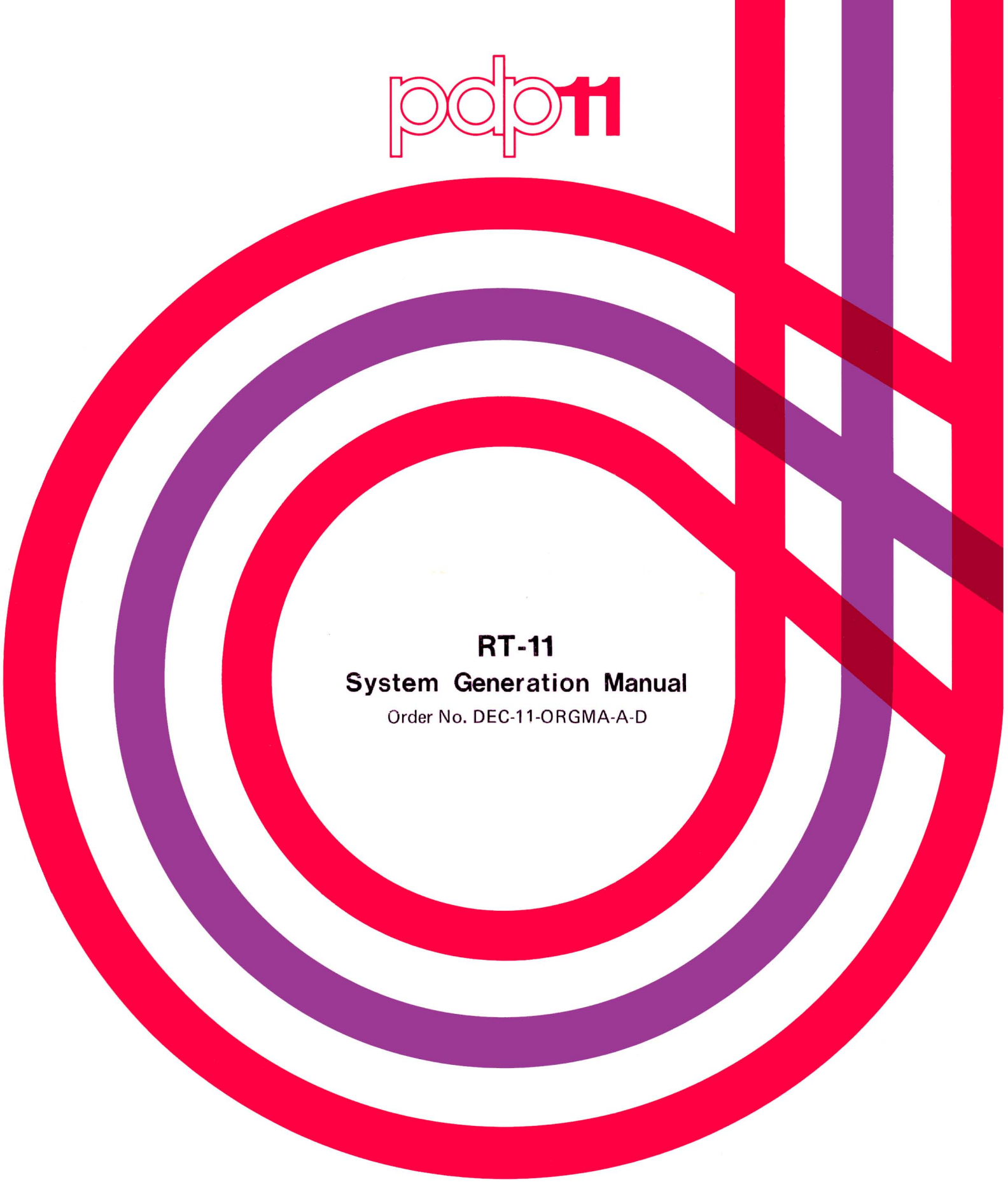

RT-11
System Generation Manual

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

digital

RT-11

System Generation Manual

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

digital equipment corporation · maynard, massachusetts

First Printing, January 1976

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

The following are trademarks of Digital Equipment Corporation:

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOMM	DECsystem-20	TYPESET-11

CONTENTS

TABLES

PREFACE

HOW TO USE THIS MANUAL

This document introduces the RT-11 V02C, RT-11 FORTRAN V01C, and BASIC/RT-11 V01B-01 software kits to users receiving them for the first time.

Chapter 1 explains the contents of the kits. Chapter 2 contains step-by-step instructions for building the systems from the distribution kits. Chapter 3 contains step-by-step instructions for executing simple demonstrations. Chapter 4 contains general building instructions and instructions for customizing RT-11 for special hardware. Chapter 5 contains instructions for assembling and linking RT-11 source files.

Immediately upon receiving your RT-11 software kits, thoroughly read this document and the RT-11 System Reference Manual (DEC-11-ORUGA-C-D,-DN1,-DN2). In addition to the RT-11 system documents, RT-11 FORTRAN users should read the PDP-11-FORTRAN Language Reference Manual (DEC-11-LFLRA-C-D) and the RT-11/RSTS/E FORTRAN IV User's Guide (DEC-11-LRRUA-A-D). Users of BASIC/RT-11 should be familiar with the BASIC/RT-11 Language Reference Manual (DEC-11-LBACA-D-D) as well as the RT-11 system documents.

Once you understand the contents of these manuals, you may build your RT-11 system according to the instructions in Chapter 2 and exercise the demonstration packages in Chapter 3. Users with special hardware considerations, or those users building nonstandard RT-11 configurations, consult Chapter 4 before building the system.

CONVENTIONS, ABBREVIATIONS, AND STANDARDS

The following are the conventions, abbreviations, and standards that are used throughout this manual.

1. All numbers are listed in octal unless otherwise indicated.

2. The following abbreviations are used:

ALT	ALTMODE or ESCAPE key
CTRL	CONTROL key
CR	RETURN key
LF	LINE FEED key

3. <CR> or <LF> indicate that the RETURN or LINE FEED key should be typed at that place in the dialogue.

4. <ALT> indicates that the ALTMODE (or ESCAPE) key should be typed at that place in the dialogue.

5. Text enclosed in square brackets, [], is optional; when including such text, do not type the square brackets unless otherwise indicated.

6. CTRL x indicates that the CONTROL key should be pressed and held down while another key, "x", is also pressed.

7. <TAB> indicates that the horizontal tab should be typed.

8. On ASR33 and ASR35 Teletype(1) terminals, special characters that are produced by holding down one key and pressing another are:

^	SHIFT N
\	SHIFT L
[	SHIFT K
]	SHIFT M
<TAB>	CTRL I

9. The sample terminal dialogue provided in this document contains version numbers where they would normally appear. The version numbers given include xx's in those fields that may vary from installation to installation. The exact contents of these fields are not of interest, as long as appropriate digits appear in the area indicated in this document. The same is true for FREE CORE messages printed by any of the system programs and for FREE BLOCKS messages included in device directories.

10. Wherever necessary, computer outputs are underlined to differentiate them from user inputs.

RELATED DOCUMENTS

See the RT-11 Documentation Directory (DEC-11-ORDDA-A-D) for information concerning related documents in the RT-11 library.

(1) Teletype is a registered trademark of the Teletype Corporation.

CHAPTER 1

OVERVIEW OF SOFTWARE KITS

1.1 RT-11 SOFTWARE KIT

The basic RT-11 Software Kit is available on six media: DECpack, DECtape, diskette, magtape, cassette, and paper tape. Each kit contains user documentation and the materials necessary to build a complete RT-11 system. The components of an RT-11 Software Kit are inventoried on checklists attached to the outside of the kit. It is recommended that the user verify the contents of the package against the checklist and report any discrepancies to DIGITAL's Software Distribution Center.

The RT-11 DECtape, DECpack, and diskette kits contain "ready-to-run" RT-11 DECtape or disk Single-Job systems; copies of the masters can be mounted and bootstrapped directly without modification.

The magtape kit contains an RT-11 magtape which is used to build an RT-11 disk system; a special program (MBUILD) is used to initialize the system disk, then to copy the requisite system files and programs over to the system disk.

The cassette kit contains RT-11 system cassettes, which are used to build an RT11 disk system; a special program (CBUILD) is used to initialize the disk, then to copy the requisite system files and programs over to the disk.

The paper tape kit consists of a special program (PT BUILD) and all system components in the form of relocatable binary object modules. PT BUILD is run to initialize the disk with a temporary monitor, linker, and paper tape handler. These tools are then used to link a complete RT-11 V02C system onto the disk.

Detailed instructions for building the system from all six media are contained in Chapter 2 of this document; general system assembly, linking, and building instructions are contained in Chapter 5. Note that the RT-11 System Reference Manual should be read thoroughly before exercising the package in Chapters 2 and 3, and the procedures in Chapters 2 and 3 should be exercised before the system is put to general use.

The contents of the RT-11 DECtape, DECpack, diskette, magtape, and cassette kits can be divided into two logical groups: system files and system programs. The system files include the monitor files and the device handlers (all with .SYS extensions), the system macro files (SYSMAC.SML, SYSMAC.8K, and VTMAC.MAC), the display support handler (VTHDLR.OBJ), and the FORTRAN system routines (SYSF4.OBJ). The monitors included are Single-Job and Foreground/Background versions

for RK11 disk, TC11 DECtape, RX11 disk, RP02 disk, RJS03/4 disk, and RF11 disk; their names are identified in Table 1-1.

Table 1-1
RT-11 Monitors

Monitor	DECpack	DECtape	Diskette	Magtape
Single-Job, RK11	MONITR.SYS	RKMNSJ.SYS	RKMNSJ.SYS	RKMNSJ.SYS
Foreground/ Background, RK11	RKMNFB.SYS	RKMNFB.SYS	RKMNFB.SYS	RKMNFB.SYS
Single-Job, RF11	RFMNSJ.SYS	RFMNSJ.SYS	RFMNSJ.SYS	RFMNSJ.SYS
Foreground/ Background, RF11	RFMNFB.SYS	RFMNFB.SYS	RFMNFB.SYS	RFMNFB.SYS
Single-Job, TC11	DTMNSJ.SYS	MONITR.SYS	*	DTMNSJ.SYS
Foreground/ Background, TC11	DTMNFB.SYS	DTMNFB.SYS	*	DTMNFB.SYS
Single-Job, RX11	DXMNSJ.SYS	DXMNSJ.SYS	MONITR.SYS	DXMNSJ.SYS
Foreground/ Background, RX11	DXMNFB.SYS	DXMNFB.SYS	DXMNFB.SYS	DXMNFB.SYS
Single-Job, RP02	DPMNSJ.SYS	DPMNSJ.SYS	DPMNSJ.SYS	DPMNSJ.SYS
Foreground/ Background, RP02	DPMNFB.SYS	DPMNFB.SYS	DPMNFB.SYS	DPMNFB.SYS
Single-Job, RJS03 RJS04	DSMNSJ.SYS	*	*	DSMNSJ.SYS
Foreground/ Background, RJS03 RJS04	DSMNFB.SYS	*	*	DSMNFB.SYS

NOTE

The monitors available on cassette are the RK11 Single-Job (MONITR.SYS) and the RK11 Foreground/Background (RKMNFB.SYS). The monitors available on paper tape are the RF11 Single-Job Monitor, the RF11 Foreground/Background Monitor, the RK11 Single-Job Monitor, and the RK11 Foreground/Background Monitor. See Section 2.7.1 for the paper tape monitor file names.

* Not available

The actual running version of the monitor is always named MONITR.SYS; to change monitors, the user must rename the monitor in use to a name other than MONITR.SYS, name the desired monitor MONITR.SYS, rewrite the bootstrap, and reboot the system. Additional instructions for modification of system files and their use can be found in Chapter 4.

The device handler files are named dev.SYS where dev is the 2-character device name for the device in question.

SYSMAC.SML, SYSMAC.8K, and VTMAC.MAC are the system macro libraries and are used when system macros are called from user-written assembly language programs. VTHDLR.OBJ is a set of display support routines used with assembly language programs that use the display processor (if any). SYSF4.OBJ is a set of FORTRAN callable subroutines that allow direct access to the RT-11 monitor capabilities from an RT-11 FORTRAN program; this set can be built into a library called SYSLIB.

Those files with .SAV extensions are the system programs; they can be executed directly with appropriate monitor commands. In all cases, these are the latest released versions of all programs, and they obsolete any other versions that may be in use. See RT-11 System Release Notes, Table 1, for the version numbers of the components in the V02C kit.

Recipients of RT-11 V02C paper tape kits must build all the files named above using the build process described in Section 2.7 of this document.

Users of RT-11 Version 2 who upgrade to V02C should upgrade their BASIC systems to V01B and their FORTRAN systems to V01C to take advantage of the improved reliability. Users of RT-11 Version 1 who upgrade to V02C directly (without using Version 2 or 2B) must upgrade their FORTRAN system to V01C and their BASIC system to V01B; unmodified FORTRAN V01 and BASIC V01 systems will not operate correctly under RT-11 V02C.

In addition to the basic system kit described above, RT-11 Source Kits and microfiche Listing Kits are available to assist in system development or modification.

1.2 FORTRAN IV SOFTWARE KIT

The basic RT-11 FORTRAN Software Kit is available on six media: DECpack, DECtape, diskette, magtape, cassette, and paper tape. Each kit contains user documentation and the materials necessary to build a complete RT-11 FORTRAN system. The components of each package are inventoried on checklists attached to the outside of the kit. It is recommended that the user verify the contents of the package against the checklist and report any discrepancies to DIGITAL's Software Distribution Center.

The RT-11 FORTRAN DECtape, DECpack, diskette, magtape, and cassette kits each contain a "ready-to-run" FORTRAN compiler and the modules necessary to build a library for each supported arithmetic option. The paper tape kit includes object modules that are used to build a running FORTRAN compiler and the appropriate library. Included in all kits is the simple demonstration program, DEMO.FOR.

Included in RT-11 FORTRAN kits, with the exception of paper tape, is the RT-11 FORTRAN Extensions Kit, which provides FORTRAN support for optional devices such as LPS-11 and VT11. Documentation for use of these functions and their installation can be found in the FORTRAN/RT-11 Extensions Manual (DEC-11-LRTEA-B-D), which is provided in the Extensions Kit.

All the kits include the PDP-11 FORTRAN Language Reference Manual (DEC-11-LFLRA-C-D), the RT-11/RSTS/E FORTRAN IV User's Guide (DEC-11-LRRUA-A-D), and this document. Specific instructions for building and exercising an RT-11 FORTRAN system are contained in Chapters 2 and 3 of this document; more general linking and assembly instructions are contained in Chapter 5. Before using RT-11 FORTRAN, it is important that the user be familiar with RT-11 itself and with the enclosed FORTRAN documentation.

RT-11 FORTRAN V01C requires RT-11 V02C to operate; it will not operate correctly under RT-11 V01-15, V02, or V02B.

Previous users of FORTRAN will notice the absence of LINKV2, LIBR, and PATCHO from this kit. The versions of these programs that were included in the V01-11 FORTRAN kit have since been obsoleted by those released with RT-11 V02, V02B, and V02C, and are no longer necessary in the FORTRAN kit. RT-11 V02 users will notice that OTSV2 and OTSV2S are not included in this FORTRAN kit; they were obsoleted by the files V2S and V2NS in this kit and are thus no longer necessary.

Included in this kit is the Software Performance Summary and recent editions of the Digital Software News; both documents list known software problems and solutions (if any) for PDP-11 software. These documents should be inspected carefully. The user should make the recommended modifications to the software and documents immediately; failure to do so may cause difficulty that could have been avoided by early correction of known problems.

In addition to the system kits described above, RT-11 FORTRAN Source Kits and microfiche Listing Kits are available to assist in system development or modification.

1.3 BASIC/RT-11 SOFTWARE KIT

BASIC/RT-11 is distributed on six media: DECpack, DECtape, diskette, magtape, cassette, and paper tape. Each kit contains the BASIC/RT-11 Language Reference Manual (DEC-11-LBACA-D-D) and all the materials necessary to build BASIC/RT-11. The components of each package are inventoried on checklists attached to the outside of the kit. It is recommended that the user verify the contents of the software package with the checklist and report any discrepancies to DIGITAL's Software Distribution Center.

The BASIC/RT-11 DECpack, DECtape, diskette, magtape, and cassette kits all contain the following "ready-to-run" versions of BASIC:

BASIC.SAV	Complete BASIC, nonoverlaid; includes string support.
BAS8K.SAV	Smallest possible version of BASIC; overlaid, no string support.

BASGT.SAV	BASIC with VT11 graphics functions included; nonoverlaid.
BASGTO.SAV	BASIC with VT11 graphics functions included; overlaid.
BASLPS.SAV	BASIC with LPS-11 support functions included; nonoverlaid.
BGTLPS.SAV	BASIC with both VT11 and LPS-11 support functions included; nonoverlaid.
BGTLPO.SAV	BASIC with both VT11 and LPS-11 support functions included; overlaid.

All these versions of BASIC are ready-to-run (under RT-11 V02C); once placed on the system device, they can be started as described in Chapter 1 of the BASIC/RT-11 Language Reference Manual. The nonoverlaid versions (BASIC.SAV, BASGT.SAV, BASLPS.SAV, and BGTLPS.SAV are linked for maximum execution speed at the expense of the extra memory necessary to keep all components resident. The overlaid versions (BAS8K.SAV, BASGTO.SAV, and BGTLPO.SAV) are linked for maximum memory efficiency at a slight expense to execution time in response to certain commands.

Files with .OBJ extensions are relocatable object modules that can be linked together to form BASIC. They are used when customizing BASIC or when adding assembly language functions (paper tape users will use these files to build a running version of BASIC on the disk). BASICR.OBJ, BASICE.OBJ, and BASICX.OBJ are object modules that have been assembled to include the alphanumeric string capability. BASNSR.OBJ, BASNSE.OBJ, and BASNSX.OBJ are the corresponding object modules assembled without the optional string support.

To build a BASIC with all features included, use the linking instructions in Section 5.12. To build a version of BASIC without string capability, substitute BASNSR for BASICR, BASNSE for BASICE, and BASNSX for BASICX.

FPMP.OBJ is the floating point package for BASIC. FPMP.EAE, FPMP.EIS, and FPMP.FPU are customized versions of that math package and support EAE, EIS, and FPU options, respectively. To build a version of BASIC that uses one of these options, use the corresponding version of FPMP wherever FPMP is called for in the build instructions.

Other files in the BASIC/RT-11 kit constitute the CALL functions for special purpose devices supported by BASIC/RT-11. For example, files labeled LPSx.xxx are the LPS function components; those labeled GTxx.xxx are used when VT11 support is desired. For additional information, consult Appendix I (for LPS-11 support) and Appendix J (for VT11 support) in the BASIC Language Reference Manual.

Instructions for building BASIC from the distribution kit can be found in Chapter 2 of this document. Users of previous releases of BASIC or RT-11 should note that the use of BASIC V01B requires RT-11 V02B or V02C for proper operation.

Users of the optional LV-11 package for BASIC should note the changes, documented in the BASIC/RT-11 Release Notes Manual (DEC-11-LBRNA-A-D), that are necessary for proper operation.

Finally, in addition to the BASIC kit described above, BASIC/RT-11 Source Kits and Listing Kits are available to assist in system development or modification. These may be ordered from the Software Distribution Center.

CHAPTER 2

SYSTEM BUILD INSTRUCTIONS

2.1 INTRODUCTION

This chapter contains the step-by-step information necessary to build an RT-11 system, including instructions for FORTRAN and BASIC users. The building procedures in this chapter assume familiarity with the RT-11 operating system; thus the user should have read the RT-11 System Reference Manual before beginning to build the system. In addition, FORTRAN users should be familiar with the contents of the PDP-11 FORTRAN Language Reference Manual and the RT-11/RSTS/E FORTRAN IV User's Guide. Users of BASIC/RT-11 should be familiar with the contents of the BASIC/RT-11 Language Reference Manual. There are six sections to this build procedure:

Section 2.2 Building and Starting from DECpack Disk (RK11)

Section 2.3 Building and Starting from DECtape (TC11)

Section 2.4 Building and Starting from Diskette (RX11/RX01)

Section 2.5 Building and Starting from Magtape (TJU16 or TM11)

Section 2.6 Building and Starting from Cassette (TA11)

Section 2.7 Building and Starting from Paper Tape (PC11)

After reading the general instructions that follow, proceed to the section that pertains to the medium upon which the system was distributed. Once the system has been built, proceed to Chapter 3 for a demonstration of the system.

Chapter 4 contains instructions for customizing RT-11 systems for specific hardware or unusual configurations (e.g., single-disk systems). Users with specific hardware considerations should refer to Chapter 4 after building their systems.

NOTE

No RT-11 system program ever HALTs with the expectation that the CONTINUE switch can be pressed to resume operation after corrective action has been taken. If the computer HALTs (the RUN light is off), a significant error has occurred and the entire section must be repeated from the beginning.

If an error not explained in this manual occurs, please refer to the RT-11 System Message Manual.

If user errors occur within a section, go back to the beginning of that particular section.

User typing errors may be corrected using the standard RT-11 input editing techniques (RUBOUT and CTRL/U).

2.2 BUILDING AND STARTING FROM DECPACK DISK

2.2.1 RT-11

This section contains instructions for those who have received RT-11 on DECpack disk. The instructions describe building an RF11, RP02, or RJS03/4 disk system from RK disk or, for RK users, duplicating the master RT-11 disk, then bootstrapping the copy for use as the system disk. RK11 and RP02 users will need a blank, formatted disk for this procedure (see Appendix B for RK11 formatting procedures).

1. Set the ENABLE/HALT switch to HALT to stop any previous program that may be running. Set the ENABLE/HALT switch to ENABLE. Mount the RT-11 master disk on Unit 0, WRITE PROTECTed.

2. If the system has a hardware bootstrap capable of bootstrapping the RK11 disk, boot the disk and proceed to Step 3; otherwise, see Section A.1, then proceed with Step 3.

3. There will be a slight pause, after which RT-11 will respond with:

```
RT-11SJ V02C-xx
.
```

```
Type:       DATE dd-mmm-yy<CR>
Response:   .
```

where dd-mmm-yy is the current date in the form 27-NOV-75.

If the master disk is being used to build an RF11 system,

```
Type:       ASSIGN RF1DK<CR>
Response:   .
```

If the master disk is being used to build an RP02 system, mount a blank, formatted disk on Unit 0, WRITE ENABLEd.

```
Type:       ASSIGN DP1DK<CR>
Response:   .
```

If the master disk is being used to build an RJS03/4 system,

Type: ASSIGN DS:DK<CR>
Response: .

If the master is being used to build an RK11 system, mount a blank, formatted disk cartridge on Unit 1, WRITE ENABLEd.

Type: ASSIGN RK1:DK<CR>
Response: .

In any case,

Type: R PIP<CR>
Response: *

Scan the master disk for readability on this drive:

Type: RK:/K<CR>
Response: *

Each block on the disk will be read and checked for errors; the process takes about four minutes.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

If the system being built is an RP02,

Type: DK:/Z/N:37<CR>

If the system being built is an RK11 or a multiple platter RJS03/4 or RF11,

Type: DK:/Z/N:12<CR>

Otherwise,

Type: DK:/Z<CR>

In any case,

Response: DK:/Z ARE YOU SURE?

Type: Y<CR>
Response: *

Type: DK:*,*=RK:*,SYS/Y/X<CR>
Response: *

4. If the system being built is RK11, go to Step 5; otherwise:

Type: DK:RKMNSJ,SYS=DK:MONITR,SYS/Y/R<CR>
Response: *

If the system being built is an RF11 system,

Type: **DK:MONITR,SYS=DK:RFMNSJ,SYS/Y/R**<CR>
Response: *

Go to Step 5.

If the system being built is an RJS03/4 system,

Type: **DK:MONITR,SYS=DK:DSMNSJ,SYS/Y/R**<CR>
Response: *

Go to Step 5.

If the system being built is an RP02 system,

Type: **DK:MONITR,SYS=DK:DPMNSJ,SYS/Y/R**<CR>
Response: *

Go to Step 6.

5. Delete all unnecessary monitor and .SYS files from the system device as follows:

Type: DK:xxMNSJ.SYS,xxMNFB.SYS/Y/D<CR>
Response: *

where xx is the designation of the device for any unnecessary monitors (as described in Table 1-1) as follows:

DP	RP02 disk
DS	RJS03/4 disk
DT	DECtape
DX	RX11 diskette
RF	RF11 disk
RK	RK11 disk

Type: DK:xx.SYS/Y/D<CR>
Response: *

where xx is the designation of the device for any unnecessary device handler on the system as follows:

Delete	If System has no
BA.SYS	more than 8K words of memory
CR.SYS	card reader
CT.SYS	cassette
DP.SYS	RP02 disk
DS.SYS	RJS03/4 disk
DX.SYS	diskette
LP.SYS	line printer
MM.SYS	TJU16 magtape
MT.SYS	TM11/TS03 or TM11/TU10 magtape
PP.SYS	paper tape punch
PR.SYS	paper tape reader
RF.SYS	RF11 disk
RK.SYS	RK11 disk

Type: **DK:/S**<CR>
Response: *

```
Type:       DK1*,*=RK1*,*/X<CR>
Response:   ?NO SYS/,BAD ACTION?
            *

Type:       DK1/K<CR>
Response:   *
```

Each block on the disk being built will be checked for errors; the process takes about four minutes on an RK11 or about one minute on a single platter RF11 or RJS03/4.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

6. Type: DK1A=DK1MONITR,SYS/U<CR>
   Response: *

7. Remove the master disk from Unit 0 and store it in a safe place. If the new system is an RK system, remove the copy from Unit 1, label it RT-11 V02C SYSTEM DISK, then mount it in Unit 0, WRITE-ENABLEd.

   If the system has a hardware bootstrap capable of bootstrapping the new system disk, boot the disk; otherwise, perform the bootstrapping procedures in the appropriate section of Appendix A.

   RT-11 should bootstrap from the new system disk and type its identifying message,

   ```
   RT-11SJ V02C-xx
   .
   ```

   If it does not, repeat this section.

8. Proceed to Section 2.2.2 if building a FORTRAN system from DECpack, Section 2.2.3 if building a BASIC system from DECpack, or consult the Table of Contents for the appropriate section if building FORTRAN or BASIC from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

2.2.2 FORTRAN IV

This section contains instructions for those who received RT-11 FORTRAN on DECpack disk. The instructions involve the transfer of the necessary FORTRAN-related files to the system disk and the creation of the FORTRAN library.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.2.1, the system is already booted and running).

   To the running RT-11 monitor,

Type: DATE dd-mmm-yy<CR>

where dd-mmm-yy is the current date in the form 27-NOV-75.

Response: .

Type: TIME hh:mm:ss<CR>

where hh:mm:ss is the current 24-hour time in hours:minutes:seconds.

Response: .

2. If an RK11 system is being built, mount the RT-11 FORTRAN IV system disk on Unit 1, WRITE-PROTECTed.

```
Type:       ASSIGN RK1↑DIS<CR>
Response:   .
```

Go to Step 3.

If an RF11, RSJ03/4, or RP02 system is being built, mount the RT-11 FORTRAN IV system disk on Unit 0, WRITE-PROTECTed.

```
Type:       ASSIGN RK0↑DIS<CR>
Response:   .
```

3.
```
Type:       R PIP<CR>
Response:   *

Type:       *,*=DIS↑*,*/X<CR>
Response:   *
```

Dismount the RT-11 FORTRAN master disk and store it in a safe place.

```
Type:       CTRL C
RESPONSE:   ↑C
            .

Type:       R LIBR<CR>
Response:   *
```

4. If the FORTRAN system will be running on a configuration that includes an EAE, proceed. Otherwise, go to Step 5.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,EAE/G<CR>
Response:   ENTRY POINT↑

Type:       $ERR↑B<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

```
Type:       FORLIB,V2S=UNI,OTSCOM,V2S,EAE/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 9.

5. If the configuration includes a PDP-11/45 processor without FPU or a PDP-11/40 processor with EIS but without FIS, proceed. Otherwise, go to step 6.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,EIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,EIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 9.

6. If the configuration includes a PDP-11/40 or PDP-11/03 processor with FIS, proceed. Otherwise, go to Step 7.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,FIS/G<CR>
Response:   ENTRY POINT!
```

```
Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S;FIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *
```

Go to Step 9.

7. If the configuration includes a PDP-11/45 processor with FPU, proceed. Otherwise, go to Step 8.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,FPU/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S;FPU/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
```

```
$FIO2   ILL INS
$FIO2   ILL INS

*
```

Go to Step 9.

8. If the configuration contains none of the above options:

```
Type:       FORLIB=UNI,OTSCOM,V2NS,NHD/G<CR>
Response:   ENTRY POINTI

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,NHD/G<CR>
Response:   ENTRY POINTI

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

9.
```
Type:       CTRL C
Response:   ^C
            .
```

Proceed to Section 2.2.3 if building a BASIC system from DECpack or consult the Table of Contents for the appropriate section if building BASIC from another media; otherwise, proceed to Chapter 3.

2.2.3 BASIC/RT-11

This section contains instructions for those who received BASIC/RT-11 on DECpack disk. The instructions involve placing a running version of BASIC on the system device.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.2.1 or 2.2.2, the system is already booted and running).

To the running RT-11 monitor,

Type: DATE dd-mmm-yy<CR>

where dd-mmm-yy is the current date in the form 27-NOV-75.

Response: .

2. If an RK11 system is being built, mount the BASIC/RT-11 system disk on Unit 1, WRITE-PROTECTed.

Type: ASSIGN RK1:DIS<CR>
Response: .

If an RF11, RJS03/4, or RP02 system is being built, mount the BASIC/RT-11 system disk on RK11 Unit 0, WRITE-PROTECTed.

Type: ASSIGN RK0:DIS<CR>
Response: .

3. Type: R PIP<CR>
Response: *

Type: *,*=DIS:BASIC.SAV,BAS8K.SAV,DEMO.BAS/X<CR>
Response: *

Dismount the BASIC/RT-11 master disk and store it in a safe place.

Type: CTRL C
Response: ^C
.

Go to Chapter 3.

2.3 BUILDING AND STARTING FROM DECTAPE

2.3.1 RT-11

This section contains instructions for those who received RT-11 on DECtape. If DECtape is to be the system device, the user is instructed how to copy the master tape. If disk (RK11, RF11, or RP02) is to be the system device, the user is instructed how to make a disk system from the master DECtape. It is important (for the user's protection) to build the system as instructed, then to store the master in a safe place.

1. Set the ENABLE/HALT switch to HALT to stop any previous program that may be running. Set the ENABLE/HALT switch to ENABLE. Mount the RT-11 System DECtape (Tape 1 of 2, DEC-11-ORTSA-E-UC1) on Unit 0, WRITE LOCKed. Ensure that no other DECtape unit is set to 0.

2. If the system has a hardware bootstrap capable of bootstrapping the DECtape, boot the DECtape and proceed to Step 3; otherwise, see Section A.2, then proceed with Step 3.

3. The DECtape should move back and forth (for several seconds) and then the following message should be displayed on the terminal:

```
RT-11SJ V02C-xx
.
```

If not, repeat this section from Step 2.

Type: DATE dd-mmm-yy<CR>

where dd-mmm-yy is the current date in the form 27-NOV-75.

Response: .

If DECtape will be the system device, proceed to Step 4. If disk will be the system device, proceed to Step 6.

4. Type: R PIP<CR>
Response: *

Mount a blank, formatted DECtape on an empty drive and set to Unit 1, WRITE ENABLEd.

Type: DT1:/Z<CR>
Response: DT1:/Z ARE YOU SURE?

Type: Y<CR>
Response: *

Type: DT1:*.*=DT0:*.*/X/Y<CR>

(The master tape will be copied onto the blank tape. The process takes about two and one half minutes.)

Response: *
Type: DT1:A=DT1:MONITR.SYS/U<CR>

(Both tapes will move as the system bootstrap is copied onto the tape on Unit 1. The process takes several seconds.)

Response: *

Type: DT1:/K<CR>
Response: *

Each block on the DECtape being built will be checked for errors. This process takes about four and one half minutes.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

Remove the master RT-11 System Tape (on Unit 0) and store in a safe place.

Remove the new DECtape from Unit 1 and label it as RT-11 SYSTEM TAPE 1.

5. Mount the RT-11 System DECtape (Tape 2 of 2, DEC-11-ORTSA-E-TC2) on Unit 0, WRITE LOCKed.

   Mount a blank, formatted DECtape on an empty drive and set to Unit 1, WRITE ENABLEd.

   Type: DT1I/Z<CR>
   Response: DT1I/Z ARE YOU SURE?

   TYPE: Y<CR>
   Response: *

   Type: DT1I*,**=DTØI*,*/X/Y<CR>

   The master tape will be copied onto the blank tape. The process takes about three minutes.

   Response: *

   Type: DT1I/K<CR>
   Response: *

   Each block on the DECtape being built will be checked for errors. This process takes about four and one half minutes.

   NOTE

   If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

   Remove the master RT-11 System Tape (on Unit 0) and store it in a safe place. Remove the new DECtape from Unit 1 and label it as RT-11 SYSTEM TAPE 2.

   Mount the newly created RT-11 SYSTEM TAPE 1 on Unit 0, WRITE-ENABLEd. If the system has a hardware bootstrap capable of bootstrapping the DECtape, boot the DECtape; otherwise, perform the bootstrapping procedures in Section A.2.

   The new system tape (now on Unit 0) will move as the system is bootstrapped from the newly made copy. RT-11 should respond by displaying the following on the terminal:

   RT-11SJ V02C-xx

   If it does not, repeat this section from Step 1.

   Proceed to Section 2.3.2 if building a FORTRAN system from DECtape, Section 2.3.3 if building a BASIC system from DECtape, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

6. Mount the RT-11 System DECtape, Tape 2 of 2 (DEC-11-ORTSA-E-UC2), on Unit 1, WRITE LOCKed.

   If the disk in use is an RK11 DECpack or RP02 disk, mount a

blank, formatted disk on Unit 0, WRITE ENABLEd. (See Appendix B for RK11 formatting procedures.)

If the system is to be built on RK11 disk,

Type: ASS RK1DK<CR>
Response: .

If the system is to be built on RF11 disk,

Type: ASS RF1DK<CR>
Response: .

If the system is to be built on an RP02 disk,

Type: ASS DP1DK<CR>
Response: .

In any case,

Type: R PIP<CR>
Response: *

If the system device will be a single platter RF11 disk,

Type: DK1/Z<CR>

If the system device will be RP02 disk,

Type: DK1/Z/N1371<CR>

Otherwise,

Type: DK1/Z/N112<CR>

In any case,

Response: DK1/Z ARE YOU SURE?

Type: Y<CR>
Response: *

Type: DK1*,*=DT01*,*/Y/X<CR>
Response: *

Type: DK1*,*=DT11*,*/Y/X<CR>
Response: *

Type: DK1DTMNSJ,SYS=DK1MONITR,SYS/Y/R<CR>
Response: *

7. If the disk being built is an RK11 disk,

Type: DK1MONITR,SYS=DK1RKMNSJ,SYS/Y/R<CR>
Response: *

Go to Step 8.

If the disk being built is an RF11 disk,

Type: DK1MONITR,SYS=DK1RFMNSJ,SYS/Y/R<CR>

Response: *

Go to Step 8.

If the disk being built is an RP02 disk,

Type: DK:MONITR.SYS=DK:DPMNSJ.SYS/Y/R<CR>
Response: *

Go to Step 9.

8. Delete all unnecessary monitor and .SYS files from the system device as follows:

Type: DK:xxMNSJ.SYS,xxMNFB.SYS/Y/D<CR>
Response: *

where xx is the designation of the device for any unnecessary monitors (as described in Table 1-1) as follows:

DP	RP02 disk
DS	RJS03/4 disk
DT	DECtape
DX	RX11 diskette
RF	RF11 disk
RK	RK11 disk

Type: DK:xx.SYS/Y/D<CR>
Response: *

where xx is the designation of the device for any unnecessary device handler on the system as follows:

Delete	If System has no
BA.SYS	more than 8K words of memory
CR.SYS	card reader
CT.SYS	cassette
DP.SYS	RP02 disk
DS.SYS	RJS03/4 disk
DX.SYS	diskette
LP.SYS	line printer
MM.SYS	TJU16 magtape
MT.SYS	TM11/TS03 or TM11/TU10 magtape
PP.SYS	paper tape punch
PR.SYS	paper tape reader
RF.SYS	RF11 disk
RK.SYS	RK11 disk

Type: DK:/S<CR>
Response: *

Type: DK:/K<CR>
Response: *

Each block on the disk being built will be checked for errors. This process takes about four minutes on an RK11 or one minute on a single platter RF11 or RJS03/4.

NOTE

If the response is anything but an asterisk, see the RT-11 System Reference Manual, Section 4.2.12.

9. Type: DK:A=DK:MONITR.SYS/U<CR>
Response: *

Type: DK:/O<CR>
Response: RT-11SJ V02C-xx
*

If it does not, repeat this entire section.

Dismount the master DECtapes from Unit 0 and Unit 1, and store in a safe place.

Proceed to Section 2.3.2 if building a FORTRAN system from DECtape, Section 2.3.3 if building a BASIC system from DECtape, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

2.3.2 FORTRAN IV

This section contains instructions for those who received RT-11 FORTRAN on DECtape. The instructions involve the transfer of the necessary FORTRAN-related files to the system device and the creation of the FORTRAN library.

To build a FORTRAN system, there must be at least 370 free blocks on the system device. If the system device is DECtape, single platter RF11 disk, or single platter RJS03/4 disk, there are not enough free blocks available; a new system device must be created before the FORTRAN files are copied.

If the system device is RK11, RP02, multiple platter RF11, or multiple platter RJS03/4, go to Step 1.

Otherwise, if the system device is DECtape, create a new DECtape as described in Section 2.3.1, then delete files so that FORTRAN will fit. If the system device is single platter RF11 or single platter RJS03/4, files must be deleted from the disk so that FORTRAN will fit.

The following files must be on the system device; all others can be deleted.

MONITR.SYS
TT.SYS (the console terminal handler)
LP.SYS (if the system has a line printer)
DT.SYS (if the system device is not DECtape)
EDIT.SAV
LIBR.SAV
LINK.SAV
PIP.SAV

The procedure for deleting files is as follows:

```
Type:       .R PIP<CR>
Response:   *
```

To delete .SYS files,

```
Type:       SY:xxxxxx.SYS/Y/D<CR>
Response:   *
```

where xxxxxx is the name of the system file to be deleted. Repeat this command string for each file to be deleted.

