FNV(5)
2200 PRINT FNX(9)
2300 PRINT FNV(6),FNX(VF1(7))
2400 LET VF5(5)-'NOWIS THE '&STR$(FNX(1))
2500 RANDOMIZE
2600 CALL  'HYPOT'(G,K,Z)
2800 CLOSE VF1
2900 CLOSE VF5
60000 END
```

Figure B-1 BASIC-11/RT-11 Version 1B Program

Figure B-2 lists Part II of the sample dialog containing the conversion program. Note that all the possible features are to be checked for Version 2.1 compatibility.

```
READY
RUN SUCVT <RET>

INPUT FILE NAME? CVTEST <RET>
OUTPUT FILE NAME? CVOUT <RET>
WHICH FEATURES WOULD YOU LIKE TO CHECK (ALL, INDIVIDUAL SELECTION)? I <RET>
VIRTUAL ARRAYS(Y OR N) ? Y <RET>
DEP STATEMENTS(Y OR N) ? Y <RET>
REM STATEMENTS'(Y OR N) ? Y <RET>
BIN AND OCT USAGE(Y OR N) ? Y <RET>
STR$ AND CHR$ USAGE(Y OR N) ? Y <RET>
DATA STATEMENTS(Y OR N) ? Y <RET>
RANDOMIZE STATEMENTS(Y OR N) ? Y <RET>
CALL STATEMENTS(Y OR N) ? Y <RET>
FUNCTION NAME: FNV
NEW NAME: ? FNV$ <RET>
FUNCTION NAME: FNX
NEW NAME: ? LIST <RET>
200 DEF FNX(T)=T+T^2
NEW NAME: ? <RET>
VIRTUAL ARRAY NAME VF1
NEW NAME ? C <RET>
CHANNEL NUMBER ? 1 <RET>
VIRTUAL ARRAY NAME VF2%
NEW NAME ? C% <RET>
CHANNEL NUMBER ? 2 <RET>
VIRTUAL ARRAY NAME VF5$
NEW NAME ? C$ <RET>
CHANNEL NUMBER ? 3 <RET>
VIRTUAL ARRAY NAME VF6$
NEW NAME ? C1$ <RET>
CHANNEL NUMBER ? 5 <RET>
LINE NUMBER GREATER THAN MAX
OLD LINE: 60000 END
NEW LINE NUMBER:? 32000 <RET>
WARNING! FUNCTION CHR$ HAS CHANGED.
OLD LINE : 100 DEF FNV$(S)=CHR$(S)
NEW LINE : ? <RET>
DATA STATEMENT NOT LAST
OLD LINE : 1000 DATA 'NOT','AT','END','OF','A','LINE'\GO TO 1100
NEW LINE : ? 1000 GOTO 1100 <RET>
NEW LINE : ? 1010 DATA 'NOT','AT','END','OF','A','LINE' <RET>
NEW LINE : ? <RET>
WARNING! FUNCTION STR$ HAS CHANGED.
OLD LINE : 2400 LET C$(5)='NOWIS THE '&STR$(FNX(1))
NEW LINE : ? <RET>
FUNCTION OF RANDOMIZE STATEMENT HAS CHANGED
OLD LINE : 2500 RANDOMIZE
NEW LINE : ? 50 RANDOMIZE <RET>
NEW LINE : ? <RET>
CALL INTERFACE HAS CHANGED
OLD LINE : 2600 CALL 'HYPOT'(G,K,Z)
NEW LINE : ? <RET>
CONVERSION COMPLETE.
ANOTHER CONVERSION? (Y OR N) ? N <RET>

READY
```

Figure B-2 Conversion Program Dialog

Figure B-3 is the listing of Part III or the resulting converted program, called CVOUT.BAS:

```
OLD CVOUT <RET>

READY
LIST <RET>

5Ø RANDOMIZE
1ØØ DEF FNV$(S)=CHR$(S)
2ØØ DEF FNX(T)=T+T^2
3ØØ OPEN 'TEST1.DAT' AS FILE #6
4ØØ OPEN 'TEST2.DAT' FOR OUTPUT AS FILE #4, FILESIZE 4
5ØØ OPEN 'TEST3.DAT' FOR OUTPUT AS FILE #1 \ DIM #1,C(25)
6ØØ OPEN 'TEST4.DAT' FOR OUTPUT AS FILE #2, FILESIZE 1Ø \ DIM #2,C%(25)
7ØØ OPEN 'TEST5.DAT' FOR OUTPUT AS FILE #3 \DIM #3,C$(64)=32
75Ø OPEN 'TEST6.DAT' AS FILE #5 \ DIM #5,C1$(6)=16
8ØØ REM TEST PROGRAM FOR V1B
9ØØ REM CONTAINS A "BACK-SLASH" FOLLOWEDBYMORETEXT
1ØØØ GO TO 11ØØ
1Ø1Ø DATA 'NOT','AT','END','OF','A','LINE'
11ØØ K=OCT('7Ø7')
12ØØ PRINT K
125Ø PRINT K+C(4)
13ØØ K=OCT('7Ø7')
14ØØ PRINT K
145Ø PRINT K+C%(3)
15ØØ K=BIN('1Ø1Ø1Ø')
16ØØ PRINT K
17ØØ K=BIN('1Ø1Ø1Ø')
18ØØ PRINT K
19ØØ K=BIN('1Ø1Ø1')+12
2ØØØ PRINT K
21ØØ P$='GTY'&FNV$(5)
22ØØ PRINT FNX(9)
23ØØ PRINT FNV$(6),FNX(C(7))
24ØØ LET C$(5)='NOWIS THE '&STR$(FNX(1))
26ØØ CALL 'HYPOT'(G,K,Z)
28ØØ CLOSE 1
29ØØ CLOSE 3
32ØØØ END

READY
```

Figure B-3 BASIC-11/RT-11 Version 2.1 Program

B.4 COMPILING THE CONVERSION PROGRAM

If you need to use the conversion program frequently, compile the program so that it executes more efficiently. See Figure B-4 for the commands to do this.

```
OLD SUCVT <RET>

READY
COMPILE <RET>

READY
OLD SUCVT1 <RET>

READY
COMPILE <RET>

READY
```

Figure B-4 Compiling the Conversion Program

APPENDIX C

CONVERSION INFORMATION

C.1 FEATURES REQUIRED TO RUN THE CONVERSION PROGRAM

The following features, listed as optional in Chapter 1 of the BASIC-11/RT-11 User's Guide, are required to run SUCVT.BAS:

1. ABORT
2. ASC
3. LEN
4. POS
5. SEG$
6. VAL

C.2 SPECIFYING FILENAME DURING CONVERSION PROGRAM

When running the conversion dialog program SUCVT, specify the name of your output file in response to the dialog question OUTPUT FILENAME:. If, instead, you hit a <CR>, the program will not default to NONAME.BAS.

Change line number 290 of SUCVT.BAS to read as follows:

```
290 IF LEN (F1$)=0 THEN GOTO 280
```

C.3 CHECKING FOR ERRORS IN LOGIC DURING CONVERSION PROGRAM

The conversion program SUCVT.BAS does not check for errors in logic although it does check for errors in syntax. When you type a new channel number, for example, the conversion program does not check to see if that channel number is already in use. The conversion program replaces the old channel number with the new channel number.

Manually check the program you are converting. Make sure the names and variables you choose are not already in use in the program.

APPENDIX D

BASIC-11/RT-11 DISTRIBUTION KIT DIRECTORY

The following files are on the BASIC-11/RT-11 distribution kit:

BASIC .SAV	53	BSPR1 .OBJ	7
BASICD.SAV	57	BSPR1N.OBJ	1
BAS16K .SAV	55	BSPTRD.OBJ	6
BSASM .MAC	5	BSPTRS.OBJ	6
BSCLI .MAC	5	BSRSQ .OBJ	2
BSCLI .OBJ	1	BSRSQN.OBJ	1
BSCLIN.OBJ	1	BSRØD .OBJ	9
BSCLLB.OBJ	5	BSRØS .OBJ	8
BSCLS .OBJ	6	BSR1D .OBJ	4
BSCLSN.OBJ	1	BSR1S .OBJ	3
BSCMP .OBJ	4	BSSUB .OBJ	2
BSERML.OBJ	8	BSSUBN.OBJ	1
BSERMS.OBJ	5	BSXØA .OBJ	11
BSERR .OBJ	2	BSXØB .OBJ	8
BSEØ .OBJ	16	BSX1A .OBJ	8
BSE1 .OBJ	8	BSX1B .OBJ	7
BSFUNC.OBJ	9	BSX2 .OBJ	7
BSKEYS.OBJ	6	SUCNFG.BAS	3
BSMAC .MAC	1Ø	SUCNFG.DAT	7
BSOTØD.EAE	12	SUCNF1.BAS	7
BSOTØD.EIS	12	SUCNF2.BAS	4
BSOTØD.FPU	7	SUCNF3.BAS	4
BSOTØD.OBJ	12	SUCNF4.BAS	4
BSOTØS.EAE	1Ø	SUCNF5.BAS	5
BSOTØS.EIS	9	SUCNF6.BAS	5
BSOTØS.FIS	7	SUCNF7.BAS	5
BSOTØS.FPU	7	SUCVT .BAS	6
BSOTØS.OBJ	9	SUCVT1.BAS	14
BSOT1D.EAE	9	SUDTCM.OBJ	2
BSOT1D.EIS	9	SUEID .OBJ	1
BSOT1D.FPU	4	SUE1ID.OBJ	1
BSOT1D.OBJ	9	SUIMP .OBJ	3
BSOT1S.EAE	6	SUIOID.OBJ	1
BSOT1S.EIS	6	SUIOPO.OBJ	11
BSOT1S.FIS	6	SUIOPR.OBJ	4
BSOT1S.FPU	3	SUOPT .OBJ	5
BSOT1S.OBJ	6	SUO1ID.OBJ	1
BSPAT .OBJ	1	SUR1ID.OBJ	1
BSPRØ .OBJ	3	SUXID .OBJ	1
BSPRØN.OBJ	3	SUX2ID.OBJ	1

8Ø Files, 594 Blocks
4168 Free Blocks

INDEX

HOW TO ORDER ADDITIONAL DOCUMENTATION

From	Call	Write
Chicago	312–640–5612 8:15 A.M. to 5:00 P.M. CT	Digital Equipment Corporation Accessories & Supplies Center 1050 East Remington Road Schaumburg, IL 60195
San Francisco	408–734–4915 8:15 A.M. to 5:00 P.M. PT	Digital Equipment Corporation Accessories & Supplies Center 632 Caribbean Drive Sunnyvale, CA 94086
Alaska, Hawaii	603–884–6660 8:30 A.M. to 6:00 P.M. ET *or* 408–734–4915 8:15 A.M. to 5:00 P.M. PT	
New Hampshire	603–884–6660 8:30 A.M. to 6:00 P.M. ET	Digital Equipment Corporation Accessories & Supplies Center P.O. Box CS2008 Nashua, NH 03061
Rest of U.S.A., Puerto Rico*	1–800–258–1710 8:30 A.M. to 6:00 P.M. ET	

*Prepaid orders from Puerto Rico must be placed with the local DIGITAL subsidiary (call 809–754–7575)

Canada		
British Columbia	1–800–267–6146 8:00 A.M. to 5:00 P.M. ET	Digital Equipment of Canada Ltd 940 Belfast Road Ottawa, Ontario K1G 4C2 Attn: A&SG Business Manager
Ottawa–Hull	613–234–7726 8:00 A.M. to 5:00 P.M. ET	
Elsewhere	112–800–267–6146 8:00 A.M. to 5:00 P.M. ET	
Elsewhere		Digital Equipment Corporation A&SG Business Manager*

*c/o DIGITAL's local subsidiary or approved distributor

BASIC–11/RT–11
Installation Guide and
Release Notes
AA–K724B–TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

— Assembly language programmer
— Higher-level language programmer
— Occasional programmer (experienced)
— User with little programming experience
— Student programmer
— Other (please specify) _____

Name _____ Date _____

Organization _____ Telephone _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear — Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG/ML PUBLICATIONS, MLO5–5/E45
DIGITAL EQUIPMENT CORPORATION
146 MAIN STREET
MAYNARD, MA 01754

Do Not Tear — Fold Here

Cut Along Dotted Line

Printed in U.S.A.