To delete system programs,

```
Type:       SY:xxxxxx.xxx/D<CR>
Response:   *
```

where xxxxxx.xxx is the name of the system program to be deleted.

When all unnecessary files have been deleted,

```
Type:       SY:/S<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            .
```

If the system has a hardware bootstrap capable of bootstrapping the new system device, bootstrap the device; otherwise, perform the bootstrapping procedures in the appropriate section of Appendix A.

1. Bootstrap an RT-11 V02C system. Ensure that the system device is WRITE-ENABLEd.

   ```
   Type:       DATE dd-mmm-yy<CR>
   ```

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   ```
   Response:   .
   ```

   Mount the RT-11 FORTRAN master DECtape (DEC-11-LRF4A-C-UC) on Unit 1, WRITE-LOCKed.

   ```
   Type:       R PIP<CR>
   Response:   *

   Type:       SY:*.*=DT1:FORTRA,SAV,FORTRA,HLP,DEMO,FOR/X<CR>
   Response:   *

   Type:       CTRL C
   Response:   ^C
               .

   Type:       R LIBR<CR>
   Response:   *
   ```

   NOTE

   In the following instructions, the square brackets [] are part of the command line and must be typed.

2. If the configuration includes an EAE, proceed. Otherwise, go to Step 3.

```
Type:       SY:FORLIB[140]=DT1:UNI,OTSCOM,V2NS,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FI02   ILL INS
            $CL02   ILL INS
            $FI02   ILL INS
            $FI02   ILL INS

            *

Type:       SY:FORLIB,V2S[140]=DT1:UNI,OTSCOM,V2S,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FI02   ILL INS
            $CL02   ILL INS
            $FI02   ILL INS
            $FI02   ILL INS

            *
```

Go to Step 7.

3. If the configuration includes a PDP-11/45 processor without FPU or a PDP-11/40 processor with EIS but without FIS proceed. Otherwise, go to Step 4.

```
Type:       SY:FORLIB[140]=DT1:UNI,OTSCOM,V2NS,EIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FI02   ILL INS
            $CL02   ILL INS
            $FI02   ILL INS
            $FI02   ILL INS

            *

Type:       SY:FORLIB,V2S[140]=DT1:UNI,OTSCOM,V2S,EIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>
```

```
Response:  $END2    ILL INS
           $END2    ILL INS
           $FIO2    ILL INS
           $CLO2    ILL INS
           $FIO2    ILL INS
           $FIO2    ILL INS

           *
```

Go to Step 7.

4. If the configuration includes a PDP-11/40 or PDP-11/03 processor with FIS, proceed. Otherwise, go to Step 5.

```
Type:      SY:FORLIB[140]=DT1:UNI,OTSCOM,V2NS,FIS/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2    ILL INS
           $END2    ILL INS
           $FIO2    ILL INS
           $CLO2    ILL INS
           $FIO2    ILL INS
           $FIO2    ILL INS

           *

Type:      SY:FORLIB,V2S[140]=DT1:UNI,OTSCOM,V2S,FIS/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2    ILL INS
           $END2    ILL INS
           $FIO2    ILL INS
           $CLO2    ILL INS
           $FIO2    ILL INS
           $FIO2    ILL INS

           *
```

Go to Step 7.

5. If the configuration includes a PDP-11/45 processor with FPU, proceed. Otherwise, go to Step 6.

```
Type:      SY:FORLIB[140]=DT1:UNI,OTSCOM,V2NS,FPU/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2    ILL INS
           $END2    ILL INS
           $FIO2    ILL INS
```

```
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

Type: `SY!FORLIB,V2S[140]=DT1!UNI,OTSCOM,V2S,FPU/G<CR>`
Response: `ENTRY POINT!`

Type:
```
$ERRTB<CR>
$ERRS<CR>
<CR>
```

Response:
```
$END2   ILL INS
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

Go to Step 7.

6. If the configuration contains none of the above options:

Type: `SY!FORLIB[140]=DT1!UNI,OTSCOM,V2NS,NHD/G<CR>`
Response: `ENTRY POINT!`

Type:
```
$ERRTB<CR>
$ERRS<CR>
<CR>
```

Response:
```
$END2   ILL INS
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

Type: `SY!FORLIB,V2S[140]=DT1!UNI,OTSCOM,V2S,NHD/G<CR>`
Response: `ENTRY POINT!`

Type:
```
$ERRTB<CR>
$ERRS<CR>
<CR>
```

Response:
```
$END2   ILL INS
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

7. Type: CTRL C
Response:
```
^C
.
```

Dismount the master tape from Unit 1 and store it in a safe place.

Proceed to Section 2.3.3 if building a BASIC system from DECtape or consult the Table of Contents for the appropriate section if building BASIC from another media. If BASIC will not be used on the system, proceed to Chapter 3.

2.3.3 BASIC/RT-11

This section contains instructions for those who received BASIC/RT-11 on DECtape. The instructions involve placing running versions of BASIC on the system device.

To build a BASIC system, there must be at least 80 free blocks on the system device. If the system device is a DECtape created in Section 2.3.1 or 2.3.2, there are not enough free blocks available; a new system device must be created before the BASIC files are copied.

If the system device is disk, go to Step 1. Otherwise, create a new DECtape as described in Section 2.3.1, then delete files so that BASIC will fit.

The following files must be on the system device; all others can be deleted.

```
MONITR.SYS
TT.SYS (the console terminal handler)
LP.SYS (if the system has a line printer)
EDIT.SAV
LIBR.SAV
LINK.SAV
PIP.SAV
```

The procedure for deleting files is as follows:

```
Type:       .R PIP<CR>
Response:   *
```

To delete .SYS files,

```
Type:       SY:xxxxxx.SYS/Y/D<CR>
Response:   *
```

where xxxxxx is the name of the system file to be deleted. Repeat this command string for each file to be deleted.

To delete system programs,

```
Type:       SY:xxxxxx.xxx/D<CR>
Response:   *
```

where xxxxxx.xxx is the name of the system program to be deleted.

When all unnecessary files have been deleted,

```
Type:       SY:/S<CR>
Response:   *
```

```
Type:       CTRL C
Response:   ^C
            .
```

If the system has a hardware bootstrap capable of bootstrapping the new system DECtape, bootstrap the DECtape; otherwise, perform the bootstrapping procedures in Section A.2.

1. Bootstrap an RT-11 V02C system. Ensure that the system device is WRITE-ENABLEd.

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

2. Mount the BASIC/RT-11 System DECtape 1 (DEC-11-LBACA-D-UC1) on Unit 1, WRITE LOCKed.

3. Type: R PIP<CR>
   Response: *

   Type: *,*=DT1:BASIC,SAV,BAS8K,SAV,DEMO,BAS/X<CR>
   Response: *

   Dismount the master DECtape and store it in a safe place.

   Type: CTRL C
   Response: ^C
   .

   Proceed to Chapter 3.

2.4 BUILDING AND STARTING ON DISKETTE

2.4.1 RT-11

This section contains instructions for those who received RT-11 on diskette. If diskette is to be the system device, the user is instructed how to copy the master diskette. If another disk (RK11, RF11, or RP02) is to be the system device, the user is instructed how to make the larger disk system from the master diskette. It is important (for the user's protection) to build the system as instructed, then store the master in a safe place.

1. Mount the RT-11 master diskette (Disk 1 of 2, DEC-11-ORTSA-E-YC1) in the left-hand drive (Unit 0). Set the ENABLE/HALT switch to HALT to stop any previous program that may be running. Set the ENABLE/HALT switch to ENABLE.

2. If the system has a hardware bootstrap capable of bootstrapping the diskette, boot the diskette and proceed to Step 3; otherwise, see Section A.3, then proceed to Step 3.

3. There is a slight pause after which the following message is displayed on the terminal:

If not, repeat this section from Step 2.

Type: DATE dd-mmm-yy<CR>

where dd-mmm-yy is the current date in the form 27-NOV-75.

Users of diskette as the system device, proceed to Step 4. Users of other disks as the system device, proceed to Step 6.

4. Type: R PIP<CR>
Response: *

Mount a blank diskette in Unit 1.

Type: DX1:/Z<CR>
Response: DX1: /Z ARE YOU SURE?

Type: Y<CR>
Response: *

Type: DX1:*.*=DX0:*.*/X/Y<CR>

(The system will be copied onto the blank diskette The process takes about one minute.)

Response: *
Type: DX1:A=DX1:MONITR.SYS/U<CR>

(The system bootstrap will be copied onto the diskette on Unit 1.)

Response: *

Type: DX1:/K <CR>
Response: *

Each block of the diskette being built will be checked for errors. This process takes about 30 seconds.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

Remove the master RT-11 System Disk and store in a safe place. Dismount the new disk from Unit 1, label it as RT-11 SYSTEM DISK 1.

5. Mount the RT-11 System Disk (Disk 2 of 2, DEC-11-ORTSA-E-YC2) on Unit 0. Mount a blank diskette in Unit 1.

Type: DX1:/Z<CR>
Response: DX1: /Z ARE YOU SURE?

Type: Y<CR>
Response: *

Type: DX1:*,*=DX0:*,*/Y/X<CR>

(The system will be copied onto the blank disk. The process takes about one minute.)

Response: *

Type: DX1:/K <CR>
Response: *

Each block of the disk being built will be checked for errors. This process takes about 30 seconds.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

Remove the master RT-11 System Disk from Unit 0 and store it in a safe place. Remove the disk from Unit 1 and label it RT-11 SYSTEM DISK 2.

Mount the newly created RT-11 SYSTEM DISK 1 on Unit 0.

Bootstrap the new disk with the hardware bootstrap or the bootstrapping procedure in Section A.3.

RT-11 will boot off the new disk and display the following on the terminal:

RT-11SJ V02C-xx
.

If it does not, repeat this section from Step 1.

Proceed to Section 2.4.2 if building a FORTRAN system from diskette, Section 2.4.3 if building a BASIC system from diskette, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

6. Mount the RT-11 System Disk, Disk 2 of 2 (DEC-11-ORTSA-E-YC2) on Unit 1.

If the disk is being used to build a system for RK11 DECpack or RP02 disk, mount a blank, formatted disk on Unit 0, WRITE ENABLEd. (See Appendix B for RK11 formatting procedures.)

If the system is to be built on RK11 disk,

Type: ASSIGN RK:DK<CR>
Response: .

If the system is to be built on RF11 disk,

Type: ASSIGN RF:DK<CR>
Response: .

If the system is to be built on RP02 disk,

Type: ASSIGN DP1DK<CR>
Response: .

In any case,

Type: R PIP<CR>
Response: *

If the system device will be RP02 disk,

Type: DK1/Z/N:37<CR>

If the system device will be a single platter RF11 disk,

Type: DK1/Z<CR>

Otherwise,

Type: DK1/Z/N:12<CR>

In any case,

Response: DK1/Z ARE YOU SURE?

Type: Y<CR>
Response: *

Type: DK1*,*=DX01*,*/Y/X<CR>
Response: *

Type: DK1*,*=DX11*,*/Y/X<CR>
Response: *

Type: DK1DXMNSJ,SYS=DK1MONITR,SYS/Y/R<CR>
Response: *

7. If the disk being built is an RK11,

Type: DK1MONITR,SYS=DK1RKMNSJ,SYS/Y/R<CR>
Response: *

Go to Step 8.

If the disk being built is an RF11,

Type: DK1MONITR,SYS=DK1RFMNSJ,SYS/Y/R<CR>
Response: *

Go to Step 8.

If the disk being built is an RP02,

Type: DK1MONITR,SYS=DK1DPMNSJ,SYS/Y/R<CR>
Response: *

Go to Step 10.

8. Delete all unnecessary monitor and .SYS files from the system device as follows:

```
Type:       DK:xxMNSJ.SYS,xxMNFB.SYS/Y/D<CR>
Response:   *
```

where xx is the designation of the device for any unnecessary monitors as follows:

DP	RP02 disk
DS	RJS03/4 disk
DT	DECtape
DX	RX11 diskette
RF	RF11 disk
RK	RK11 disk

```
Type:       DK:xx.SYS/Y/D<CR>
Response:   *
```

where xx is the designation of the device for any unnecessary device handler on the system as follows:

Delete	If System has no
BA.SYS	more than 8K words of memory
CR.SYS	card reader
CT.SYS	cassette
DP.SYS	RP02 disk
DS.SYS	RJS03/4 disk
DT.SYS	DECtape
LP.SYS	line printer
MM.SYS	TJU16 magtape
MT.SYS	TM11/TS03 or TM11/TU10 magtape
PP.SYS	paper tape punch
PR.SYS	paper tape reader
RF.SYS	RF11 disk
RK.SYS	RK11 disk

```
Type:       DK!/S<CR>
Response:   *
```

9.
```
Type:       DK!/K<CR>
Response:   *
```

Each block of the disk being built will be checked for errors. This process takes about four minutes for RK11 or about one minute for single platter RF11 or RJS03/4.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

10.
```
Type:       DK!A=DK!MONITR.SYS/U<CR>
Response:   *

Type:       DK!/O<CR>
Response:   RT-11SJ V02C-xx
            .
```

If it does not, repeat this entire section.

Dismount the master diskettes from Units 0 and 1 and store them in a safe place..

Proceed to Section 2.4.2 if building a FORTRAN system from diskette, Section 2.3.3 if building a BASIC system from diskette, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

2.4.2 FORTRAN IV

This section contains instructions for those who received RT-11 FORTRAN on diskette. The instructions involve the transfer of the necessary FORTRAN-related files to the system device and the creation of the FORTRAN library.

To build a FORTRAN system, there must be at least 370 free blocks on the system device. If the system device is diskette, single platter RF11 disk, or single platter RJS03/4 disk, there are not enough free blocks available; a new system device must be created before the FORTRAN files are copied.

If the system device is RK11, RP02, multiple platter RF11, or multiple platter RJS03/4, go to Step 1.

Otherwise, if the system device is diskette, create a new diskette as described in Section 2.4.1, then delete files so that FORTRAN will fit. If the system device is single platter RF11 or single platter RJS03/4, files must be deleted from the disk so that FORTRAN will fit.

The following files must be on the system device; all others can be deleted.

```
MONITR.SYS
TT.SYS (the console terminal handler)
LP.SYS (if the system has a line printer)
DX.SYS (if the system device is not diskette)
EDIT.SAV
LIBR.SAV
LINK.SAV
PIP.SAV
```

The procedure for deleting files is as follows:

```
Type:      .R PIP<CR>
Response:  *
```

To delete .SYS files,

```
Type:      SY:xxxxxx.SYS/Y/D<CR>
Response:  *
```

where xxxxxx is the name of the system file to be deleted. Repeat this command string for each file to be deleted.

To delete system programs,

```
Type:      SY:xxxxxx.xxx/D<CR>
```

Response: *

where xxxxxx.xxx is the name of the system program to be deleted.

When all unnecessary files have been deleted,

Type: SY:/S<CR>
Response: *

Type: CTRL C
Response: ^C
.

If the system has a hardware bootstrap capable of bootstrapping the new system device, bootstrap the device; otherwise, perform the bootstrapping procedures in the appropriate section of Appendix A.

1. Bootstrap an RT-11 V02C system.

   Mount the RT-11 FORTRAN master diskette (DEC-11-LRF4A-C-YC) on Unit 1.

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

   Type: TIME hh:mm:ss<CR>

   where hh:mm:ss is the current 24-hour time in the form hours:minutes:seconds.

   Response: .

   Type: R PIP<CR>
   Response: *

   Type: SY:*.*=DX1:FORTRA.SAV,FORTRA.HLP,DEMO.FOR/X<CR>
   Response: *

   Type: CTRL C
   Response: ^C
   .

   Type: R LIBR<CR>
   Response: *

NOTE

In the following instructions, the square brackets [] are part of the command line and must be typed.

2. If the configuration includes an EAE, proceed. Otherwise, go to Step 3.

   Type: SY:FORLIB[140]=DX1:UNI,CTSCOM,V2NS,EAE/G<CR>
   Response: ENTRY POINT:

```
Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FI02    ILL INS
            $CL02    ILL INS
            $FI02    ILL INS
            $FI02    ILL INS

            *
```

If the system device is diskette, go to Step 7; otherwise,

```
Type:       SY:FORLIB,V2S[140]=DX1:UNI,OTSCOM,V2S,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FI02    ILL INS
            $CL02    ILL INS
            $FI02    ILL INS
            $FI02    ILL INS

            *
```

Go to Step 7.

3. If the configuration includes a PDP-11/45 processor without FPU or a PDP-11/40 processor with EIS but without FIS, proceed. Otherwise, go to Step 4.

```
Type:       SY:FORLIB[140]=DX1:UNI,OTSCOM,V2NS,EIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FI02    ILL INS
            $CL02    ILL INS
            $FI02    ILL INS
            $FI02    ILL INS

            *
```

If the system device is diskette, go to Step 7; otherwise,

```
Type:       SY:FORLIB,V2S[140]=DX1:UNI,OTSCOM,V2S,EIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>
```

```
Response:  $END2    ILL INS
           $END2    ILL INS
           $FI02    ILL INS
           $CL02    ILL INS
           $FI02    ILL INS
           $FI02    ILL INS

           *
```

Go to Step 7.

4. If the configuration includes a PDP-11/40 or PDP-11/03 processor with FIS, proceed. Otherwise, go to Step 5.

```
Type:      SY:FORLIB[140]=DX1:UNI,OTSCOM,V2NS,FIS/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2    ILL INS
           $END2    ILL INS
           $FI02    ILL INS
           $CL02    ILL INS
           $FI02    ILL INS
           $FI02    ILL INS

           *
```

If the system device is diskette, go to Step 7; otherwise,

```
Type:      SY:FORLIB,V2S[140]=DX1:UNI,OTSCOM,V2S,FIS/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2    ILL INS
           $END2    ILL INS
           $FI02    ILL INS
           $CL02    ILL INS
           $FI02    ILL INS
           $FI02    ILL INS

           *
```

Go to Step 7.

5. If the configuration includes a PDP-11/45 processor with FPU, proceed.

```
Type:      SY:FORLIB[140]=DX1:UNI,OTSCOM,V2NS,FPU/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2    ILL INS
```

```
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

If the system device is diskette, go to Step 7; otherwise,

Type: SY:FORLIB,V2S[140]=DX1:UNI,OTSCOM,V2S,FPU/G<CR>
Response: ENTRY POINT:

Type:
```
$ERRTB<CR>
$ERRS<CR>
<CR>
```

Response:
```
$END2   ILL INS
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

Go to Step 7.

6. If the configuration contains none of the above options:

Type: SY:FORLIB[140]=DX1:UNI,OTSCOM,V2NS,NHD/G<CR>
Response: ENTRY POINT:

Type:
```
$ERRTB<CR>
$ERRS<CR>
<CR>
```

Response:
```
$END2   ILL INS
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

If the system device is diskette, go to Step 7; otherwise,

Type: SY:FORLIB,V2S[140]=DX1:UNI,OTSCOM,V2S,NHD/G<CR>
Response: ENTRY POINT:

Type:
```
$ERRTB<CR>
$ERRS<CR>
<CR>
```

Response:
```
$END2   ILL INS
$END2   ILL INS
$FI02   ILL INS
$CL02   ILL INS
$FI02   ILL INS
$FI02   ILL INS

*
```

7. Type: CTRL C
   Response: ^C
   .

Proceed to Section 2.4.3 if building a BASIC system from diskette or consult the Table of Contents for the appropriate section if building BASIC from another media. If BASIC will not be used on the system, go to Chapter 3.

2.4.3 BASIC/RT-11

This section contains instructions for those who received BASIC/RT-11 on diskette. The instructions involve placing running versions of BASIC on the system device.

To build a BASIC system, there must be at least 80 free blocks on the system device. If the system device is a diskette created in Section 2.4.1 or 2.4.2, there are not enough free blocks available; a new system device must be created before the BASIC files are copied.

If the system device is not diskette, go to Step 1. Otherwise, create a new diskette as described in Section 2.4.1, then delete files so that BASIC will fit.

The following files must be on the system device; all others can be deleted.

```
MONITR.SYS
TT.SYS (the console terminal handler)
LP.SYS (if the system has a line printer)
EDIT.SAV
LIBR.SAV
LINK.SAV
PIP.SAV
```

The procedure for deleting files is as follows:

```
Type:      .R PIP<CR>
Response:  *
```

To delete .SYS files,

```
Type:      SY:xxxxxx.SYS/Y/D<CR>
Response:  *
```

where xxxxxx is the name of the system file to be deleted. Repeat this command string for each file to be deleted.

To delete system programs,

```
Type:      SY:xxxxxx.xxx/D<CR>
Response:  *
```

where xxxxxx.xxx is the name of the system program to be deleted.

When all unnecessary files have been deleted,

```
Type:      SY:/S<CR>
Response:  *
```

```
Type:       CTRL C
Response:   ^C
            .
```

If the system has a hardware bootstrap capable of bootstrapping the new system diskette, bootstrap the diskette; otherwise, perform the bootstrapping procedures in Section A.3.

1. Bootstrap an RT-11 V02C system.

2. Mount the BASIC/RT-11 master diskette 1 of 2 (DEC-11-LBACA-C-YC1) on Unit 1.

   ```
   Type:       DATE dd-mmm-yy<CR>
   ```

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   ```
   Response:   .
   ```

3. ```
   Type:       R PIP<CR>
   Response:   *
   ```

   ```
   Type:       *,*=DX1:BASIC.SAV,BAS8K.SAV,DEMO.BAS/X<CR>
   Response:   *
   ```

   Dismount the master disk and store it in a safe place.

   ```
   Type:       CTRL C
   Response:   ^C
               .
   ```

   Proceed to Chapter 3.

2.5 BUILDING AND STARTING FROM MAGTAPE

2.5.1 RT-11

This section contains instructions for those who received RT-11 on magtape. The instructions describe building an RK11, RF11, RP02, or RJS03/4 disk system from the master magtape. It is important (for the user's protection) to build the system as instructed, then to store the master in a safe place.

1. Set the ENABLE/HALT switch to HALT to stop any previous program that may be running. Set the ENABLE/HALT switch to ENABLE.

   Mount the RT-11 System magtape (DEC-11-ORTSA-E-MC7 for 7-track tape or DEC-11-ORTSA-E-MC9 for 9-track tape) on Unit 0, ensuring that no write ring is inserted in the back of the tape reel. If TM11, ensure that the magtape is positioned at the load point; if it is not, manually rewind the tape.

2. If the system has a hardware bootstrap capable of bootstrapping the magtape, boot the magtape and proceed to Step 3; otherwise, see Section A.4, then proceed with Step 3.
   ```

3. The magtape will move as the magtape build program is loaded.

   A program called MSBOOT should respond by displaying the following message on the terminal:

```
MSBOOT V01-xx
*
```

   If not, repeat this section from Step 2.

   Type: MBUILD.xxx

   where xxx indicates the type of interface and is one of the following:

   MT1 builds RK11, RP02, or RJS03/4 from TM11/TS03 and TM11/TU10 magtapes

   MT2 builds RF11 disk from TM11/TS03 and TM11/TU10 magtapes

   MM1 builds RK11, RP02, or RJS03/4 from TJU16 magtape

   MM2 builds RF11 disk from TJU16 magtape

4. The magtape will move as the specified MBUILD program is loaded. MBUILD should respond by displaying the following message:

```
MBUILD V02-xx
*
```

   If it does not, repeat this Section from Step 2.

5. If the system being built is an RK11 DECpack or an RP02 disk cartridge, mount a blank, formatted disk on Unit 0, WRITE ENABLEd. Formatting procedures are explained in Appendix B.

   If the system device will be RK11 disk,

   Type: RK1/Z/N:12<CR>
   Response: RK1 /Z ARE YOU SURE?

   Type: Y<CR>
   Response: *

   Go to Step 6.

   If the system device will be a single platter RF11 disk,

   Type: RF1/Z<CR>

   If the system device will be multiple platter RF11 disk,

   Type: RF1/Z/N:12<CR>

   In either case,

   Response: RF1 /Z ARE YOU SURE?

   Type: Y<CR>
   Response: *

   Go to Step 6.

If the system device will be RP02 disk,

```
Type:       DP1/Z/N:37<CR>
Response:   DP1 /Z ARE YOU SURE?

Type:       Y<CR>
Response:   *
```

Go to Step 6.

If the system device will be a single platter RJS03/4 disk,

```
Type:       DS1/Z<CR>
```

If the system device will be a multiple platter RJS03/4 disk,

```
Type:       DS1/Z/N:12<CR>
```

In either case,

```
Response:   DS1 /Z  ARE YOU SURE?

Type:       Y<CR>
Response:   *
```

6. If the master magtape is on TM11,

```
Type:       xx*.*=MT0:*.*/Y/X/Q/M:1000<CR>
```

If the master magtape is on TJU16,

```
Type:       xx*.*=MM0:*.*/X/Y/Q/M:1000<CR>
```

where xx is the designation of the device to be the system device as follows:

DP	RP02 disk
DS	RJS03/4 disk
RF	RF11 disk
RK	RK11 disk

MBUILD will respond by printing the names of each system component. Type Y<CR> after the component name if you desire to transfer it to the system device. Type <CR> after the component name if you do not desire to transfer it to the system device.

Minimally, the system device must contain MBOOT.BOT, MSBOOT.BOT, MTINIT.SAV, the appropriate MBUILD file (selected in Step 3), the appropriate monitors (as described in Table 1-1), TT.SYS and the other necessary device handlers, and the desired system programs. The following example shows the building of a typical RK11 disk as the system device from TJU16 magtape master:

```
MSBOOT.BOT?Y<CR>
MBUILD.MT1?<CR>
MBUILD.MT2?<CR>
MBUILD.MM1?Y<CR>
MBUILD.MM2?<CR>
RKMNSJ.SYS?Y<CR>
RKMNFB.SYS?Y<CR>
RFMNSJ.SYS?<CR>
RFMNFB.SYS?<CR>
```

```
DTMNSJ,SYS?<CR>
DTMNFB,SYS?<CR>
DPMNSJ,SYS?<CR>
DPMNFB,SYS?<CR>
DXMNSJ,SYS?<CR>
DXMNFB,SYS?<CR>
DSMNSJ,SYS?<CR>
DSMNFB,SYS?<CR>
DX     .SYS?<CR>
DP     .SYS?<CR>
DT     ,SYS?<CR>
RK     ,SYS?<CR>
RF     ,SYS?<CR>
TT     ,SYS?Y<CR>
LP     ,SYS?Y<CR>
CR     ,SYS?<CR>
MT     .SYS?<CR>
MM     ,SYS?Y<CR>
PR     ,SYS?<CR>
PP     ,SYS?<CR>
CT     ,SYS?<CR>
DS     ,SYS?<CR>
BA     ,SYS?Y<CR>
SYSMAC.SML?Y<CR>
SYSMAC,8K ?<CR>
BATCH  ,SAV?Y<CR>
EDIT   ,SAV?Y<CR>
MACRO  ,SAV?Y<CR>
ASEMBL,SAV?Y<CR>
EXPAND,SAV?Y<CR>
CREF   ,SAV?Y<CR>
LINK   ,SAV?Y<CR>
LIBR   ,SAV?Y<CR>
PIP    ,SAV?Y<CR>
FILEX  ,SAV?Y<CR>
SRCCOM,SAV?Y<CR>
DUMP   ,SAV?Y<CR>
PATCH  ,SAV?Y<CR>
PATCHO,SAV?Y<CR>
MTINIT,SAV?Y<CR>
ODT    ,OBJ?Y<CR>
VTHDLR,OBJ?Y<CR>
SYSF4  ,OBJ?Y<CR>
VTMAC  ,MAC?Y<CR>
DEMOFG,MAC?Y<CR>
DEMOBG,MAC?Y<CR>
KB     ,MAC?Y<CR>
MBOOT  ,BOT?Y<CR>
```

NOTE

A maximum of 54 files can be tranferred with this procedure. If more than 54 files are specified to be transferred, a ?COR OVR? message will appear and you must restart the selection procedure.

7. If the system device will be RK11 disk,

   Response:
   ```
   ?REBOOT?
   *
   ```

```
Type:       RK1MONITR,SYS=RK1RKMNSJ,SYS/Y/R<CR>
Response:   ?REBOOT?
            *

Type:       RK1A=RK1MONITR,SYS/U<CR>
Response:   *

Type:       RK1/O<CR>
```

Go to Step 8.

If the system device will be RF11 disk,

```
Response:   *

Type:       RF1MONITR,SYS=RF1RFMNSJ,SYS/Y/R<CR>
Response:   *

Type:       RF1A=RF1MONITR,SYS/U<CR>
Response:   *

Type:       RF1/O<CR>
```

Go to Step 8.

If the system device will be RJS03/4 disk,

```
Response:   *

Type:       DS1MONITR,SYS=DS1DSMNSJ,SYS/Y/R<CR>
Response:   *

Type:       DS:A=DS:MONITR.SYS/U<CR>
Response:   *

Type:       DS1/O<CR>
```

Go to Step 8.

If the system device will be RP02 disk,

```
Response:   *

Type:       DP1MONITR,SYS=DP1DPMNSJ,SYS/Y/R<CR>
Response:   *

Type:       DP1A=DP1MONITR,SYS/U<CR>
Response:   *

Type:       DP1/O<CR>
```

8. RT-11 should bootstrap off the new disk and respond with its identifying message:

```
RT-11SJ V02C-xx
.
```

If it does not, repeat this entire section.

If it does, remove the master magtape from Unit 0 and store it in a safe place.

Type: DATE dd-mmm-yy

where dd-mmm-yy is the current date in the form 27-NOV-75.

If RP02 is the new system device, go to Step 10.

If RK11 is the new system device,

Type: ASSIGN RK1DK<CR>

Go to Step 9.

If RF11 is the new system device,

Type: ASSIGN RF1DK<CR>

Go to Step 9.

If RSJ03/4 is the new system device,

Type: ASSIGN DS1DK<CR>

9. Response: .

Type: R PIP<CR>
Response: *

Type: DK1/K<CR>
Response: *

Each block of the new system device will be read for errors. This process takes about four minutes on RK11 or one minute on a single platter RF11 or RJS03/4.

NOTE

If the response is anything but the above, see the RT-11 System Reference Manual, Section 4.2.12.

Type: CTRL C
Response: ^C
.

10. Proceed to Section 2.5.2 if building a FORTRAN system from magtape, Section 2.5.3 if building a BASIC system from magtape, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

2.5.2 FORTRAN IV

This section contains instructions for those who received RT-11 FORTRAN on magtape. The instructions involve the transfer of the necessary FORTRAN-related files to the system device and the creation of the FORTRAN library.

The system device onto which FORTRAN is being built must have at least 370 free blocks. The monitor, the appropriate magtape handler (MM.SYS or MT.SYS), TT.SYS, LP.SYS (if the configuration includes a line printer), PIP.SAV, LINK.SAV, and LIBR.SAV are the only system components necessary on the system device.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.5.1, the system is already booted and running).

   Mount the RT-11 FORTRAN master magtape (DEC-11-LRF4A-A-MC7 for 7-track tape or DEC-11-LRF4A-A-MC9 for 9-track tape) on Unit 0.

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

   Type: TIME hh:mm:ss<CR>

   where hh:mm:ss is the current 24-hour time in the form hours:minutes:seconds.

   Response: .

   Type: R PIP<CR>
   Response: *

   If the master magtape is on TM11,

   Type: SY:*.*=MT0:*.*/X/M:1000<CR>

   If the master magtape is on TJU16,

   Type: SY:*.*=MM0:*.*/X/M:1000<CR>

   In either case,

   Response: *

   Type: CTRL C
   Response: *C
   .

   Type: R LIBR<CR>
   Response: *

   NOTE

   In the following instructions, the square brackets [] are part of the command line and must be typed.

2. If the configuration includes an EAE, proceed. Otherwise, go to Step 3.

   Type: SY:FORLIB[140]=SY:UNI,OTSCOM,V2NS,EAE/G<CR>
   Response: ENTRY POINT:

   Type: $ERRTB<CR>
   $ERRS<CR>
   <CR>

```
Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *

Type:      SY:FORLIB,V2S[140]=SY:UNI,OTSCOM,V2S,EAE/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

Go to Step 7.

3. If the configuration includes a PDP-11/45 processor without FPU or a PDP-11/40 processor with EIS but without FIS, proceed. Otherwise, go to Step 4.

```
Type:      SY:FORLIB[140]=SY:UNI,OTSCOM,V2NS,EIS/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *

Type:      SY:FORLIB,V2S[140]=SY:UNI,OTSCOM,V2S,EIS/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

Go to Step 7.

4. If the configuration includes a PDP-11/40 or PDP-11/03 processor with FIS, proceed. Otherwise, go to Step 5.

```
Type:       SY:FORLIB[140]=SY:UNI,OTSCOM,V2NS,FIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *

Type:       SY:FORLIB,V2S[140]=SY:UNI,OTSCOM,V2S,FIS/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *
```

Go to Step 7.

5. If the configuration includes a PDP-11/45 processor with FPU, proceed.

```
Type:       SY:FORLIB[140]=SY:UNI,OTSCOM,V2NS,FPU/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *

Type:       SY:FORLIB,V2S[140]=SY:UNI,OTSCOM,V2S,FPU/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>
```

```
Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

Go to Step 7.

6. If the configuration contains none of the above options:

```
Type:      SY:FORLIB[140]=SY:UNI,OTSCOM,V2NS,NHD/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *

Type:      SY:FORLIB,V2S[140]=SY:UNI,OTSCOM,V2S,NHD/G<CR>
Response:  ENTRY POINT:

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

7.
```
Type:      CTRL C
Response:  ^C
           .
```

Proceed to Section 2.5.3 if building a BASIC system from magtape or consult the Table of Contents for the appropriate section if building BASIC from another media. If BASIC will not be used on the system, go to Chapter 3.

2.5.3 BASIC/RT-11

This section contains instructions for those who received BASIC/RT-11 on magtape. The instructions involve placing running versions of BASIC on the system device.

The system device onto which BASIC is being built must have at least 80 free blocks. The monitor, the appropriate magtape handler (MM.SYS or MT.SYS), and PIP.SAV are the only system components necessary on this system device.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.5.1 or 2.5.2, the system is already booted and running).

2. Mount the BASIC/RT-11 master magtape 1 of 2 (DEC-11-LBACA-A-MC7 for 7-track tape or DEC-11-LBACA-A-MC9 for 9-track tape) on Unit 0.

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

3. Type: R PIP<CR>
   Response: *

   If the master magtape is on MT11,

   Type: *,*=MT0:BASIC.SAV,BAS8K.SAV,DEMO.BAS/X/M:1000<CR>

   If the master magtape is on TJU16,

   Type: *,*=MM0:BASIC.SAV,BAS8K.SAV,DEMO.BAS/X/M:1000<CR>

   In either case,

   Response: *

   Dismount the master magtape and store it in a safe place.

   Type: CTRL C
   Response: *C
   .

   Proceed to Chapter 3.

2.6 BUILDING AND STARTING FROM CASSETTE

2.6.1 RT-11

This section contains instructions for those who received RT-11 on cassettes. The instructions describe the building of an RK11 disk system from cassette. RK11 users will need a blank, formatted disk for this procedure. See Appendix B for formatting procedures.

Prior to inserting any master cassette during this procedure, WRITE PROTECT the data by placing the orange tabs so that the holes are uncovered. Cassette Unit 0 is on the left and cassette Unit 1 is on the right.

1. Set the ENABLE/HALT switch to HALT to stop any previous program that may be running. Set the ENABLE/HALT switch to ENABLE.

Mount a blank, formatted disk cartridge in Unit 0, WRITE ENABLEd. (See Appendix B for formatting procedures.)

Insert the RT-11 System Cassette, Tape 1 of 5 (DEC-11-ORTSA-E-TC1), into Unit 0.

2. If the system has a hardware bootstrap capable of bootstrapping the cassette, boot the cassette and proceed to Step 3; otherwise, see Section A.5, then proceed with Step 3.

3. The cassette will move as the cassette build program is loaded.

   CBUILD should respond by displaying the following message on the terminal:

```
CBUILD V01-02
*
```

   If it does not, repeat this section from Step 1.

   Press the rewind button on the cassette drive.

```
Type:       RK1/Z/N:12<CR>
Response:   ARE YOU SURE?

Type:       Y<CR>
Response:   *

Type:       RK:*,*=CT0:*,*/Y/X<CR>
Response:   ?REBOOT?
            *

Type:       RK:A=CT0:MONITR.SYS/U<CR>
Response:   *
```

4. Insert the RT-11 System Cassette, Tape 2 of 5 (DEC-11-ORTSA-E-TC2), into cassette Unit 1. WRITE PROTECT it before insertion (as described above).

   Press the rewind button on the cassette drive.

```
Type:       RK:*,*=CT1:*,*/Y/X<CR>
Response:   ?REBOOT?
            *
```

5. Remove the RT-11 System Cassette, Tape 2 of 5, from Unit 1 and mount the RT-11 System Cassette, Tape 3 of 5 (DEC-11-ORTSA-E-TC3), in Unit 1, WRITE PROTECTed.

```
Type:       RK:*,*=CT1:*,*/X<CR>
Response:   *
```

6. Remove the RT-11 System Cassette, Tape 3 of 5, from Unit 1 and mount the RT-11 System Cassette, Tape 4 of 5 (DEC-11-ORTSA-E-TC4), in Unit 1, WRITE PROTECTed.

```
Type:       RK:*,*=CT1:*,*/X<CR>
Response:   *
```

7. Remove the System Cassette, Tape 4 of 5, from Unit 1 and mount the RT-11 System Cassette, Tape 5 of 5 (DEC-11-ORTSA-E-TC5), in Unit 1, WRITE PROTECTed.

```
Type:       RK1*,*=CT1:*,*/X/Y<CR>
Response:   ?REBOOT?
            *
```

8. Remove the master tapes from Units 0 and 1.

```
Type:       RK1/O<CR>
Response:   RT-11SJ V02C-xx
            .
```

The system is now running from the disk. If the above message is not displayed, repeat this section.

Proceed to Section 2.6.2 if building a FORTRAN system from cassette, Section 2.6.3 if building a BASIC system from cassette, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

2.6.2 FORTRAN IV

This section contains instructions for those who received RT-11 FORTRAN on cassette. The instructions involve the transfer of the necessary FORTRAN-related files to the system device and the creation of the FORTRAN library.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.6.1, the system is already booted and running).

```
Type:       DATE dd-mmm-yy<CR>
```

where dd-mmm-yy is the current date in the form 27-NOV-75.

```
Response:   .

Type:       R PIP<CR>
Response:   *
```

Mount the RT-11 FORTRAN IV System Cassette, Tape 1 of 4 (DEC-11-LRF4A-C-TC1), in cassette Unit 0. Before loading the cassette, make certain that it is WRITE PROTECTed.

```
Type:       *,*=CT0:*,*/X/M:1000<CR>
Response:   *
```

2. Remove the system cassette, Tape 1 of 4, from Unit 0 and mount the RT-11 FORTRAN IV System Cassette, Tape 2 of 4 (DEC-11-LRF4A-C-TC2), in Unit 0, WRITE PROTECTed.

```
Type:       *,*=CT0:*,*/X/M:1000<CR>
Response:   *
```

3. Remove the system cassette, Tape 2 of 4, from Unit 0 and mount the RT-11 FORTRAN IV System Cassette, Tape 3 of 4 (DEC-11-LRF4A-C-TC3), in Unit 0, WRITE PROTECTed.

```
Type:       *,*=CT0:*,*/X/M:1000<CR>
Response:   *
```

4. Remove the system cassette, Tape 3 of 4, from Unit 0 and mount the RT-11 FORTRAN IV System Cassette, Tape 4 of 4 (DEC-11-LRF4A-C-TC4), in Unit 0, WRITE PROTECTed.

```
Type:       *,*=CT0:*,*/X/M:1000<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            *
```

5. Store all four master cassettes in a safe place.

```
Type:       R LIBR<CR>
Response:   *
```

6. If the FORTRAN system will be running on a configuration that includes an EAE, proceed. Otherwise, go to Step 7.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 11.

7. If the configuration includes a PDP-11/45 processor without FPU or a PDP-11/40 processor with EIS but without FIS, proceed. Otherwise, go to Step 8.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,EIS/G<CR>
Response:   ENTRY POINT:
```

```
Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,EIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 11.

8. If the configuration includes a PDP-11/40 or PDP-11/03 processor with FIS, proceed. Otherwise, go to Step 9.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,FIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,FIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 11.

9. If the configuration includes a PDP-11/45 processor with FPU, proceed. Otherwise, go to step 10.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,FPU/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,FPU/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *
```

Go to Step 11.

10. If the configuration contains none of the above options:

```
Type:       FORLIB=UNI,OTSCOM,V2NS,NHD/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2    ILL INS
            $END2    ILL INS
            $FIO2    ILL INS
            $CLO2    ILL INS
            $FIO2    ILL INS
            $FIO2    ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,NHD/G<CR>
Response:   ENTRY POINT!
```

Type: $ERR↑B<CR>
$ERRS<CR>
<CR>

Response: $END2 ILL INS
$END2 ILL INS
$FIO2 ILL INS
$CLO2 ILL INS
$FIO2 ILL INS
$FIO2 ILL INS

*

11. Type: CTRL C
Response: ↑C
.

Proceed to Section 2.6.3 if building a BASIC system from cassette or consult the Table of Contents for the appropriate section if building BASIC from another media. If BASIC will not be used on the system, go to Chapter 3.

2.6.3 BASIC/RT-11

This section contains instructions for those who received BASIC/RT-11 on cassettes. The instructions involve placing running versions of BASIC on the system device.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.6.1 or 2.6.2, the system is already booted and running).

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

2. WRITE PROTECT the BASIC/RT-11 master cassettes by placing the orange tabs so that the holes are uncovered.

3. Place the BASIC/RT-11 Cassette 1 of 6 (DEC-11-LBACA-C-TC1) into Unit 0.

   Press the rewind button on the cassette drive.

4. Type: R PIP<CR>
   Response: *

   Type: *,*=CT0:*,*/X/M:1000<CR>
   Response: *

5. Dismount the master cassette on Unit 0 and store it in a safe place.

   Type: CTRL C
   Response: ↑C
   .

   Go to Chapter 3.

2.7 BUILDING AND STARTING ON PAPER TAPE

2.7.1 RT-11

This section contains instructions for those who received RT-11 on paper tape. The instructions first describe loading the Absolute Loader into memory at the 8K boundary, then detail the building of an RK11 or RF11 system from paper tape. RK11 users will need a blank, formatted disk for this procedure.

1. Check to see if the Bootstrap Loader is in memory, starting at 37744. If not, deposit it starting at 37744. (Instructions for checking and depositing the Bootstrap Loader are in Section A.6.)

   Load the Absolute Loader (DEC-11-UABLA-A-PO) into the paper tape reader. Press the FEED button until the special leader is over the read head.

   Set 37744 in the Switch Register. (Set switches 13 through 5 and 2 to the up (1) position. Set all other switches to the down (0) position.)

   Press the LOAD ADDR switch.

   Press the START switch.

   The paper tape should pass through the reader and then the computer should halt.

   If the system being built is an RK11 system, mount a formatted disk cartridge in Unit 0, WRITE ENABLEd. Place the paper tape labeled RK PT BUILD Tape 1 of 3 (DEC-11-ORPBA-D-PB1) in the reader.

   If the system being built is an RF11 system, place the tape labeled RF PT BUILD Tape 3 of 3 (DEC-11-ORPBA-D-PB3) in the reader.

   Set the starting address of the Absolute Loader, 37500, in the Switch Register. (Set switches 13 through 8 and 6 to the up (1) position. Set all other switches to the down (0) position.)

   Press the LOAD ADDR switch.

   Press the START switch.

   The paper tape should pass through the reader and the following message should be displayed on the terminal:

   ```
   PT BUILD Vxx-xx
   ```

   There is a slight pause, after which PT BUILD displays the following on the terminal.)

   ```
   PLACE SECOND TAPE IN READER,
   STRIKE ANY CHARACTER TO CONTINUE,
   ```

   Place the paper tape labeled PT BUILD Tape 2 of 3 (DEC-11-ORPBA-D-PB2) in the reader. Press the FEED button until blank leader (unpunched tape preceding the punched

data) is over the read head. (PT BUILD Tape 2 of 3 serves as the second tape for both RK and RF systems.)

Strike any character on the keyboard to start the read operation.

The paper tape passes through the reader and the following message is displayed on the terminal. (There is a slight pause between reading the tape and displaying the message.)

```
                RT-11 BUILD COMPLETE
                RT-11 V01-15    (A rudimentary V01-15 monitor
                .                is used to build V02C)
     Type:      DATE dd-mmm-yy<CR>
```

(where dd-mmm-yy is the current date in the form 27-NOV-76.)

Response: .

If the console terminal is an ASR or KSR Teletype, a VT50 Alphanumeric Display, a parallel LA30, or an LA36 DECwriter II, go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 300 baud, go to Step 2.

If the console terminal is a serial LA30 DECwriter operating at 300 baud,

Type: **D 56=5015**<CR>
Response: .

Go to Step 2.

If the console terminal is a serial LA30 DECwriter operating at 150 baud,

Type: **D 56=2015**<CR>
Response: .

Go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 2400 baud,

Type: **D 56=2012**<CR>
Response: .

Go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 1200 baud,

Type: **D 56=1012**<CR>
Response: .

Go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 600 baud,

Type: **D 56=412**<CR>
Response: .

2. Type: R LINK<CR>
   Response: *

Place the paper tape labeled OLDPIP.OBJ (DEC-11-OROPA-D-PR) in the reader. Press the FEED button until blank leader is over the read head.

Type: OLDPIP=PR:<CR>
Response: ↑

(The V01 system being used to build V02C will respond with an up-arrow or circumflex whenever it is ready to read a tape.)

Strike any character on the keyboard to initiate the read operation.

The paper tape should pass through the reader.

Response: ↑

Reload OLDPIP.OBJ in the reader so it can be read a second time.

Strike any character on the keyboard to initiate the read operation.

The paper tape should pass through the reader.

Response: *

Type: CTRL C
Response: ↑C
↑

Type: R OLDPIP<CR>
Response: *

3. Place the paper tape labeled PATCH.OBJ (DEC-11-ORPAA-E-PR) in the reader. Press the FEED button until blank leader is over the read head.

For this and all the following paper tape instructions in Step 4, the convention xxxxxx.xxx is used to represent the name on the tape label underneath the RT-11 version number. In this first example, the command below will look like PATCH.OBJ=PR:/B<CR>.

Type: xxxxxx.xxx=PR:/B<CR>
Response: ↑

Strike any character on the keyboard to initiate the read operation.

The paper tape should pass through the reader.

Response: *

NOTE

During the following operations, a ?CHK SUM? message displayed during a paper tape input operation means an input error has occurred. Retry the tape if such a message is received.

4. Repeat Step 3 for the each of the following paper tapes. Note that only those system components that are needed should be transferred to the system device; see Section 4.5 for further information.

EDIT.OBJ	(DEC-11-ORTEA-E-PR1)
ODT.OBJ	(DEC-11-ORODA-E-PR)
TT.OBJ	(DEC-11-ORTTA-E-PR)
LP.OBJ	(DEC-11-ORTLA-E-PR)
PP.OBJ	(DEC-11-ORTPA-E-PR)
PREXEC.OBJ	(DEC-11-OREXA-E-PR1)
PREPAS.OBJ	(DEC-11-OREXA-E-PR2)
SMEXEC.OBJ	(DEC-11-ORTAA-E-PR1)
SMMAC.OBJ	(DEC-11-ORTAA-E-PR2)
SMPST.OBJ	(DEC-11-ORTAA-E-PR3)
RTEXEC.OBJ	(DEC-11-ORMAA-E-PR1)
RTMAC.OBJ	(DEC-11-ORMAA-E-PR2)
RTPST.OBJ	(DEC-11-ORMAA-E-PR3)
VTCED1.OBJ	(DEC-11-ORTEA-E-PR2)
VTCED4.OBJ	(DEC-11-ORTEA-E-PR3)
VTBEDT.OBJ	(DEC-11-ORTEA-E-PR4)
LINK0.OBJ	(DEC-11-ORLLA-E-PR1)
LNKOV1.OBJ	(DEC-11-ORLLA-E-PR2)
LNKOV2.OBJ	(DEC-11-ORLLA-E-PR3)
LNKOV3.OBJ	(DEC-11-ORLLA-E-PR4)
LNKOV4.OBJ	(DEC-11-ORLLA-E-PR5)
LNKOV5.OBJ	(DEC-11-ORLLA-E-PR6)
LNKV2B.OBJ	(DEC-11-ORLLA-E-PR7)
LIBR0.OBJ	(DEC-11-ORLBA-E-PR1)
LIBR1.OBJ	(DEC-11-ORLBA-E-PR2)
LIBR2.OBJ	(DEC-11-ORLBA-E-PR3)
LIBR3.OBJ	(DEC-11-ORLBA-E-PR4)
LIBR4.OBJ	(DEC-11-ORLBA-E-PR5)
DUMP.OBJ	(DEC-11-ORDMA-E-PR)
SRCCOM.OBJ	(DEC-11-OSRCA-E-PR)
FILEX.OBJ	(DEC-11-ORFLA-E-PR)
PIP.OBJ	(DEC-11-ORPPA-E-PR)
PIP1.OBJ	(DEC-11-ORPPA-E-PR2)
CREF.OBJ	(DEC-11-ORCFA-E-PR)
VTHDLR.OBJ	(DEC-11-OVTHA-E-PR)
RKBTSJ.OBJ	(DEC-11-ORBTA-E-PR4)
RKBTFB.OBJ	(DEC-11-ORBTA-E-PR2)
RFBTSJ.OBJ	(DEC-11-ORBTA-E-PR3)
RFBTFB.OBJ	(DEC-11-ORBTA-E-PR1)
RT11SJ.OBJ	(DEC-11-ORMNA-E-PR2)
RT11FB.OBJ	(DEC-11-ORMNA-E-PR1)
RK.OBJ	(DEC-11-ORKHA-E-PR)
RF.OBJ	(DEC-11-ORFHA-E-PR)
PR.OBJ	(DEC-11-ORPHA-E-PR)
MT.OBJ	(DEC-11-OMTHA-E-PR)
CT.OBJ	(DEC-11-OCAHA-E-PR)
CR.OBJ	(DEC-11-OCRHA-E-PR)
DS.OBJ	(DEC-11-ORJSA-E-PR)
MM.OBJ	(DEC-11-OTUMA-E-PR)
BA.OBJ	(DEC-11-ORBHA-E-PR)
BATCH.OBJ	(DEC-11-ORBCA-E-PR)
SYSF4.OBJ	(DEC-11-ORLIA-E-PR)

Repeat Step 3 for each of the following paper tapes, using /A in place of /B in the command string.

VTMAC.MAC	(DEC-11-OVTMA-E-PA)
SYSMAC.SML	(DEC-11-ORSYA-E-PA1)
SYSMAC.8K	(DEC-11-ORSYA-E-PA2)

```
DEMOFG.MAC    (DEC-11-ORDFA-E-PA)
DEMOBG.MAC    (DEC-11-ORDSA-E-PA)

Type:       CTRL C
Response:   ^C
            .

Type:       R LINK<CR>
Response:   *

Type:       LINK=LINK0/B:500/C<CR>
Response:   *

Type:       LNK0V1/0:1/C<CR>
Response:   *

Type:       LNK0V2/0:1/C<CR>
Response:   *

Type:       LNKV2B/0:1/C<CR>
Response:   *

Type:       LNK0V3/0:1/C<CR>
Response:   *

Type:       LNK0V4/0:1/C<CR>
Response:   *

Type:       LNK0V5/0:1<CR>
```

(At this point, many ADDITIVE REF messages will appear on the terminal. These are expected and should be ignored.)

```
Response:   *

Type:       CTRL C
Response:   ^C
            .

Type:       R LINK<CR>
Response:   *

Type:       LIBR=LIBR0/C<CR>
Response:   *

Type:       LIBR1/0:1/C<CR>
Response:   *

Type:       LIBR2/0:1/C<CR>
Response:   *

Type:       LIBR3/0:1/C<CR>
Response:   *

Type:       LIBR4/0:1<CR>
Response:   *

Type:       PATCH=PATCH<CR>
Response:   *

Type:       TT.SYS=TT<CR>
Response:   *
```

```
Type:       PR,SYS=PR<CR>
Response:   *

Type:       PP,SYS=PP<CR>
Response:   *
```

If the configuration includes a line printer,

```
Type:       LP,SYS=LP<CR>
Response:   *
```

If the configuration includes a card reader,

```
Type:       CR,SYS=CR<CR>
Response:   *
```

If the configuration includes TM11/TS03 or TM11/TU10 magtape,

```
Type:       MT,SYS=MT<CR>
Response:   *
```

If the configuration includes TM02/TJU16 magtape,

```
Type:       MM,SYS=MM<CR>
Response:   *
```

If the configuration includes RF11 disk,

```
Type:       RF,SYS=RF<CR>
Response:   UNDEF GLBLS
            *
```

If the configuration includes RJS03 or RJS04 disk,

```
Type:       DS,SYS=DS<CR>
Response:   UNDEF GLBLS
            *
```

If the configuration includes RK11 disk,

```
Type:       RK,SYS=RK<CR>
Response:   UNDEF GLBLS
            *
```

If the configuration includes cassette,

```
Type:       CT,SYS=CT<CR>
Response:   *
```

In any case,

```
Type:       PIP=PIP/C<CR>
Response:   *

Type:       PIP1/O11<CR>
Response:   *

Type:       EXPAND=PREXEC,PREPAS<CR>
Response:   *

Type:       ASEMBL=SMEXEC,SMMAC,SMPST<CR>
Response:   *
```

```
Type:       DUMP=DUMP<CR>
Response:   *

Type:       FILEX=FILEX<CR>
Response:   *

Type:       SRCCOM=SRCCOM<CR>
Response:   *

Type:       EDIT=VTCED1,VTCED4,VTBEDT,EDIT<CR>
Response:   *
```

If the system includes more than 8K words of memory,

```
Type:       MACRO=RTEXEC,RTMAC,RTPST<CR>
Response:   *

Type:       CREF=CREF<CR>
Response:   *

Type:       BATCH=BATCH<CR>
Response:   *

Type:       BA,SYS=BA<CR>
Response:   *
```

If the system being built is on an RK11 disk,

```
Type:       MONITR,SYS=RKBTSJ,RT11SJ,RK<CR>
Response:   *

Type:       RKMNFB,SYS=RKBTFB,RT11FB,RK<CR>
Response:   *
```

If the system being built is on an RF11 disk,

```
Type:       MONITR,SYS=RFBTSJ,RT11SJ,RF<CR>
Response:   *

Type:       RFMNFB,SYS=RFBTFB,RT11FB,RF<CR>
Response:   *
```

In either case,

```
Type:       CTRL C
Response:   ^C
            .

Type:       R OLDPIP<CR>
Response    *
```

If the new system is on RF11 disk,

```
Type:       RF1A=RF1MONITR,SYS/U<CR>
Response:   *
```

If the new system is on RK11 disk,

```
Type:       RK1A=RK1MONITR,SYS/U<CR>
```

In either case,

```
Response:   *
```

5. If the system has a hardware bootstrap capable of bootstrapping the new system disk, boot the disk; otherwise, perform the bootstrapping procedures in the appropriate section of Appendix A.

   RT-11 should bootstrap off the new disk and respond with its identifying message:

```
RT-11SJ V02C-xx
.
```

   If the response is not the system identification message, repeat the entire section.

```
Type:       DATE dd-mmm-yy
Response:   .
```

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   At this point, the files listed below may be deleted from the disk as they are no longer needed. To delete a disk file named xxxxxx.xxx:

```
Type:       R PIP<CR>
Response:   *

Type:       xxxxxx.xxx/D<CR>
Response:   *
```

   The deletable files are:

```
OLDPIP.SAV          LIBR1.OBJ
EDIT.OBJ            LIBR2.OBJ
TT.OBJ              LIBR3.OBJ
LP.OBJ              LIBR4.OBJ
PP.OBJ              DUMP.OBJ
PREXEC.OBJ          SRCCOM.OBJ
PREPAS.OBJ          FILEX.OBJ
SMEXEC.OBJ          PIP.OBJ
SMMAC.OBJ           PIP1.OBJ
SMPST.OBJ           CREF.OBJ
RTEXEC.OBJ          RKBTSJ.OBJ
RTMAC.OBJ           RKBTFB.OBJ
RTPST.OBJ           RFBTSJ.OBJ
VTCED1.OBJ          RFBTFB.OBJ
VTCED4.OBJ          RT11SJ.OBJ
VTBEDT.OBJ          RT11FB.OBJ
LINK0.OBJ           RK.OBJ
LNKOV1.OBJ          RF.OBJ
LNKOV2.OBJ          PR.OBJ
LNKV2B.OBJ          MT.OBJ
LNKOV3.OBJ          CT.OBJ
LNKOV4.OBJ          CR.OBJ
LNKOV5.OBJ          DS.OBJ
LIBR0.OBJ           MM.OBJ
BATCH.OBJ           BA.OBJ
```

When all the unnecessary files have been deleted,

```
Type:       CTRL C
Response:   ^C
            .
```

Paper tape users who do not have FORTRAN cannot build the program PATCHO, as PATCHO requires the FORTRAN library to link. Paper tape users who do have FORTRAN systems should proceed to Section 2.7.2 and then link PATCHO as described in Chapter 5.

Paper tape users should take special note of the program DIREXT which is described in Section 4.5.1. It can be used to extend the size of a disk that has already been built.

Proceed to Section 2.7.2 if building a FORTRAN system from paper tape, Section 2.7.3 if building a BASIC system from paper tape, or consult the Table of Contents for the appropriate section if building from another media. If neither FORTRAN nor BASIC will be used on the system, go to Chapter 3.

2.7.2 FORTRAN IV

This section contains instructions for those who received RT-11 FORTRAN on paper tape. The instructions involve the transfer of the FORTRAN-related paper tapes to the system device, then the linking of the FORTRAN files from their components. Final instructions involve creation of the FORTRAN library.

The system device onto which FORTRAN is being built must have at least 370 free blocks. The monitor, TT.SYS, LP.SYS (if the configuration includes a line printer), PIP.SAV, LINK.SAV, and LIBR.SAV are the only system components that must be on this system device.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.7.1, the system is already booted and running).

   To the running RT-11 monitor,

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

   Type: R PIP<CR>
   Response: *

2. Place the paper tape labeled FROOT.OBJ (DEC-11-LRF4A-C-PR1) in the reader. Press the FEED button until blank leader is over the read head.

   For this, and all the following paper tape instructions in Step 3, the convention xxxxxx.xxx is used to represent the name on the tape label below the FORTRAN version number. In this first example, the command below would be FROOT.OBJ=PR:/B<CR>.

   Type: xxxxxx.xxx=PR:/B<CR>

   The tape is read.

   Response: *

NOTE

A ?CHK SUM? message displayed during a paper tape input operation means an input error has occurred. Retry the operation if such a message is received.

3. Repeat Step 2 for the paper tapes labeled:

```
F0.OBJ        (DEC-11-LRF4A-C-PR2)
F1.OBJ        (DEC-11-LRF4A-C-PR3)
F2.OBJ        (DEC-11-LRF4A-C-PR4)
F3.OBJ        (DEC-11-LRF4A-C-PR5)
F4.OBJ        (DEC-11-LRF4A-C-PR6)
F5.OBJ        (DEC-11-LRF4A-C-PR7)
F6.OBJ        (DEC-11-LRF4A-C-PR8)
F7.OBJ        (DEC-11-LRF4A-C-PR9)
F8.OBJ        (DEC-11-LRF4A-C-PR10)
F9.OBJ        (DEC-11-LRF4A-C-PR11)
F10.OBJ       (DEC-11-LRF4A-C-PR12)
F11.OBJ       (DEC-11-LRF4A-C-PR13)
F12.OBJ       (DEC-11-LRF4A-C-PR14)
F13.OBJ       (DEC-11-LRF4A-C-PR15)
F14.OBJ       (DEC-11-LRF4A-C-PR16)
F15.OBJ       (DEC-11-LRF4A-C-PR17)
F16.OBJ       (DEC-11-LRF4A-C-PR18)
F17.OBJ       (DEC-11-LRF4A-C-PR19)
OTSCOM.OBJ    (DEC-11-LRF4A-C-PR20)
UNI.OBJ       (DEC-11-LRF4A-C-PR26)
V2S.OBJ       (DEC-11-LRF4A-C-PR27)
V2NS.OBJ      (DEC-11-LRF4A-C-PR28)
```

Repeat Step 2 for the following paper tapes, using /A in place of /B in the command string.

```
FORTRA.HLP    (DEC-11-LRF4A-C-PA1)
DEMO.FOR      (DEC-11-LRF4A-C-PA2)
```

4. If the configuration contains an EAE, repeat Step 2 for the paper tape:

```
EAE.OBJ       (DEC-11-LRF4A-C-PR22)
```

If the configuration contains a PDP-11/40 with EIS or a PDP-11/45 without FPU, repeat Step 2 for the paper tape:

```
EIS.OBJ       (DEC-11-LRF4A-C-PR23)
```

If the configuration contains a PDP-11/40 or PDP-11/03 with FIS, repeat Step 2 for the paper tape:

```
FIS.OBJ       (DEC-11-LRF4A-C-PR24)
```

If the configuration contains a PDP-11/45 with FPU, repeat Step 2 for the paper tape:

```
FPU.OBJ       (DEC-11-LRF4A-C-PR25)
```

If the configuration does not contain any of the above options, repeat Step 2 for the paper tape:

NHD.OBJ (DEC-11-LRF4A-C-PR21)

5. Type: CTRL C
Response: ^C
*

Type: R LINK<CR>
Response: *

Type: FORTRA=FROOT/C<CR>
Response: *

Type: F0/011/C<CR>
Response: *

Type: F1/011/C<CR>
Response: *

Type: F2/011/C<CR>
Response: *

Type: F3/011/C<CR>
Response: *

Type: F4/011/C<CR>
Response: *

Type: F5/011/C<CR>
Response: *

Type: F6/011/C<CR>
Response: *

Type: F7/011/C<CR>
Response: *

Type: F8/011/C<CR>
Response: *

Type: F9/011/C<CR>
Response: *

Type: F10/011/C<CR>
Response: *

Type: F11/011/C<CR>
Response: *

Type: F12/011/C<CR>
Response: *

Type: F13/011/C<CR>
Response: *

Type: F14/011/C<CR>
Response: *

Type: F15/011/C<CR>
Response: *

```
Type:       F16/011/C<CR>
Response:   *

Type:       F17/011<CR>
Response:   ADDITIVE REF OF WRNBAS
            AT SEGMENT # 000003
            ADDITIVE REF OF WRNBAS
            AT SEGMENT # 000004
            ADDITIVE REF OF WRNBAS
            AT SEGMENT # 000004
            ADDITIVE REF OF WRNBAS
            AT SEGMENT # 000004
            ADDITIVE REF OF WRNBAS
            AT SEGMENT # 000004

            *

TYPE:       CTRL C
Response:   ^C
             '

Type:       R LIBR<CR>
Response:   *
```

If the FORTRAN system will be running on a configuration that includes EAE, proceed. Otherwise, go to Step 6.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,EAE/G<CR>
Response:   ENTRY POINT:

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 10.

6. If the configuration includes a PDP-11/45 processor without FPU or a PDP-11/40 processor with EIS but without FIS, proceed. Otherwise, go to Step 7.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,EIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,EIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

Go to Step 10.

7. If the configuration includes a PDP-11/40 or PDP-11/03 processor with FIS, proceed. Otherwise, go to Step 8.

```
Type:       FORLIB=UNI,OTSCOM,V2NS,FIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *

Type:       FORLIB,V2S=UNI,OTSCOM,V2S,FIS/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>
```

```
Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

Go to Step 10.

8. If the configuration includes a PDP-11/45 processor with FPU, proceed. Otherwise, go to Step 9.

```
Type:      FORLIB=UNI,OTSCOM,V2NS,FPU/G<CR>
Response:  ENTRY POINT

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *

Type:      FORLIB,V2S=UNI,OTSCOM,V2S,FPU/G<CR>
Response:  ENTRY POINT

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

Go to Step 10.

9. If the configuration contains none of the above options:

```
Type:      FORLIB=UNI,OTSCOM,V2NS,NHD/G<CR>
Response:  ENTRY POINT

Type:      $ERRTB<CR>
           $ERRS<CR>
           <CR>

Response:  $END2   ILL INS
           $END2   ILL INS
           $FIO2   ILL INS
           $CLO2   ILL INS
           $FIO2   ILL INS
           $FIO2   ILL INS

           *
```

```
Type:       FORLIB,V2S=UNI,OTSCOM,V2S;NHD/G<CR>
Response:   ENTRY POINT!

Type:       $ERRTB<CR>
            $ERRS<CR>
            <CR>

Response:   $END2   ILL INS
            $END2   ILL INS
            $FIO2   ILL INS
            $CLO2   ILL INS
            $FIO2   ILL INS
            $FIO2   ILL INS

            *
```

10. Type: CTRL C
    Response: ^C
    .

Proceed to Section 2.7.3 if building a BASIC system from paper tape or consult the Table of Contents for the appropriate section if building BASIC from another media. If the system will not include BASIC, go to Chapter 3.

2.7.3 BASIC/RT-11

This section contains instructions for those who received BASIC/RT-11 on paper tape. The instructions involve placing running versions of BASIC on the system device.

The system device onto which BASIC is being built must have at least 70 free blocks. The monitor and PIP.SAV are the only system components that must be on this system device.

1. Bootstrap an RT-11 V02C system (if you have completed Section 2.7.1 or 2.7.2, the system is already booted and running).

   To the running RT-11 monitor,

   Type: DATE dd-mmm-yy<CR>

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

   Type: R PIP<CR>
   Response: *

2. Place the paper tape labeled DEMO.BAS (DEC-11-LBACA-C-PA1) into the paper tape reader. Press the FEED button until blank leader is over the tape head.

   For this, and all the following paper tape instructions in Step 3, the convention xxxxxx.xxx is used to represent the name on the tape label below the BASIC version number. In this first example, the command below would be DEMO.BAS=PR:/A<CR>.

   Type: xxxxxx.xxx=PR:/A<CR>

   (the tape is read)

Response: *

NOTE

For the following steps, a ?CHK SUM? message indicates that an input error occurred during the reading of the paper tape. Retry the operation.

3. Repeat Step 2 for the following paper tapes, using /B in place of /A in the command string.

```
BASICR.OBJ    (DEC-11-LBACA-C-PR1)
BASICE.OBJ    (DEC-11-LBACA-C-PR2)
BASICX.OBJ    (DEC-11-LBACA-C-PR3)
BASNSR.OBJ    (DEC-11-LBACA-C-PR4)
BASNSE.OBJ    (DEC-11-LBACA-C-PR5)
BASNSX.OBJ    (DEC-11-LBACA-C-PR6)
FPMP.OBJ      (DEC-11-LBACA-C-PR7)
BASICH.OBJ    (DEC-11-LBACA-C-PR8)
```

4. Type: CTRL C
   Response: ^C
   *

   Type: R LINK<CR>
   Response: *

   Type: BAS8K=BASNSR,FPMP/T/B:400/C<CR>

NOTE

The above command string is for a configuration that has no devices with vectors above 400. User's with 8K whose configurations have interrupt vectors above 400 should relink BAS8K.SAV with a bottom address of 500, e.g.,

BAS8K=BASNSR,FPMP/T/B:500/C<CR>

See BASIC/RT-11 Release Notes, Section 3.6, for further information.

Response: TRANSFER ADDRESS *

Type: GO<CR>
Response: *

Type: BASNSE/O:1/C<CR>
Response: *

Type: BASNSX/O:1/C<CR>
Response: *

Type: BASICH/O:2<CR>
Response: *

Type: BASIC=BASICR,FPMP,BASICE,BASICX/B:400/C<CR>
Response: *

```
Type:       BASICH<CR>
Response:   *

Type:       CTRL C
Response:   *C
            *
```

The BASIC system is now built on disk. Proceed to Chapter 3.

CHAPTER 3

DEMONSTRATION PACKAGE

3.1 RUNNING THE RT-11 SINGLE-JOB MONITOR

For purposes of this demonstration, a small program (DEMOBG.MAC) is edited, assembled, linked, and executed. When executed, DEMOBG displays a message on the terminal.

1. If the system is not running, bootstrap an RT-11 V02C system. Ensure that the system device is WRITE-ENABLEd.

   Response: RT-11 V02C-xx
   .

   Type: DATE dd-mmm-yy

   where dd-mmm-yy is the current date in the form 27-NOV-75.

   Response: .

2. If the console terminal is an ASR or KSR Teletype, a parallel LA30 DECwriter, a VT50 Alphanumeric Display, or an LA36 DECwriter II, go to Step 3.

   If the console terminal is a VT05 Alphanumeric Display operating at 300 baud, go to Step 3.

   If the console terminal is a serial LA30 DECwriter operating at 300 baud,

   Type: **D 56=5015**<CR>
   Response: .

   Go to Step 3.

   If the console terminal is a serial LA30 DECwriter operating at 150 baud,

   Type: **D 56=2015**<CR>
   Response: .

   Go to Step 3.

   If the console terminal is a serial LA30 DECwriter operating at 110 baud,

   Type: **D 56=1015**<CR>
   Response: .

   Go to Step 3.

If the console terminal is a VT05 Alphanumeric Display operating at 2400 baud,

Type: D 56=2012<CR>
Response: .

Go to Step 3.

If the console terminal is a VT05 Alphanumeric Display operating at 1200 baud,

Type: D 56=1012<CR>
Response: .

Go to Step 3.

If the console terminal is a VT05 Alphanumeric Display operating at 600 baud,

Type: D 56=412<CR>
Response: .

3. If the system configuration does not include a GT40 or VT11 display processor and scope or does not have more than 8K words of memory, go to Step 4. Otherwise, verify that the scope is on by turning the BRIGHTNESS knob to an adequate level.

   Build the VT11 support library as follows:

   Type: R LIBR<CR>
   Response: *

   Type: VTLIB=SY:VTHDLR<CR>
   Response: *

   Type: CTRL C
   Response: *C
   .

   Type: GT ON<CR>

   The system output will shift to the display scope. Commands are still entered at the keyboard, but echo will be on the screen.

   Response: .(on screen)

4. If the system device is diskette, go to Step 5. If the system device is RF11, RK11, RP02, or RJS03/4 disk, go to Step 6.

   If the system device is DECtape, mount a blank tape on Unit 1, WRITE ENABLEd.

   Type: ASSIGN DT1:DK<CR>
   Response: .

   Type: R PIP<CR>
   Response: *

```
Type:       DT1:/Z<CR>
Response:   DT1:/Z ARE YOU SURE?

Type:       Y<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            .
```

Go to Step 6.

5. If the system device is diskette, mount a blank diskette on Unit 1, WRITE ENABLEd.

```
Type:       ASSIGN DX1:DK<CR>
Response:   .

Type:       R PIP<CR>
Response:   *

Type:       DX1:/Z<CR>
Response:   DX1:/Z ARE YOU SURE?

Type:       Y<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            .
```

6. Display the directory of the system device on the terminal.

```
Type:       R PIP<CR>
Response:   *

Type:       SY:/F<CR>
Response:   MONITR.SYS
            DT    .SYS
            LP    .SYS
            PP    .SYS
            PR    .SYS
            TT    .SYS
            PATCH .SAV
            EDIT  .SAV
            MACRO .SAV
            EXPAND.SAV
            ASEMBL.SAV
            SYSMAC.SML
            LINK  .SAV
            ODT   .OBJ
            PIP   .SAV
            *
```

NOTE

Depending on the medium from which RT-11 is built and the individual system configuration, the preceding directory will vary. As long as a directory is printed, its exact content is not of concern.

7. Use the Text Editor to modify the demonstration program, DEMOBG.MAC.

```
Type:       CTRL C
Response:   ^C
            .

Type:       R EDIT<CR>
Response:   *

Type:       EBSY:DEMOBG.MAC<ALT>R<ALT><ALT>
Response:   *

Type:       G:<TAB>,ASCII<ALT><ALT>
Response:   *

Type:       ØAD<ALT><ALT>
Response:   *

Type:       EX<ALT><ALT>
Response:   .
```

Assemble DEMOBG.MAC with EXPAND and ASEMBL and obtain a listing.

```
Type:       R EXPAND<CR>
Response:   *

Type:       DEMOBG=SY:DEMOBG<CR>
Response:   ERRORS DETECTED:  Ø
            *

Type:       CTRL C
Response:   ^C
            .

Type:       R ASEMBL<CR>
Response:   *
```

If a line printer is available, go to Step 9.

8. If running on an 8K DECtape-, RF11-, or Diskette-based system,

```
Type:       DEMOBG=DEMOBG<CR>
Response:   ERRORS DETECTED:  0
            FREE CORE:  xxx WORDS
            *

Type:       ,TTI=DEMOBG<CR>
Response:   (see following)
```

All other systems,

```
Type:       DEMOBG,TTI=DEMOBG<CR>
Response:   (see following)
```

```
.MAIN.   RT-11 MACRO V5.02-09   PAGE 1


1                ; RT-11 MACRO EXPAND V02-02
24               ;DEMOBG.MAC
25               ;DEMONSTRATION PROGRAM TO PRINT DEMONSTRATION MESSAGE, THEN
26               ;RING BELL IF FG JOB SENDS A MESSAGE.
27
28               ;        .MCALL  ..V2..,.REGDEF,.RCVDC,.PRINT
29               ;.MACRO ..V2..
30               ;...V2=1
31               ;.ENDM
32               ;.MACRO .PRINT .ADD
33               ;.IF NB .ADD
34               ;                MOV     .ADD,%0
35               ;.ENDC
36               ;                EMT     ^O351
37               ;.ENDM
38               ;.MACRO .RCVDC .AREA,.BUFF,.WCNT,.CRTN
39               ;.IF NB .AREA
40               ;                MOV     .AREA,%0
41               ;                MOV     #13000,(0)
42               ;.ENDC
43               ;.IF NB .BUFF
44               ;                MOV     .BUFF,4.(0)
45               ;.ENDC
46               ;.IF NB .WCNT
47               ;                MOV     .WCNT,6.(0)
48               ;.ENDC
49               ;.IF NB .CRTN
50               ;                MOV     .CRTN,8.(0)
51               ;.ENDC
52               ;                EMT     ^O375
53               ;.ENDM
54               ;.MACRO .REGDEF
55               ;R0=%0
56               ;R1=%1
57               ;R2=%2
58               ;R3=%3
59               ;R4=%4
60               ;R5=%5
61               ;SP=%6
62               ;PC=%7
63               ;.ENDM
64
65               ;        .REGDEF
66        000000 R0=%0
67        000001 R1=%1
68        000002 R2=%2
69        000003 R3=%3
70        000004 R4=%4
71        000005 R5=%5
72        000006 SP=%6
73        000007 PC=%7
74
75 00000        START:; .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST REQUEST FOR MESS
76                      IF NB #AREA
77 00000 012700                  MOV     #AREA,%0
         000226'
78 00004 012710                  MOV     #13000,(0)
.MAIN.   RT-11 MACRO V5.02-09   PAGE 1+


         013000
```

```
79                      .ENDC
80                      .IF NB #BUFFER
81 00010 012760                         MOV     #BUFFER,4.(0)
         000236'
         000004
82                      .ENDC
83                      .IF NB #400
84 00016 012760                         MOV     #400,6.(0)
         000400
         000006
85                      .ENDC
86                      .IF NB #MSGIN
87 00024 012760                         MOV     #MSGIN,8.(0)
         000044'
         000010
88                      .ENDC
89 00032 104375                         EMT     ^O375
90                      ;       .PRINT  #MSG            ;PRINT DEMONSTRATION MESSAGE
91                      .IF NB #MSG
92 00034 012700                         MOV     #MSG,%0
         000112'
93                      .ENDC
94 00040 104351                         EMT     ^O351
95 00042 000777                 BR      .                       ;AND LOOP
96
97                      ;COMPLETION ROUTINE ENTERED WHEN FG SEBDS MESSAGE
98 00044                MSGIN:; .PRINT  #BELL           ;RING BELL IN RESPONSE TO MESSAG
99                      .IF NB #BELL
100 0044 012700                         MOV     #BELL,%0
         000110'
101                     .ENDC
102 0050 104351                         EMT     ^O351
103                     ;       .RCVDC  #AREA,#BUFFER,#400,#MSGIN ;POST ANOTHER MESSAGE
104                     .IF NB #AREA
105 0052 012700                         MOV     #AREA,%0
         000226'
106 0056 012710                         MOV     #13000,(0)
         013000
107                     .ENDC
108                     .IF NB #BUFFER
109 0062 012760                         MOV     #BUFFER,4.(0)
         000236'
         000004
110                     .ENDC
111                     .IF NB #400
112 0070 012760                         MOV     #400,6.(0)
         000400
         000006
113                     .ENDC
114                     .IF NB #MSGIN
115 0076 012760                         MOV     #MSGIN,8.(0)
         000044'
         000010
116                     .ENDC
117 0104 104375                         EMT     ^O375
118 0106 000207                 RTS     PC                      ;AND RETURN FROM COMPLETION ROUT
.MAIN.  RT-11 MACRO V5.02-09   PAGE 1+


119
120                     ;ASCII MESSAGES
121 0110    007 BELL:   .BYTE   7,200           ;MESSAGE THAT RINGS BELL
    0111    200
122
123 0112    122 MSG:    .ASCII  /RT-11 DEMONSTRATION PROGRAM/
    0113    124
```

```
     0114    055
     0115    061
     0116    061
     0117    040
     0120    104
     0121    105
     0122    115
     0123    117
     0124    116
     0125    123
     0126    124
     0127    122
     0130    101
     0131    124
     0132    111
     0133    117
     0134    116
     0135    040
     0136    120
     0137    122
     0140    117
     0141    107
     0142    122
     0143    101
     0144    115
124  0145    015          .BYTE   15,12
     0146    012
125  0147    111          .ASCII  /IF INCORRECTLY EDITED,THIS IS THE LAST LINE./
     0150    106
     0151    040
     0152    111
     0153    116
     0154    103
     0155    117
     0156    122
     0157    122
     0160    105
     0161    103
     0162    124
     0163    114
     0164    131
     0165    040
     0166    105
     0167    104
     0170    111
     0171    124
     0172    105
     0173    104
     0174    054
     0175    124
.MAIN.  RT-11 MACRO V502-09   PAGE 1+


     0176    110
     0177    111
     0200    123
     0201    040
     0202    111
     0203    123
     0204    040
     0205    124
     0206    110
     0207    105
     0210    040
     0211    114
     0212    101
```

```
    0213    123
    0214    124
    0215    040
    0216    114
    0217    111
    0220    116
    0221    105
    0222    056
126 0223    015          .BYTE   15,12
    0224    012
127             ;        .ASCII  /WELL DONE./
128 0225    000          .BYTE   0
129
130             ;RCVDC PACKET AREA
131 0226 000236'AREA:    .=.+10
132             ;RCVDC MESSAGE AREA
133 0236        BUFFER:
134      000000'         .END    START
.MAIN.  RT-11 MACRO V5B2-09   PAGE 1+
SYMBOL TABLE

AREA    000226R         BELL    000110R         BUFFER  000236R
MSG     000112R         MSGIN   000044R         PC     =%000007
R0     =%000000         R1     =%000001         R2     =%000002
R3     =%000003         R4     =%000004         R5     =%000005
SP     =%000006         START   000000R
. ABS.  000000     000
        000236     001
ERRORS DETECTED: 0
FREE CORE: 20488. WORDS

DEMOBG,TT:=DEMOBG

ERRORS DETECTED: 0
FREE CORE: 20488. WORDS
```

Response: *

Go to Step 10.

9. Ensure that the line printer is on and set to on-line. If running on an 8K DECtape-, RF11-, or Diskette-based system,

   Type: **DEMOBG=DEMOBG**<CR>
   Response: ERRORS DETECTED = 0
   FREE CORE: xxx WORDS
   *

   Type: **,LP:=DEMOBG**<CR>
   Response: *(The listing on the printer is similar to the response for Step 8, except that the FREE CORE message appears only once.)

   All other systems,

   Type: **DEMOBG,LP:=DEMOBG**<CR>
   Response: ERRORS DETECTED: 0
   FREE CORE: xxx WORDS
   *(The listing on the printer is similar to the response for Step 8, except that the FREE CORE message appears only once.)

10. Type: CTRL C
Response: ^C
.

Link the program DEMOBG.

Type: R LINK<CR>
Response: *

Type: SY:DEMOBG=DEMOBG<CR>
Response: *

Type: CTRL C
response: ^C
.

Execute the demonstration program.

Type: R DEMOBG<CR>
Response: RT-11 DEMONSTRATION PROGRAM,
IF INCORRECTLY EDITED, THIS IS THE LAST LINE,
WELL DONE.

Type: CTRL C
CTRL C
Response: ^C
^C
.
.

Type: GT OFF<CR>
Response: .

If the file was incorrectly edited, the procedure may be repeated, although this is not necessary for successful continuation. If repeating the editing procedure, first,

Type: R PIP<CR>
Response: *

Type: DEMOBG,MAC=DEMOBG,BAK/R<CR>
Response: *

Type: CTRL C
Response: ^C
.

Then go to Step 7.

Otherwise, if the system will be running the Foreground/Background Monitor, proceed to Section 3.2. If the system will be running FORTRAN IV but will not be running F/B, proceed to Section 3.3. If the system will be running BASIC but will not be running F/B or FORTRAN IV, proceed to Section 3.4. Otherwise the demonstration is complete.

Before continuing to use the system, make all patches and corrections documented in the Digital Software News and Software Performance Summary, and note the restrictions documented in the RT-11 System Release Notes manual. In addition, you may wish to permanently customize the system for special hardware; instructions for customizing the system are in Section 4.6.

3.2 RUNNING THE RT-11 FOREGROUND/BACKGROUND MONITOR

For the purposes of this demonstration, a second program (DEMOFG.MAC) is assembled, linked for the foreground, and executed in conjunction with DEMOBG. This portion of the demonstration requires 16K words of memory and a clock to run. DEMOFG is a small foreground program that sends a message every two seconds to DEMOBG, running in the background, telling it to ring the terminal bell. Besides printing the terminal message used in Section 3.1, DEMOBG recognizes these messages and rings the bell once for each message sent.

Although DEMOFG is always active, sending messages to the background every two seconds, this demonstration will execute other programs in the background besides DEMOBG. Only when DEMOBG is active, however, is the circuit complete and messages successfully received and honored. During those periods when DEMOBG is not running, DEMOFG will enter the messages in the message queue; when DEMOBG is started, all the messages queued since the last forced exit will be dequeued immediately, resulting in many successive bell rings. When the queue is empty, the normal send/receive cycle resumes and the bell is rung every two seconds, as each current message is sent and honored.

1. To the running RT-11 Single-Job Monitor,

   Type: R PIP<CR>
   Response: *

   If running an RK11 system,

   Type: SY:RKMNSJ,SYS=SY:MONITR,SYS/Y/R<CR>

   If running an RF11 system,

   Type: SY:RFMNSJ,SYS=SY:MONITR,SYS/Y/R<CR>

   If running a DECtape system,

   Type: SY:DTMNSJ,SYS=SY:MONITR,SYS/Y/R<CR>

   If running an RJS03/4 system,

   Type: SY:DSMNSJ,SYS=SY:MONITR,SYS/Y/R<CR>

   If running an RP11/RP02 system,

   Type: SY:DPMNSJ,SYS=SY:MONITR,SYS/Y/R<CR>

   If running a diskette system,

   Type: SY:DXMNSJ,SYS=SY:MONITR,SYS/Y/R<CR>

   In any case, the response is:

   Response: ?REBOOT?
   *

   If running an RK11 system,

   Type: SY:MONITR,SYS=SY:RKMNFB,SYS/Y/R<CR>

   If running an RF11 system,

   Type: SY:MONITR,SYS=SY:RFMNFB,SYS/Y/R<CR>

If running an RP11/RP02 system,

Type: **SY:MONITR,SYS=SY:DPMNFB,SYS/Y/R**<CR>

If running a diskette system,

Type: **SY:MONITR,SYS=SY:DXMNFB,SYS/Y/R**<CR>

If running an RJS03/4 system,

Type: **SY:MONITR,SYS=SY:DSMNFB,SYS/Y/R**<CR>

If running a DECtape system,

Type: **SY:MONITR,SYS=SY:DTMNFB,SYS/Y/R**<CR>

In any case, the response is:

Response: **?REBOOT?**
*****

Type: **SY:A=SY:MONITR,SYS/U**<CR>
Response: *****

Type: **SY:/O**<CR>
Response: RT-11FB V02C-xx
.

The F/B Monitor is now running.

If the console terminal is an ASR or KSR Teletype, a VT50 Alphanumeric Display, an LA36 DECwriter II, or a parallel LA30 DECwriter, go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 300 baud, go to Step 2.

If the console terminal is a serial LA30 DECwriter operating at 300 baud,

Type: **D 56=5015**<CR>
Response: .

Go to Step 2.

If the console terminal is a serial LA30 DECwriter operating at 150 baud,

Type: **D 56=2015**<CR>
Response: .

Go to Step 2.

If the console terminal is a serial LA30 DECwriter operating at 110 baud,

Type: **D 56=1015**<CR>
Response: .

Go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 2400 baud,

```
Type:       D 56=2012<CR>
Response:   .
```

Go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 1200 baud.

```
Type:       D 56=1012<CR>
Response:   .
```

Go to Step 2.

If the console terminal is a VT05 Alphanumeric Display operating at 600 baud,

```
Type:       D 56=412<CR>
Response:   .
```

2. If the system configuration does not include a GT40 or VT11 display processor and scope, go to Step 3. Otherwise, verify that the scope is on by turning the BRIGHTNESS knob to an adequate level.

```
Type:       GT ON<CR>
```

The system output will shift to the display scope. Commands are still entered at the keyboard, but echo will be on the screen.

```
Response:   .(on screen)
```

3. If the system device is diskette, go to Step 4. If the system device is RF11, RK11, RJS03/4, or RP02, go to Step 5.

If the system device is DECtape, mount a blank, formatted DECtape on Unit 1, WRITE ENABLEd.

```
Type:       ASSIGN DT1:DK<CR>
Response:   .

Type:       R PIP<CR>
Response:   *

Type:       DT1:/Z<CR>
Response:   DT1:/Z ARE YOU SURE?

Type:       Y<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            .
```

Go to Step 5.

4. If the system device is diskette, mount a blank diskette on Unit 1, WRITE ENABLEd.

```
Type:       ASSIGN DX1:DK<CR>
Response:   .
```

```
Type:       R PIP<CR>
Response:   *

Type:       DX1:/Z<CR>
Response:   DX1:/Z ARE YOU SURE?

Type:       Y<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            .
```

5. Enter the date on which this demonstration is being run.

```
Type:       DATE dd-mmm-yy<CR>
```

where dd-mmm-yy is the current date in the form 27-NOV-75.

```
Response:   .
```

Enter the time of day.

```
Type:       TIME hh:mm:ss<CR>
```

where hh:mm:ss is the current hour, minutes, and seconds in the form 13:12:50 (1:12 p.m.).

6. Assemble the foreground demonstration program, DEMOFG.MAC.

```
Type:       R MACRO<CR>
Response:   *

Type:       DEMOFG=SY:DEMOFG<CR>
Response:   ERRORS DETECTED:  0
            FREE CORE xxxxx.  WORDS
            *

Type:       CTRL C
Response:   ^C
            .
```

7. Link DEMOFG for the foreground.

```
Type:       R LINK<CR>
Response:   *

Type:       SY:DEMOFG=DEMOFG/R<CR>
Response:   *

Type:       CTRL C
Response:   ^C
            .
```

8. Start DEMOFG as the foreground job.

```
Type:       FRUN SY:DEMOFG<CR>
Response:   F>
            FOREGROUND DEMONSTRATION PROGRAM, SENDS A
            MESSAGE TO THE BACKGROUND PROGRAM "DEMOBG"
            EVERY 2 SECONDS, TELLING IT TO RING THE
            BELL.
            B>
```

DEMOFG is now running and queueing the messages for DEMOBG every two seconds.

9. Execute DEMOBG and receive the messages.

```
Type:       R DEMOBG<CR>
            (The bell will ring quickly several times,
            then will ring once every two seconds.)
Response:   RT-11 DEMONSTRATION PROGRAM
            IF INCORRECTLY EDITED, THIS IS THE LAST LINE,
            WELL DONE.
```

10. Execute PIP in the background to get a directory listing.

```
Type:       CTRL C
            CTRL C (the bell will stop)
Response:   ^C
            ^C
            .

Type:       R PIP<CR>
Response:   *

Type:       /L<CR>
Response:   dd-mmm-yy
            (The directory of the device DK: is printed
            on the terminal.  Its exact contents are not
            of consequence.)
            *

Type:       CTRL C
Response:   ^C
            .
```

11. Rerun DEMOBG to collect all the foreground messages queued while PIP was running.

```
Type:       R DEMOBG<CR>
            (The bell will ring several times in rapid
            succession, then will begin ringing once
            every two seconds.)
Response:   RT-11 DEMONSTRATION PROGRAM
            IF INCORRECTLY EDITED, THIS IS THE LAST LINE,
            WELL DONE.

TYPE:       CTRL C
            CTRL C
Response:   ^C
            ^C
            .
            (The bell will stop ringing.)
```

12. Stop the foreground program and remove it from memory.

```
Type:       CTRL/F
Response:   F>

Type:       CTRL C
            CTRL C
Response:   ^C
            ^C
            B>
```

```
Type:       UNL FG<CR>
Response:   .

Type:       GT OFF<CR>
Response:   .
```

Proceed to Section 3.3 if running FORTRAN IV, or to Section 3.4 if running BASIC; otherwise, the demonstration is complete.

Before continuing to use the system, make all patches and corrections documented in the Digital Software News and Software Performance Summary, and note the restrictions documented in the RT-11 System Release Notes. In addition, you may wish to permanently customize the system for special hardware; instructions for customizing the system are in Section 4.6.

3.3 RUNNING FORTRAN IV

This section contains instructions for compiling the sample program (DEMO.FOR), linking it, and executing it. The program is a simple FORTRAN program to calculate a Fibonacci Series (each term is the sum of the preceding two terms) based on user inputs.

1. Bootstrap the FORTRAN system device created in Chapter 2; enter the date this demonstration is being run.

   ```
   Type:       DATE dd-mmm-yy<CR>
   ```

   where dd-mmm-yy is the currect date in the form 27-NOV-75.

   ```
   Response:   .
   ```

   If the configuration does not include a line printer, proceed to Step 2. Otherwise,

   ```
   Type:       ASS LP:TT<CR>
   Response:   .
   ```

2. If the system device is diskette or disk, go to Step 3.

   If the system device is DECtape, mount a blank tape on Unit 1, WRITE ENABLEd.

   ```
   Type:       ASSIGN DT1:DK<CR>
   Response:   .

   Type:       R PIP<CR>
   Response:   *

   Type:       DT1:/Z<CR>
   Response:   DT1:/Z ARE YOU SURE?

   Type:       Y<CR>
   Response:   *

   Type:       CTRL C
   Response:   ^C
               .
   ```

3. Compile the FORTRAN program DEMO.FOR as follows:

   Type: R FORTRA<CR>
   Response: *

   Type: DEMO,TT:=DEMO<CR>

If the configuration includes a line printer, the source listing is printed on the printer. If not, the listing appears on the console terminal.

Response:

```
FORTRAN IV  V01C-XX      THUR 27-NOV-75     13:51:35          PAGE 001

      !C      RT-11 FORTRAN PROGRAM TO GENERATE N TERMS OF
      !C      A FIBONACCI SERIES, THE FIRST TWO TERMS OF
      !C      WHICH ARE SPECIFIED BY THE USER.
      !C
      !C      PRINT IDENTIFYING MESSAGE
0001          WRITE (5,1000)
      !C
      !C      GET THE LENGTH AND FIRST TWO TERMS OF THE SERIES
0002  100     WRITE (5,1010)
0003          READ (5,1060) LENGTH
0004          IF (LENGTH) 150,125,150 !A LENGTH OF ZERO MEANS FINISHED
0005  125     STOP
0006  150     WRITE (5,1020)
0007          READ (5,1060) ITERM1
0008          WRITE (5,1030)
0009          READ (5,1060) ITERM2
      !C
      !C      MAKE SURE THE LENGTH TYPED WASN'T NEGATIVE OR TOO LARGE
0010          IF (LENGTH -3) 200,250,250
0011  200     WRITE (5,1040) LENGTH
0012          GOTO 100
0013  250     IF (LENGTH -50) 300,300,200
      !C
      !C      PRINT THE FIRST TWO TERMS OF THE SERIES
0014  300     WRITE (5,1050)
0015          WRITE (5,1060) ITERM1
0016          WRITE (5,1060) ITERM2
0017          LENGTH = LENGTH -2
      !C
      !C      CALCULATE THE NEXT TERM AND PRINT IT
0018  400     ITNEW = ITERM1 + ITERM2
0019          ITERM1 = ITERM2
0020          ITERM2 = ITNEW
0021          WRITE (5,1060) ITNEW
      !C
      !C      DETERMINE IF SERIES IS FINISHED.  IF SO, DO NEXT ONE.
0022          LENGTH = LENGTH -1
0023          IF (LENGTH) 100,100,400
      !C
0024  1000    FORMAT ('0PROGRAM TO GENERATE A FIBONACCI SERIES')
0025  1010    FORMAT ('0HOW MANY TERMS DO YOU WANT GENERATED? ')
0026  1020    FORMAT (' WHAT IS THE FIRST TERM? ')
0027  1030    FORMAT (' WHAT IS THE SECOND TERM? ')
0028  1040    FORMAT (I5,' TERMS DOES NOT REALLY MAKE SENSE,')
0029  1050    FORMAT ('0THE REQUESTED SERIES IS:')
0030  1060    FORMAT (I10)
0031          END
```

```
FORTRAN IV      STORAGE MAP

NAME    OFFSET  ATTRIBUTES

LENGTH  000334  INTEGER*2 VARIABLE
ITERM1  000336  INTEGER*2 VARIABLE
ITERM2  000340  INTEGER*2 VARIABLE
ITNEW   000342  INTEGER*2 VARIABLE
*
```

Type: CTRL C
Response: ↑C
'

Type: ASSIGN<CR>
Response: '

4. Link the program with the FORTRAN library, FORLIB.OBJ, as follows:

Type: R LINK<CR>
Response: *

Type: DEMO=DEMO/F<CR>
Response: *

Type: CTRL C
Response: ↑C
'

5. Execute the program.

Type: R DEMO<CR>
Response:
```
PROGRAM TO GENERATE A FIBONACCI SERIES
HOW MANY TERMS DO YOU WANT GENERATED?
```

Type: 10<CR>
Response:
```
WHAT IS THE FIRST TERM?
```

Type: 2<CR>
Response:
```
WHAT IS THE SECOND TERM?
```

Type: 4<CR>
Response:
```
THE REQUESTED SERIES IS:
         2
         4
         6
        10
        16
        26
        42
        68
       110
       178

HOW MANY TERMS DO YOU WANT GENERATED?
```

Type: 10<CR>
Response:
```
WHAT IS THE FIRST TERM?
```

Type: 17<CR>
Response:
```
WHAT IS THE SECOND TERM?
```

Type: ↑C<CR>

```
Response:  THE REQUESTED SERIES IS:
             17
             -3
             14
             11
             25
             36
             61
             97
            158
            255

Response:  HOW MANY TERMS DO YOU WANT GENERATED?

Type:      0<CR>
Response:  STOP --
           .
```

Proceed to Section 3.4 if running BASIC; otherwise the demonstration is complete.

Before continuing to use the system, make all patches and corrections documented in the Digital Software News and Software Performance Summary, and note the restrictions documented in the RT-11 System Release Notes. In addition, you may wish to permanently customize the system for special hardware; instructions for customizing the system are in Section 4.6.

Before using FORTRAN, carefully read the RT-11 FORTRAN Release Notes manual and note the restrictions and clarifications for RT-11 FORTRAN V01C.

3.4 RUNNING BASIC/RT-11

The following instructions are for running the simple BASIC program, DEMO.BAS, which is the same program as DEMO.FOR, written in BASIC. It calculates a Fibonnacci series (each term is the sum of the previous two) based on user inputs.

1. To the running RT-11 monitor,

```
Type:      R BAS8K<CR>
Response:  BASIC V01B-01

Type:      <CR>
Response:  READY
```

2. Load the demonstration program into memory.

```
Type:      OLD "DEMO.BAS"<CR>
Response:  READY
```

3. List the program.

```
Type:      LIST<CR>
Response:
```

```
DEMO     27-NOV-75  BASIC V01B-01

10 REM BASIC PROGRAM TO GENERATE N TERMS OF A FIBONACCI SERIES,
20 REM THE FIRST TWO TERMS OF WHICH ARE SPECIFIED BY THE USER,
30 REM
40 REM PRINT THE IDENTIFYING MESSAGE
50 PRINT "PROGRAM TO GENERATE A FIBONACCI SERIES"
60 REM
70 REM GET THE LENGTH AND FIRST TWO TERMS OF THE SERIES
80 PRINT "HOW MANY TERMS DO YOU WANT GENERATED";
90 INPUT L
100 IF L<>0 THEN 130
110 REM IF USER REQUESTS 0 TERMS, TERMINATE EXECUTION
120 STOP
130 PRINT "WHAT IS THE FIRST TERM";
140 INPUT T1
150 PRINT "WHAT IS THE SECOND TERM";
160 INPUT T2
170 REM MAKE SURE L IS NOT NEGATIVE OR TOO LARGE
180 IF L<3 THEN 200
190 IF L<50 THEN 220
200 PRINT L;"TERMS DO NOT REALLY MAKE SENSE,"
210 GO TO 80
220 REM PRINT THE FIRST TWO TERMS OF THE SERIES
230 PRINT "THE REQUESTED SERIES IS"
240 PRINT T1
250 PRINT T2
260 L=L-2
270 REM CALCULATE NEXT TERM AND PRINT IT
280 N=T1+T2
290 T1=T2
300 T2=N
310 PRINT N
320 REM DETERMINE IF SERIES IS FINISHED, IF SO, DO NEXT ONE,
330 L=L-1
340 IF L<=0 THEN 80
350 GO TO 280
360 END

READY
```

4. Execute the program.

   Type: RUN<CR>
   Response:
   ```
   DEMO dd-mmm-yy  BASIC V01B-01
   PROGRAM TO GENERATE A FIBONACCI SERIES
   HOW MANY TERMS DO YOU WANT GENERATED?
   ```

   Type: 10<CR>
   Response:
   ```
   WHAT IS THE FIRST TERM?
   ```

   Type: 2<CR>
   Response:
   ```
   WHAT IS THE SECOND TERM?
   ```

   Type: 4<CR>

```
Response:  THE REQUESTED SERIES IS:
           2
           4
           6
           10
           16
           26
           42
           68
           110
           178
           HOW MANY TERMS DO YOU WANT GENERATED?

Type:      CTRL C
Response:  ^C
           .

Type:      RE<CR>
Response:  READY
```

5. BASIC/RT-11 may be run in immediate mode, described in Chapter 4 of the BASIC/RT-11 Language Reference Manual. Statements without line numbers are executed as soon as they are entered.

```
Type:      FOR I=1 TO 5\PRINT I,SQR(I)\NEXT I<CR>
Response:  1     1
           2     1.41421
           3     1.73205
           4     2
           5     2.23607

Type:      CTRL C
Response:  ^C
           .
```

The demonstration is complete.

Before continuing to use the system, make all patches and corrections documented in the Digital Software News and Software Performance Summary, and note the restrictions documented in the RT-11 System Release Notes. In addition, you may wish to permanently customize the system for special hardware; instructions for customizing the system are in Section 4.6.

Before using BASIC, carefully read the BASIC/RT-11 Release Notes manual and note the restrictions and clarifications for BASIC/RT-11 V01B.

CHAPTER 4

RT-11 SYSTEM CUSTOMIZATION

4.1 GENERAL BUILDING INSTRUCTIONS

RT-11 is designed so that the monitor and device handlers that comprise the system are files on the system device. These files (called system files) all have the extension .SYS and can be transferred between devices just like any other RT-11 file.

The running version of the monitor must be named MONITR.SYS; other versions of the monitor may reside on the system device, but they must be named something other than MONITR.SYS. The handlers for the system must be named xx.SYS, where xx is the device mnemonic as used in command strings. For example, the high-speed reader handler must be named PR.SYS. There may be many versions of a given handler on the system device, but the one that is in use must be named as above.

Once copies of the system files have been obtained, the procedure for building an RT-11 system consists of the following basic steps:

1. Initializing the target device with an RT-11 directory.

2. Transferring the appropriate monitor file to the target device and giving it the name MONITR.SYS.

3. Transferring the appropriate handler files to the target device.

4. Writing the appropriate bootstrap on the target device.

5. Transferring the desired system components (EDIT, LINK, etc.) to the target device.

6. Check the target device with the PIP /K switch to read all the blocks and verify that all the data written is good.

After step 4 above, the target device may be bootstrapped and the remainder of the build procedure may be carried out while executing the system from the new (and perhaps faster) system device. Because the above build steps involve standard RT-11 file operations, system programs are used for the build procedure. When building from DECtape or disk, PIP is used; from magtape, MBUILD is used; from cassette, CBUILD is used; and from paper tape, LINK and PIP are used. RT-11 V02C can be bootstrapped WRITE-PROTECTed and will run PIP WRITE-PROTECTed as well. System building should always be carried out with the master WRITE-PROTECTed.

Once the system is built, any necessary patches to the system should be installed as noted in the Digital Software News, Software Performance Summary, and RT-11 System Release Notes. Then the system should be customized for specific hardware, if any, as detailed in Section 4.6. When all patches are made and the system customized, the new system device should be backed-up immediately.

The following files on the distribution media have special significance:

MONITR.SYS	On a DECtape master this is a Single-Job DECtape Monitor; on a DECpack or TA11 cassette master, this is a Single-Job RK11 Disk Monitor; on diskette, this is the Single-Job RX11 Monitor. When the volume is booted, this file is the monitor used. (Not available on magtape or paper tape.)
DTMNSJ.SYS	This is a Single-Job DECtape Monitor, used for building DECtape-based Single-Job systems (not available on cassette, diskette, or paper tape).
DTMNFB.SYS	This is a Foreground/Background DECtape Monitor and is used when it is desired to run the F/B system from DECtape (not available on cassette, diskette, or paper tape).
RKMNSJ.SYS	This a Single-Job RK11 Monitor. This file is called MONITR.SYS on RK11 and cassette masters.
RKMNFB.SYS	This is a Foreground/Background RK11 Monitor. If an RK11 F/B system is desired, this file becomes MONITR.SYS on the disk.
RFMNSJ.SYS	This is a Single-Job RF11 Monitor (not available on cassette).
RFMNFB.SYS	This is a Foreground/Background RF11 Monitor (not available on cassette).
DXMNSJ.SYS	This is a Single-Job RX11 Monitor (not available on cassette or paper tape).
DXMNFB.SYS	This is a Foreground/Background RX11 Monitor (not available on cassette or paper tape).
DPMNSJ.SYS	This is a Single-Job RP11/RP02 Monitor (not available on cassette or paper tape).
DPMNFB.SYS	This is a Foreground/Background RP11/RP02 Monitor (not available on cassette or paper tape).
DSMNSJ.SYS	This is a Single-Job RJS03/4 Monitor (not available on cassette, DECtape, diskette, or paper tape).
DSMNFB.SYS	This is a Foreground/Background RJS03/4 Monitor (not available on cassette, DECtape, diskette, or paper tape).

DX.SYS	This is the RX11/RX01 diskette handler, which allows RT-11 systems running from a disk other than the RX to access a diskette (not available on cassette or paper tape).
DP.SYS	This is the RP11/RP02 disk handler; it allows systems based on another device to access the RP02 (not available on cassette or paper tape).
RF.SYS	This is an RF11/RS11 device handler; it allows RT-11 systems running from other devices to access the RF11 disk (not available on cassette).
RK.SYS	This is an RK11 device handler which allows RT-11 to read and write RK11 disks while running from an alternate system device.
DT.SYS	This is a TC11 DECtape handler which allows RT-11 to read and write DECtape while running disk systems (not available on cassette or paper tape).
DS.SYS	This is the RJS03/4 fixed-head disk handler.
MT.SYS	This is the TM11/TS03 or TM11/TU10 magtape handler.
MM.SYS	This is the TJU16 magtape handler.
CT.SYS	This is the TA11 cassette handler.
PR.SYS	This is the PC11 high-speed reader handler.
PP.SYS	This is the PC11 high-speed punch handler.
TT.SYS	This is the general terminal handler for the S/J Monitor. The F/B terminal handler is included in the MONITR.SYS file.
LP.SYS	This is the line printer (LP11, LS11, LV11) handler.
CR.SYS	This is the CR11 card reader handler.
BA.SYS	This is the BATCH run-time handler.

4.2 BUILDING RT-11 SYSTEMS FROM DECTAPE, DECPACK, AND DISKETTE

On DECtape, DECpack, and diskette, RT-11 is distributed as a ready-to-run Single-Job system. When bootstrapped (as described in the RT-11 System Reference Manual, Chapter 2), the system is running and may be used to build other DECtape and disk systems.

To build another DECtape or disk system from a running DECtape or disk system, the following set of commands to PIP can be used. These commands assume that the device to built has been assigned the logical device name DK1:. (For example, the command ASSIGN DT0:DK1 assigns the logical name DK1 to DECtape unit 0.

Command	Explanation
.R PIP<CR>	
*DK1:/Z<CR>	Initialize the new DECtape or disk.
DK1:/Z ARE YOU SURE?Y<CR>	
*DK1:A=DK0:/S<CR>	Copy all files from DK0 to DK1.
*DK1:A=DK1:MONITR.SYS/U<CR>	Write the hardware bootstrap on DK1.
*DK1:/K<CR>	Check for bad blocks.

DK1 is now a ready-to-run copy of the system device.

In the preceding example, the system files were copied to DK1: via the /S option in PIP; the /S option makes the directory on DK1: exactly the same as the directory on DK0:. In actuality, any method of copying the files to the new device would have sufficed. For example, the command:

```
*DK1:A=DK0:/S
```

could be replaced by:

```
DK1:/Z/N:12
*DK1:*.*=DK0:*.*/Y/X
```

or

```
*DK1:MONITR.SYS=MONITR.SYS/Y
*DK1:LP.SYS=LP.SYS/Y
      .
      .
      .
```

etc., until all desired files are transferred.

When a disk is initialized for RT-11, the PIP /Z/N switches can be used to determine the maximum number of files that can be entered in the directory. (See the RT-11 System Reference Manual, Section 4.2.7 for details.) The PIP /Z command allows the following approximate number of files per disk:

```
*DK:/Z = DK:/Z/N:4 =  280 files allowed
       *DK:/Z/N:10 = 560 files allowed
       *DK:/Z/N:37 = 2170 files allowed
```

The procedures given in this document are for single platter disks; if more platters are on the system, the disk initialization command can be changed.

For example, to build an RK11 disk system from a running DECtape system, use the following commands:

Commands	Explanation
.R PIP<CR>	
*RK1:/Z/N:12<CR>	Initialize the disk.
RK1:/Z ARE YOU SURE?Y<CR>	
*RK1:A=DT0:/S <CR>	Copy the DECtape files onto disk.
*RK1:DTMNSJ.SYS=RK1:MONITR.SYS/Y/R<CR>	Rename the DECtape monitor on the disk to an appropriate name.

Command	Explanation
*RK:MONITR,SYS=RK:RKMNSJ,SYS/Y/R<CR>	Rename the disk monitor on the disk to MONITR.SYS (in this case, the Single-Job Monitor).
*RK:A=RK:MONITR,SYS/U<CR>	Write the system bootstrap on the disk.
*RK:/K<CR>	Check for bad blocks.

The RK11 disk may now be bootstrapped as described in the RT-11 System Reference Manual, Chapter 2, or with the PIP /O switch. The original DECtape MONITR.SYS file (on the disk) must be renamed to another name before the RK monitor is named to MONITR.SYS, or it will be automatically deleted when the RK monitor rename is performed.

As another example, to build an RF11 disk system from a running RK11 system, use the following commands:

Command	Explanation
.R PIP<CR>	
*RF:/Z<CR>	Initialize the disk to allow room for 280 files.
RF:/Z ARE YOU SURE?Y<CR>	
RF:,*=*,*/X/Y<CR>	Copy all files from the running system to the new system.
*RF:RKMNSJ,SYS=RF:MONITR,SYS/Y/R<CR>	Rename the RK monitor on the RF to an appropriate name.
*RF:MONITR,SYS=RF:RFMNFB,SYS/Y/R<CR>	Rename the desired RF monitor on the RF disk to MONITR.SYS (in this case, the F/B Monitor).
*RF:A=RF:MONITR,SYS/U<CR>	Write the system bootstrap on the RF disk.
*RF:/K<CR>	Check for bad blocks.

The RF11 disk may now be bootstrapped.

Note that in the above examples, all files were copied from the running system to the system being built. Although this is a common practice, it is not necessary, and is usually not possible when the running system is disk and the target device is DECtape or a smaller disk. Only the following elements must be transferred.

1. A monitor file
2. The handlers for the desired devices
3. The hardware bootstrap
4. Those programs and files that will be used with the new system.

When building a system for a configuration that contains only LA36 and diskette or DECtape, it is wise to avoid transferring any handler files that are not required, since they require space on the system device, yet serve no purpose. Note that the system device handler is built into the monitor, therefore the file DT.SYS need not be transferred to a DECtape system since both DTMNSJ.SYS and DTMNFB.SYS contain the DECtape handler.

Users of 8K machines may choose to build systems without the files MACRO.SAV, BA.SYS, and BATCH.SAV, while EXPAND.SAV and ASEMBL.SAV can be eliminated from Single-Job systems with 16K or more of memory.

Although the system is distributed with 12 monitor files, the system only requires 1 monitor for day-to-day operation. An application would seldom need more than two (the Single-Job and Foreground/Background Monitors for a particular system device often reside on the same volume for convenient switching).

The following example details the building of a working diskette-based system from another diskette. This system configuration includes diskette, console terminal, and 16K words of memory.

Command	Explanation
.R PIP<CR>	
*DX1:/Z<CR>	Initialize the disk to be built.
DX1:/Z ARE YOU SURE?Y<CR>	
*DX1:MONITR.SYS=DX0:MONITR.SYS/Y/X<CR>	Copy the RX11 Single Job Monitor to the new system.
*DX1:DXMNFB.SYS=DX0:DXMNFB.SYS/Y/X<CR>	Copy the RX11 F/B monitor to the new system.
DX1:,*=DX0:TT.SYS,BA.SYS/Y/X<CR>	Copy the console terminal and BATCH handlers.
DX1:,*=DX0:BATCH.SAV,EDIT.SAV/X<CR>	Copy the BATCH compiler and the Editor.
DX1:,*=DX0:MACRO.SAV,CREF.SAV/X<CR>	Copy other needed system
DX1:,*=DX0:LINK.SAV,PIP.SAV/X<CR>	files to the new system.
DX1:,*=DX0:SYSMAC.SML/X<CR>	
*DX1:A=DX1:MONITR.SYS/U<CR>	Write the system bootstrap onto the new system device.
*DX1:/K<CR>	Check for bad blocks.
*DX1:/O<CR>	Bootstrap the new system device.

4.3 BUILDING RT-11 SYSTEMS FROM MAGTAPE (MBUILD)

The magtape on which RT-11 is distributed is a bootable magtape that contains RT-11 files. The following files on the system magtape have special significance:

MBOOT.BOT	Used to read the secondary bootstrap MSBOOT.BOT into memory.
MSBOOT.BOT	Used to load an RT-11 file (normally MBUILD) from magtape.
MBUILD.MT1	Used to build RK11, RJS03/4, or RP11/RP02 disk systems from TM11/TS03 or TM11/TU10 magtape.
MBUILD.MT2	Used to build RF11 disk systems from TM11/TS03 or TM11/TU10 magtape.
MBUILD.MM1	Used to build RK11, RJS03/4, and RP11/RP02 disk systems from TJU16 magtape.

MBUILD.MM2	Used to build RF11 disk systems from TJU16 magtape.
MTINIT.SAV	Used to initialize a bootable magtape. MTINIT writes the bootstrap on the magtape and zeroes the tape. MTINIT is used as part of the procedure to backup a system device on a bootable magtape.

The remaining .SYS files are the monitor and handler files as described in Section 4.1. Step-by-step instructions for building RT-11 with MBUILD are contained in Section 2.5.1.

4.3.1 Building RT-11 Sytems with MBUILD

MBUILD is the special system-build version of PIP which is loaded via MSBOOT. This program is used to initialize the disk and transfer the remaining files from magtape to disk. To build an RT-11 system from magtape, the bootable magtape is booted via a hardware or software bootstrap which reads MBOOT.BOT into memory. MBOOT.BOT then automatically reads the secondary bootstrap MSBOOT.BOT (which must be the first file on the tape) into memory.

MSBOOT.BOT types its version number, then prints an asterisk and waits for the user to enter the file name of the MBUILD file to be used. As noted above, the file name entered depends on the type of disk being built and the type of magtape interface (TM11 or TJU16). When the file name is entered, MSBOOT searches the magtape for the file, reads it into memory, and starts it.

MBUILD prints its version number, then prints an asterisk. In response to the asterisk, the user enters an I/O specification in the standard RT-11 PIP format. The various operations that can be performed by MBUILD are summarized in Table 4-1; they are a subset of PIP commands. If no switch is specified, MBUILD assumes that the operation is a file transfer in image mode; the files are not concatenated.

NOTE

If more than 50 files at a time are to be transferred to the disk, do not use the *.* construction to perform the operation. Use MBUILD to transfer the necessary magtape handlers, monitors, and PIP.SAV; then use PIP to transfer the remaining files.

Table 4-1
MBUILD Switches

Switch	Meaning
/C	Used with another switch to cause only files with the current date (as designated using the monitor DATE command) to be included in the specified operation.
/E	Lists the entire specified directory, including unused spaces, on the terminal.
/F	Lists the short specified directory (file names only) on the terminal.
/I	Transfers files in image mode.
/L	Lists the specified directory on the terminal; this directory includes the file names and dates for each entry.
/M:n	Used to prevent the magtape from rewinding between each file involved in the operation; n should be a large positive number.
/N:n	Used with /Z to specify the number of directory segments (n) to allocate to the disk directory.
/O	Boots the RT-11 system from the specified disk.
/Q	When used with another switch, causes MBUILD to print each file name that is eligible for a wild card operation and to ask for a confirmation of its inclusion in the operation. Typing a Y causes the named file to be included in the operation; typing anything else excludes the file. The command line is not processed until the user has confirmed each file in the operation.
/R	Renames the specified file.
/U	Copies the bootstrap from the specified file into the boot blocks of the specified volume.
/X	When combined with wild card (*) file specification, causes all files that satisfy conditions to be individually transferred.
/Y	Causes system files and .BAD files to be operated on by the command specified. Attempted modifications of .SYS or .BAD files without /Y are not done and cause the message ?NO .SYS/.BAD ACTION? to be printed.
/Z[:n]	Zeroes (initializes) the directory of the specified disk; n is used to allocate extra words per directory entry. When used with /N, the number of directory segments for entries may be specified.

4.3.2 Creating Bootable Magtapes with MTINIT

MTINIT is a small program used to initialize a magtape so that it can be bootstrapped from RT-11. This program writes the bootstrap on the magtape and zeroes the magtape. MTINIT is used primarily to create a bootable magtape backup of the system disk.

The system disk being used to create the magtape must contain the following files: MBOOT.BOT, MSBOOT.BOT, MTINIT.SAV, the appropriate MBUILD file, the appropriate monitors, the appropriate device handlers, and the desired system programs.

To call MTINIT from the RT-11 system device, type:

```
R MTINIT<CR>
```

in response to the dot printed by the Keyboard Monitor. MTINIT prints:

```
MTINIT Vxx-xx
*
```

and the user responds:

```
MTn:=<CR>
```

where n is the magtape unit number on which the bootable magtape is to be created. MTINIT rewinds the magtape on MTn and prints:

```
MTn:/ZERO/BOOT - ARE YOU SURE?
```

the user responds,

```
Y<CR>
```

When MTINIT has written the bootstrap and zeroed the magtape, it returns to the keyboard monitor which prompts the user with a dot at the terminal.

PIP can then be used to transfer the remaining files to the magtape. Files must be copied to the magtape in the following order:

1. MSBOOT.BOT
2. MBUILD.xxx files
3. Monitor files
4. Device handlers
5. Programs

4.4 BUILDING RT-11 SYSTEMS FROM CASSETTE (CBUILD)

On cassette, RT-11 is distributed as a series of RT-11 files on several cassettes, each cassette labeled DEC-11-ORTSA-E-TCn. The following files on the system cassettes have special significance:

CBUILD.SYS	CBUILD is the special system-build version of PIP which is loaded via the cassette bootstrap. This program is used to initialize the disk and transfer the remaining files from cassette to disk. CBUILD.SYS is used to build RK11 systems only.
MONITR.SYS	The RT-11 Single-Job RK11 Monitor file.

The remaining .SYS files are the monitor and handler files as described in Section 4.1. Step-by-step instructions for building RT-11 with CBUILD are contained in Section 2.6.1.

CBUILD is designed to allow cassettes to serve as backup devices. CBUILD can be used to transfer files between standard cassettes and other RT-11 devices, delete cassette files, and list cassette directories. Cassette to cassette transfers, however, are not allowed.

To build an RT-11 system from cassette, bootstrap an RT-11 cassette (with the file CBUILD.SYS as the first file) on Unit 0 and perform the following operations:

In response to the asterisk printed by CBUILD, type an I/O specification in the standard RT-11 format.

CBUILD accepts up to six input files and three output files. The contents of the input file are transferred to the output file in image mode.

The various operations that can be performed by CBUILD are summarized in Table 4-2. If no switch is specified, CBUILD assumes that the operation is a file transfer in image mode; the files are not concatenated.

Table 4-2
CBUILD Switches

Switch	Meaning
/A	Transfers files in ASCII mode (nulls are ignored).
/B	Transfers files in formatted binary mode.
/D	Deletes the file specified from the output cassette. The /D switch is valid only if the output device is a cassette. For example: `*CT1:OFILE.MAC/D` deletes OFILE.MAC from the cassette on drive 1.
/E	Lists the entire directory including spaces called *EMPTY which result from deleted files. For example: `*CT0:/E` lists the directory of the cassette mounted on unit 0 including *EMPTY entries.

(continued on next page)

Table 4-2 (Cont.)
CBUILD Switches

Switch	Meaning
/F	Prints the short directory of the specified cassette (file names only). For example: `*LP:=CT1:/F` prints the directory (file names only) of cassette unit 1 on the line printer.
/G	Copies a file and ignores input errors. When copying from a cassette to another device, input errors in data and the header are ignored.
/I or none	Transfers files in image mode. For example: `*CT0:FA,FB,FC=FD,FE,FF/I` transfers files FD, FE, and FF from device DK to cassette unit 0 as files FA, FB, FC.
/L	Reads the input cassette directory and writes it on the output device. The directory includes file names and dates for each entry. Notice that in this case the input file itself is not transferred, only the directory. The /L switch is valid only if the input device is a cassette.
/M	Reads multi-volume cassette files.
/N	Used with /Z to specify the number of directory blocks to allocate on the disk or DECtape. The number of directory blocks governs how many files can be stored on a volume.
/O	Boots the RT-11 system from the specified disk or DECtape unit 0. For example: `*RK:/O`
/Q	Reads after write; writes a block then reads that block and checks for write errors.
/U	Transfers the system bootstrap to the specified device. For example: `*DK:A=CT1:MONITR.SYS/U` transfers the bootstrap in the file MONITR.SYS from cassette unit 1 to the boot blocks of the device DK. The file name A is a dummy specification.
/V	Displays the version number of CBUILD.
/X	When combined with wild card (*) file specification, causes all files that satisfy conditions to be individually transferred.

(continued on next page)

Table 4-2 (Cont.)
CBUILD Switches

Switch	Meaning
/Z	Clears the directory of the specified device. The message: ARE YOU SURE? is displayed. Enter Y and a carriage return to zero the directory; any other response causes CBUILD to ignore the command. /Z allows specification of extra words in the directory entry when used to zero a disk or DECtape. For example: DT1:/Z:2 provides two extra words per directory entry. A value given to the /Z switch when used to zero a cassette has no effect.

Use the following set of commands (or their equivalent) to build the RK11 disk system.

Commands	Explanation
*RK:/Z/N:12<CR> ARE YOU SURE? Y<CR>	Initialize disk directory.
RK:.*=CT0:*.*/Y/X<CR>	Write system files on disk. Repeat this command for each cassette necessary to build the desired system.
*RK:A=CT0:MONITR,SYS/U<CR>	Write bootstrap on disk.
*RK:/O<CR>	Bootstrap the disk system.

RT-11 is then running from the RK11 disk and may be used in the normal fashion to copy any other desired files from cassette to disk. Note that no devices other than disk or cassette can be used until their handler files have been added to the disk and the system has been rebooted.

4.5 BUILDING DISK SYSTEMS FROM PAPER TAPE (PT BUILD)

RT-11 is distributed as object modules on paper tape. Two of the tapes (PT BUILD Tapes 1 and 2) are used to place a rudimentary V01-15 Monitor and Linker on the disk. The disk system is then started and the Linker is used to link OLDPIP from paper tape onto the disk. Once linked, OLDPIP is used to copy the remaining paper tapes onto the disk, where they can be linked to complete the system.

The following paper tapes have special significance:

DEC-11-ORPBA-D-PB1 This paper tape is for RK11 systems only.
RK PT BUILD Tape 1

DEC-11-ORPBA-D-PB2 This paper tape is for both RK11 and RF11
PT BUILD Tape 2 systems.

DEC-11-ORPBA-D-PB3 This paper tape is for RF11 systems only.
RF PT BUILD Tape 1

To build an RT-11 system from paper tape, perform the following operations:

1. Load the Bootstrap Loader (see Section A.6) at 37744, then use it to load the Absolute Loader.

2. Using the Absolute Loader, load the appropriate PT BUILD Tape 1 for the system device desired. It self starts and prints:

   PT BUILD Version number

3. There is a 10-15 second pause, after which PT BUILD prints:

   ```
   PLACE SECOND TAPE IN READER,
   STRIKE ANY CHARACTER TO CONTINUE.
   ```

4. Place the tape PT BUILD Tape 2 into the reader, then strike any character to start the tape.

   There is a slight pause, after which the following is printed:

   ```
   DISK BUILD COMPLETE,

   RT-11 V01-15
   ```

   During the build procedure, only a

   ```
   WRITE FAILED
   ```

   error message has exact meaning. If encountered, check to ensure that the system disk is not WRITE-PROTECTed. Any other error indicates a user error or hardware problem.

5. Link OLDPIP as follows:

   ```
   .R LINK<CR>
   *OLDPIP=PR1
   ***
   ```

   For each occurrence of the prompting ^, place the tape OLDPIP (DEC-11-OROPA-D-PR) in the reader, then strike a character to read the tape. Type CTRL C when the second * is printed.

6. Run OLDPIP to copy the remaining tapes onto the system disk (use the /B switch for all files but SYSMAC.SML, SYSMAC.8K, and VTMAC.MAC, which require /A.) Unlike the V02C system when built, OLDPIP prompts with an up-arrow or circumflex prior to reading each tape, and proceeds when a character is struck at the keyboard.

   Section 2.7.1 lists the tapes distributed with RT-11. If a specific component is not required, it is not necessary to transfer the corresponding tapes.

   PREXEC.OBJ and PREPAS.OBJ are the object modules for EXPAND. SMEXEC.OBJ, SMMAC.OBJ, and SMPST.OBJ are linked for ASEMBL, while RTEXEC.OBJ, RTMAC.OBJ, and RTPST.OBJ are the MACRO object modules. PIP.OBJ is the object module for PIP.

VTHDLR, after a pass by the Librarian, becomes the display handler. CT.OBJ through MM.OBJ are linked into handlers of the same name; RFBTFB.OBJ through RT11SJ.OBJ are the RF11 and RK11 monitor components. SYSF4.OBJ becomes SYSLIB after a pass through the Librarian; BA.OBJ and BATCH.OBJ are the BATCH system components.

EDIT.OBJ and VTLEDT.OBJ make the Editor, while LINK0 through LNKOV5 are the components for the V02C Linker. DUMP.OBJ, SRCCOM.OBJ, FILEX.OBJ, and CREF.OBJ are linked for programs of the same name, while LIBR0 through LIBR4 become the Librarian. PAT0 through R50ASC are linked with RT-11 FORTRAN (if available) to make PATCHO.

7. Run LINK to generate the V02C .SYS files and program .SAV files as described in Chapter 5. Linking the new V02C Linker will generate ADDITIVE REF messages, which should be ignored. Once the V02C Linker is built, it should be used for subsequent linking operations.

8. Rename the desired monitor file to MONITR.SYS, then copy the bootstrap with the PIP /U switch.

9. Reboot the system with a hardware bootstrap or use the bootstrapping procedures in the appropriate section of Appendix A. The .OBJ files (except for ODT, SYSLIB, and VTLIB) can be deleted as well as OLDPIP.SAV, since they are no longer required.

4.5.1 Directory Extension

When a device is zeroed, additional directory segments can be specified by use of the PIP /N switch. Once files are written on the device, however, the directory size cannot be changed without using the /Z option in PIP, which destroys existing directory information.

Although it is recommended that, whenever possible, the PIP /Z/N combination be used to determine directory size, the following program may be used to change the size of a directory on-line without reinitializing the disk. It can be used if there is no other alternative, and will be of special interest to paper tape users who cannot specify the size of their disk directory at system build time. The program should be used only as a last resort, and only on a disk which has been thoroughly backed-up. It should be entered as instructed in the comments. Instructions for its use are also in the comments.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;DIREXT-PROGRAM TO ADD SEGMENTS TO AN RT-11 DIRECTORY
;TO USE, THE FIRST ENTRY IN THE DIRECTORY MUST BE <UNUSED>, AND ITS LENGTH
;MUST BE TWICE AS MANY BLOCKS AS THE NUMBER OF SEGMENTS THAT ARE TO
;BE ADDED.  THE FIRST ENTRY MAY BE MADE <UNUSED> SIMPLY BY CALLING
;PIP TO MOVE THE FIRST FILE ON THE DEVICE.  ONCE THE UNUSED AREA HAS BEEN
;CREATED, DIREXT IS RUN BY TYPING:
;
;               .R DIREXT
;THE RESPONSE:  *
;
;A COMMAND LINE IS ENTERED OF THE FORM "DEV:/N:M" WHERE DEV IS
;THE DEVICE WHOSE DIRECTORY IS TO BE EXPANDED AND "M" IS
;THE NUMBER OF SEGMENTS (NOT BLOCKS) TO ADD TO THE DIRECTORY.
;
```

```
;FOR EXAMPLE:
;               *RK:/N:14       WILL ADD 12 (DECIMAL) SEGMENTS TO THE
;                               DIRECTORY ON RK0, FOR WHICH THE FIRST
;                               FILE MUST BE AN <UNUSED> AREA OF AT
;                               LEAST 24 BLOCKS
;
;THERE ARE TWO POSSIBLE ERROR MESSAGES:
;
;       ?HARD I/O ERROR?        WILL OCCUR IF THE DIRECTORY FOR THE
;                               DEVICE CANNOT BE READ OR WRITTEN WITHOUT
;                               HARDWARE ERROR.
;       ?ILLEGAL COMMAND?       WILL OCCUR ON ANY OF THE FOLLOWING CONDITIONS
;                                 1) INAPPROPRIATE COMMAND STRING
;                                 2) FIRST ENTRY IN DIRECTORY NOT <UNUSED>
;                                 3) REQUESTED NUMBER OF SEGMENTS WOULD
;                                    CAUSE TOTAL TO EXCEED ALLOWABLE 31(10)
;                                 4) SIZE OF <UNUSED> AREA TOO SMALL TO
;                                    ACCOMODATE REQUESTED DIRECTORY INCREASE
;
;IF THE ?ILLEGAL COMMAND? MESSAGE OCCURS, THE DIRECTORY IS UNMODIFIED
;AND THE PROGRAM IS RESTARTED.
;
;TO CREATE DIREXT, TYPE THIS TEXT INTO THE FILE "DIREXT.MAC" WITH THE EDITOR.
;
;TO ASSEMBLE,   .R MACRO
;               *DIREXT=DIREXT
;
;TO LINK,       .R LINK
;               *DIREXT=DIREXT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


        .MCALL  ..V1..,.CSIGEN,.READW,.WRITW,.EXIT,.PRINT
        ..V1..
        R0=%0
        R1=%1
        R2=%2
        R3=%3
        R4=%4
        R5=%5
        SP=%6

ILLCMD: .PRINT  #MSG1           ;TYPE "?ILLEGAL COMMAND?"
START:  .CSIGEN #DSPACE,#DEXT,#0        ;GET COMMAND STRING
        DEC     (SP)+           ;ANY SWITCHES?
        BNE     ILLCMD          ;NO-BAD COMMAND
        CMP     (SP)+,#101516   ;/N:M ON FIRST INPUT FILE?
        BNE     ILLCMD          ;NO-BAD COMMAND
        MOV     (SP)+,R1        ;NUMBER OF SEGS TO ADD INTO R1
        MOV     #6,R2           ;BLOCK FOR CURRENT SEG IN R2
        MOV     #DIRBUF,R3      ;R3 POINTS INTO DIRECTORY BUFFER
        .READW  3,R3,#256.,R2   ;READ FIRST DIRECTORY SEG
        BCS     HERR            ;READ ERROR
        CMP     12(R3),#1000    ;FIRST ENTRY <UNUSED>?
        BNE     ILLCMD          ;NO-BAD COMMAND
        MOV     (R3),R4         ;NUMBER OF SEGS NOW IN DIR INTO R4
        ADD     R1,R4           ;TOTAL SEGS FOR NEW DIR IN R4
        MOV     R4,R5           ;REMEMBER NEW SEG TOTAL
        CMP     R4,#37          ;IS NEW TOTAL TOO LARGE?
        BHI     ILLCMD          ;YES-BAD COMMAND
        ASL     R1              ;# OF BLOCKS NEEDED FOR NEW SEGS
        CMP     R1,22(R3)       ;<UNUSED> LARGE ENOUGH TO TAKE NEW SEGS?
        BGT     ILLCMD          ;NO-BAD COMMAND
        ADD     R1,10(R3)       ;UPDATE FILE START ADDRESS
        SUB     R1,22(R3)       ;REDUCE <UNUSED> SIZE
```

```
LOOP:   MOV     R4,(R3)         ;UPDATE NUMBER OF SEGS TO NEW TOTAL
        .WRITW  3,R3,#256.,R2   ;WRITE SEG BACK OUT
        BCS     HERR            ;WRITE ERROR
        DEC     R5              ;ALL SEGS UPDATED?
        BLE     DONE            ;YES
        TST     (R2)+           ;NO-ADD 2 TO BLOCK NUMBER
        .READW  3,R3,#256.,R2   ;READ NEXT SEG
        BCC     LOOP            ;AND UPDATE
HERR:   .PRINT  #MSG2           ;PRINT "?HARD I/O ERROR?"
DONE:   .EXIT
MSG1:   .ASCIZ  "?ILLEGAL COMMAND?"
MSG2:   .ASCIZ  "?HARD I/O ERROR?"
        .EVEN
DEXT:   0,0,0,0
DIRBUF: .=.+1000
DSPACE:
        .END    START
```

4.6 CUSTOMIZATION FOR SPECIAL HARDWARE

This section contains instructions on patching various RT-11 system components to customize your system for specific hardware. The patching instructions contain certain mnemonics in place of actual values. For example, the mnemonic $HSIZE is used in place of the actual value for the handler size. The actual values for these mnemonics are listed in RT-11 System Release Notes, Table 2.

4.6.1 High Baud Rate Serial Console Devices

The serial LA30 (LA30S) requires that filler characters follow each carriage return; the 600, 1200, and 2400 baud VT05's require that filler characters follow each line feed. RT-11 has established a mechanism by which any number of fills may follow any character. The byte at location 56 (octal) contains the character to be followed by fillers and the byte at location 57 (octal) contains the number of null fills to be used. These locations are initially set to zero, which results in no fillers being generated (normal operation for LT33, LA30P, LA36, VT50, and VT52).

Depending on the terminal, modify the locations as follows:

	Loc 56	Loc 57	Resulting Word (octal)
LA30S 110 baud	015(8)	002(8)	1015
LA30S 150 baud	015(8)	004(8)	2015
LA30S 300 baud	015(8)	012(8)	5015
VT05 600 baud	012(8)	001(8)	412
VT05 1200 baud	012(8)	002(8)	1012
VT05 2400 baud	012(8)	004(8)	2012

The proper octal word can be changed permanently in the monitor by using PATCH to modify locations 56 and 57 in the monitor file. For example:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*56\0          15<LF>        (Fill after <CR>
57\0           4<CR>          with 4 nulls)
*E
```

Once the change is made, the bootstrap recopied with the PIP /U switch, and the monitor rebooted with the PIP /O switch, all programs that use the monitor for console I/O will operate correctly.

4.6.2 Magtape Parity, Density, Number of Tracks

4.6.2.1 TM11 (MT.SYS) -- The RT-11 TM11 magtape handler is distributed such that it will correctly handle both 7- and 9-track drives without modification. It does so at 800 bpi, using the TM11 dump mode for 7-track drives. Seven-track drives can also be written (in hardware mode only; see the RT-11 System Reference Manual, Appendix H) at 200, 556, and 800 bpi (non-dump mode) by modifying the handler as described below.

To alter the magtape density used by the handler, the following patches must be made:

1. Patch the handler density word (location PARDEN in MT.SYS).

2. When changing from 800 bpi 9- or 7-track dump mode to 200, 556, or 800 bpi 7-track, you must patch the handler to default to hardware mode (location HW in MT.SYS) and patch the $STAT entry in the monitor for MT.SYS (location MTSTAT in MONITR.SYS) so that the special device bit (bit 12) is off (zero).

3. When changing to 800 bpi 9- or 7-track dump mode from 200, 556, or 800 bpi 7-track, you must patch the handler to default to software mode and patch the $STAT entry in the monitor for MT.SYS so that the special device bit is on (one).

NOTE

In the 200, 556, and 800 bpi 7-track modes, with the $STAT table patched so that bit 12 is off, the MT handler will not get control when a .LOOKUP, .ENTER, .CLOSE, or .DELETE is issued for MT. Doing a .LOOKUP, .ENTER, or .CLOSE in this state simply opens or closes a channel associated with MT. The magtape is not rewound for any of these functions in this state; the only operations that are passed to the magtape handler are .READ, .WRITE, and special functions (.SPFUN).

The following table describes the patches for the various magtape densities:

Location	200 bpi 7-track	556 bpi 7-track	800 bpi 7-track	7-track dump mode (800 bpi) or 800 bpi 9-track
PARDEN (MT.SYS)	0	20000	40000	60000
HW (MT.SYS)	377	377	377	0
MTSTAT (MONITR.SYS)	2011	2011	2011	12011

For example, to cause the MT handler to write 7-track tapes at 556 bpi:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MT.SYS<CR>
*PARDEN/60000  20000<CR>
*HW    /00000    377<CR>
*F

FILE NAME--
*MONITR.SYS/M<CR>
*MSTAT /12011  2011<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact addresses of PARDEN, HW, and MTSTAT.

4.6.2.2 TJU16 (MM.SYS) -- The TJU16 allows five possible tape modes in file-structured operation. Since the user may wish to transfer between tapes written in various modes, the TJU16 handler includes a table that defines the default mode for each magtape unit (MM0 to MM7). The user can change the default mode for a particular unit simply by patching that unit's entry in the mode table. The table is located at UNIMOD in the magtape handler and is eight words long. Each word contains the mode for a given unit (0-7).

The default mode for a particular unit may also be changed dynamically under program control. A nonfile-structured LOOKUP with a file count between 1 and 5 causes a mode change for the unit accessed. The following list shows the possible modes and corresponding table value for patching and file count for dynamic modification:

Mode	Table Value	File Count
200 bpi	0	1
556 bpi	400	2
800 bpi (odd parity)	1000	3
800 bpi (even parity)	1400	4
1600 bpi (phase-encoded)	2000	5

Default for all drives is initially 800 bpi even parity. The handler can distinguish between 7- and 9-track drives, so the user need not concern himself with the type of drive in use.

For example, to cause unit 1 to write in 1600 bpi mode (the first word of the table, location UNIMOD, is mode for unit 0; location UNIMOD+2 is mode for unit 1):

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MM.SYS<CR>
*UNIMOD+2/1000      2000<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

NOTE

When dynamically modifying the mode with a nonfile-structured LOOKUP, position the magtape at the beginning of the tape (BOT). See RT-11 Release Notes, Table 2, for the exact address of UNIMOD.

4.6.3 Specifying the Number of RF11 Platters

RT-11 is distributed with fixed-head disk support initialized for one platter. To allow RT-11 to make use of more than one platter, the device size table in the various monitor files must be modified as follows:

Number of Platters	New Value of Table
1	2000
2	4000
3	6000
4	10000

For example, to modify the RF11 F/B Monitor for three RF/RS11 platters, type:

```
.R PATCH

PATCH Version number

FILE NAME--
*RFMNFB.SYS/M<CR>
*RFSIZ/2000      6000<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

NOTE

For all monitors, the address to modify for RF disk is RFSIZ. The address of RFSIZ for the current version of the monitor can be found in RT-11 System Release Notes, Table 2.

Once the above change has been made, zeroing the disk (using the PIP /Z switch) will adjust the directory to the appropriate size. If the system is already running off fixed-head disk as the system device and the disk cannot be zeroed without destroying the system, compressing the disk (with the PIP /S switch) will automatically re-adjust the directory size.

4.6.4 Specifying a 50-Cycle Clock Rate

RT-11 is distributed with the Keyboard Monitor TIME command calculations based on a 60-cycle clock rate. To cause the TIME command to base calculations on a 50-cycle clock rate, modify the monitor such that bit 5 (40(octal)) is set in the monitor configuration word (see the RT-11 System Reference Manual, Section 9.2.6). For example:

For the F/B Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*CONFIG/1       41<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2 for the exact address of CONFIG in the current monitor.

For the S/J Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*CONFIG/0       40<CR>
*E
```

```
.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

4.6.5 Interfacing RJS03/4 DISKS TO RT-11

RT-11 is distributed with the monitor device tables for RJS03/4 disk initialized for RJS03. To allow complete use of all the space available on an RJS04 disk, modify the device size table in the monitor as follows:

Location	RJS03 Value	RJS04 Value
DSSIZ	2000	4000

For example, to modify the RT11 DS monitor to support an RJS04 disk rather than RJS03, type:

```
.R PATCH

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*DSSIZ/2000        4000<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact address of DSSIZ in the current monitor.

Once the above change has been made, zeroing the disk (using the PIP /Z switch) will adjust the directory to the appropriate size. If the system is already running from fixed-head disk as the system device and the disk cannot be zeroed without destroying the system, compressing the disk (with the PIP /S switch) will automatically re-adjust the directory size.

4.6.6 Interfacing RP03 Disks to RT-11

The RP02 support provided in the distribution kit is initialized for RP02 only. The RT-11 file structure can accommodate a maximum of 64000(10) blocks. The 40000(10)-block RP02 cartridge, therefore, can be accommodated as a single logical unit, while an RP03 cannot.

The RT-11 RP02 support can easily be altered, however to accommodate RP03s as follows:

1. Each RP03 drive must be considered as two logical units of 40000 blocks each; in essence, a single RP03 drive looks like two RP02 drives to the system. The cartridge on physical unit n is accessed as logical DPn and DPn+4; thus, drive 0 is referenced as DP0: and DP4:, drive 1 is DP1: and DP5:, etc. Note that although an RP03 is physically one device, it is two separate devices to the system. Each logical unit has its own complete directory and data space.

2. The DP.SYS handler must be patched to change location RP23 from 404 (octal) to 1404 (octal).

3. The DP monitors must be patched to alter location RP23 from 404 (octal) to 1404 (octal).

For example, to allow RT-11 to support RP03s:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*DP.SYS<CR>
*RP23/   404    1404<CR>
*E

.R PATCH<CR>

PATCH Version number

FILE NAME--
*DPMNFB.SYS<CR>
*RP23/   404    1404<CR>
*E

.R PATCH<CR>

PATCH Version number

FILE NAME--
*DPMNSJ.SYS<CR>
*RP23/   404    1404<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the actual location of RP23 in each monitor and in DP.SYS.

Note that a maximum of four RP03s can be supported on the system. RP02s and RP03s can be mixed as long as the total number of units (physical drives) on the system does not exceed four. If the system contains only RP02s, the above changes must not be made and the system can support as many as eight units.

4.6.7 Interfacing Card Readers to RT-11

Although the CR11 device handler is included in the RT-11 V02C kits, this device is not included in the monitor device tables due to lack of available space. To install the card reader in the V02C system, the user must examine Table 3 in RT-11 System Release Notes, select a device that is not used on the system, and replace that device with CR. Four tables must be patched: the handler size table, the device size table, the physical name table, and the device status word table. The corresponding table values for the CR driver are:

```
handler size:   1326
device size:    0
physical name:  12620 (.RAD50 /CR/)
status word:    40014
```

The table entries to be patched are:

```
$HSIZE + 2*(table index for driver to be replaced)
$DVSIZ + 2*(table index for driver to be replaced)
$PNAME + 2*(table index for driver to be replaced)
$STAT + 2*(table index for driver to be replaced)
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact addresses of $HSIZE, $DVSIZ, $PNAME, and $STAT.

For example, suppose the PC11 reader/punch is not used on the system. In this case, the PR and PP drivers are not needed. The PR driver is selected to be replaced by the CR driver. The PR driver's table index is 5, which is equivalent to an octal byte offset of 10 into each table. The following patches can then be applied to the appropriate monitors.

For the Single-Job Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,13634/174        1326<CR>
*0,13670/0          0<CR>
*0,16500/63320      12620<CR>
*0,16534/40007      40014<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

For the F/B Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,14566/174      1326<CR>
*0,14622/0        0<CR>
*0,17640/63320    12620<CR>
*0,17674/40007    40014<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<Cr>
```

NOTE

See RT-11 System Release Notes, Table 2 for the actual address of BASE.

4.6.8 Changing the Location of VT11 Floating Vectors

The floating vector region on the PDP-11 is situated in locations 300 to 476; VT11 display processor vectors are normally located at 320 to 332. However, the VT11 display vectors may be forced to float by the addition of other devices. For example, VT11 device vectors may be changed if a DL11 device (a communications device) is added.

In such a case, a patch to the RT-11 monitor (at GTVECT), to reflect the new location of the VT11 display vectors, is necessary. The value of the VT11 display stop vector (called NEW in the following example) must be determined by consulting the configuration data for the particular installation.

For the F/B Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*GTVECT/320   NEW<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact address of GTVECT in the current monitor.

For example, if NEW is 340 and the value of GTVECT for the F/B Monitor is 37354:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*37354/320   340<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

This patch enables the text scroller (GT ON), the display file handler, EDIT, BASIC/GT, and FORTRAN/GT to function properly on the system without further patching.

4.6.9 Interfacing a Second Diskette Handler

RT-11 V02C is distributed with only two diskettes supported. If additional diskettes are needed, an additional device handler for each two extra diskettes must be installed into the RT-11 monitor. A system with an additional diskette handler must have at least 16K words of memory.

Perform the following operations to install a second diskette handler in the system.

1. Use PIP to create a copy of the floppy handler:

```
.R PIP<CR>
*DY.SYS=DX.SYS/Y/X<CR>
*^C
```

2. Use PATCH to modify the new handler (DY.SYS) as follows:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*DY.SYS<CR>
*1000/264          xxx<CR>
*DXIOP/177170      yyyyyy<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

where xxx is a 3-digit value for the interrupt vector address for the second diskette controller and yyyyyy is a 6-digit value for the I/O page address of the second diskette controller.

NOTE

See RT-11 System Release Notes, Table 2, for the actual address of DXIOP.

3. Use PATCH to insert the new handler into the system tables in place of another handler. Four tables must be patched: handler size, device size, physical name, and device status word. The corresponding table values for the DY driver are:

```
handler size:   670
device size:    756
physical name:  16350 (.RAD50 /DY/)
status word:    102022
```

The table entries to be patched are:

```
$HSIZE + 2*(table index for driver to be replaced)
$DVSIZ + 2*(table index for driver to be replaced)
$PNAME + 2*(table index for driver to be replaced)
$STAT + 2*(table index for driver to be replaced)
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact addresses of $HSIZE, $DVSIZ, $PNAME, and $STAT.

For example, suppose that the TU60 cassette handler is not used on the system. Since the CT driver is not needed, it is selected to be replaced by the DY handler. The CT driver's table index is 10, which is equivalent to an octal byte offset of 20 into each table. The following patches can then be applied to the appropriate monitors:

For the Single-Job Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,13644/3710      670<CR>
*0,13700/0         756<CR>
*0,16510/12740     16350<CR>
*0,16544/12013     102022<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

For the F/B Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,14576/3710      670<CR>
*0,14632/0         756<CR>
*0,17650/12740     16350<CR>
*0,17704/12013     102022<CR>
*E
```

```
.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact address of BASE.

4. The diskettes on the second controller may now be referenced as DY0 and DY1. The monitor ASSIGN command may be used to assign a user logical name such as DX2 or DX3 as follows:

```
.ASSIGN DY0:DX2<CR>
.ASSIGN DY1:DX3<CR>
```

When the name DX2 is used, operations will be performed on DY0; when the name DX3 is used, operations will be performed on DY1. Note that the ASSIGN command is temporary and must be reissued each time the system is bootstrapped.

4.6.10 Reducing the Size of Text Window Displayed

The Editor is constructed in such a way that when the scope is in use, the window into the buffer and the scrolled commands lines are separate "pictures". On rare occasions, if the text window around the cursor contains long lines and several line feeds (or form feed characters), the window can "overflow" onto the scrolled editing commands, making that portion of the screen difficult to read.

In most applications, this problem does not occur; when it does, the obscure lines can be seen by advancing the cursor several lines to bring them into clear view.

If the problem is troublesome for a particular application, it can be removed by reducing the size of the window displayed as follows:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*EDIT.SAV<CR>
*EBASE;0R<CR>
*0,DSARG/____12____n<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

where n is the number of lines to be displayed above and below the cursor; n should be smaller than the value currently in DSARG to eliminate the problem.

NOTE

See RT-11 System Release Notes, Table 2, for the exact addresses of EBASE and DSARG.

4.6.11 Using "}" and "~" Characters on LA36

The LA36 contains two characters (} and ~) which generate ASCII codes 175 (octal) and 176 (octal). Because many older terminals generate 175 (octal) and 176 (octal) for ALTMODE (or ESCAPE), RT-11 EDIT treats 175 (octal) and 176 (octal) as ALTMODE, making the new characters impossible to insert as text.

The Editor may be patched as follows to remove the special-case check for these characters so that they may be used on an LA36 or other terminal capable of handling them.

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*EDIT.SAV<CR>
*ALTTST/   175    33<CR>
*ALTTST+6/   176   33<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the actual address of ALTTST.

Once EDIT is altered in the preceding manner, "}" and "~" can be used normally.

4.6.12 Setting an Upper Limit on File Size

RT-11 is distributed such that the maximum size of a file allocated in a general .ENTER request is half the largest space, or the entire second largest space available, whichever is larger. This is satisfactory for most applications and should be left unchanged. It is possible that certain applications require that an upper limit be set on the size of a file; for these applications (and these only), the following change can be made.

For Single-Job Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,MAXBLK/ 177777    n<CR>
*E
```

where n is an octal number of blocks defining the maximum file size for a general .ENTER.

```
.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the actual values of BASE and MAXBLK.

For Foreground/Background Monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,MAXBLK/  177777    n<CR>
*E
```

where n is an octal number of blocks defining the maximum file size for a general .ENTER.

```
.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

4.6.13 Running RT-11 in Less Memory Than That Available

The V02C monitors have bootstraps which allow the system to run in less memory than is available on the system (e.g., RT-11 can be bootstrapped to run in the lower 12K or 8K of a 16K machine). Most applications require that RT-11 make use of all memory available, and the system is distributed such that it automatically does so.

If (and only if) an application requires that RT-11 run in less memory than is available, the following change can be made.

For all monitors:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BHALT/   407      0<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
```

NOTE

BHALT is a value obtained from Table 2 of RT-11 System Release Notes.

Once the change has been made and a new system bootstrap written on the device (with the PIP /U switch), a halt occurs whenever the system is booted.

At this point, set the switch register to one of the following values and press CONTINUE; the bootstrap operation completes for the specified memory size.

```
  40000 = 8K
  60000 = 12K
 100000 = 16K
 120000 = 20K
 140000 = 24K
 160000 = 28K
>160000 = Use all available memory.
```

4.6.14 Accessing Nonsystem Disks on Single-Disk Systems

Source disks and other nonsystem disks can be accessed on single-disk systems with more than 8K words of memory as follows:

1. Boot the system disk and enter the date.
2. LOAD the handler for the device to which the desired files will be transferred.
3. Run PIP. When the prompting * appears, dismount the system disk.
4. Mount the source disk, WRITE LOCKed, in place of the system disk.
5. Transfer the desired files to the backup device (such as magtape or cassette).
6. Remount the system disk.
7. Type CTRL C to return to the monitor.

PIP always keeps the USR resident and, with the handler loaded, the system disk is not required if no other devices are referenced.

For example to transfer RT-11 sources from the source disk to TM11 magtape on a single-disk system:

1. Boot system and enter the date.
2.
```
Type:      LOAD MT<CR>
Response:  .

Type:      R PIP<CR>
Response:  *
```
3. Dismount system disk, mount source disk.
4.
```
Type:      MT0:*.*,*.*/X/M11<CR>
```
5. When done, remount system disk.

The sources can now be manipulated from magtape.

4.6.15 Reassigning Device Names for RK11 and RF11

Users of other DIGITAL operating systems will notice that RT-11 uses the controller names (RK and RF) rather than the more common user-level names (DK and DF) for these devices. This is due to the fact that RT-11 uses the name DK to refer to the default storage device, which may not necessarily be the RK11.

If you find this situation annoying, the device names can be reassigned with the monitor ASSIGN command, as follows:

```
.ASSIGN RK:DK<CR>
.ASSIGN RF:DF<CR>
```

Note, however, that when DK is reassigned in this manner, all default storage goes to the device name DK, and you may not wish to use the physical device RK as the default storage device.

4.6.16 Interfacing a Foreground Terminal

Applications for the F/B Monitor frequently require that the foreground program dialogue appear on a separate terminal, independent of the console terminal.

To facilitate development of these applications, or any others requiring multiple terminals, a source (KB.MAC) for a device-independent terminal handler has been included in the distribution kit. The source is provided for user convenience and can be used as a model for handler development, modified to meet specific needs, or assembled and used as is to provide support for a second terminal.

Documentation for the use of KB.MAC is contained in the comments at the beginning of the source; a listing appears in Appendix B of the RT-11 Software Support Manual.

4.6.17 Modifying the Line Count in MACRO and CREF

RT-11 MACRO and CREF set the number of lines printed per listing page at 60. This line count is satisfactory for applications with line printers that use paper 10.5 inches long. Applications that use paper of a different size (e.g., 8.5 inches long) and applications without line printers should modify MACRO and CREF as follows.

For MACRO:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MACRO.SAV<CR>
*MACR1;0R<CR>
*0,12364/   74   n<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

where n is the new line count specified in octal.

For CREF:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*CREF.SAV<CR>
*CREF1;0R<CR>
*0,3114/____74____n<CR>
*E

.R PIP<CR>
*SY:/O<CR>
```

where n is the new line count specified in octal.

NOTE

See RT-11 System Release Notes, Table 2, for the actual addresses of MACR1 and CREF1.

4.6.18 Changing the DUMP Default Output Device

The DUMP utility program uses LP as its default output device. Systems that do not have line printers will want to change the default device, normally to TT.

DUMP.SAV can be easily altered to change the default output device as follows:

1. Patch to change location LP to the .RAD50 code for the new default output device.
2. Patch to change location MSGO+1 to the .ASCII code for the first letter of the new default output device.
3. Patch to change location MSGO+2 to the .ASCII code for the second letter of the new default output device.

For example, to change the DUMP default output device to TT (the console terminal),

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*DUMP.SAV<CR>
*LP/___46600___10040<CR>          [.RAD50 for TT]
*MSGO+1\114_____124<CR>           [.ASCII for T]
*MSGO+2\120_____124<CR>           [.ASCII for T]
*E

.R PIP<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the actual values of LP and MSGO. See RT-11 System Release Notes, Table 3, for the .ASCII and .RAD50 codes for all RT-11 devices.

4.6.19 Using UNLOAD When a Foreground Job is Running

The keyboard monitor .UNLOAD command cannot be used to unload handlers from a background job when a foreground job is running. The following patch allows users to reclaim the space occupied by .LOADed background job handlers while a foreground job is running. When this patch is made, users have no protection against the possibility of .UNLOADing handlers required by the foreground job. If a handler required by a foreground job is inadvertently .UNLOADed, an ?M-BAD FETCH error on a .FETCH or an ?M-NO DEV error on a .LOOKUP, .ENTER, .READ, or .WRITE will occur.

The correction to the RT-11 V02C Foreground/Background Monitor is:

```
.R PATCH<CR>

PATCH Version number

FILE NAME--
*MONITR.SYS/M<CR>
*BASE;0R<CR>
*0,41046/____1403____403<CR>
*E

.R PIP<CR>
*SY:A=MONITR.SYS/U<CR>
*SY:/O<CR>
```

NOTE

See RT-11 System Release Notes, Table 2, for the exact address of BASE.

4.7 OPTIMIZING THE SYSTEM DEVICE

When building RT-11 systems, performance can be optimized by proper placement of .SYS files on the system device.

Optimal file placement is:

```
MONITR.SYS
Most frequently used handler
      .
      .
      .
Least frequently used handler

SYSMAC.SML                    (if many assembly operations
                              are performed)
```

Most frequently used program
.
.
.
Least frequently used program

Considerations for the above placements are:

1. Positioning the monitor immediately after the directory optimizes device motion during monitor swapping operations.

2. Positioning PIP.SAV immediately after the monitor ensures that it will not move when the device is compressed or files are deleted.

3. Positioning the handlers and programs in descending order related to frequency of usage reduces the access times for those files.

4. In systems that will be used for frequent assembly operations, placing SYSMAC.SML near the beginning of the device improves assembler performance.

Diskette and DECtape users can also conserve time and space by placing only those files needed on the system disk or DECtape. Users of 8K systems need not place files such as MACRO.SAV, SYSMAC.SML, CREF.SAV and BA.SYS on the system device, since these files cannot be used in 8K systems.

4.8 SWITCHING BETWEEN SINGLE-JOB AND FOREGROUND/BACKGROUND MONITORS

For an application that requires frequent switching between the F/B and Single-Job Monitors, use the following procedure:

1. Both monitors reside on the same volume, the one running is named MONITR.SYS and the other is called xxxxxx.SYS. (The actual name xxxxxx is not significant.)

2. When a change-over is desired, PIP is used to:

   a. Preserve the running monitor by renaming it to yyyyyy.SYS

   b. Rename the desired monitor to MONITR.SYS

   c. Write the new bootstrap from the new MONITR.SYS file

   d. Reboot the system

For example, assume an RX11 system is running the Single-Job Monitor; the Foreground/Background Monitor on the system is named RKMNFB.SYS. The following commands are used to switch from Single-Job to Foreground/Background:

```
.R PIP<CR>
*DXMNSJ.SYS=MONITR.SYS/Y/R<CR>      Preserve the S/J Monitor.
?REBOOT?
*MONITR.SYS=DXMNFB.SYS/Y/R<CR>      Activate the F/B Monitor by
?REBOOT?                            renaming it to MONITR.SYS.
*DX1A=MONITR.SYS/U<CR>              Write the new bootstrap.
*DX1/G<CR>                          Reboot the system.
```

The following is a faster method of switching between monitors; this method may be used with all system devices except RX11 diskette and DECtape. This method presents no danger if the user forgets to rename the monitor or does the rename in the wrong order.

This example shows the switching of monitors on an RK11 system currently running the foreground/background monitor. Before using this method, ensure that both monitor files are present on the disk and named RKMNSJ.SYS and RKMNFB.SYS. The monitor file MONITR.SYS is also present.

```
.R PIP<CR>
*MONITR.SYS=RKMNSJ.SYS/X/Y<CR>      Create a copy of the RK11 S/J
?REBOOT?                            Monitor.
*A=MONITR.SYS/U<CR>                 Write the new bootstrap.
*RK1/O<CR>                          Reboot the system.
```

CHAPTER 5

ASSEMBLY AND LINK INSTRUCTIONS

5.1 GENERAL INSTRUCTIONS

All RT-11 components, except MACRO, ASEMBL, and the monitors, require 16K words of memory to be assembled. MACRO and ASEMBL require 20K words; the monitors require 24K words. RT-11 MACRO is used as the assembler, and RT-11 LINK is used as the Linker in all cases. All assemblies (except ODT) and all links (except where otherwise noted) should be error free.

Throughout this chapter, the following conventions are used:

1. Default extensions are not explicitly specified. For all the source files, the extensions are .MAC. The assembler output is .OBJ and Linker output is .SAV.

2. The system macro library, SYSMAC.SML, must be on the system device during all assembles given below.

3. In the example command strings, the sources are kept on logical device SRC:, binary output is to device BIN:, and listing and map files are output to LST:. In actual practice, any appropriate device can be used. The assembly and link operations were run as a BATCH stream; command lines in the following sections were taken from the output of the BATCH stream, which accounts for lack of prompting asterisks and periods in some cases.

4. The example command strings were executed on a 28K computer and the FREE CORE error messages reflect that fact. The actual number of free memory words in each V02C installation will vary, and is not important.

All RT-11 system assembling and linking operations are normal operations, and the command strings in the descriptions below can be altered to take full advantage of all RT-11 MACRO and LINK command features.

5.2 ASSEMBLING AND LINKING THE SYSTEM FILES

The result of the operations below is the 12 monitors and 14 handler files. (The BATCH run-time handler is assembled and linked in Section 5.3.14.) The UNDEF GLBLS messages resulting from RK.SYS, RF.SYS, DX.SYS, DS.SYS, DP.SYS, and DT.SYS linking are expected. The reason for this undefined global is to prevent the accidental linking of the

wrong bootstrap and system devices when linking a monitor. An undefined global will result if, e.g., an RF boot is linked with an RK driver. The monitor files are named as described in Table 1-1.

```
R MACRO
*BIN:RT11SJ,LST:RT11SJ=SRC:KMON,USR,RMONSJ,KMOVLY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 6144. WORDS

*BIN:RT11FB,LST:RT11FB=SRC:BFDEF,KMON,USR,RMONFB,KMOVLY/C/N:CND:TTM
ERRORS DETECTED: 0
FREE CORE: 4329. WORDS

*BIN:RKBTSJ,LST:RKBTSJ=SRC:BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12016. WORDS

*BIN:RFBTSJ,LST:RFBTSJ=SRC:RFSYS,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12008. WORDS

*BIN:DTBTSJ,LST:DTBTSJ=SRC:DTSYS,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12000. WORDS

*BIN:DPBTSJ,LST:DPBTSJ=SRC:DPSYS,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11979. WORDS

*BIN:DXBTSJ,LST:DXBTSJ=SRC:DXSYS,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11936. WORDS

*BIN:DSBTSJ,LST:DSBTSJ=SRC:DSSYS,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12024. WORDS

*BIN:RKBTFB,LST:RKBTFB=SRC:BFDEF,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11988. WORDS

*BIN:RFBTFB,LST:RFBTFB=SRC:RFSYS,BFDEF,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11980. WORDS

*BIN:DTBTFB,LST:DTBTFB=SRC:DTSYS,BFDEF,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11972. WORDS

*BIN:DPBTFB,LST:DPBTFB=SRC:DPSYS,BFDEF,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11951. WORDS

*BIN:DXBTFB,LST:DXBTFB=SRC:DXSYS,BFDEF,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11908. WORDS

*BIN:DSBTFB,LST:DSBTFB=SRC:DSSYS,BFDEF,BSTRAP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11996. WORDS
```

```
*BIN:DP,LST:DP=SRC:DP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12824. WORDS

*BIN:DX,LST:DX=SRC:DX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12788. WORDS

*BIN:RK,LST:RK=SRC:RK/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12965. WORDS

*BIN:RF,LST:RF=SRC:RF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12968. WORDS

*BIN:DT,LST:DT=SRC:DT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12961. WORDS

*BIN:TT,LST:TT=SRC:TT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12927. WORDS

*BIN:LP,LST:LP=SRC:LP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12945. WORDS

*BIN:PR,LST:PR=SRC:PR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13047. WORDS

*BIN:PP,LST:PP=SRC:PP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13044. WORDS

*BIN:CR,LST:CR=SRC:CR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12581. WORDS

*BIN:MT,LST:MT=SRC:MT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11869. WORDS

*BIN:MM,LST:MM=SRC:TUDEF,MT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11473. WORDS

*BIN:CT,LST:CT=SRC:CT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11970. WORDS

*BIN:DS,LST:DS=SRC:DS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12992. WORDS
```

```
R LINK
BIN:RKMNFB.SYS,LST:RKMNFB=BIN:RKBTFB,RT11FB,RK

BIN:RKMNSJ.SYS,LST:RKMNSJ=BIN:RKBTSJ,RT11SJ,RK

BIN:RFMNFB.SYS,LST:RFMNFB=BIN:RFBTFB,RT11FB,RF

BIN:RFMNSJ.SYS,LST:RFMNSJ=BIN:RFBTSJ,RT11SJ,RF

BIN:DTMNFB.SYS,LST:DTMNFB=BIN:DTBTFB,RT11FB,DT

BIN:DTMNSJ.SYS,LST:DTMNSJ=BIN:DTBTSJ,RT11SJ,DT

BIN:DPMNFB.SYS,LST:DPMNFB=BIN:DPBTFB,RT11FB,DP

BIN:DPMNSJ.SYS,LST:DPMNSJ=BIN:DPBTSJ,RT11SJ,DP

BIN:DXMNFB.SYS,LST:DXMNFB=BIN:DXBTFB,RT11FB,DX

BIN:DXMNSJ.SYS,LST:DXMNSJ=BIN:DXBTSJ,RT11SJ,DX

BIN:DSMNFB.SYS,LST:DSMNFB=BIN:DSBTFB,RT11FB,DS

BIN:DSMNSJ.SYS,LST:DSMNSJ=BIN:DSBTSJ,RT11SJ,DS

BIN:DP.SYS,LST:DP=BIN:DP
UNDEF GLBLS

BIN:DX.SYS,LST:DX=BIN:DX
UNDEF GLBLS

BIN:RK.SYS,LST:RK=BIN:RK
UNDEF GLBLS

BIN:RF.SYS,LST:RF=BIN:RF
UNDEF GLBLS

BIN:DT.SYS,LST:DT=BIN:DT
UNDEF GLBLS

BIN:TT.SYS,LST:TT=BIN:TT

BIN:LP.SYS,LST:LP=BIN:LP
```

```
BIN:PR.SYS,LST:PR=BIN:PR

BIN:PP.SYS,LST:PP=BIN:PP

BIN:CR.SYS,LST:CR=BIN:CR

BIN:MT.SYS,LST:MT=BIN:MT

BIN:MM.SYS,LST:MM=BIN:MM

BIN:CT.SYS,LST:CT=BIN:CT

BIN:DS.SYS,LST:DS=BIN:DS
UNDEF GLBLS
```

5.3 ASSEMBLING AND LINKING THE UTILITIES

5.3.1 EDIT

```
R MACRO
*BIN:VTCED1,LST:VTCED1=SRC:EDITDF,VTCAL1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15083. WORDS

*BIN:VTCED4,LST:VTCED4=SRC:EDITDF,VTCAL4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14592. WORDS

*BIN:VTBEDT,LST:VTBEDT=SRC:EDITDF,VTBASE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14844. WORDS

*BIN:EDIT,LST:EDIT=SRC:VTMAC,EDIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11198. WORDS


R LINK
BIN:EDIT,LST:EDIT=BIN:VTCED1,VTCED4,VTBEDT,EDIT
```

5.3.2 MACRO

```
R MACRO
*BIN:RTEXEC,LST:RTEXEC=SRC:RTPAR,RPARAM,RCIOCH,RTEXEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12182. WORDS

*BIN:RTMAC,LST:RTMAC=SRC:RTPAR,RPARAM,RCIOCH,MACRO3,MACRO5/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 7603. WORDS
```

```
*BIN:RTPST,LST:RTPST=SRC:RTPAR,PST/N:TTM:CND/C
ERRORS DETECTED: 0
FREE CORE: 14687. WORDS


R LINK
BIN:MACRO,LST:MACRO=BIN:RTEXEC,RTMAC,RTPST
```

5.3.3 EXPAND

```
R MACRO
*BIN:PREXEC,LST:PREXEC=SRC:PREPAR,PPARAM,PCIOCH,PREXEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13586. WORDS

*BIN:PREPAS,LST:PREPAS=SRC:PREPAR,PPARAM,PCIOCH,PREPAS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13322. WORDS


R LINK
BIN:EXPAND,LST:EXPAND=BIN:PREXEC,PREPAS
```

5.3.4 ASEMBL

```
R MACRO
*BIN:SMEXEC,LST:SMEXEC=SRC:SMPAR,RPARAM,RCIOCH,RTEXEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12424. WORDS

*BIN:SMMAC,LST:SMMAC=SRC:SMPAR,RPARAM,RCIOCH,MACRO3,MACRO5/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 8350. WORDS

*BIN:SMPST,LST:SMPST=SRC:SMPAR,PST/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14683. WORDS


R LINK
BIN:ASEMBL,LST:ASEMBL=BIN:SMEXEC,SMMAC,SMPST
```

5.3.5 CREF

```
R MACRO
*BIN:CREF,LST:CREF=SRC:CREF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13221. WORDS


R LINK
BIN:CREF,LST:CREF=BIN:CREF
```

5.3.6 LINK

```
R MACRO
*BIN:LINK0,LST:LINK0=SRC:LINK0/B:500/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12433. WORDS

*BIN:LNKOV1,LST:LNKOV1=SRC:LNKOV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12896. WORDS

*BIN:LNKOV2,LST:LNKOV2=SRC:LNKOV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13035. WORDS

*BIN:LNKV2B,LST:LNKV2B=SRC:LNKV2B/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14817. WORDS

*BIN:LNKOV3,LST:LNKOV3=SRC:LNKOV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12870. WORDS

*BIN:LNKOV4,LST:LNKOV4=SRC:LNKOV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13099. WORDS

*BIN:LNKOV5,LST:LNKOV5=SRC:LNKOV5/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12803. WORDS

R LINK
BIN:LINK,LST:LINK=BIN:LINK0/C/B:500
BIN:LNKOV1/O:1/C
BIN:LNKOV2/O:1/C
BIN:LNKOV3/O:1/C
BIN:LNKOV4/O:1/C
BIN:LNK0V5/O:1
```

5.3.7 LIBR

```
R MACRO
*BIN:LIBR0,LST:LIBR0=SRC:LIBR0/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12853. WORDS

*BIN:LIBR1,LST:LIBR1=SRC:LIBR1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12992. WORDS

*BIN:LIBR2,LST:LIBR2=SRC:LIBR2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13200. WORDS

*BIN:LIBR3,LST:LIBR3=SRC:LIBR3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13449. WORDS
```

```
*BIN:LIBR4,LST:LIBR4=SRC:LIBR4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13365. WORDS


R LINK
BIN:LIBR,LST:LIBR=BIN:LIBR0/C
BIN:LIBR1/O:1/C
BIN:LIBR2/O:1/C
BIN:LIBR3/O:1/C
BIN:LIBR4/O:1
```

5.3.8 PIP

```
R MACRO
*BIN:PIP,LST:PIP=SRC:PIP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11769. WORDS

*BIN:PIP1,LST:PIP1=SRC:PIP1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE:

R LINK
BIN:PIP,LST:PIP=BIN:PIP/C
BIN:PIP1/O:1
```

5.3.9 FILEX

```
R MACRO
*BIN:FILEX,LST:FILEX=SRC:FILEX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11391. WORDS


R LINK
BIN:FILEX,LST:FILEX=BIN:FILEX
```

5.3.10 SRCCOM

```
R MACRO
*BIN:SRCCOM,LST:SRCCOM=SRC:SRCCOM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13698. WORDS


R LINK
BIN:SRCCOM,LST:SRCCOM=BIN:SRCCOM
```

5.3.11 DUMP

```
R MACRO
*BIN:DUMP,LST:DUMP=SRC:DUMP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13986. WORDS


R LINK
BIN:DUMP,LST:DUMP=BIN:DUMP
```

5.3.12 PATCH

```
R MACRO
*BIN:PATCH,LST:PATCH=SRC:PATCH/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13909. WORDS


R LINK
BIN:PATCH,LST:PATCH=BIN:PATCH
```

5.3.13 ODT

ODT assembles with one error (a Z error which flags an ODT instruction that is machine dependent). The error is necessary and should be ignored.

```
R MACRO
*BIN:ODT,LST:ODT=SRC:ODT/C/N:TTM:CND
ERRORS DETECTED: 1
FREE CORE: 14159. WORDS
```

5.3.14 BATCH

```
R MACRO
*BIN:BA,LST:BA=SRC:BA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12238. WORDS

*BIN:BATCH,LST:BATCH=SRC:BATCH/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 5551. WORDS

R LINK
BIN:BA,SYS,LST:BA=BIN:BA


BIN:BATCH,LST:BATCH=BIN:BATCH
```

5.4 COMPILING AND LINKING PATCHO

PATCHO is written in FORTRAN, and requires RT-11 FORTRAN IV for compilation and linking. The warning messages should be ignored.

```
R FORTRA
*BIN:PAT0,LST:PAT0=SRC:PAT0/S/P/N:5/R:120
[,MAIN,] ERRORS: 000, WARNINGS: 002
*BIN:PAT1,LST:PAT1=SRC:PAT1/S/P
*BIN:PAT2,LST:PAT2=SRC:PAT2/S/P
*BIN:PAT3,LST:PAT3=SRC:PAT3/S/P
[INBLK ] ERRORS: 000, WARNINGS: 002
*BIN:PAT4,LST:PAT4=SRC:PAT4/S/P
[LIST  ] ERRORS: 000, WARNINGS: 004
*BIN:PAT5,LST:PAT5=SRC:PAT5/S/P
*BIN:PAT6,LST:PAT6=SRC:PAT6/S/P
[DUMP  ] ERRORS: 000, WARNINGS: 003

R MACRO
*BIN:IRAD50,LST:IRAD50=SRC:IRAD50/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15148. WORDS

*BIN:R50ASC,LST:R50ASC=SRC:R50ASC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15145. WORDS

*BIN:CMFCV,LST:CMFCV=SRC:CMFCV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15142. WORDS
```

The following linking instructions require that the FORTRAN library, FORLIB.OBJ, be available on the system device. FORLIB.OBJ is available as part of the RT-11 FORTRAN software kit. The UNDEF GLBLS message from the link should be ignored.

```
R LINK
BIN:PATCHO,LST:PATCHO=BIN:PAT0,CMFCV/F/I/C
BIN:PAT1,IRAD50/O:1/C
BIN:PAT3,R50ASC/O:1/C
BIN:PAT2/O:2/C
BIN:PAT4/O:2/C
BIN:PAT5/O:2/C
BIN:PAT6/O:2


LIBRARY SEARCH:
SSWORT

UNDEF GLBLS
```

5.5 ASSEMBLING AND BUILDING THE VT11 DISPLAY HANDLER LIBRARY (VTLIB)

```
R MACRO
*BIN:VTCAL1,LST:VTCAL1=SRC:VTCAL1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15087. WORDS

*BIN:VTCAL2,LST:VTCAL2=SRC:VTCAL2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15151. WORDS

*BIN:VTCAL3,LST:VTCAL3=SRC:VTCAL3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15122. WORDS

*BIN:VTCAL4,LST:VTCAL4=SRC:VTCAL4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14612. WORDS

*BIN:VTBASE,LST:VTBASE=SRC:VTBASE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14784. WORDS


R PIP
BIN:VTHDLR.OBJ=BIN:VTCAL1.OBJ,VTCAL2.OBJ,VTCAL3.OBJ/B
BIN:VTHDLR.OBJ=BIN:VTHDLR.OBJ,VTCAL4.OBJ,VTBASE.OBJ/B

R LIBR
BIN:VTLIB=BIN:VTHDLR
```

5.6 ASSEMBLING AND BUILDING THE SYSTEM SUBROUTINE LIBRARY (SYSLIB)

```
R MACRO
*BIN:LEN.SBJ,LST:LEN=SRC:LEN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15188. WORDS

*BIN:TRIM.SBJ,LST:TRIM=SRC:TRIM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15176. WORDS

*BIN:STRPAD.SBJ,LST:STRPAD=SRC:STRPAD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15161. WORDS

*BIN:VERIFY.SBJ,LST:VERIFY=SRC:VERIFY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15112. WORDS

*BIN:INSERT.SBJ,LST:INSERT=SRC:INSERT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15159. WORDS

*BIN:CONCAT.SBJ,LST:CONCAT=SRC:CONCAT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15149. WORDS

*BIN:REPEAT.SBJ,LST:REPEAT=SRC:REPEAT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15161. WORDS
```

```
*BIN:IADDR,SBJ,LST:IADDR=SRC:IADDR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15183. WORDS

*BIN:INDEX,SBJ,LST:INDEX=SRC:INDEX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15157. WORDS

*BIN:TRANSL,SBJ,LST:TRANSL=SRC:TRANSL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15163. WORDS

*BIN:SCOMP,SBJ,LST:SCOMP=SRC:SCOMP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15155. WORDS

*BIN:SUBSTR,SBJ,LST:SUBSTR=SRC:SUBSTR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15163. WORDS

*BIN:SCOPY,SBJ,LST:SCOPY=SRC:SCOPY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15169. WORDS

*BIN:CLOSEC,SBJ,LST:CLOSEC=SRC:CLOSEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ICMKT,SBJ,LST:ICMKT=SRC:ICMKT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ICSTAT,SBJ,LST:ICSTAT=SRC:ICSTAT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:GTIM,SBJ,LST:GTIM=SRC:GTIM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:IDELET,SBJ,LST:IDELET=SRC:IDELET/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15175. WORDS

*BIN:IDSTAT,SBJ,LST:IDSTAT=SRC:IDSTAT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15191. WORDS

*BIN:IENTER,SBJ,LST:IENTER=SRC:IENTER/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15174. WORDS

*BIN:IFREEC,SBJ,LST:IFREEC=SRC:IFREEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15167. WORDS

*BIN:IGETC,SBJ,LST:IGETC=SRC:IGETC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15161. WORDS

*BIN:GTJB,SBJ,LST:GTJB=SRC:GTJB/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS
```

```
*BIN:LOOKUP,SBJ,LST:LOOKUP=SRC:LOOKUP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15174. WORDS

*BIN:MWAIT,SBJ,LST:MWAIT=SRC:MWAIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:PRINT,SBJ,LST:PRINT=SRC:PRINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15183. WORDS

*BIN:PURGE,SBJ,LST:PURGE=SRC:PURGE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:RCTRLO,SBJ,LST:RCTRLO=SRC:RCTRLO/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15192. WORDS

*BIN:IREOPN,SBJ,LST:IREOPN=SRC:IREOPN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:RESUME,SBJ,LST:RESUME=SRC:RESUME/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ISAVES,SBJ,LST:ISAVES=SRC:ISAVES/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15178. WORDS

*BIN:SUSPND,SBJ,LST:SUSPND=SRC:SUSPND/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ITLOCK,SBJ,LST:ITLOCK=SRC:ITLOCK/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ITTINR,SBJ,LST:ITTINR=SRC:ITTINR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15188. WORDS

*BIN:ITTOUR,SBJ,LST:ITTOUR=SRC:ITTOUR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15183. WORDS

*BIN:ITWAIT,SBJ,LST:ITWAIT=SRC:ITWAIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:UNLOCK,SBJ,LST:UNLOCK=SRC:UNLOCK/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15192. WORDS

*BIN:IWAIT,SBJ,LST:IWAIT=SRC:IWAIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15179. WORDS

*BIN:LOCK,SBJ,LST:LOCK=SRC:LOCK/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15192. WORDS
```

```
*BIN:TIMSUB,SBJ,LST:TIMSUB=SRC:TIMSUB/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15164. WORDS

*BIN:ISLEEP,SBJ,LST:ISLEEP=SRC:ISLEEP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15178. WORDS

*BIN:JTIME,SBJ,LST:JTIME=SRC:JTIME/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15181. WORDS

*BIN:IUNTIL,SBJ,LST:IUNTIL=SRC:IUNTIL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15178. WORDS

*BIN:IREAD,SBJ,LST:IREAD=SRC:IREAD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15166. WORDS

*BIN:ISDAT,SBJ,LST:ISDAT=SRC:ISDAT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15170. WORDS

*BIN:ISPFN,SBJ,LST:ISPFN=SRC:ISPFN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15153. WORDS

*BIN:IWRITE,SBJ,LST:IWRITE=SRC:IWRITE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15162. WORDS

*BIN:IRCVD,SBJ,LST:IRCVD=SRC:IRCVD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15170. WORDS

*BIN:CVTTIM,SBJ,LST:CVTTIM=SRC:CVTTIM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15177. WORDS

*BIN:DIV60,SBJ,LST:DIV60=SRC:DIV60/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15164. WORDS

*BIN:R50ASC,SBJ,LST:R50ASC=SRC:R50ASC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15145. WORDS

*BIN:ISPY,SBJ,LST:ISPY=SRC:ISPY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15179. WORDS

*BIN:TIMASC,SBJ,LST:TIMASC=SRC:TIMASC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15165. WORDS

*BIN:IFETCH,SBJ,LST:IFETCH=SRC:IFETCH/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15155. WORDS

*BIN:IQSET,SBJ,LST:IQSET=SRC:IQSET/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15167. WORDS
```

```
*BIN:DEVICE,SBJ,LST:DEVICE=SRC:DEVICE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ICDFN,SBJ,LST:ICDFN=SRC:ICDFN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15162. WORDS

*BIN:TIME1,SBJ,LST:TIME1=SRC:TIME1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15174. WORDS

*BIN:ICHCPY,SBJ,LST:ICHCPY=SRC:ICHCPY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15178. WORDS

*BIN:SYSLBV,SBJ,LST:SYSLBV=SRC:SYSLBV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15224. WORDS

*BIN:MRKT,SBJ,LST:MRKT=SRC:MRKT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15182. WORDS

*BIN:ISDATF,SBJ,LST:ISDATF=SRC:ISDATF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15177. WORDS

*BIN:ISPFNF,SBJ,LST:ISPFNF=SRC:ISPFNF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15177. WORDS

*BIN:IWRITF,SBJ,LST:IWRITF=SRC:IWRITF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15177. WORDS

*BIN:IRCVDF,SBJ,LST:IRCVDF=SRC:IRCVDF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15177. WORDS

*BIN:ISCHED,SBJ,LST:ISCHED=SRC:ISCHED/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15162. WORDS

*BIN:IRENAM,SBJ,LST:IRENAM=SRC:IRENAM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15178. WORDS

*BIN:JADD,SBJ,LST:JADD=SRC:JADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15173. WORDS

*BIN:JSUB,SBJ,LST:JSUB=SRC:JSUB/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15173. WORDS

*BIN:JMUL,SBJ,LST:JMUL=SRC:JMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15149. WORDS

*BIN:JDIV,SBJ,LST:JDIV=SRC:JDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15133. WORDS
```

```
*BIN:JICVT,SBJ,LST:JICVT=SRC:JICVT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15181. WORDS

*BIN:IJCVT,SBJ,LST:IJCVT=SRC:IJCVT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15161. WORDS

*BIN:JFIX,SBJ,LST:JFIX=SRC:JFIX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15153. WORDS

*BIN:JFLT,SBJ,LST:JFLT=SRC:JFLT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15137. WORDS

*BIN:JMOV,SBJ,LST:JMOV=SRC:JMOV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15175. WORDS

*BIN:IREADF,SBJ,LST:IREADF=SRC:IREADF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15177. WORDS

*BIN:CMPLT,SBJ,LST:CMPLT=SRC:CMPLT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14955. WORDS

*BIN:IRAD50,SBJ,LST:IRAD50=SRC:IRAD50/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15148. WORDS

*BIN:CHAIN,SBJ,LST:CHAIN=SRC:CHAIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15174. WORDS

*BIN:RCHAIN,SBJ,LST:RCHAIN=SRC:RCHAIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15175. WORDS

*BIN:SECNDS,SBJ,LST:SECNDS=SRC:SECNDS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15170. WORDS

*BIN:JCMP,SBJ,LST:JCMP=SRC:JCMP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15167. WORDS

*BIN:ILUN,SBJ,LST:ILUN=SRC:ILUN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15159. WORDS

*BIN:IASIGN,SBJ,LST:IASIGN=SRC:IASIGN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15092. WORDS

*BIN:TIME,SBJ,LST:TIME=SRC:TIME/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15178. WORDS

*BIN:JJCVT,SBJ,LST:JJCVT=SRC:JJCVT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15183. WORDS
```

```
*BIN:INTSET,SBJ,LST:INTSET=SRC:INTSET/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15094. WORDS

*BIN:ICSI,SBJ,LST:ICSI=SRC:ICSI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15137. WORDS


R FORTRA
*BIN:PUTSTR,SBJ,LST:PUTSTR=SRC:PUTSTR/S
*BIN:GETSTR,SBJ,LST:GETSTR=SRC:GETSTR/S

R PIP
BIN:SYSF4,OBJ=BIN:*,SBJ/B

R LIBR
BIN:SYSLIB=BIN:SYSF4
```

5.7 ASSEMBLING AND LINKING MBUILD

```
R MACRO
*BIN:MBOOT,LST:MBOOT=SRC:MBOOT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 16435. WORDS

*BIN:MSBOOT,LST:MSBOOT=SRC:MSBOOT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 16267. WORDS

*BIN:MTINIT,LST:MTINIT=SRC:MTINIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 15377. WORDS

*BIN:MBUILD,LST:MBUILD=SRC:MBUILD,PIP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13471. WORDS

*BIN:MBLD1,LST:MBLD1=SRC:MBUILD,PIP1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 12582. WORDS

R LINK
BIN:MBOOT,BOT,LST:MBOOT=BIN:MBOOT


BIN:MSBOOT,BOT,LST:MSBOOT=BIN:MSBOOT


BIN:MTINIT,LST:MTINIT=BIN:MTINIT


BIN:MBUILD,LST:MBUILD=BIN:MBLD1,MBUILD
```

5.8 ASSEMBLING AND LINKING THE FORTRAN IV COMPILER

This section provides assembly and linking instructions for the RT-11 FORTRAN Compiler. This information applies only to those users who received the source versions of the Compiler.

All assemblies listed in this section were actually run as a BATCH stream under the RT-11 BATCH Compiler. BIN, LST, and SRC are logical device names used for the binary output device, listing output device, and source input device, respectively. The user can employ any legal MACRO assembly command that suits the need; if no listing is desired, the list file specification may be omitted entirely.

All assemblies require that the system macro file SYSMAC.SML be present on the system device.

5.8.1 Compiler Assembly

Below is an example of the Compiler assembly procedure.

```
$JOB/TIME/RT11

$RT11

00:48:16

        TTYIO

R MACRO
*BIN:FROOT,LST:FROOT=SRC:FORTHD,FROOT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14486. WORDS

*BIN:F0,LST:F0=SRC:FORTHD,F0/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13057. WORDS

*BIN:F1,LST:F1=SRC:FORTHD,F1,F1S/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13507. WORDS

*BIN:F2,LST:F2=SRC:FORTHD,F2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13594. WORDS

*BIN:F3,LST:F3=SRC:FORTHD,F3,F3S/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13502. WORDS

*BIN:F4,LST:F4=SRC:FORTHD,F4,F3S/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13479. WORDS

*BIN:F5,LST:F5=SRC:FORTHD,F5,F3S/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13454. WORDS

*BIN:F6,LST:F6=SRC:FORTHD,F6/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13741. WORDS

*BIN:F7,LST:F7=SRC:F7/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13624. WORDS

*BIN:F8,LST:F8=SRC:F8/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13525. WORDS
```

```
*BIN:F9,LST:F9=SRC:F9/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 13718. WORDS

*BIN:F10,LST:F10=SRC:F10/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14546. WORDS

*BIN:F11,LST:F11=SRC:F11/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 14335. WORDS

*BIN:F12,LST:F12=SRC:F12P,FBEGIN,F12/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9905. WORDS

*BIN:F13,LST:F13=SRC:F13P,FBEGIN,F13/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10013. WORDS

*BIN:F14,LST:F14=SRC:F14P,FBEGIN,FDRIVE,FWRT,F14A,F14B/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10742. WORDS

*BIN:F15,LST:F15=SRC:F15P,FBEGIN,FDRIVE,F15,FWRT,FCODE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9840. WORDS

*BIN:F16,LST:F16=SRC:F16P,FBEGIN,FDRIVE,F16,FWRT,FCODE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9812. WORDS

*BIN:F17,LST:F17=SRC:F17P,FBEGIN,FDRIVE,F17A,F17B,FCODE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9668. WORDS

*
$EOJ

TIME
01:08:52
```

5.8.2 Compiler Linking

Below is an example of the Compiler linking procedures.

```
$JOB/TIME/RT11

$RT11

01:08:55

        TTYIO

R LINK
BIN:FORTRA,LST:FORTRA=BIN:FROOT/C
BIN:F0/O:1/C
BIN:F1/O:1/C
BIN:F2/O:1/C
BIN:F3/O:1/C
BIN:F4/O:1/C
BIN:F5/O:1/C
```

```
BIN:F6/O:1/C
BIN:F7/O:1/C
BIN:F8/O:1/C
BIN:F9/O:1/C
BIN:F10/O:1/C
BIN:F11/O:1/C
BIN:F12/O:1/C
BIN:F13/O:1/C
BIN:F14/O:1/C
BIN:F15/O:1/C
BIN:F16/O:1/C
BIN:F17/O:1


$EOJ

TIME
01:09:29
```

5.9 ASSEMBLING COMMON FORTRAN OTS MODULES

This section provides assembly instructions for the RT-11 FORTRAN Object Time System (OTS) library modules that are hardware independent and are therefore common to all OTS libraries. This information applies only to those users who received the source versions of OTS.

Below is an example of the assembly procedures for the common OTS library modules.

```
$JOB/TIME/RT11

$RT11

23:55:53

        TTYIO

R PIP
BIN:F,MAC=SRC:V2DEF,PRE,FINIT,PRE,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE/A

R MACRO
*BIN:ABS,ALL,LST:ABS=BIN:F,SRC:ABS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:IABS,ALL,LST:IABS=BIN:F,SRC:IABS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11189. WORDS

*BIN:DABS,ALL,LST:DABS=BIN:F,SRC:DABS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:CABS,ALL,LST:CABS=BIN:F,SRC:CABS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11341. WORDS

*BIN:FLOAT,ALL,LST:FLOAT=BIN:F,SRC:FLOAT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS
```

```
*BIN:DIM,ALL,LST:DIM=BIN:F,SRC:DIM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11381. WORDS

*BIN:IDIM,ALL,LST:IDIM=BIN:F,SRC:IDIM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11181. WORDS

*BIN:CEXP,ALL,LST:CEXP=BIN:F,SRC:CEXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11357. WORDS

*BIN:CSIN,ALL,LST:CSIN=BIN:F,SRC:CSIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11309. WORDS

*BIN:TANH,ALL,LST:TANH=BIN:F,SRC:TANH/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11301. WORDS

*BIN:CONJG,ALL,LST:CONJG=BIN:F,SRC:CONJG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:IFIX,ALL,LST:IFIX=BIN:F,SRC:IFIX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:DBLE,ALL,LST:DBLE=BIN:F,SRC:DBLE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:REAL,ALL,LST:REAL=BIN:F,SRC:REAL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:AIMAG,ALL,LST:AIMAG=BIN:F,SRC:AIMAG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:CMPLX,ALL,LST:CMPLX=BIN:F,SRC:CMPLX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:INT,ALL,LST:INT=BIN:F,SRC:INT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:AMOD,ALL,LST:AMOD=BIN:F,SRC:AMOD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11361. WORDS

*BIN:DMOD,ALL,LST:DMOD=BIN:F,SRC:DMOD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11361. WORDS

*BIN:MOD,ALL,LST:MOD=BIN:F,SRC:MOD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11381. WORDS

*BIN:MAX0,ALL,LST:MAX0=BIN:F,SRC:MAX0/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS
```

```
*BIN:AMIN0,ALL,LST:AMIN0=BIN:F,SRC:AMIN0/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11357. WORDS

*BIN:MIN0,ALL,LST:MIN0=BIN:F,SRC:MIN0/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:DMIN1,ALL,LST:DMIN1=BIN:F,SRC:DMIN1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11358. WORDS

*BIN:SIGN,ALL,LST:SIGN=BIN:F,SRC:SIGN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:ISIGN,ALL,LST:ISIGN=BIN:F,SRC:ISIGN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11185. WORDS

*BIN:DSIGN,ALL,LST:DSIGN=BIN:F,SRC:DSIGN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:CSQRT,ALL,LST:CSQRT=BIN:F,SRC:CSQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:SNGL,ALL,LST:SNGL=BIN:F,SRC:SNGL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11185. WORDS

*BIN:ENDERR,ALL,LST:ENDERR=BIN:F,SRC:ENDERR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11029. WORDS

*BIN:CONV5,ALL,LST:CONV5=BIN:F,SRC:CONV5/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11371. WORDS

*BIN:WAIT,ALL,LST:WAIT=BIN:F,SRC:WAIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11298. WORDS

*BIN:EOF,ALL,LST:EOF=BIN:F,SRC:EOF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10808. WORDS

*BIN:FNEG,ALL,LST:FNEG=BIN:F,SRC:FNEG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11361. WORDS

*BIN:XCI,ALL,LST:XCI=BIN:F,SRC:XCI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11133. WORDS

*BIN:RETD,ALL,LST:RETD=BIN:F,SRC:RETD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11162. WORDS

*BIN:IFW,ALL,LST:IFW=BIN:F,SRC:IFW/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10809. WORDS
```

```
*BIN:IFR,ALL,LST:IFR=BIN:F,SRC:IFR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10821. WORDS

*BIN:IBR,ALL,LST:IBR=BIN:F,SRC:IBR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*BIN:IBW,ALL,LST:IBW=BIN:F,SRC:IBW/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*BIN:FCHNL,ALL,LST:FCHNL=BIN:F,SRC:FCHNL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10817. WORDS

*BIN:CMPF,ALL,LST:CMPF=BIN:F,SRC:CMPF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11293. WORDS

*BIN:CMPD,ALL,LST:CMPD=BIN:F,SRC:CMPD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11295. WORDS

*BIN:INITIO,ALL,LST:INITIO=BIN:F,SRC:INITIO/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10825. WORDS

*BIN:UIO,ALL,LST:UIO=BIN:F,SRC:UIO/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10688. WORDS

*BIN:REWIND,ALL,LST:REWIND=BIN:F,SRC:REWIND/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11017. WORDS

*BIN:DUMPLA,ALL,LST:DUMPLA=BIN:F,SRC:DUMPLA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11205. WORDS

*BIN:CLOSE,ALL,LST:CLOSE=BIN:F,SRC:CLOSE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10924. WORDS

*BIN:VTRAN,ALL,LST:VTRAN=BIN:F,SRC:VTRAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:GETFIL,ALL,LST:GETFIL=BIN:F,SRC:GETFIL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10837. WORDS

*BIN:EOL,ALL,LST:EOL=BIN:F,SRC:EOL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11021. WORDS

*BIN:CALL,ALL,LST:CALL=BIN:F,SRC:CALL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:ISNLSN,ALL,LST:ISNLSN=BIN:F,SRC:ISNLSN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11164. WORDS
```

```
*BIN:IPMOVS,ALL,LST:IPMOVS=BIN:F,SRC:IPMOVS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11365. WORDS

*BIN:IADDS,ALL,LST:IADDS=BIN:F,SRC:IADDS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:IPADDS,ALL,LST:IPADDS=BIN:F,SRC:IPADDS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11377. WORDS

*BIN:ISUBS,ALL,LST:ISUBS=BIN:F,SRC:ISUBS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:IPSUBS,ALL,LST:IPSUBS=BIN:F,SRC:IPSUBS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11377. WORDS

*BIN:INCR,ALL,LST:INCR=BIN:F,SRC:INCR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11377. WORDS

*BIN:INEG,ALL,LST:INEG=BIN:F,SRC:INEG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:TESTS,ALL,LST:TESTS=BIN:F,SRC:TESTS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11361. WORDS

*BIN:ICMPS,ALL,LST:ICMPS=BIN:F,SRC:ICMPS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:IPCMPS,ALL,LST:IPCMPS=BIN:F,SRC:IPCMPS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11377. WORDS

*BIN:BRAS,ALL,LST:BRAS=BIN:F,SRC:BRAS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11362. WORDS

*BIN:NXT1,ALL,LST:NXT1=BIN:F,SRC:NXT1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11351. WORDS

*BIN:NXT2,ALL,LST:NXT2=BIN:F,SRC:NXT2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11351. WORDS

*BIN:NXT3,ALL,LST:NXT3=BIN:F,SRC:NXT3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11351. WORDS

*BIN:NXT4,ALL,LST:NXT4=BIN:F,SRC:NXT4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11379. WORDS

*BIN:AIF,ALL,LST:AIF=BIN:F,SRC:AIF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11398. WORDS
```

```
*BIN:FMOVR,ALL,LST:FMOVR=BIN:F,SRC:FMOVR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:FMOV1,ALL,LST:FMOV1=BIN:F,SRC:FMOV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:FMOV2,ALL,LST:FMOV2=BIN:F,SRC:FMOV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11401. WORDS

*BIN:FMOV3,ALL,LST:FMOV3=BIN:F,SRC:FMOV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11389. WORDS

*BIN:FMOV4,ALL,LST:FMOV4=BIN:F,SRC:FMOV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:FMOV5,ALL,LST:FMOV5=BIN:F,SRC:FMOV5/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11405. WORDS

*BIN:FMOV6,ALL,LST:FMOV6=BIN:F,SRC:FMOV6/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:FMOV7,ALL,LST:FMOV7=BIN:F,SRC:FMOV7/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:FMOV8,ALL,LST:FMOV8=BIN:F,SRC:FMOV8/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11389. WORDS

*BIN:FMOV9,ALL,LST:FMOV9=BIN:F,SRC:FMOV9/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11365. WORDS

*BIN:LOADS,ALL,LST:LOADS=BIN:F,SRC:LOADS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11367. WORDS

*BIN:DMOVR,ALL,LST:DMOVR=BIN:F,SRC:DMOVR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:DMOV1,ALL,LST:DMOV1=BIN:F,SRC:DMOV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11405. WORDS

*BIN:DMOV2,ALL,LST:DMOV2=BIN:F,SRC:DMOV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:DMOV3,ALL,LST:DMOV3=BIN:F,SRC:DMOV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11369. WORDS

*BIN:DMOV4,ALL,LST:DMOV4=BIN:F,SRC:DMOV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS
```

```
*BIN:DMOV5,ALL,LST:DMOV5=BIN:F,SRC:DMOV5/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11381. WORDS

*BIN:DMOV6,ALL,LST:DMOV6=BIN:F,SRC:DMOV6/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11365. WORDS

*BIN:DMOV7,ALL,LST:DMOV7=BIN:F,SRC:DMOV7/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:LMOVS,ALL,LST:LMOVS=BIN:F,SRC:LMOVS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11345. WORDS

*BIN:BITDID,ALL,LST:BITDID=BIN:F,SRC:BITDID/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:LTEST,ALL,LST:LTEST=BIN:F,SRC:LTEST/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:LNOTS,ALL,LST:LNOTS=BIN:F,SRC:LNOTS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11389. WORDS

*BIN:LCMPS,ALL,LST:LCMPS=BIN:F,SRC:LCMPS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:LCMPSP,ALL,LST:LCMPSP=BIN:F,SRC:LCMPSP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:LPCMPS,ALL,LST:LPCMPS=BIN:F,SRC:LPCMPS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:QVEC,ALL,LST:QVEC=BIN:F,SRC:QVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:QVECP,ALL,LST:QVECP=BIN:F,SRC:QVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:QPVEC,ALL,LST:QPVEC=BIN:F,SRC:QPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11401. WORDS

*BIN:SUBR,ALL,LST:SUBR=BIN:F,SRC:SUBR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11161. WORDS

*BIN:LCMPSI,ALL,LST:LCMPSI=BIN:F,SRC:LCMPSI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11401. WORDS

*BIN:CONV6,ALL,LST:CONV6=BIN:F,SRC:CONV6/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11378. WORDS
```

```
*BIN:XFF,ALL,LST:XFF=BIN:F,SRC:XFF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:XDD,ALL,LST:XDD=BIN:F,SRC:XDD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11109. WORDS

*BIN:RETS,ALL,LST:RETS=BIN:F,SRC:RETS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:ASFRET,ALL,LST:ASFRET=BIN:F,SRC:ASFRET/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11383. WORDS

*BIN:RETDSF,ALL,LST:RETDSF=BIN:F,SRC:RETDSF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11170. WORDS

*BIN:TCMPLX,ALL,LST:TCMPLX=BIN:F,SRC:TCMPLX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11145. WORDS

*BIN:TRARY,ALL,LST:TRARY=BIN:F,SRC:TRARY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11125. WORDS

*BIN:FUD,ALL,LST:FUD=BIN:F,SRC:FUD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11197. WORDS

*BIN:IMOVS,ALL,LST:IMOVS=BIN:F,SRC:IMOVS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11333. WORDS

*BIN:IMOVR,ALL,LST:IMOVR=BIN:F,SRC:IMOVR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11389. WORDS

*BIN:LMOVR,ALL,LST:LMOVR=BIN:F,SRC:LMOVR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:GOTO,ALL,LST:GOTO=BIN:F,SRC:GOTO/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11185. WORDS

*BIN:PUTREC,ALL,LST:PUTREC=BIN:F,SRC:PUTREC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10768. WORDS

*BIN:RAN,ALL,LST:RAN=BIN:F,SRC:RAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11169. WORDS

*BIN:RANDU,ALL,LST:RANDU=BIN:F,SRC:RANDU/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11169. WORDS

*BIN:FIND,ALL,LST:FIND=BIN:F,SRC:FIND/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10809. WORDS
```

```
*BIN:OPEN,ALL,LST:OPEN=BIN:F,SRC:OPEN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10694. WORDS

*BIN:VCTRAN,ALL,LST:VCTRAN=BIN:F,SRC:VCTRAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:CONVI,ALL,LST:CONVI=BIN:F,SRC:CONVI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11275. WORDS

*BIN:CONVL,ALL,LST:CONVL=BIN:F,SRC:CONVL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11368. WORDS

*BIN:IDATE,ALL,LST:IDATE=BIN:F,SRC:IDATE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10853. WORDS

*BIN:DATE,ALL,LST:DATE=BIN:F,SRC:DATE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10993. WORDS

*BIN:SETERR,ALL,LST:SETERR=BIN:F,SRC:SETERR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10965. WORDS

*BIN:ASSIGN,ALL,LST:ASSIGN=BIN:F,SRC:ASSIGN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10681. WORDS

*BIN:RIO,ALL,LST:RIO=BIN:F,SRC:RIO/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10709. WORDS

*BIN:CLS,ALL,LST:CLS=BIN:F,SRC:CLS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11033. WORDS

*BIN:AMAX1,ALL,LST:AMAX1=BIN:F,SRC:AMAX1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11329. WORDS

*BIN:4K,SIZ,LST:4K=BIN:F,SRC:4K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:8K,SIZ,LST:8K=BIN:F,SRC:8K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:12K,SIZ,LST:12K=BIN:F,SRC:12K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:16K,SIZ,LST:16K=BIN:F,SRC:16K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:20K,SIZ,LST:20K=BIN:F,SRC:20K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS
```

```
*BIN:24K,SIZ,LST:24K=BIN:F,SRC:24K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:28K,SIZ,LST:28K=BIN:F,SRC:28K/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11406. WORDS

*BIN:SIMRT,UNI,LST:SIMRT=BIN:F,SRC:SIMRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10765. WORDS

*BIN:POPR,ALL,LST:POPR=BIN:F,SRC:POPR/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11397. WORDS

*BIN:CNEG,ALL,LST:CNEG=BIN:F,SRC:CNEG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11377. WORDS

*BIN:COPY,ALL,LST:COPY=BIN:F,SRC:COPY/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11389. WORDS

*BIN:GETREC,ALL,LST:GETREC=BIN:F,SRC:GETREC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10725. WORDS

*BIN:FCALL,ALL,LST:FCALL=BIN:F,SRC:FCALL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11401. WORDS

*BIN:PAUSE,ALL,LST:PAUSE=BIN:F,SRC:PAUSE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11260. WORDS

*BIN:BACKSP,ALL,LST:BACKSP=BIN:F,SRC:BACKSP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10973. WORDS

*BIN:OBJFMT,ALL,LST:OBJFMT=BIN:F,SRC:OBJFMT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10558. WORDS

*BIN:RWBLK,ALL,LST:RWBLK=BIN:F,SRC:RWBLK/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10773. WORDS

*BIN:CLOG,ALL,LST:CLOG=BIN:F,SRC:CLOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11381. WORDS

*BIN:STOP,ALL,LST:STOP=BIN:F,SRC:STOP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11047. WORDS

*BIN:EXIT,ALL,LST:EXIT=BIN:F,SRC:EXIT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11395. WORDS

*BIN:ENCODE,ALL,LST:ENCODE=BIN:F,SRC:ENCODE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10809. WORDS
```

```
*BIN:OBJENC,ALL,LST:OBJENC=BIN:F,SRC:OBJENC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10825. WORDS

*BIN:USEREX,ALL,LST:USEREX=BIN:F,SRC:USEREX/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10977. WORDS

*BIN:FIO,ALL,LST:FIO=BIN:F,SRC:FIO/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10516. WORDS

*BIN:TESTC,ALL,LST:TESTC=BIN:F,SRC:TESTC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*
$EOJ

TIME
00:32:39
```

5.9.1 Subscripting Modules Assembly

The FORTRAN library modules used for subscripting purposes are grouped into two categories, V2NS and V2S. V2NS modules (Section 5.9.1.1) are the normal library modules used to link debugged programs for production purposes or when execution time is critical. V2S modules (Section 5.9.1.2) constitute a special version of the library containing subscript checking to determine whether an array subscript reference is within the bounds of the program and are used for debugging or running in a Foreground/Background environment.

5.9.1.1 V2NS OTS Assembly

Below is an example of the assembly procedures for the V2NS modules.

```
$JOB/RT11/TIME ASSEMBLE V2NS MODULES
        TTYIO

R PIP
SRC:F,MAC=SRC:FINIT,PRE,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE,V2DEF,PRE/A

R MACRO
*BIN:IVEC,V2N,LST:IVEC,VLN=SRC:F,IVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:FVEC,V2N,LST:FVEC,VLN=SRC:F,FVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:DVEC,V2N,LST:DVEC,VLN=SRC:F,DVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11371. WORDS

*BIN:LVEC,V2N,LST:LVEC,VLN=SRC:F,LVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS
```

```
*BIN:IVECP,V2N,LST:IVECP,VLN=SRC:F,IVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:FVECP,V2N,LST:FVECP,VLN=SRC:F,FVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:DVECP,V2N,LST:DVECP,VLN=SRC:F,DVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11371. WORDS

*BIN:LVECP,V2N,LST:LVECP,VLN=SRC:F, VECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:FPVEC,V2N,LST:FPVEC,VLN=SRC:F,FPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11381. WORDS

*BIN:IPVEC,V2N,LST:IPVEC,VLN=SRC:F,IPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:DPVEC,V2N,LST:DPVEC,VLN=SRC:F,DPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11383. WORDS

*BIN:LPVEC,V2N,LST:LPVEC,VLN=SRC:F,LPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11393. WORDS

*BIN:SHORT,E2N,LST:SHORT,VLN=SRC:F,SHORT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9610. WORDS

*BIN:ERRS,V2N,LST:ERRS,VLN=SRC:F,ERRS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9430. WORDS

*BIN:FUDGE,V2N,LST:FUDGE,VLN=SRC:F,FUDGE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11402. WORDS

*
$EOJ

TIME
00:40:45
```

5.9.1.2 V2S OTS Assembly

Below is an example of the assembly procedures for the V2S modules.

```
$JOB/RT11/TIME ASSEMBLE V2S MODULES
        TTYIO

R PIP
SRC:F,MAC=SRC:FINIT.PRE,OTSWA,PRE.FBLOCK,PRE,ERRORS,PRE,V2SDEF,PRE/A
```

```
R MACRO
*BIN:IVEC,V2S,LST:IVEC,VLS=SRC:F,IVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10941. WORDS

*BIN:FVEC,V2S,LST:FVEC,VLS=SRC:F,FVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11365. WORDS

*BIN:DVEC,V2S,LST:DVEC,VLS=SRC:F,DVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11363. WORDS

*BIN:LVEC,V2S,LST:LVEC,VLS=SRC:F,LVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11371. WORDS

*BIN:IVECP,V2S,LST:IVECP,VLS=SRC:F,IVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11371. WORDS

*BIN:FVECP,V2S,LST:FVECP,VLS=SRC:F,FVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11365. WORDS

*BIN:DVECP,V2S,LST:DVECP,VLS=SRC:F,DVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11363. WORDS

*BIN:LVECP,V2S,LST:LVECP,VLS=SRC:F,LVECP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11371. WORDS

*BIN:FPVEC,V2S,LST:FPVEC,VLS=SRC:F,FPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:IPVEC,V2S,LST:IPVEC,VLS=SRC:F,IPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11379. WORDS

*BIN:DPVEC,V2S,LST:DPVEC,VLS=SRC:F,DPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11375. WORDS

*BIN:LPVEC,V2S,LST:LPVEC,VLS=SRC:F,LPVEC/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11379. WORDS

*BIN:SHORT,E2S,LST:SHORT,VLS=SRC:F,SHORT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9602. WORDS

*BIN:ERRS,V2S,LST:ERRS,VLS=SRC:F,ERRS/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 9422. WORDS
```

```
*BIN:FUDGE,V2S,LST:FUDGE,VLS=SRC:F,FUDGE/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11398. WORDS

*
$EOJ

TIME
00:36:45
```

5.10 ASSEMBLING HARDWARE DEPENDENT FORTRAN OTS MODULES

Certain OTS library modules are hardware dependent and therefore need to be selectively assembled. These modules include bare machine (Section 5.10.1), EIS (Section 5.10.2), FIS (Section 5.10.3), EAE (Section 5.10.4), and FPU (Section 5.10.5). Refer to the section applicable to your hardware configuration.

5.10.1 Bare Machine OTS Assembly

The following is an example of the assembly procedures for the hardware dependent modules on a machine with none of the optional arithmetic hardware extensions.

```
$JOB/TIME/RT11

$RT11

23:44:42

        TTYIO

R PIP
BIN:F,MAC=SRC:V2DEF,PRE,FINIT,PRE,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE/A

R MACRO
*BIN:OTI,NHD,LST:OTI,NHL=BIN:F,SRC:OTI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10293. WORDS

*BIN:ATAN,NHD,LST:ATAN,NHL=BIN:F,SRC:ATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11276. WORDS

*BIN:XII,NHD,LST:XII,NHL=BIN:F,SRC:XII/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11133. WORDS

*BIN:CONVF,NHD,LST:CONVF,NHL=BIN:F,SRC:CONVF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11036. WORDS

*BIN:CONV2,NHD,LST:CONV2,NHL=BIN:F,SRC:CONV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11137. WORDS

*BIN:CMUL,NHD,LST:CMUL,NHL=BIN:F,SRC:CMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11328. WORDS
```

```
*BIN:CDIV,NHD,LST:CDIV,NHL=BIN:F,SRC:CDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11085. WORDS

*BIN:DMUL,NHD,LST:DMUL,NHL=BIN:F,SRC:DMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11069. WORDS

*BIN:DADD,NHD,LST:DADD,NHL=BIN:F,SRC:DADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10981. WORDS

*BIN:DDIV,NHD,LST:DDIV,NHL=BIN:F,SRC:DDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11077. WORDS

*BIN:XDI,NHD,LST:XDI,NHL=BIN:F,SRC:XDI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11133. WORDS

*BIN:CONV1,NHD,LST:CONV1,NHL=BIN:F,SRC:CONV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11385. WORDS

*BIN:CONV3,NHD,LST:CONV3,NHL=BIN:F,SRC:CONV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11361. WORDS

*BIN:CADD,NHD,LST:CADD,NHL=BIN:F,SRC:CADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11337. WORDS

*BIN:ALOG,NHD,LST:ALOG,NHL=BIN:F,SRC:ALOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11093. WORDS

*BIN:DEXP,NHD,LST:DEXP,NHL=BIN:F,SRC:DEXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11053. WORDS

*BIN:EXP,NHD,LST:EXP,NHL=BIN:F,SRC:EXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11113. WORDS

*BIN:DSQRT,NHD,LST:DSQRT,NHL=BIN:F,SRC:DSQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11157. WORDS

*BIN:SIN,NHD,LST:SIN,NHL=BIN:F,SRC:SIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11312. WORDS

*BIN:DSIN,NHD,LST:DSIN,NHL=BIN:F,SRC:DSIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11272. WORDS

*BIN:DATAN,NHD,LST:DATAN,NHL=BIN:F,SRC:DATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11272. WORDS

*BIN:AINT,NHD,LST:AINT,NHL=BIN:F,SRC:AINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11367. WORDS
```

```
*BIN:DINT,NHD,LST:DINT,NHL=BIN:F,SRC:DINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11367. WORDS

*BIN:ADDA,NHD,LST:ADDA,NHL=BIN:F,SRC:ADDA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11347. WORDS

*BIN:SQRT,NHD,LST:SQRT,NHL=BIN:F,SRC:SQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11153. WORDS

*BIN:FMUL,NHD,LST:FMUL,NHL=BIN:F,SRC:FMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11121. WORDS

*BIN:FDIV,NHD,LST:FDIV,NHL=BIN:F,SRC:FDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11097. WORDS

*BIN:ADDM,NHD,LST:ADDM,NHL=BIN:F,SRC:ADDM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11347. WORDS

*BIN:ADDP,NHD,LST:ADDP,NHL=BIN:F,SRC:ADDP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11347. WORDS

*BIN:DLOG,NHD,LST:DLOG,NHL=BIN:F,SRC:DLOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11085. WORDS

*BIN:FADD,NHD,LST:FADD,NHL=BIN:F,SRC:FADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11057. WORDS

*BIN:XFI,NHD,LST:XFI,NHL=BIN:F,SRC:XFI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11133. WORDS

*BIN:CONV4,NHD,LST:CONV4,NHL=BIN:F,SRC:CONV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11185. WORDS

*BIN:IMUL,NHD,LST:IMUL,NHL=BIN:F,SRC:IMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:IDIV,NHD,LST:IDIV,NHL=BIN:F,SRC:IDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11141. WORDS

*
$EOJ

TIME
23:54:17
```

5.10.2 EIS OTS Assembly

Below is an example of the assembly procedures for the hardware dependent modules on a machine with the EIS hardware option.

```
$JOB/TIME/RT11

$RT11

23:15:57

        TTYIO

R PIP
BIN:F,MAC=SRC:V2DEF,PRE,FINIT,EIS,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE/A

R MACRO
*BIN:OTI,EIS,LST:OTI,EIL=BIN:F,SRC:OTI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10285. WORDS

*BIN:ATAN,EIS,LST:ATAN,EIL=BIN:F,SRC:ATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11268. WORDS

*BIN:XII,EIS,LST:XII,EIL=BIN:F,SRC:XII/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11141. WORDS

*BIN:CONVF,EIS,LST:CONVF,EIL=BIN:F,SRC:CONVF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11036. WORDS

*BIN:CONV2,EIS,LST:CONV2,EIL=BIN:F,SRC:CONV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11121. WORDS

*BIN:CMUL,EIS,LST:CMUL,EIL=BIN:F,SRC:CMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11320. WORDS

*BIN:CDIV,EIS,LST:CDIV,EIL=BIN:F,SRC:CDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11077. WORDS

*BIN:DMUL,EIS,LST:DMUL,EIL=BIN:F,SRC:DMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11077. WORDS

*BIN:DADD,EIS,LST:DADD,EIL=BIN:F,SRC:DADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10977. WORDS

*BIN:DDIV,EIS,LST:DDIV,EIL=BIN:F,SRC:DDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11069. WORDS

*BIN:XDI,EIS,LST:XDI,EIL=BIN:F,SRC:XDI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11125. WORDS

*BIN:CONV1,EIS,LST:CONV1,EIL=BIN:F,SRC:CONV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11377. WORDS
```

```
*BIN:CONV3,EIS,LST:CONV3,EIL=BIN:F,SRC:CONV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11353. WORDS

*BIN:CADD,EIS,LST:CADD,EIL=BIN:F,SRC:CADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11329. WORDS

*BIN:ALOG,EIS,LST:ALOG,EIL=BIN:F,SRC:ALOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11085. WORDS

*BIN:DEXP,EIS,LST:DEXP,EIL=BIN:F,SRC:DEXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11045. WORDS

*BIN:EXP,EIS,LST:EXP,EIL=BIN:F,SRC:EXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11105. WORDS

*BIN:DSQRT,EIS,LST:DSQRT,EIL=BIN:F,SRC:DSQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:SIN,EIS,LST:SIN,EIL=BIN:F,SRC:SIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11304. WORDS

*BIN:DSIN,EIS,LST:DSIN,EIL=BIN:F,SRC:DSIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11264. WORDS

*BIN:DATAN,EIS,LST:DATAN,EIL=BIN:F,SRC:DATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11264. WORDS

*BIN:AINT,EIS,LST:AINT,EIL=BIN:F,SRC:AINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11367. WORDS

*BIN:DINT,EIS,LST:DINT,EIL=BIN:F,SRC:DINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11367. WORDS

*BIN:ADDA,EIS,LST:ADDA,EIL=BIN:F,SRC:ADDA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11339. WORDS

*BIN:SQRT,EIS,LST:SQRT,EIL=BIN:F,SRC:SQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11145. WORDS

*BIN:FMUL,EIS,LST:FMUL,EIL=BIN:F,SRC:FMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11085. WORDS

*BIN:FDIV,EIS,LST:FDIV,EIL=BIN:F,SRC:FDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11081. WORDS

*BIN:ADDM,EIS,LST:ADDM,EIL=BIN:F,SRC:ADDM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11339. WORDS
```

```
*BIN:ADDP,FIS,LST:ADDP,EIL=BIN:F,SRC:ADDP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11339. WORDS

*BIN:DLOG,FIS,LST:DLOG,EIL=BIN:F,SRC:DLOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11077. WORDS

*BIN:FADD,FIS,LST:FADD,EIL=BIN:F,SRC:FADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11049. WORDS

*BIN:XFI,FIS,LST:XFI,EIL=BIN:F,SRC:XFI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11125. WORDS

*BIN:CONV4,FIS,LST:CONV4,EIL=BIN:F,SRC:CONV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11177. WORDS

*BIN:IMUL,FIS,LST:IMUL,EIL=BIN:F,SRC:IMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*BIN:IDIV,FIS,LST:IDIV,EIL=BIN:F,SRC:IDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*
$EOJ

TIME
23:25:11
```

5.10.3 FIS OTS Assembly

Below is an example of the assembly procedures for the hardware dependent modules on a machine with the FIS hardware option.

```
$JOB/TIME/RT11

$RT11

23:25:22

        TTYIO

R PIP
BIN:F,MAC=SRC:V2DEF,PRE,FINIT,FIS,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE/A

R MACRO
*BIN:OTI,FIS,LST:OTI,FIL=BIN:F,SRC:OTI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10261. WORDS

*BIN:ATAN,FIS,LST:ATAN,FIL=BIN:F,SRC:ATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11264. WORDS

*BIN:XII,FIS,LST:XII,FIL=BIN:F,SRC:XII/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11137. WORDS
```

```
*BIN:CONVF,FIS,LST:CONVF,FIL=BIN:F,SRC:CONVF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11032. WORDS

*BIN:CONV2,FIS,LST:CONV2,FIL=BIN:F,SRC:CONV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11117. WORDS

*BIN:CMUL,FIS,LST:CMUL,FIL=BIN:F,SRC:CMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11316. WORDS

*BIN:CDIV,FIS,LST:CDIV,FIL=BIN:F,SRC:CDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11073. WORDS

*BIN:DMUL,FIS,LST:DMUL,FIL=BIN:F,SRC:DMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11073. WORDS

*BIN:DADD,FIS,LST:DADD,FIL=BIN:F,SRC:DADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10973. WORDS

*BIN:DDIV,FIS,LST:DDIV,FIL=BIN:F,SRC:DDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11065. WORDS

*BIN:XDI,FIS,LST:XDI,FIL=BIN:F,SRC:XDI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11121. WORDS

*BIN:CONV1,FIS,LST:CONV1,FIL=BIN:F,SRC:CONV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11373. WORDS

*BIN:CONV3,FIS,LST:CONV3,FIL=BIN:F,SRC:CONV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11349. WORDS

*BIN:CADD,FIS,LST:CADD,FIL=BIN:F,SRC:CADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11325. WORDS

*BIN:ALOG,FIS,LST:ALOG,FIL=BIN:F,SRC:ALOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11081. WORDS

*BIN:DEXP,FIS,LST:DEXP,FIL=BIN:F,SRC:DEXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11041. WORDS

*BIN:EXP,FIS,LST:EXP,FIL=BIN:F,SRC:EXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11101. WORDS

*BIN:DSQRT,FIS,LST:DSQRT,FIL=BIN:F,SRC:DSQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11145. WORDS

*BIN:SIN,FIS,LST:SIN,FIL=BIN:F,SRC:SIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11300. WORDS
```

```
*BIN:DSIN,FIS,LST:DSIN,FIL=BIN:F,SRC:DSIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11260. WORDS

*BIN:DATAN,FIS,LST:DATAN,FIL=BIN:F,SRC:DATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11260. WORDS

*BIN:AINT,FIS,LST:AINT,FIL=BIN:F,SRC:AINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11363. WORDS

*BIN:DINT,FIS,LST:DINT,FIL=BIN:F,SRC:DINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11363. WORDS

*BIN:ADDA,FIS,LST:ADDA,FIL=BIN:F,SRC:ADDA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11333. WORDS

*BIN:SQRT,FIS,LST:SQRT,FIL=BIN:F,SRC:SQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11145. WORDS

*BIN:FMUL,FIS,LST:FMUL,FIL=BIN:F,SRC:FMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*BIN:FDIV,FIS,LST:FDIV,FIL=BIN:F,SRC:FDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*BIN:ADDM,FIS,LST:ADDM,FIL=BIN:F,SRC:ADDM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11341. WORDS

*BIN:ADDP,FIS,LST:ADDP,FIL=BIN:F,SRC:ADDP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11341. WORDS

*BIN:DLOG,FIS,LST:DLOG,FIL=BIN:F,SRC:DLOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11073. WORDS

*BIN:FADD,FIS,LST:FADD,FIL=BIN:F,SRC:FADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11141. WORDS

*BIN:XFI,FIS,LST:XFI,FIL=BIN:F,SRC:XFI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11121. WORDS

*BIN:CONV4,FIS,LST:CONV4,FIL=BIN:F,SRC:CONV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11173. WORDS

*BIN:IMUL,FIS,LST:IMUL,FIL=BIN:F,SRC:IMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11161. WORDS
```

```
*BIN:IDIV,FIS,LST:IDIV,FIL=BIN:F,SRC:IDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11161. WORDS

*
$EOJ

TIME
23:34:36
```

5.10.4 EAE OTS Assembly

Below is an example of the assembly procedures for the hardware dependent modules on a machine with the EAE hardware option.

```
$JOB/TIME/RT11

$RT11

23:03:03

        TTYIO

R PIP
BIN:F,MAC=SRC:V2DEF,PRE,FINIT,EAE,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE/A

R MACRO
*BIN:OTI,EAE,LST:OTI,EAL=BIN:F,SRC:OTI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10285. WORDS

*BIN:ATAN,EAE,LST:ATAN,EAL=BIN:F,SRC:ATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11272. WORDS

*BIN:XII,EAE,LST:XII,EAL=BIN:F,SRC:XII/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:CONVF,EAE,LST:CONVF,EAL=BIN:F,SRC:CONVF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11032. WORDS

*BIN:CONV2,EAE,LST:CONV2,EAL=BIN:F,SRC:CONV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11117. WORDS

*BIN:CMUL,EAE,LST:CMUL,EAL=BIN:F,SRC:CMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11324. WORDS

*BIN:CDIV,EAE,LST:CDIV,EAL=BIN:F,SRC:CDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11081. WORDS

*BIN:DMUL,EAE,LST:DMUL,EAL=BIN:F,SRC:DMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11081. WORDS

*BIN:DADD,EAE,LST:DADD,EAL=BIN:F,SRC:DADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10981. WORDS
```

```
*BIN:DDIV,EAE,LST:DDIV,EAL=BIN:F,SRC:DDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11073. WORDS

*BIN:XDI,EAE,LST:XDI,EAL=BIN:F,SRC:XDI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:CONV1,EAE,LST:CONV1,EAL=BIN:F,SRC:CONV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11381. WORDS

*BIN:CONV3,EAE,LST:CONV3,EAL=BIN:F,SRC:CONV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11349. WORDS

*BIN:CADD,EAE,LST:CADD,EAL=BIN:F,SRC:CADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11333. WORDS

*BIN:ALOG,EAE,LST:ALOG,EAL=BIN:F,SRC:ALOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11089. WORDS

*BIN:DEXP,EAE,LST:DEXP,EAL=BIN:F,SRC:DEXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11049. WORDS

*BIN:EXP,EAE,LST:EXP,EAL=BIN:F,SRC:EXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11109. WORDS

*BIN:DSQRT,EAE,LST:DSQRT,EAL=BIN:F,SRC:DSQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11153. WORDS

*BIN:SIN,EAE,LST:SIN,EAL=BIN:F,SRC:SIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11308. WORDS

*BIN:DSIN,EAE,LST:DSIN,EAL=BIN:F,SRC:DSIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11268. WORDS

*BIN:DATAN,EAE,LST:DATAN,EAL=BIN:F,SRC:DATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11268. WORDS

*BIN:AINT,EAE,LST:AINT,EAL=BIN:F,SRC:AINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11363. WORDS

*BIN:DINT,EAE,LST:DINT,EAL=BIN:F,SRC:DINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11363. WORDS

*BIN:ADDA,EAE,LST:ADDA,EAL=BIN:F,SRC:ADDA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11343. WORDS

*BIN:SQRT,EAE,LST:SQRT,EAL=BIN:F,SRC:SQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS
```

```
*BIN:FMUL,EAE,LST:FMUL,EAL=BIN:F,SRC:FMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11077. WORDS

*BIN:FDIV,EAE,LST:FDIV,EAL=BIN:F,SRC:FDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11089. WORDS

*BIN:ADDM,EAE,LST:ADDM,EAL=BIN:F,SRC:ADDM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11343. WORDS

*BIN:ADDP,EAE,LST:ADDP,EAL=BIN:F,SRC:ADDP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11343. WORDS

*BIN:DLOG,EAE,LST:DLOG,EAL=BIN:F,SRC:DLOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11081. WORDS

*BIN:FADD,EAE,LST:FADD,EAL=BIN:F,SRC:FADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11053. WORDS

*BIN:XFI,EAE,LST:XFI,EAL=BIN:F,SRC:XFI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:CONV4,EAE,LST:CONV4,EAL=BIN:F,SRC:CONV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11181. WORDS

*BIN:IMUL,EAE,LST:IMUL,EAL=BIN:F,SRC:IMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11157. WORDS

*BIN:IDIV,EAE,LST:IDIV,EAL=BIN:F,SRC:IDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*
$EOJ

TIME
23:12:15
```

5.10.5 FPU OTS Assembly

Below is an example of the assembly procedures for the hardware dependent modules on a machine with the FPU hardware option.

```
$JOB/TIME/RT11

$RT11

23:34:49

        TTYIO

R PIP
BIN:F,MAC=SRC:V2DEF,PRE,FINIT,FPU,OTSWA,PRE,FBLOCK,PRE,ERRORS,PRE/A
```

```
R MACRO
*BIN:OTI,FPU,LST:OTI,FPL=BIN:F,SRC:OTI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 10237. WORDS

*BIN:ATAN,FPU,LST:ATAN,FPL=BIN:F,SRC:ATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11281. WORDS

*BIN:XII,FPU,LST:XII,FPL=BIN:F,SRC:XII/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11133. WORDS

*BIN:CONVF,FPU,LST:CONVF,FPL=BIN:F,SRC:CONVF/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11028. WORDS

*BIN:CONV2,FPU,LST:CONV2,FPL=BIN:F,SRC:CONV2/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11134. WORDS

*BIN:CMUL,FPU,LST:CMUL,FPL=BIN:F,SRC:CMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11325. WORDS

*BIN:CDIV,FPU,LST:CDIV,FPL=BIN:F,SRC:CDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11101. WORDS

*BIN:DMUL,FPU,LST:DMUL,FPL=BIN:F,SRC:DMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:DADD,FPU,LST:DADD,FPL=BIN:F,SRC:DADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11069. WORDS

*BIN:DDIV,FPU,LST:DDIV,FPL=BIN:F,SRC:DDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:XDI,FPU,LST:XDI,FPL=BIN:F,SRC:XDI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11137. WORDS

*BIN:CONV1,FPU,LST:CONV1,FPL=BIN:F,SRC:CONV1/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11374. WORDS

*BIN:CONV3,FPU,LST:CONV3,FPL=BIN:F,SRC:CONV3/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11357. WORDS

*BIN:CADD,FPU,LST:CADD,FPL=BIN:F,SRC:CADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11326. WORDS

*BIN:ALOG,FPU,LST:ALOG,FPL=BIN:F,SRC:ALOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:DEXP,FPU,LST:DEXP,FPL=BIN:F,SRC:DEXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11101. WORDS
```

```
*BIN:EXP,FPU,LST:EXP,FPL=BIN:F,SRC:EXP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11121. WORDS

*BIN:DSQRT,FPU,LST:DSQRT,FPL=BIN:F,SRC:DSQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:SIN,FPU,LST:SIN,FPL=BIN:F,SRC:SIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11325. WORDS

*BIN:DSIN,FPU,LST:DSIN,FPL=BIN:F,SRC:DSIN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11325. WORDS

*BIN:DATAN,FPU,LST:DATAN,FPL=BIN:F,SRC:DATAN/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11281. WORDS

*BIN:AINT,FPU,LST:AINT,FPL=BIN:F,SRC:AINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11365. WORDS

*BIN:DINT,FPU,LST:DINT,FPL=BIN:F,SRC:DINT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11362. WORDS

*BIN:ADDA,FPU,LST:ADDA,FPL=BIN:F,SRC:ADDA/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11349. WORDS

*BIN:SQRT,FPU,LST:SQRT,FPL=BIN:F,SRC:SQRT/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11149. WORDS

*BIN:FMUL,FPU,LST:FMUL,FPL=BIN:F,SRC:FMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11165. WORDS

*BIN:FDIV,FPU,LST:FDIV,FPL=BIN:F,SRC:FDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11161. WORDS

*BIN:ADDM,FPU,LST:ADDM,FPL=BIN:F,SRC:ADDM/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11349. WORDS

*BIN:ADDP,FPU,LST:ADDP,FPL=BIN:F,SRC:ADDP/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11349. WORDS

*BIN:DLOG,FPU,LST:DLOG,FPL=BIN:F,SRC:DLOG/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11129. WORDS

*BIN:FADD,FPU,LST:FADD,FPL=BIN:F,SRC:FADD/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11141. WORDS

*BIN:XFI,FPU,LST:XFI,FPL=BIN:F,SRC:XFI/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11137. WORDS
```

```
*BIN:CONV4,FPU,LST:CONV4,FPL=BIN:F,SRC:CONV4/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11169. WORDS

*BIN:IMUL,FPU,LST:IMUL,FPL=BIN:F,SRC:IMUL/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11157. WORDS

*BIN:IDIV,FPU,LST:IDIV,FPL=BIN:F,SRC:IDIV/C/N:TTM:CND
ERRORS DETECTED: 0
FREE CORE: 11157. WORDS

*
$EOJ

TIME
23:43:54
```

5.10.6 OTS Library Component Building

This section contains the final OTS library preparation. The FORTRAN object modules have been concatenated, using PIP, into larger common object modules to simplify the library build process.

```
$JOB/RT11/TIME  CONCATENATE F4 OTS OBJ'S INTO LARGER FILES

$RT11

00:40:48

        TTYIO

R PIP
BIN:OTSCOM,OBJ=BIN:*,ALL/B
BIN:FPU,OBJ=BIN:*,FPU/B
BIN:FIS,OBJ=BIN:*,FIS/B
BIN:EAE,OBJ=BIN:*,EAE/B
BIN:EIS,OBJ=BIN:*,EIS/B
BIN:NHD,OBJ=BIN:*,NHD/B
BIN:UNI,OBJ=BIN:*,UNI,*,SIZ/B
BIN:V2S,OBJ=BIN:SHORT,E2S,*,V2S/B
BIN:V2NS,OBJ=BIN:SHORT,E2N,*,V2N/B

$EOJ

TIME
00:43:12
```

Refer to the appropriate section of Chapter 2 for specific library build procedures.

Section 2.2.2	building from DECpack
Section 2.3.2	building from DECtape
Section 2.4.2	building from diskette
Section 2.5.2	building from magtape
Section 2.6.2	building from cassette
Section 2.7.2	building from paper tape

5.11 ASSEMBLING BASIC/RT-11

This section provides assembly instructions for BASIC/RT-11; this information applies only to those users who received the source versions of BASIC/RT-11. A 16K system is required to assemble BASIC. The source program of BASIC/RT-11 consists of three source files:

```
BASICL.MAC
BASICH.MAC
FPMP.MAC
```

It is necessary to create the .MAC files BASICR, BASICE, and BASICX which consist of only one line of code each. They specify the conditionals necessary to assemble BASICL into three object modules: BASICR.OBJ, BASICE.OBJ, and BASICX.OBJ.

1. Create the object modules using EDIT. To the running RT-11 monitor,

```
Type:       R EDIT<CR>
Response:   *

Type:       EWBASICR,MAC<ALT><ALT>
Response:   *

Type:       IBASICR=1<CR>
            <ALT>EX<ALT><ALT>
Response:   .

Type:       R EDIT<CR>
Response:   *

Type:       EWBASICE,MAC<ALT><ALT>
Response:   *

Type:       IBASICE=1<CR>
            <ALT>EX<ALT><ALT>
Response:   .

Type:       R EDIT<CR>
Response:   *

Type:       EWBASICX,MAC<ALT><ALT>
Response:   *

Type:       IBASICX=1<CR>
            <ALT>EX<ALT><ALT>
Response:   .
```

2. If any other options are desired, include the conditionals for them in these files. For example:

```
$NOSTR=1        ;NO STRINGS
$LONGER=1       ;LONGER ERROR MESSAGES
$NOVF=1         ;NO VIRTUAL MEMORY FILES
$NOPOW=1        ;NO POWER-FAIL OPTION
$STKSZ=n        ;STACK SIZE IN BYTES (DEFAULT IS 200
                 (OCTAL) BYTES)
```

NOTE

BASIC uses part of the user area for its working stack. See BASIC/RT-11 Release Notes, Section 3.9, for further information.

If BASIC is to run on an 8K system, the $NOSTR conditional must be specified.

For example, to create a BASIC with no strings, no virtual memory files, and a stack size of 300 (octal) bytes, create the BASICR, BASICE, and BASICX files using the EDIT program, as follows:

```
Type:       R EDIT<CR>
Response:   *

Type:       EWBASICR,MAC<ALT><ALT>
Response:   *

Type:       IBASICR=1<CR>
            $NOSTR=1<CR>
            $NOVF=1<CR>
            $STKSZ=300<CR>
            <ALT>EX<ALT><ALT>
Response:   '

Type:       R EDIT<CR>
Response:   *

Type:       EWBASICE,MAC<ALT><ALT>
Response:   *

Type:       IBASICE=1<CR>
            $NOSTR=1<CR>
            $NOVF=1<CR>
            $STKSZ=300<CR>
            <ALT>EX<ALT><ALT>
Response:   '

Type:       R EDIT<CR>
Response:   *

Type:       EWBASICX,MAC<ALT><ALT>
Response:   *

Type:       IBASICX=1<CR>
            $NOSTR=1<CR>
            $NOVF=1<CR>
            $STKSZ=300<CR>
            <ALT>EX<ALT><ALT>
Response:   '
```

3. To assemble BASIC,

```
Type:       R MACRO<CR>
Response:   *

Type:       BASICR=BASICR,BASICL<CR>
Response:   *

Type:       BASICE=BASICE,BASICL<CR>
Response:   *

Type:       BASICX=BASICX,BASICL<CR>
Response:   *
```

```
Type:       BASICH=BASICH<CR>
Response:   *

Type:       FPMP=FPMP<CR>
Response:   *

Type:       CTRL C
            ^C
            .
```

The preceding instructions produce five object modules:

BASICR.OBJ	BASIC Root section
BASICE.OBJ	BASIC Edit overlay
BASICX.OBJ	BASIC EXecution overlay
FPMP.OBJ	Floating Point Math Package
BASICH.OBJ	BASIC High section, with once-only code and optional functions

5.11.1 Floating Point Math Package

Assembly of the FPMP source file produces a standard FPMP for BASIC. This standard FPMP runs on any PDP-11 but does not make use of special arithmetic hardware; it includes all of the routines needed for the full complement of BASIC arithmetic functions. A non-standard FPMP may be specified, as outlined in Table 5-1.

1. To assemble the Floating Point Math Package with conditionals, it is necessary to use the EDIT program to either insert the conditionals in the beginning of the FPMP.MAC file or create a new file, FPMPC.MAC which will be assembled with FPMP.MAC. For example, to create the FPMP with the SIN, COS, and SQR functions and EAE hardware but without the ATN function, create the file FPMPC.MAC with EDIT as follows:

```
Type:       R EDIT<CR>
Response:   *

Type:       EWFPMPC.MAC<ALT><ALT>
Response:   *

Type:       IMIN=1<CR>
            EAE=1<CR>
            CND$37=1<CR>
            CND$41=1<CR>
            <ALT>EX<ALT><ALT>
Response:   .
```

2. Assemble BASIC with MACRO as follows:

```
Type:       R MACRO<CR>
Response:   *

Type:       BASICR=BASICR,BASICL<CR>
Response:   *
```

Table 5-1
FPMP Assembly Parameters

Parameter	Default Value	Description
MIN	undefined	Define to eliminate code for BASIC functions SIN, COS, ATN, and SQR. When linked, the functions are listed as undefined references. However, when executed by a BASIC program, they produce a ?UFN (UNDEFINED FUNCTION) error.
FPU	undefined	Define to assemble a version for the PDP-11/45 FPU hardware.
EAE	undefined	Define to assemble for the EAE hardware.
MULDIV	undefined	Define to assemble for the PDP-11/40 extended instruction set (EIS) or the PDP-11/45 processor.
If MIN is defined, then the following parameters may be specified to include the SIN, COS, ATN, and SQR functions, selectively.		
CND$37	1	Define (only if MIN is specified) to include the code for the SIN and COS functions.
CND$39	1	Define (only if MIN is specified) to include the code for the ATN function.
CND$41	1	Define (only if MIN is specified) to include the code for the SQR function.

```
Type:       BASICE=BASICE,BASICL<CR>
Response:   *

Type:       BASICX=BASICX,BASICL<CR>
Response:   *

Type:       FPMP=FPMPC,FPMP<CR>
response:   *

Type:       CTRL C
Response:   ^C
            .
```

5.12 LINKING BASIC/RT-11

The five object modules (BASICR, BASICE, BASICX, FPMP, BASICH) may be linked with or without an overlay structure. The overlay option has the advantage that sections of BASIC that are not required at the same time occupy the same core space alternately when they are used; the

disadvantage is that BASIC will run somewhat slower and I/O time will be spent when switching overlay segments in and out of core. When BASIC is linked to run in an 8K system, it must use the overlay option and the $NOSTR conditional must be used during assembly.

1. To link BASIC without overlays,

```
Type:       R LINK<CR>
Response:   *

Type:       BASIC,BASIC=BASICR,FPMP,BASICE,BASICX,BASICH/B:400<CR>
Response:   *
```

NOTE

The above command string is for a configuration that has no devices with vectors above 400. User's with 8K whose configurations have interrupt vectors above 400 should relink BASIC with a bottom address of 500. See BASIC/RT-11 Release Notes, Section 3.6, for further information.

2. To link BASIC with overlays,

```
Type:       R LINK<CR>
Response:   *

Type:       BAS8K,BAS8K=BASICR,FPMP/T/B:400/C<CR>
Response:   TRANSFER ADDRESS =
```

NOTE

The above command string is for a configuration that has no devices with vectors above 400. User's with 8K whose configurations have interrupt vectors above 400 should relink BASIC with a bottom address of 500. See BASIC/RT-11 Release Notes, Section 3.6, for further information.

```
Type:       GO<CR>
Response:   *

Type:       BASICE/O11/C<CR>
Response:   *

Type:       BASICX/O11/C<CR>
Response:   *

Type:       BASICH/O12<CR>
Response:   *
```

5.12.1 Linking BASIC/RT-11 with User Functions

The System Function Table address used by the CALL statement to link the user's assembly language routines must be set in the first word of the BASICR control section.

The source code for the System Function Table and the actual function routines must be broken into two separate source files. The source file FUN1 consists of the System Function Table definition, with this general outline:

```
          Function Entry Points

          .GLOBL  FN1, FN2
          .CSECT  BASICR
          .WORD   FUNTAB


          .CSECT  FUN1
FUNTAB:   (function table entries for FN1,FN2,...)
```

The source file FUN2 consists of the code for the function routines, with the following general outline:

```
          .GLOBL  FN1,FN2,...
          .CSECT  FUN2

FN1:
               (the user function routines)
FN2:
```

1. To link BASIC with the user functions in a non-overlay system, type the following to the running RT-11 monitor:

```
Type:      R LINK<CR>
Response:  *

Type:      BASIC=BASICR,FPMP,BASICE,BASICX/B:500/C<CR>
Response:  *

Type:      FUN1,FUN2E,GETARG],BASICH<CR>
Response:  *
```

   GETARG is the general argument interface module listed in Appendix H of the BASIC/RT-11 Language Reference Manual.

2. In an overlay system, there are two possible ways to link BASIC with the user functions.

   If the user function routines contain no data that must be preserved from one function call to the next, that is, if the code for the routines may be refreshed at the beginning of each function call, then the routines may be incorporated into the execution overlay by using the following commands to the Linker:

```
Type:      BASIC,BASIC=BASICR,FPMP,FUN1/T/B:400/C<CR>
Response:  TRANSFER ADDRESS =
```

NOTE

The above command string is for a configuration that has no devices with vectors above 400. User's with 8K whose configurations have interrupt vectors above 400 should relink BASIC with a bottom address of 500. See BASIC/RT-11 Release Notes, Section 3.6, for further information.

```
Type:       GO<CR>
Response:   *

Type:       BASICE/O11/C<CR>
Response:   *

Type:       BASICX,FUN2[,GETARG]/O11/C<CR>
Response:   *

Type:       BASICH/O12<CR>
```

In this case, the function routines (in the module FUN2) occupy space in the first overlay segment that is normally unused, since the Edit overlay segment (BASICE) is about 250 words longer in the 8K no-string system than the Execution overlay segment (BASICX). These first 250 words of storage are free in this case.

3. If FUN2 may not be read in anew whenever it is used, enter the following to the Linker:

```
Type:       BASIC=BASICR,FPMP,FUN1,FUN2/T/B:1400/C<CR>
Response:   TRANSFER ADDRESS =
```

NOTE

The above command string is for a configuration that has no devices with vectors above 400. User's with 8K whose configurations have interrupt vectors above 400 should relink BASIC with a bottom address of 500. See BASIC/RT-11 Release Notes, Section 3.6, for further information.

```
Type:       GO<CR>
Response:   *

Type:       BASICE/O11/C<CR>
Response:   *

Type:       BASICX[,GETARG]/O11/C<CR>
Response:   *

Type:       BASICH/O12<CR>
```

Three additional object modules (FPMP.FPU, FPMP.EAE, FPMP.EIS) allow BASIC/RT-11 to be linked for special arithmetic hardware.

Optional Hardware	Replace FPMP.OBJ with
EAE hardware	FPMP.EAE
PDP-11/40 extended processor or PDP-11/45 processor	FPMP.EIS
PDP-11/45 FPU hardware	FPMP.FPU

APPENDIX A

INSTRUCTIONS FOR LOADING SOFTWARE BOOTSTRAPS

A.1 RK11 DECPACK BOOTSTRAP LOADER

1. Deposit the basic RK11 disk bootstrap loader into memory as follows:

   a. Set the ENABLE/HALT switch to HALT, then set the first address, 001000, in the Switch Register. (Set switch 9 to the up (1) position and all others to the down (0) position.)

   b. Press the LOAD ADDR switch.

   c. Set the proper contents for Table A-1 in the Switch Register and lift the DEP switch.

   d. Repeat Step c until all the instructions have been deposited.

2. Verify that the bootstrap program has been deposited properly as follows:

   a. Set the starting address in the Switch Register as in Step 1a above.

   b. Press the LOAD ADDR switch.

   c. Display the contents of that address in the Data Register by pressing the EXAM switch.

   d. Compare the number in the Data Register with the value in Table A-1.

   e. If they are the same, repeat Step 2c until all words have been examined.

   f. If not the same, repeat Step 1.

3. Set the starting address, 001000, in the Switch Register as in Step 1a above, then press the LOAD ADDR switch.

4. Set the ENABLE/HALT switch to ENABLE, then press the START switch.

Table A-1
RK11 Bootstrap Loader

Location	Contents	Like This
001000	012700	Set switches 12, 10, 8, 7, and 6 to the up (1) position. Set all others to the down (0) position.
001002	177406	Set switches 15 through 8, 2, and 1 to the up (1) position. Set all others to the down (0) position.
001004	012710	Set switches 12, 10, 8, through 6, and 3 to the up (1) position. Set all others to the down (0) position.
001006	177400	Set switches 15 through 8 to the up (1) position. Set all others to the down (0) position.
001010	012740	Set switches 12, 10, and 8 through 5 to the up (1) position. Set all others to the down (0) position.
001012	000005	Set switches 2 and 0 to the up (1) position. Set all others to the down (0) position.
001014	105710	Set switches 15, 11, 9 through 6, and 3 to the up (1) position. Set all others to the down (0) position.
001016	100376	Set switches 15 and 7 through 1 to the up (1) position. Set all others to the down (0) position.
001020	005007	Set switches 11, 9, and 2 through 0 to the up (1) position. Set all others to the down (0) position.

A.2 TC11 DECTAPE BOOTSTRAP LOADER

1. Deposit the basic DECtape bootstrap loader into memory as follows:

   a. Set the ENABLE/HALT switch to HALT, then set the first address, 001000, in the Switch Register. (Set switch 9 to the up (1) position and all others to the down (0) position.

   b. Press the LOAD ADDR switch.

   c. Set the proper contents from Table A-2 in the Switch

Register and lift the DEP switch.

d. Repeat Step c until all the instructions have been deposited.

Table A-2
TC11 Bootstrap Loader

Location	Contents	Like This
001000	012700	Set switches 12, 10, 8, 7, and 6 to the up (1) position. Set all others to the down (0) position.
001002	177344	Set switches 15 through 9, 7, 6, 5, and 2 to the up (1) position. Set all others to the down (0) position.
001004	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001006	177400	Set switches 15 through 8 to the up (1) position. Set all others to the down (0) position.
001010	012740	Set switches 12, 10, 8, 7, 6, and 5 to the up (1) position. Set all others to the down (0) position.
001012	004002	Set switches 11 and 1 to the up (1) position. Set all others to the down (0) position.
001014	005710	Set switches 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001016	100376	Set switches 15, and 7 through 1 to the up (1) position. Set all others to the down (0) position.
001020	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001022	000003	Set switches 1 and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-2 (Cont.)
TC11 Bootstrap Loader

Location	Contents	Like This
001024	105710	Set switches 15, 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001026	100376	Set switches 15, and 7 through 1 to the up (1) position. Set all others to the down (0) position.
001030	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001032	000005	Set switches 2 and 0 to the up (1) position. Set all others to the down (0) position.
001034	105710	Set switches 15, 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001036	100376	Set switches 15 and 7 through 1 to the up (1) position. Set all others to the down (0) position.
001040	005007	Set switches 11, 9, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.

2. Verify that the bootstrap program has been deposited properly as follows:

   a. Set the starting address in the Switch Register as in Step 1a above.

   b. Press the LOAD ADDR switch.

   c. Display the contents of that address in the Data Register by pressing the EXAM switch.

   d. Compare the number in the Data Register with the value in Table A-2.

   e. If they are the same, repeat Step 2c until all words have been examined.

   f. If not the same, repeat Step 1.

3. Set the starting address 001000 in the Switch Register as in Step 1a above, then press the LOAD ADDR switch.

4. Set the ENABLE/HALT switch to ENABLE, then press the START switch.

A.3 RX11/RX01 DISKETTE BOOTSTRAP LOADER

1. If the computer is a PDP-11V/03, perform the following; if the computer is a PDP-11/03 or LSI-11, see the LSI-11, PDP-11/03 User's Manual (EK-LSI11-TM-002); otherwise, go to Step 2.

   a. Put all three switches in the up position.

   b. Move the DC ON/OFF switch down and up.

   c. Response: $
      Type: DX<CR>

   The bootstrapping procedure is complete.

2. Deposit the basic diskette bootstrap loader into memory as follows:

   a. Set the ENABLE/HALT switch to HALT, then set the first address, 001000, in the Switch Register. (Set switch 9 to the up (1) position and all others to the down (0) position.)

   b. Press the LOAD ADDR switch.

   c. Set the proper contents from Table A-3 in the Switch Register and lift the DEP switch.

   d. Repeat Step 2c until all the instructions have been deposited.

Table A-3
RX11 Bootstrap Loader

Location	Contents	Like This
001000	012702	Set switches 12, 10, 8, 7, 6, and 1 to the up (1) position. Set all others to the down (0) position.
001002	1002n7 (n=4 for unit 0 and n=6 for unit 1)	Set switches 15, 7, 2, 1, and 0 to the up (1) position and all others to the down (0) position. For unit 0, set switch 5 to the up (1) position; for unit 1, set switches 5 and 4 to the up (1) position.
001004	012701	Set switches 12, 10, 8, 7, 6, and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-3 (Cont.)
RX11 Bootstrap Loader

Location	Contents	Like This
001006	177170	Set switches 15 through 9, and 6 through 3 to the up (1) position. Set all others to the down (0) position.
001010	130211	Set switches 15, 13, 12, 7, 3, and 0 to the up (1) position. Set all others to the down (0) position.
001012	001776	Set switches 9 through 1 to the up (1) position. Set all others to the down (0) position.
001014	112703	Set switches 15, 12, 10, 8, 7, 6, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001016	000007	Set switches 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001020	010100	Set switches 12 and 6 to the up (1) position. Set all others to the down (1) position.
001022	010220	Set switches 12, 7, and 4 to the up (1) position. Set all others to the down (0) position.
001024	000402	Set switches 8 and 1 to the up (1) position. Set all others to the down (0) position.
001026	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001030	000001	Set switch 0 to the up (1) position. Set all others to the down (0) position.
001032	006203	Set switches 11, 10, 7, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001034	103402	Set switches 15, 10, 9, 8, and 1 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-3 (Cont.)
RX11 Bootstrap Loader

Location	Contents	Like This
001036	112711	Set switches 15, 12, 10 , 8, 7, 6, 3, and 0 to the up (1) position. Set all others to the down (0) position.
001040	111023	Set switches 15, 12, 9, 4, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001042	030211	Set switches 13, 12, 7, 3, and 0 to the up (1) position. Set all others to the down (0) position.
001044	001776	Set switches 9 through 1 to the up (1) position. Set all others to the down (0) position.
001046	100756	Set switches 15, 8, 7, 6, 5, and 3, 2, and 1 to the up (1) position. Set all others to the down (0) position.
001050	103766	Set switches 15, 10 through 4, 2, and 1 to the up (1) position. Set all others to the down (0) position.
001052	105711	Set switches 15, 11, 9 through 6, 3, and 0 to the up position. Set all others to the down (0) position.
001054	100771	Set switches 15, 8 through 3, and 0 to the up (1) position. Set all others to the down (0) position.
001056	005000	Set switches 11 and 9 to the up (1) position. Set all others to the down (0) position.
001060	022710	Set switches 13, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) postion.
001062	000240	Set switches 7 and 5 to the up (1) position. Set all others to the down (0) position.
001064	001347	Set switches 9, 7, 6, 5, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-3 (Cont.)
RX11 Bootstrap Loader

Location	Contents	Like This
001066	122702	Set switches 15, 13, 10, 8, 7, 6, and 1 to the up (1) position. Set all others to the down (0) position.
001070	000247	Set switches 7, 5, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001072	005500	Set switches 11, 9, 8, and 6 to the up (1) position. Set all others to the down (0) position.
001074	005007	Set switches 11, 9, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.

3. Verify that the bootstrap program has been deposited properly as follows:

   a. Set the starting address in the Switch Register as in Step 2a above.

   b. Press the LOAD ADDR switch.

   c. Display the contents of that address in the Data Register by pressing the EXAM switch.

   d. Compare the number in the Data Register with the value in Table A-3.

   e. If they are the same, repeat Step 3c until all words have been examined.

   f. If not the same, repeat Step 2.

3. Set the starting address 001000 in the Switch Register as in Step 2a above, then press the LOAD ADDR switch.

4. Set the ENABLE/HALT switch to ENABLE, then press the START switch.

A.4 MAGTAPE BOOTSTRAP LOADERS

1. Deposit the basic magtape bootstrap into memory as follows:

   a. Set the ENABLE/HALT switch to HALT, then set the first address, 010000, in the Switch Register. (Set switch 12 to the up (1) position and all others to the down (0) position.)

b. Press the LOAD ADDR switch.

c. Set the proper contents for TJU16 magtape (Table A-4) or for TM11 magtape (Table A-5) in the Switch Register and lift the DEP switch.

d. Repeat Step c until all the instructions have been deposited.

Table A-4
TJU16 Bootstrap Loader

Location	Contents	Like This
010000	012700	Set switches 12, 10, 8, 7, and 6 to the up (1) position. Set all others to the down (0) position.
010002	172440	Set switches 15 through 12, 10, 8, and 5 to the up (1) position. Set all others to the down (0) position.
010004	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010006	000021	Set switches 4 and 0 to the up (1) position. Set all others to the down (0) position.
010010	012760	Set switches 12, 10, and 8 through 4 to the up (1) position. Set all others to the down (0) position.
010012	001300	Set switches 9, 7, and 6 to the up (1) position. Set all others to the down (0) position.
010014	000032	Set switches 4, 3, and 1 to the up (1) position. Set all others to the down (0) position.
010016	012760	Set switches 12, 10, and 8 through 4 to the up (1) position. Set all others to the down (0) position.
010020	177777	Set all switches to the up (1) position.
010022	000006	Set switches 2 and 1 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-4 (Cont.)
TJU16 Bootstrap Loader

Location	Contents	Like This
010024	012720	Set switches 12, 10, 8, 7, 6, and 4 to the up (1) position. Set all others to the down (0) position.
010026	000031	Set switches 4, 3, and 0 to the up (1) position. Set all others to the down (0) position.
010030	105760	Set switches 15, 11, and 9 through 4 to the up (1) position. Set all others to the down (0) position.
010032	000010	Set switch 3 to the up (1) position. Set all others to the down (0) position.
010034	100375	Set switches 15, 7 through 2, and 0 to the up (1) position. Set all others to the down (0) position.
010036	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010040	177000	Set switches 15 through 9 to the up (1) position. Set all others to the down (0) position.
010042	012740	Set switches 12, 10, and 8 through 5 to the up (1) position. Set all others to the down (0) position.
010044	000071	Set switches 5, 4, 3, and 0 to the up (1) position. Set all others to the down (0) position.
010046	032710	Set switches 13, 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010050	100200	Set switches 15 and 7 to the up (1) position. Set all others to the down (0) position.
010052	001775	Set switches 9 through 2 and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-4 (Cont.)
TJU16 Bootstrap Loader

Location	Contents	Like This
010054	100007	Set switches 15, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.
010056	022760	Set switches 13, 10, and 8 through 4 to the up (1) position. Set all others to the down (0) position.
010060	001000	Set switch 9 to the up (1) position. Set all others to the down (0) position.
010062	000014	Set switches 3 and 2 to the up (1) position. Set all others to the down (0) position.
010064	001403	Set switches 9, 8, 1, and 0 to the up (1) position. Set all others to the down (0) position.
010066	000005	Set switches 2 and 0 to the up (1) position. Set all others to the down (0) position.
010070	000167	Set switches 6, 5, 4, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.
010072	177704	Set switches 15 through 6 and 2 to the up (1) position. Set all others to the down (0) position.
010074	005007	Set switches 11, 9, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.

Table A-5
TM11 Bootstrap Loader

Location	Contents	Like This
010000	012700	Set switches 12, 10, 8, 7, and 6 to the up (1) position. Set all others to the down (0) position.
010002	172524	Set switches 15 through 12, 10, 8, 6, 4, and 2 to the up (1) position. Set all others to the down (0) position.
010004	005310	Set switches 11, 9, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010006	012740	Set switches 12, 10, and 8 through 5 to the up (1) position. Set all others to the down (0) position.
010010	060011	Set switches 14, 13, 3, and 0 to the up (1) position. Set all others to the down (0) position.
010012	105710	Set switches 15, 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010014	100376	Set switches 15 and 7 through 1 to the up (1) position. Set all others to the down (0) position.
010016	005710	Set switches 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010020	100767	Set switches 15, 8 through 4, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.
010022	012710	Set switches 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010024	060003	Set switches 14, 13, 1, and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-5 (Cont.)
TM11 Bootstrap Loader

Location	Contents	Like This
010026	105710	Set switches 15, 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010030	100376	Set switches 15 and 7 through 1 to the up (1) position. Set all others to the down (0) position.
010032	005710	Set switches 11, 9, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
010034	100777	Set switches 15 and 8 through 0 to the up (1) position. Set all others to the down (0) position.
010036	005007	Set switches 11, 9, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.

2. Verify that the bootstrap program has been deposited properly as follows:

   a. Set the starting address, 010000, in the Switch Register as in Step 1a above.

   b. Press the LOAD ADDR switch.

   c. Display the contents of that address in the Data Register by pressing the EXAM switch.

   d. Compare the number in the Data Register with the value in the appropriate table (A-4 or A-5).

   e. If they are the same, repeat Step 2c until all words have been examined.

   f. If not the same, repeat Step 1.

3. If TM11 magtape is being used, ensure that the magtape is positioned at the load point; if it is not, manually rewind the magtape.

4. Set the starting address, 010000, in the Switch Register as in Step 1a above, then press the LOAD ADDR switch.

5. Set the ENABLE/HALT switch to ENABLE, then press the START switch.

A.5 TA11 CASSETTE BOOTSTRAP LOADER

Two cassette bootstraps are available: CBOOT (Table A-6) is the standard version, QCBOOT (Table A-7) is a shorter version that may optionally be loaded. QCBOOT does not provide some of the error checking and handling that the longer CBOOT does, but it allows a faster means of manually booting the system.

1. Deposit the cassette software bootstrap into memory as follows:

   a. Set the first address, 001000, in the Switch Register. (Set switch 9 to the up (1) position and all others to the down (0) position.)

   b. Press the LOAD ADDR switch.

   c. Set the proper contents for CBOOT (Table A-6) or QCBOOT (Table A-7) in the Switch Register and lift the DEP switch.

   d. Repeat Step c until all values have been deposited.

Table A-6
TA11 CBOOT Bootstrap Loader

Location	Contents	Like This
001000	012700	Set switches 12, 10, 8, 7, and 6 to the up (1) position. Set all others to the down (0) position.
001002	177500	Set switches 15 through 8, and 6 to the up (1) position. Set all others to the down (0) position.
001004	005010	Set switches 11, 9, and 3 to the up (1) position. Set all others to the down (0) position.
001006	010701	Set switches 12, 8, 7, 6, and 0 to the up (1) position. Set all others to the down (0) position.
001010	062701	Set switches 14, 13, 10, 8, 7, 6, and 0 to the up (1) position. Set all others to the down (0) position.
001012	000052	Set switches 5, 3, and 1 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-6 (Cont.)
TA11 CBOOT Bootstrap Loader

Location	Contents	Like This
001014	012702	Set switches 12, 10, 8, 7, 6, and 1 to the up (1) position. Set all others to the down (0) position.
001016	000375	Set switches 7 through 2 and 0 to the up position. Set all others to the down (0) position.
001020	112103	Set switches 15, 12, 10, 6, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001022	112110	Set switches 15, 12, 10, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001024	100413	Set switches 15, 8, 3, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001026	130310	Set switches 15, 13, 12, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001030	001776	Set switches 9 through 1 in the up (1) position. Set all others in the down (0) position.
001032	105202	Set switches 15, 11, 9, 7, and 1 to the up (1) position. Set all others to the down (0) position.
001034	100772	Set switches 15, 8 through 3, and 1 to the up (1) position. Set all others to the down (0) position.
001036	116012	Set switches 15, 12, 11, 10, 3, and 1 to the up (1) position. Set all others to the down (0) position.
001040	000002	Set switch 1 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-6 (Cont.)
TA11 CBOOT Bootstrap Loader

Location	Contents	Like This
001042	120337	Set switches 15, 13, 7, 6, and 4 through 0 to the up (1) position. Set all others to the down (0) position.
001044	000000	Set all switches to the down (0) position.
001046	001767	Set switches 9 through 4, and 2 through 0 to the up (1) position. Set all others to the down (0) position.
001050	000000	Set all switches to the down (0) position.
001052	000755	Set switches 8 through 5, 3, 2, and 0 to the up (1) position. Set all others to the down (0) position.
001054	005710	Set switches 11, 9 through 6, and 3 to the up (1) position. Set all others to the down (0) position.
001056	100774	Set switches 15, and 8 through 2 to the up (1) position. Set all others to the down (0) position.
001060	005007	Set switches 11, 9, and 2 through 0 to the up (1) position. Set all others to the down (0) position.
001062	017640	Set switches 12 through 7 and 5 to the up (1) position. Set all others to the down (0) position.
001064	002415	Set switches 10, 8, 3, 2, and 0 to the up (1) position. Set all others to the down (0) position.
001066	112024	Set switches 15, 12, 10, 4, and 2 to the up (1) position. Set all others to the down (0) position.

Table A-7
TA11 QCBOOT Bootstrap Loader

Location	Contents	Like This
001000	012700	Set switches 12, 10, 8, 7, and 6 to the up (1) position. Set all others to the down (0) position.
001002	177500	Set switches 15 through 8, and 6 to the up (1) position. Set all others to the down (0) position.
001004	005010	Set switches 11, 9, and 3 to the up (1) position. Set all others to the down (0) position.
001006	010701	Set switches 12, 8, 7, 6, and 0 to the up (1) position. Set all others to the down (0) position.
001010	062701	Set switches 14, 13, 10, 8, 7, 6, and 0 to the up (1) position. Set all others to the down (0) position.
001012	000034	Set switches 4, 3, and 2 to the up (1) position. Set all others to the down (0) position.
001014	112102	Set switches 15, 12, 10, 6, and 1 to the up (1) position. Set all others to the down (0) position.
001016	112110	Set switches 15, 12, 10, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001020	032710	Set switches 13, 12, 10, 8, 7, 6, and 3 to the up (1) position. Set all others to the down (0) position.
001022	100240	Set switches 15, 7, and 5 to the up (1) position. Set all others to the down (0) position.
001024	001775	Set switches 9 through 2 and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-7 (Cont.)
TA11 QCBOOT Bootstrap Loader

Location	Contents	Like This
001026	100001	Set switches 15 and 0 to the up (1) position. Set all others to the down (0) position.
001030	005007	Set switches 11, 9, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001032	005202	Set switches 11, 9, 7, and 1 to the up (1) position. Set all others to the down (0) position.
001034	100770	Set switches 15 and 8 through 3 to the up (1) position. Set all others to the down (0) position.
001036	116012	Set switches 15, 12, 11, 10, 3, and 1 to the up (1) position. Set all others to the down (0) position.
001040	000002	Set switch 1 to the up (1) position. Set all others to the down (0) position.
001042	000766	Set switches 8 through 4, 2, and 1 to the up (1) position. Set all others to the down (0) position.
001044	017775	Set switches 12 through 2, and 0 to the up (1) position. Set all others to the down (0) position.
001046	002415	Set switches 10, 8, 3, 2, and 0 to the up (1) position. Set all others to the down (0) position.

2. Verify that the bootstrap is properly in memory as follows:

   a. Set the first address in the Switch Register as in Step 1a above.

   b. Press the LOAD ADDR switch.

   c. Display the contents of that address in the Data Register by pressing the EXAM switch.

   d. Compare the number in the Data Register with the value in the appropriate table (A-6 or A-7).

   e. If they are the same, repeat Step 2c until all words have been examined.

f. If not the same, repeat Step 1.

3. Set the starting address in the Switch Register as in Step 1a above.

4. Press the LOAD ADDR switch.

5. Press the rewind button on the cassette drives.

6. Press the START switch.

A.6 PC11 PAPER TAPE BOOTSTRAP LOADER

1. Checking the Bootstrap Loader

   a. Set the ENABLE/HALT switch to HALT to stop any previous program that may be running.

   b. Set the ENABLE/HALT switch to ENABLE.

   c. Set the first address, 37744, in the Switch Register. (Set switches 13 through 5 and switch 2 to the up (1) position; set all other switches to the down (0) position.)

   d. Press the LOAD ADDR switch.

   e. Press the EXAM switch. The data is displayed in the Data Register.

   f. Compare the data displayed to the value given in Table A-8.

   g. If they are the same, repeats Steps e through f until the entire Bootstrap Loader has been checked. If they are not the same, go to Step 2 and enter the loader correctly. If all locations are the same as in Table A-8, the Bootstrap Loader is correctly in memory and Step 2 can be bypassed.

2. Deposit the Bootstrap Loader into memory as follows:

   a. Set the first address, 37744, in the Switch Register as in Step 1c above.

   b. Press the LOAD ADDR switch.

   c. Set the proper contents from Table A-8 in the Switch Register and lift the DEP switch.

   d. Repeat Step 2c until all the instructions have been deposited.

Table A-8
Paper Tape Bootstrap Loader

Location	Contents	Like This
37744	016701	Set switches 12, 11, 10, 8, 7, 6, and 0 to the up (1) position. Set all others to the down (0) position.
37746	000026	Set switches 4, 2, and 1 to the up (1) position. Set all others to the down (0) position.
37750	012702	Set switches 12, 10, 8, 7, 6, and 1 to the up (1) position. Set all others to the down (0) position.
37752	000352	Set switches 7, 6, 5, 3, and 1 to the up (1) position. Set all others to the down (0) position.
37754	005211	Set switches 11, 9, 7, 3, and 0 to the up (1) position. Set all others to the down (0) position.
37756	105711	Set switches 15, 11, 9, 8, 7, 6, 3, and 0 to the up (1) position. Set all others to the down (0) position.
37760	100376	Set switches 15 and 7 through 1 to the up (1) position. Set all others to the down (0) position.
37762	116162	Set switches 15, 12, 11, 10, 6, 5, 4, and 1 to the up (1) position. Set all others to the down (0) position.
37764	000002	Set switch 1 to the up (1) position. Set all others to the down (0) position.
37766	037400	Set switches 13 through 8 to the up (1) position. Set all others to the down (0) position.
37770	005267	Set switches 11, 9, 7, 5, 4, 2, 1, and 0 to the up (1) position. Set all others to the down (0) position.

(continued on next page)

Table A-8 (Cont.)
Paper Tape Bootstrap Loader

Location	Contents	Like This
37772	177756	Set switches 15 through 5, 3, 2, and 1 to the up (1) position. Set all others to the down (0) position.
37774	000765	Set switches 8 through 4, 2, and 0 to the up (1) position. Set all others to the down (0) position.
37776	177550	Set switches 15 through 8, 6, 5, and 3 to the up (1) position. Set all others to the down (0) position.

Repeat Step 1 to verify that the Bootstrap Loader has been correctly deposited.

APPENDIX B

FORMATTING THE RK05 DISK

The following instructions detail the procedure for formatting the RK05 DECpack disk for use on the RK11.

1. Mount the disk to be formatted in Unit 0. The following formatting procedure will work only on Unit 0.

2. Set the ENABLE/HALT switch to HALT to stop any previous program which may be running.

3. Deposit the formatting program into memory as follows:

   a. Set the first address, 001000, in the Switch Register. (Set switch 9 to the up (1) position; set all others to the down (0) position.)

   b. Press the LOAD ADDR switch.

   c. Set the proper contents from Table B-1 in the Switch Register and lift the DEP switch.

   d. Repeat Step c until all the values have been deposited.

4. Verify that the formatting program is properly in memory as follows:

   a. Set the starting address in the Switch Register as in Step 3a above.

   b. Press the LOAD ADDR switch.

   c. Display the contents of that address in the Data Register by pressing the EXAM switch.

   d. Compare the number in the Data Register with the value in Table B-1.

   e. If they are the same, repeat Step 4c until all words have been examined.

   f. If they are not the same, repeat from Step 1.

5. Set the starting address in the Switch Register as in Step 3a above, then press the LOAD ADDR switch.

6. Set the ENABLE/HALT switch to ENABLE.

7. Verify that the disk mounted in Unit 0 is the one to be formatted. If so, WRITE ENABLE Unit 0.

8. Press the START switch. Let the program execute for 60 seconds, then set the ENABLE/HALT switch to HALT to stop the program.

   The disk is now formatted and ready for use.

Table B-1
RK05 Disk Formatting Program

Location	Contents	Like This
001000	012737	Set switches 12, 10, 8, 7, 6, and 4 through 0 to the up (1) position Set all others to the down (0) position..
001002	006003	Set switches 11, 10, 1, and 0 to the up (1) position. Set all others to the down (0) position.
001004	177404	Set switches 15 through 8 and 2 to the up (1) position. Set all others to the down (0) position.
001006	105737	Set switches 15, 11, 9 through 6, and 4 through 0 to the up (1) position. Set all others to the down (0) position.
001010	177404	Set switches 15 through 8 and 2 to the up (1) position. Set all others to the down (0) position.
001012	100375	Set switches 15, 7 through 2, and 0 to the up (1) position. Set all others to the down (0) position.
001014	000137	Set switches 6 and 4 through 0 to the up (1) position. Set all others to the down (0) position.
001016	001000	Set switch 9 to the up (1) position. Set all others to the down (0) position.

Please cut along this line.

READER'S COMMENTS

NOTE: This form is for document comments only. Problems with software should be reported on a Software Problem Report (SPR) form.

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

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

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

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

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code or Country _____

If you require a written reply, please check here. ☐

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Communications
P. O. Box F
Maynard, Massachusetts 01754

digital equipment corporation

Printed in U.S.A.